Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2020

Adaptive Energy Management Strategies for Series Hybrid Electric Wheel Loaders

Carolina Pahkasalo and André Sollander

LINKÖPING
UNIVERSITY

Master of Science Thesis in Electrical Engineering

Adaptive Energy Management Strategies for Series Hybrid Electric Wheel Loaders:

Carolina Pahkasalo and André Sollander

LiTH-ISY-EX--20/5300--SE

Supervisor: **Iman Shafikhani**
ISY, Linköping University
George Babu Jithin
Volvo Construction Equipment

Examiner: **Jan Åslund**
ISY, Linköping University

Division of Vehicular Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

An emerging technology is the hybridization of wheel loaders. Since wheel loaders commonly operate in repetitive cycles it should be possible to use this information to develop an efficient energy management strategy that decreases fuel consumption. The purpose of this thesis is to evaluate if and how this can be done in a real-time online application. The strategy that is developed is based on pattern recognition and Equivalent Consumption Minimization Strategy (ECMS), which together is called Adaptive ECMS (A-ECMS). Pattern recognition uses information about the repetitive cycles and predicts the operating cycle, which can be done with Neural Network or Rule-Based methods. The prediction is then used in ECMS to compute the optimal power distribution of fuel and battery power. For a robust system it is important with stability implementations in ECMS to protect the machine, which can be done by adjusting the cost function that is minimized. The result from these implementations in a quasistatic simulation environment is an improvement in fuel consumption by 7.59 % compared to not utilizing the battery at all.

Acknowledgments

We would like to express our gratitude to our supervisor Iman Shafikhani for the commitment and guidance as well as fast responses with valuable inputs when it was needed the most. We are also grateful for the assistance and encouragement provided by our examiner Jan Åslund during the thesis. We are very thankful for the support given by the both of you, especially during the consequences of the occurring pandemic.

We are also deeply thankful for the opportunity provided by Volvo Construction Equipment to conduct a thesis within such a relevant and interesting area, everything we have learnt during this thesis is invaluable. We would like to thank our supervisor George Babu Jithin for the contributions, assistance and interesting discussions. We would also like to thank Thai Do Hoang, Anders Fröberg and the rest of the staff at the department of Electromobility Systems.

Eskilstuna, June 2020
Carolina Pahkasalo and André Sollander

Contents

Notation

Nomenclature

Notation	Meaning
a	Activation Value Matrix / Tuning Parameter
b	Bias Vector
C	Cost Function
f	State Equation
f_0	Running Cost
f_{ICE}	Engine Map Function
$f(SOC)$	Cost Function Penalty
h	Time Step
H	Hamiltonian
k	Tuning Parameter
m_{fuel}	Fuel Consumption
$\dot{m}_{fuel}$	Fuel Consumption Rate
$I_{battery}$	Battery Current
J^*	Cost-To-Go Function
J	Cost Function
$P_{battery}$	Battery Power
P_{demand}	Power Demand
P_{fuel}	Fuel Power
P_{gen}	Generator Power
P_{ICE}	Engine Power
Q	Battery Capacity
q_{LHV}	Lower Heating Value of Fuel
s	Equivalence factor
$\dot{s}$	Time Derivative of Equivalence factor
$S\dot{O}C$	Time Derivative of SOC
T_{ICE}	Engine Torque
U	System Voltage

Nomenclature

Notation	Meaning
v	Longitudinal Velocity
w	Weight Matrix
X	Characteristics Vector
x	Characteristic / state
y	Prediction
z	Node Value Matrix
α	Learning Rate
β	Tuning Parameter
δ	Node Error
η_{em}	Efficiency of Electric Machine
η_{genset}	Efficiency of Generator and ICE
θ_1	Model Parameters
θ_2	Model Parameters
λ	Adjoint variable
$\dot{\lambda}$	Derivative of Adjoint variable
σ	Sigmoid Function
τ	Threshold
ϕ	Terminal Cost
ω_{ICE}	Engine Speed
$\dot{\omega}_{ICE}$	Angular Acceleration

Abbreviations

Abbreviation	Meaning
A-ECMS	Adaptive ECMS
DP	Dynamic Programming
ECMS	Equivalent Consumption Minimization Strategy
LAC	Load and Carry
LVQ	Learning Vector Quantization
MLP	Multilayer Perceptron
ICE	Internal Combustion Engine
PMP	Pontryagin's Minimum Principle
PR	Pattern Recognition
RB	Rule-Based
SLC	Short Loading Cycle
SOC	State of Charge

1 Introduction

1.1 Background

Higher demands are set on the automotive industry as the requirements for emissions are getting tougher. The automotive industry is under constant development to achieve innovative products that meet said requirements while being highly efficient and satisfy the costumer needs. Hybrid technology is a modern and popular solution to achieve the results needed. A hybrid electric vehicle is a common hybrid technology where a conventional driveline is complemented with an electrical driveline. This means that the hybrid electric vehicle often has an internal combustion engine and a battery, meaning that there are two power sources; carbon based fuel and electric energy.

Heavy machinery is also greatly affected by the stricter requirements which leads to hybrid machines being developed. The hybridization within this area means that emissions can be decreased significantly as well as the fuel consumption, which is beneficial for both the environment and the costumers. Since heavy machinery is used to a greater extent than conventional vehicles it is important that the full potential of the hybrid technology is utilized. To do so a control strategy is needed, which optimally manages the power distribution.

It is common for heavy machinery to operate in the same site for long periods of time where the machine follows the same path continuously. Thus, the machine is mostly used in a similar way, with operation cycles that are repetitive. Repetitive cycles means that the energy usage is similar for each cycle. The prior knowledge of the energy usage for a cycle can be used to develop a control strategy. For a hybrid machine it is possible to predict the optimal battery usage based on the knowledge, which leads to an optimal power usage for the given cycle.

1.2 Problem Formulation

The aim of the master thesis is to develop a control strategy that uses information about the machine's operating cycle to deliver the most energy efficient solution possible. An energy efficient solution means that the best possible system efficiency is used so that the least amount of fuel is needed. Information about the operating cycle is obtained by identifying the cycle using pattern recognition, which is possible due to repetitive operating cycles. Different methods for the control strategy are compared to find an optimal control strategy for the battery usage. An analysis of the available approaches for online implementation is provided as well as a comparison of simulation results from the different strategies. The following questions are to be answered in order to solve the problem formulation:

- How can a repetitive cycle be identified from current and past vehicle states?
- How can information about current drive cycle be used to manage energy consumption?
- How could such an algorithm be implemented in a real-time online application?

1.3 Delimitations

The main delimitations of the thesis are presented below.

- All simulations are Quasistatic.
- The studied machine is a series hybrid electric wheel loader.
- Optimization is only with respect to fuel economy and not components life expectancy nor NO_x emissions.
- The system in consideration only includes internal combustion engine, generator and battery. The effects of the driveline and hydraulics are only present as power demands.
- The final strategy is only tested in a simulation environment and not in a real machine.
- Variations of two different repetitive operating cycles are used.
- The available computational power is limited.

1.4 Outline

The outline of the thesis is shortly described below.

Chapter 2, Vehicle System Description
Relevant theory and information about the vehicle system, i.e. powertrain components, hybrid electric vehicles and wheel loader machines.

Chapter 3, Theoretical Preliminaries
Theory about possible approaches for pattern recognition and control strategies that can be found in related research.

Chapter 4, Implementation
Detailed description of the implementation of the different pattern recognition and control strategy methods.

Chapter 5, Results and Discussion
Tests and validation of the implemented methods, presentation of the results and discussion.

Chapter 6, Conclusion
Drawn conclusions and suggested future work.

2 Vehicle System Description

This chapter presents a general description of the vehicle system, which is needed for understanding the thesis. Information about powertrain components, hybrid configurations and wheel loader operations is presented, as well as relevant academic research in the area of hybrid vehicles and heavy machinery.

2.1 Powertrain Components

The main objective of the powertrain is to convert stored energy into a propulsion force. A hybrid powertrain can be built up including many different components. The components that are used in this thesis are briefly described in the sections below.

2.1.1 Internal Combustion Engine

An Internal Combustion Engine (ICE), also called engine in this thesis, converts carbon based fuel into mechanical power and emissions. There are two main categories of ICE; spark ignited and compression ignited. Spark ignited engines uses a spark plug to ignite the air and fuel mixture. Fuels used in this type of combustion are gasoline, gas and ethanol. Compression ignited engines uses the heat created from the compression to self ignite the air and fuel mixture. Diesel is used as fuel in this type of combustion engine. The theory and models used in this thesis for ICE is based on [1], where modeling and control of engines are described in greater detail.

The main advantages for a diesel engine over a gasoline engine are that it generally has a higher compression ratio, less pumping losses and a leaner air fuel mixture. This leads to a higher efficiency but also more NOx and particle emissions than a gasoline engine.

Model

The internal combustion engine can be modeled with equations (2.1), (2.2) and (2.4). The power delivered by the engine can be described with the following equation, where ω_{ICE} is the engine speed, T_{ICE} is the engine torque and P_{ICE} is the engine power.

$$P_{ICE} = \omega_{ICE} \cdot T_{ICE} \tag{2.1}$$

The engine speed dynamics can be described according Newton's second law as the following equation, where $\dot{\omega}_{ICE}$ is the angular acceleration of the engine, J_{ICE} is the total inertia of the engine and T_{load} the torque load on the engine.

$$\dot{\omega}_{ICE} = \frac{T_{ICE} - T_{load}}{J_{ICE}} \tag{2.2}$$

The fuel consumption of the engine is described by a look up table with engine speed and torque as inputs. The look up table can be described as a non linear function f_{ICE} that can be seen in the following equation, where $\dot{m}_{fuel}$ is the fuel consumption rate.

$$\dot{m}_{fuel} = f_{ICE}(w_{ICE}, T_{ICE}) \tag{2.3}$$

The fuel power needed by the engine can be derived using the lower heating value of the fuel as done in [2]. The lower heating value is measured in Joule per kg and corresponds to the amount of heat that is produced during complete combustion of the fuel. The fuel power is calculated according to the following equation, where P_{fuel} is the fuel power and q_{LHV} is the lower heating value of the fuel.

$$P_{fuel} = \dot{m}_{fuel} \cdot q_{LHV} \tag{2.4}$$

2.1.2 Electric Machines

Most electric machines can be operated as both motors and generators. In motor operation electrical power is converted to mechanical power. In generator operation it converts mechanical power to electrical power. This is a very useful trait for hybrid vehicles where braking energy can be recovered. There are two main categories of electric motors; direct current and alternating current.

Direct current motors are commonly used as starter motors in conventional vehicles and can today also be found in electric and hybrid vehicles. This type of motor is relatively simple and inexpensive. It also requires less complicated control and can use the existing direct current power system. The main disadvantages are that it requires more maintenance and has lower efficiency.

Alternating current motors are better suited for use in electric and hybrid vehicles to produce tractive force. This is discussed in [2] as well as the modelling of electric motors. There are two main categories of alternating current motors; synchronous alternating current motors and asynchronous motors. The synchronous motors rotate with the same speed as the rotating magnetic field. Asynchronous motors do not and this type of motor usually have higher efficiency but requires more complex control and an inverter in order to run it.

Model

The electric machines can be modeled as the following equation, where $P_{mechanical}$ is the mechanical power, $P_{electrical}$ is the electrical power, η_{em} the efficiency of the electric machine.

$$P_{mechanical} = P_{electrical} \cdot \eta_{em}^{sign(P_{electrical})} \tag{2.5}$$

The $sign$ function returns a −1 if the electrical power is negative and a +1 if positive, which means that the efficiency factor switches so that it is either multiplied or divided by depending on the operation mode. This simple model can be used to describe both generator and motor operation.

2.1.3 Batteries

One of the key components in a hybrid electric propulsion system is the electrochemical battery. The theory that is used about batteries in this thesis, which also is used to derive model equations, is presented and discussed in [2]. A battery converts chemical energy into electrical energy and its main purpose is to act as storage for electrical energy. For hybrid electric vehicles it is of interest to know the specific power for the battery, measured in Watts per kilogram. The specific power can be used to calculate maximum acceleration and speed that can be achieved by the vehicle. To describe the remaining capacity of the battery a parameter called State Of Charge (SOC) is used. This parameter is dimensionless and expressed as a percentage of the battery's nominal capacity. The SOC is normally limited to be between 20% and 80% to not damage the battery and extend its life time.

Rechargeable batteries are used for hybrid propulsion systems since it is important for the battery to be used both as energy supplier and storage for recuperated energy. The most common battery technologies according to [2] are lead-acid, nickel-metal hybride, lithium-based, molten salt and metal air. Lead-acid batteries are commonly found in conventional vehicles and also in some early hybrid electric vehicles. The reason for this is that these batteries are robust and reliable while having a low cost. Although, they are limited in their use since the cycle life is low as well as the energy density. Batteries with higher energy density and cycle life are nickel-metal hybride batteries which are suitable for hybrid electric vehicles, but this comes at a higher price. The standard battery for hybrid electric vehicles are lithium-ion based batteries. An interesting battery alternative is the sodium-nickel which is low cost, high cycle life and high specific energy and power, however it is not suitable since it must be operated at high temperatures.

Model

In the powertrain model, a simple battery model is used with the assumption of relatively small and slow changes in SOC over time. Since the importance of the model is to capture the general behaviour rather than having high precision, such a simple model is acceptable. The battery is modeled using the time derivative of SOC, which is described in the following equation that is based on the battery model equations presented in [2], with the assumption that there are no inner losses in the battery.

$$\frac{d}{dt}SOC = S\dot{O}C = -\frac{P_{battery}}{UQ_0} = -\frac{I_{battery}}{Q_0} \tag{2.6}$$

In the equation, $P_{battery}$ is the power demand on the battery, $I_{battery}$ the demanded current on the battery, U the system voltage and Q_0 the battery capacity. The battery capacity determines what electric charge that can be delivered at a certain voltage, which is measured in Ampere-Seconds.

2.2 Hybrid Electric Vehicles

A hybrid vehicle is a vehicle with more than one source of power. Hybrid vehicles are divided in two main categories depending on the configuration of the powertrain; parallel and series. The hybrid vehicle studied in this thesis uses diesel and electric power in a series configuration. One of the main benefits of a hybrid electric vehicle is the ability to use regenerative braking that can recuperate the braking energy. This can be achieved by letting the electric motor work as a generator and apply the braking torque, which results in a current that charges the battery. Hybrid vehicles are briefly described in the following sections, a more detailed description can be found in [2].

Charge sustaining behaviour is of great importance for hybrid vehicles. Charge sustenance means that the initial and terminal SOC level is equal over a drive cycle. If this is not true, there will be instability in the system since the SOC level is accumulated with each cycle, meaning that the battery is either depleted or over-charged.

2.2.1 Parallel

Parallel hybrids are most common in road cars. In the parallel hybrid configuration both the electric motor and the internal combustion engine are working in parallel to produce power to the driveline. This is usually done through a mechanical connection in the gearbox. In Figure 2.1 a simplified parallel configuration is illustrated.

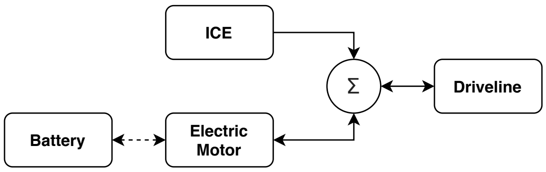


Figure 2.1: *A simplified schematic of a parallel hybrid configuration. A solid line indicates mechanical connection and dashed electrical. The direction of the arrow indicates direction of allowed power flow.*

2.2.2 Series

In the series hybrid configuration the electric motor and the internal combustion engine are working in series. The engine is connected only to the generator which is in turn powering the electric motor together with the battery. This means that the generator is in parallel with the battery and the engine in series with the electric motor. A simplified series configuration is illustrated in Figure 2.2.

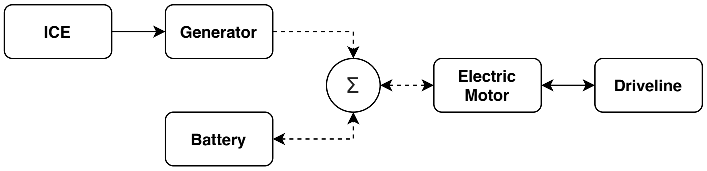


Figure 2.2: *A simplified schematic of a series hybrid configuration. A solid line indicates mechanical connection and dashed electrical. The direction of the arrow indicates direction of allowed power flow.*

2.3 Wheel Loader Operation

Wheel loader machines are used in many different applications including moving material, pallets and timber in varying work sites. The most common operation is moving material such as gravel, sand or rocks. Two common operating cycles are the short loading cycle (SLC) and the load and carry cycle (LaC), both of which are described in the following sections.

2.3.1 Short Loading Cycle

The short loading cycle is a common repetitive drive cycle for wheel loaders. It is used when the load carrier is located in close proximity of the material that is loaded. This cycle is highly transient with a lot of direction changes. In [3] and [4] the cycle is explained further and different detection methods for the cycle are implemented. In Figure 2.3 a simplified schematic of a wheel loader operating in the short loading cycle is illustrated.

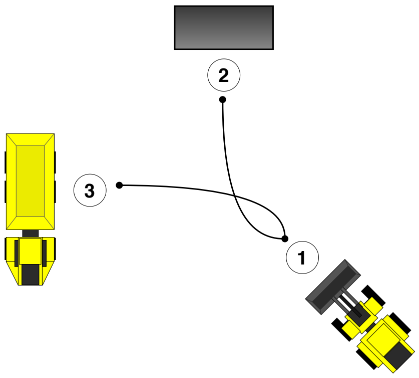


Figure 2.3: *A simplified schematic of a wheel loader operating in the short loading cycle. The motions of the cycle are described in Table 2.1.*

The short loading cycle consists of the motions described in Table 2.1, where the steps correspond to the numbers in Figure 2.3.

Table 2.1: *Motions of the short loading cycle.*

Step	Motion
1-2	Drive into the pile
2-1	Reverse back to the start point
1-3	Drive to the carrier and unload
3-1	Reverse back to the start point

2.3.2 Load and Carry Cycle

Load and carry is another common repetitive operating cycle for wheel loaders. It is used when the drop off point for the material is not located in close proximity to the material that is moved, which means that this type of cycle has higher mean velocity than a short loading cycle. This cycle is treated in [3] as well. A simplified schematic of a wheel loader operating in the load and carry cycle can be seen in Figure 2.4.

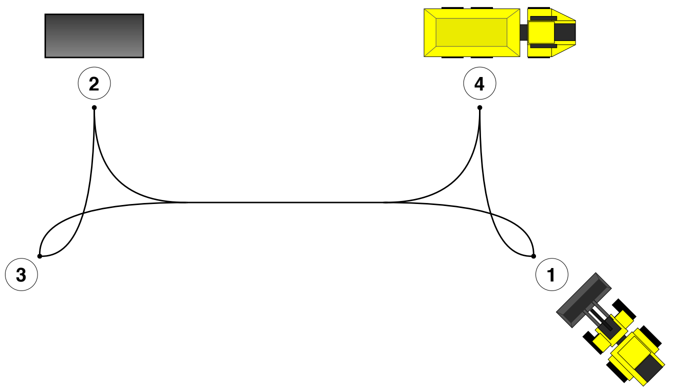


Figure 2.4: *A simplified schematic of a wheel loader operate in the load and carry cycle. The motions of the cycle are described in Table 2.2.*

The load and carry cycle consists of the motions described in Table 2.2, where the steps correspond to the numbers in Figure 2.4.

Table 2.2: *Motions of the load and carry cycle*

Step	Motion
1-2	Drive into the pile
2-3	Reverse back and turn around
3-4	Drive to the carrier and unload
4-1	Reverse back to the start point

3 Theoretical Preliminaries

Pattern recognition together with a control strategy is used to derive an optimal adaptive energy management strategy for a hybrid machine. Theory based on related research is presented in this chapter to describe different possible approaches and methods that can be used.

3.1 Pattern Recognition

Wheel loaders work in many different cycles with varying demands. In order to optimize the machine for multiple cycles the machine needs to identify what cycle the machine is undergoing. There are many methods that can be used for this application, in this thesis two main categories of methods of pattern recognition are discussed; rule-based and neural networks. The main idea is to use logged machine data to predict what cycle the machine is undergoing.

3.1.1 Rule-Based

Rule-based (RB) cycle identification is the most intuitive approach and can be simple to implement. The main idea is to use logged data and comparing it to characteristics that is considered to represent different cycles. The cycle that represents the data best is selected as the prediction. The main drawback with this approach is that it requires expert knowledge in what differentiates the different cycles in order the get good performance. A simple method for wheel loader applications that is described in [4] is to continuously integrate the velocity signal and compare it to a predetermined threshold. An even simpler rule-based method could be to continuously calculate the mean of some logged data and compare it to a predetermined threshold.

A more sophisticated rule-based pattern detection algorithm is also developed in [4] for identification and localization of the short loading cycle for wheel loaders. By identifying predetermined events such as changing directions or tilting the bucket and saving it in sequence, the cycle can be identified by comparing it to predefined automata cycles. An automata cycle is a predefined graph of how events are connected in a known drive cycle.

3.1.2 Neural Networks

Neural networks is a well studied field with many different methods to choose from, some of which are described in [5]. A neural network can be described as a universal function approximator. In this thesis two different methods are analyzed. The first method is multilayer perceptron network (MLP), which is the standard method for neural networks. MLP networks are not commonly used for cycle detection in automotive research, however it has shown potential in other applications, such as in [6] where a MLP network is successfully used to control a spark ignition engine to predict the engine brake power. The second method is learning vector quantization (LVQ), which has been used in similar applications in multiple research studies. In both [7] and [8] the LVQ network has been used in vehicle application with fuel consumption improvements.

An extensive comparison between these two neural networks is executed in [9] based on the performance when used for automatic speech recognition. In this comparison the radial basis function network is also treated. Radial basis function networks are not discussed in this thesis but could be an alternative approach since it shows potential in [10] where it predicts the energy demand for a plugin hybrid electric vehicle. The conclusions drawn about LVQ and MLP from the comparison study are that:

- LVQ is fast to train and to use online, but has lower accuracy.
- MLP has higher accuracy, but requires more computational time.
- MLP is better than LVQ at learning from very nonlinear data, due to its activation function and the ability to add more hidden layers.

Training a neural network can be done with many different methods depending on the choice of network. Figure 3.1 shows a simplified flowchart of how a neural network is trained in general. Each step in the flowchart is described by different functions for different methods. In the first step the network makes a prediction on a certain set of inputs. In the next step the prediction error is calculated. Lastly, systematic changes to the network are made depending on the error. This procedure is repeated as many times as the user requires for the network to learn the data satisfactory.

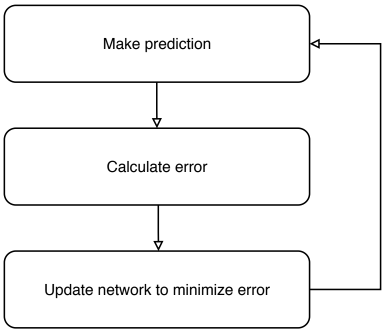


Figure 3.1: *A flowchart of how a neural network is trained.*

3.1.2.1 Multilayer Perceptron

A multilayer perceptron neural network, sometimes referred to as a standard neural network or artificial neural network, is the most common way of implementing a neural network today. The multilayer perceptron architecture is described in details in [5] and [11]. There are three main types of layers in a MLP network; input, hidden and output layers. A network can contain an arbitrary number of nodes and hidden layers, where each layer contains a predetermined amount of nodes. An example of a MLP network can be seen in Figure 3.2. The input layer contains all input nodes where the vector input enters the network. The output layer contains all the output nodes where the output prediction exits the network. All nodes are assigned biases and each of the connections between the nodes have a corresponding weight, which are used to make predictions. Backpropagation and feed forward calculations are performed in both the hidden and output layer in order to obtain the optimal weights and biases.

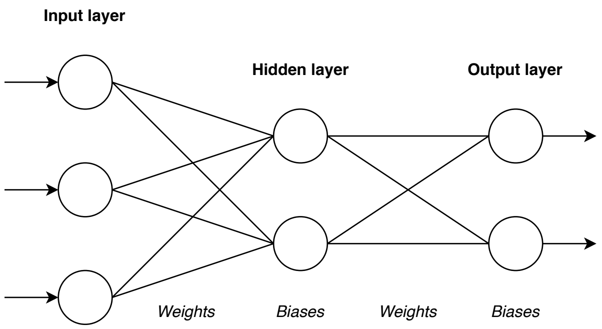


Figure 3.2: *An example of a multilayer perceptron neural network, with 3 nodes in the input layer, 2 nodes in the single hidden layer and 2 nodes in the output layer.*

Feed Forward

Feed forward describes how the input propagates through the network. The equations used for feed forward are based on the method described in [5]. Each connection between nodes in different layers has an associated weight w_{ij} and each node has a bias b_j, where the layers are denoted i and nodes j. The weights and biases are used to calculate the value at each node z_i changes, using the following formula.

$$z_i = \sum w_{ij} a_{ij} + b_j \tag{3.1}$$

The value calculated in each node is then filtered through an activation function in order to obtain its output. The activation function is important since it makes the network able to model non-linear behavior. A function often used for this is a sigmoid function that can be seen in the following equation.

$$a_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}} \tag{3.2}$$

A cost function is needed to determine the performance of the network. This function compares the correct value to the predicted output from the output layer, where $y_{predicted} = a_L$. The mean square error is a general cost function that works for most cases. This cost function is often used for regression problems, such a cost function can be seen in the following equation, where n is the number of training cases.

$$C = \frac{1}{n} \sum \left(y_{true} - y_{predicted} \right)^2 \tag{3.3}$$

For multiple classification problems the cross entropy cost function is often used instead of mean square error. The cross entropy function is described in the following equation, where log is the natural logarithm and n the number of training cases.

$$C = -\frac{1}{n} \sum \left(y_{true} \log(y_{predicted}) + (1 - y_{true}) \log(1 - y_{predicted}) \right) \tag{3.4}$$

Backpropagation

Backpropagation is used to train the network by changing the weights and biases depending on the performance of the network. The equations that are used for backpropagation are based on the method that is described in [5]. The error δ

in each node in the output layer L can be calculated with the following equation, where the chain rule is used to split the derivations into two analytically derivable parts.

$$\delta^L = \frac{\partial C}{\partial z^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \tag{3.5}$$

The error is also calculated for each of the hidden layers L, which is done with the following equation, where T is the transpose and $.*$ denotes element wise multiplication.

$$\delta^L = ((w^{L+1})^T \delta^{L+1}) .* \frac{\partial a^L}{\partial z^L} \tag{3.6}$$

The error in each node can then be used to calculate the gradient of the cost function with respect to the weights and biases which is needed to determine how the weights and biases should be changed. The gradients of the cost function are calculated using the following equations.

$$\frac{\partial C}{\partial w_j^L} = a_i^{L-1} \delta_j^L \tag{3.7}$$

$$\frac{\partial C}{\partial b_j^L} = \delta_j^L \tag{3.8}$$

The method of steepest decent is used to calculate how much the weights and biases should change, which includes the gradient of the cost function. The following equations describe how they are changed.

$$w_j^L = w_j^L - \alpha \frac{\partial C}{\partial w_j^L} \tag{3.9}$$

$$b_j^L = b_j^L - \alpha \frac{\partial C}{\partial b_j^L} \tag{3.10}$$

The step length in the negative gradient direction is called the learning rate α. A large learning rate leads to fast learning but lower accuracy and a too small value might not lead to weights and biases converging in a reasonable time frame. The backpropagation algorithm is used for each training input. The number of times the network is trained on the same input data is called epochs. Too many epochs might lead to the data being overfitted and too few can lead to a network that has not learned enough.

3.1.2.2 Learning Vector Quantization

Learning Vector Quantization (LVQ) is a special case of a neural network in a family called competitive networks. In this type of network the winning node takes it all, where all output nodes except the winner will output zero. More theory about the LVQ architecture and other competitive networks can be found in [5].

There are multiple articles related to the area of implementing a LVQ network for driving cycle identification. In both [7] and [8] LVQ is used with 10 input parameters to characterize 4 different drive cycles with success for road vehicles. The main difference between these are how the training data is used, which is discussed later in this section.

An example of a LVQ network can be seen in Figure 3.3. The input layer contains all input nodes where the vector input enters the network. The output layer contains all the output nodes where the output prediction exits the network. Calculations are made in both the competitive and output layer. The output layer is often a feed forward layer, similar to MLP.

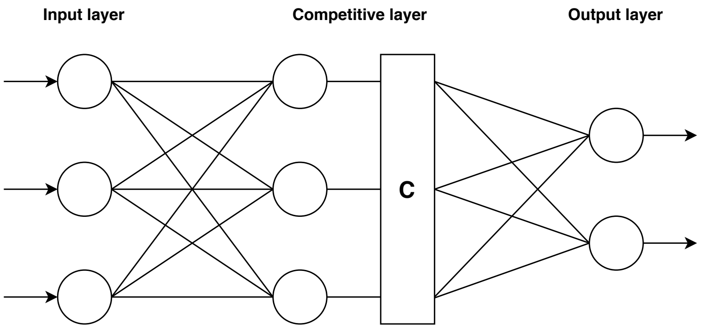


Figure 3.3: *An example of a LVQ network, with 3 nodes in the input layer, 3 nodes in the competitive layer and 2 nodes in the output layer.*

Competitive Layer

The competitive layer is generally calculated according to equation (3.11) as mentioned in [5], where the euclidean distance z between the input vector x and the weights w is calculated for each node j . The distance is often calculated with the second norm, but in some application there might be other alternatives. The main benefit of using the second norm is that there is no need to normalize the input data.

$$z_j = \|x - w_j\| = \sqrt{\sum (x - w_j)^2} \tag{3.11}$$

The node with the weights that have the shortest distance to input vector, i.e. the smallest value is the winning node. All nodes will give outputs according to the following equation.

$$a_j = \begin{cases} 1 & \text{for} \quad j = argmin(z_j) \\ 0 & \text{else} \end{cases} \tag{3.12}$$

It is an essential part of the competitive layer that only the winning node has the output 1. All output is then sent to an output feed forward layer, where the final prediction can be made.

Kohonen Rule

Depending on if the prediction is correct or not the weight for each connection is changed according to equation (3.13). This method is called the Kohonen Rule, as described in [5].

$$w_{ij} = \begin{cases} w_j + \alpha(x_j - w_j) & \text{if} \quad a = y_{true} \\ w_j - \alpha(x_j - w_j) & \text{else} \end{cases} \tag{3.13}$$

In the equation, α is the learning rate, w the weights, x the input, a the prediction and y_{true} the correct answer. This rule moves the weights closer to the input if the prediction is correct and further away if it is wrong. The training process can either be done for a predetermined amount of iterations or dynamically with a decreasing learning rate.

The main disadvantage with Kohonen rule is that it is very sensitive to the initial values of the weights. This is due to it getting stuck in local optimums instead of finding the global optimum. This can be solved by using random initial weights and retraining the network until it produces satisfactory predictions. The initial weights can also be set by looking at the means of each characteristic and for each class in the training data. Another method that can be used for setting better initial weights are bees algorithm that is described in the following paragraph.

Bees Algorithm

Another algorithm for training a LVQ network is the bees algorithm that is presented in [12]. This algorithm tries to imitate how bees learn and search for nectar. The algorithm is comprised by the following steps.

1. Generate an initial population of networks.

2. Run all of the training data and calculate the error for each network.

3. Create a new population of networks in the neighbourhoods of the best performing networks in the previous population and a few networks randomly as scouts.

4. Go to step 2 and repeat until the error of the best network is small enough.

The purpose of the scout networks is to search for the global optimum instead of getting stuck in a local optimum. The mean square error function described in equation (3.3) can be used to calculate the error for each network. This method of training is best suited for neural networks that are computationally fast. This method can be used when the Kohonen rule has problem with converging.

3.1.3 Training Data

The set of training data needed to train a neural network can be created from the actual cycle data using different methods. In all methods the common step is to determine the cycle characteristics by analyzing parameters calculated from the cycle data. The cycle characteristics are used to reduce the time series data into a set of parameters that represent the cycles, which then can be used to identify the cycles. Often mean, max and standard deviation of velocity and acceleration is used as characteristics. Different forces and powers can also be used. There are two ways of choosing how to use the characteristics and both of these are discussed in the following paragraphs.

The first way is to use statistical analysis of all the characteristic parameters for all full cycles, this method is implemented in [7]. The statistical analysis is used to classify each parameter into one of four levels, each level is represented by a unique vector containing different combinations of the values -1 and 1. Each parameter is represented according to the levels presented in equation (3.14), where β is a tuning parameter usually between 1 and 0, p the parameter, p_{std} the standard deviation of the parameter and p_{avg} the average. This method has the benefit of increasing the number of inputs to the network by using the level vector representation of each parameter as input, i.e. increasing the number of inputs by a factor of 3.

$$\begin{pmatrix} L1 \\ L2 \\ L3 \\ L4 \end{pmatrix} = \begin{cases} [1,1,1] & \text{if} \quad p > p_{avg} + \beta p_{std} \\ [1,1,-1] & \text{if} \quad p_{avg} < p \leq p_{avg} + \beta p_{std} \\ [1,-1,-1] & \text{if} \quad p_{avg} - \beta p_{std} < p \leq p_{avg} \\ [-1,-1,-1] & \text{if} \quad p \leq p_{avg} - \beta p_{std} \end{cases} \tag{3.14}$$

The different levels of the input data can be illustrated as in Figure 3.4 when the data is normally distributed.

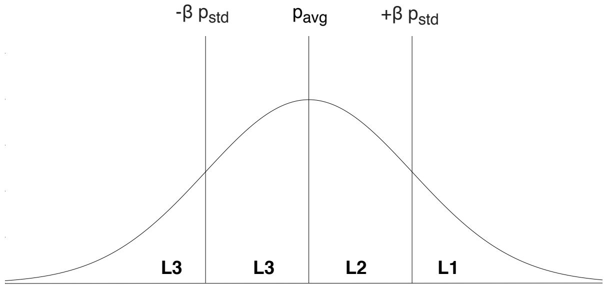


Figure 3.4: *An illustration of the different levels of input data when it is normally distributed.*

The second way is to use a set number of randomly selected subsections of the cycle with fixed time length and calculate the parameters for each subsection. This method shows potential when implemented in [8]. The main benefit of this method is that an arbitrary amount of input data can be generated. The drawbacks are that the risk of overfitting increases and the parameters in each subsection might not be representative for the whole cycle.

3.1.4 Performance Indices

Unified performance indices have to be used in order to quantify the performance of different pattern recognition algorithms. The indices described in this subsection are commonly used in the pattern recognition field and are presented in [11]. To calculate the performance of the algorithm each prediction must be classified according to Table 3.1 first. The classified predictions are then counted for each class, e.g. the total number of true positive predictions.

Table 3.1: *Classification table for predictions, which is used to determine the performance of an algorithm.*

	Actual: 1	Actual: 0
Prediction: 1	True Positive	False Positive
Prediction: 0	False Negative	True Negative

The accuracy of an algorithm is calculated according to the following equation. The accuracy determines how often the algorithm makes the correct prediction.

$$\text{Accuracy} = \frac{\sum \text{True Positive} + \sum \text{True Negative}}{\text{Total Number of Predictions}} \tag{3.15}$$

Precision describes how often the positive predictions by the algorithm are correct. This can tell if the algorithm has a bias towards making positive predictions. The precision of an algorithm is described by the following equation.

$$\text{Precision} = \frac{\sum \text{True Positive}}{\sum \text{True Positive} + \sum \text{False Positive}} \tag{3.16}$$

Recall describes how often positive examples are miss-classified as negative by the algorithm. This can show if the algorithm has a bias towards making negative predictions. The recall of an algorithm is described by the following equation.

$$\text{Recall} = \frac{\sum \text{True Positive}}{\sum \text{True Positive} + \sum \text{False Negative}} \tag{3.17}$$

By combining precision and recall in a harmonic mean according to the following equation a performance index called F_1 Score can be derived. The F_1 Score generates an index between 0 and 1, where 1 is the best possible outcome and 0 the worst. This is one of the most fair comparison indices for different algorithms.

$$\text{F}_1 \text{ Score} = 2 * \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.18}$$

3.2 Control Strategy

To obtain an optimal energy management strategy for the hybrid machine the control strategy must be an optimal control method. The general optimal control problem consists of a cost function that is either minimized or maximized with respect to a set of constraints. In discrete time an optimal control problem is generally formulated as

$$\begin{aligned} &\text{minimize } \phi(x_N) + \textstyle\sum_{k=0}^{N-1} f_0(k, x_k, u_k) \\ &\text{subject to } \begin{cases} x_{k+1} = f_k(x_k, u_k) \\ x_0 \text{ given,} \\ u_k \in U(k, x_k) \\ x_k \in X(k, x_k) \end{cases} \end{aligned} \tag{3.19}$$

where x_k is the states and u_k the control signal for the time step k. The final cost $\phi(x_N)$ defines the terminal cost at the final stage. The state equations $f_k(x_k, u_k)$ describe the system dynamics and $f_0(k, x_k, u_k)$ is the running cost.

Two of the most common optimal control methods are Dynamic Programming (DP) and Pontryagin's Minimum Principle (PMP), which are discussed in [2]. For real-time implementations where computational efficiency is critical, PMP provides a beneficial framework. Since DP is computationally heavy it is not suitable for such applications. However, the DP solution is globally optimal and can therefore be used as a benchmark.

Another powerful control strategy that could be of interest is Model Predictive Control that is described in details in [13]. This control strategy solves an optimization problem and predicts future control trajectories for a given prediction horizon. Model Predicitve Control has been successfully implemented in some research studies for hybrid vehicles, e.g. [14] and [15]. However, this control strategy is not included in this thesis.

3.2.1 Dynamic Programming

Dynamic Programming is a powerful optimal control strategy. As presented in [16], the strategy finds the global optimal solution for a given system by minimizing a certain cost function. It is commonly used in automotive research, as discussed in [2]. A disadvantage with DP is that all information, including disturbances, must be known a priori. This, in combination with the computational complexity, makes DP unsuitable for real-time online applications. However, it is useful to use DP as a benchmark to compare and validate other derived control strategies. The advantage of DP is that it guarantees a globally optimal solution and has the ability to take into consideration constraints on states and inputs. Both deterministic and stochastic approaches can be used and the problem can be either discrete or continuous.

If there are uncertainties in the problem formulation it is necessary to use Stochastic Dynamic Programming. This approach assumes that there are random disturbances, which results in an optimal policy of how to operate depending on the uncertainties. This method has been implemented in three different ways and tested on hybrid wheel loaders in [17] with promising results. Another interesting implementation of Stochastic Dynamic Programming is to take into consideration the prices for fuel and electricity as done in [18]. However, such an approach leads to results from a more financial point of view.

Method

The DP method considers the cost from each state at one time stage to every possible state in the next stage, as illustrated in Figure 3.5. This is done for all stages from start, $k = 0$, to finish, $k = N$.

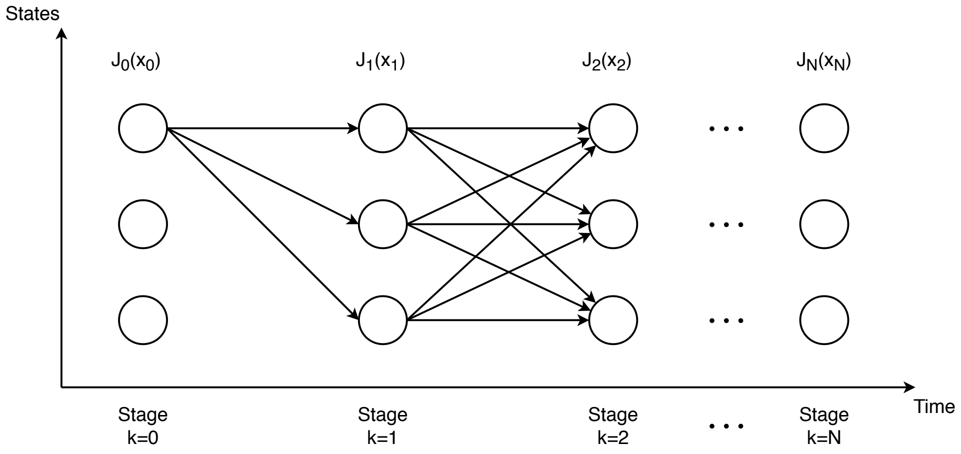


Figure 3.5: *An illustration of how the dynamic programming method works. The cost, $J_k(x_k)$, from each stage k to the next is calculated.*

DP solves an optimal control problem which in discrete time generally is formulated as presented in equation (3.19). A function that describes the cost between all stages is defined as the optimal cost-to-go function as follows

$$J^*(n, x) = \text{minimize } \phi(x_N) + \sum_{k=n}^{N-1} f_0(k, x_k, u_k), \tag{3.20}$$

The cost-to-go function minimizes the expression for all admissible controls. The function is then stored for all admissible controls. The DP equations can be derived with the principle of optimality and the Hamiltonian Jacobian Bellman equation. This derivation shows that the problem can be solved by computing backwards, which is done using the backwards dynamic programming recursion. The definition of the backwards recursion is

$$\begin{aligned} J(N, x) &= \phi(x) \\ J(n, x) &= \min_{u \in U(n,x)} \{f_0(n, x, u) + J(n+1, f(n, x, u))\} \end{aligned} \tag{3.21}$$

For each stage the cost is minimized and the location of the minimum is stored as a path containing the indices. A vector that describes how the value of the cost function changes with the index is stored as well. DP requires some post processing which is how the optimal policy is obtained. First, the initial boundaries are set. The optimal policy is obtained by iterating through the path that is provided by the DP solver, until the final state is reached. A flowchart for the DP method is presented in Figure 3.6 to illustrate the process of the method.

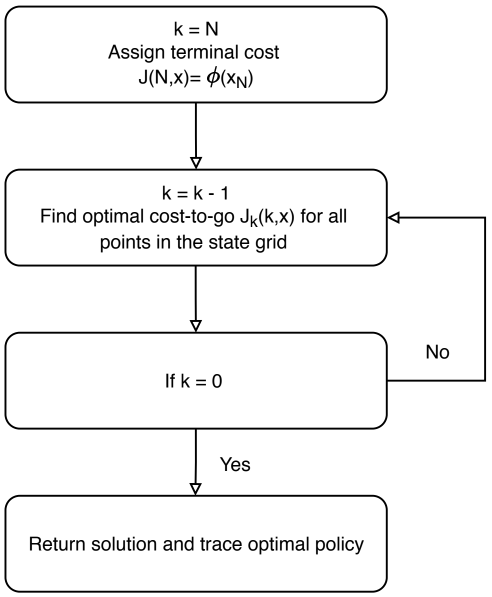

k = N
Assign terminal cost
J(N,x)= ϕ(x$_N$)
k = k - 1
Find optimal cost-to-go J$_k$(k,x) for all points in the state grid
If k = 0
No
Yes
Return solution and trace optimal policy

Figure 3.6: *Flowchart for dynamic programming.*

3.2.2 Pontryagin's Minimum Principle

Pontryagin's Minimum Principle finds the optimal solution to go from one state to the next with respect to the constraints in the problem formulation. PMP is explained in greater detail in [19], where it is also demonstrated how the method can be used for energy management in hybrid electric vehicles. In short, the optimal solution is obtained by performing pointwise minimization followed by solving a two point boundary problem. In a PMP problem the cost function is the Hamiltonian, which in discrete time is defined as

$$H(k,x,u,\lambda) = f_0(k,x,u) + \lambda^T f(k,x,u) \tag{3.22}$$

where $f_0(k,x,u)$ and $f(k,x,u)$ are defined as in equation (3.19) and λ is the adjoint variable. The adjoint variable can be considered a constant as long as the adjoint equation is zero. The adjoint equation is presented in the following equation and is the derivative of the Hamiltonian with respect to the states.

$$\dot{\lambda} = -\frac{\partial H(k,x,u,\lambda)}{\partial x} \tag{3.23}$$

3.2.3 Equivalent Consumption Minimization Strategy

Equivalent Consumption Minimization Strategy (ECMS) is one of the most commonly used control strategies for real time energy management strategies in automotive research for hybrid vehicles. It is a method with much less computational burden than DP and various adaptivity approaches can be used to reach better performance. For example, in [20] the ECMS strategy is used with cycle prediction, resulting in significant fuel consumption improvements. Successful results are also obtained in [10] by predicting the energy demand together with ECMS.

Theory about the strategy is described and discussed in [2]. The strategy is derived from PMP which guarantees an optimal solution for a certain cost function. The cost function that is used in ECMS is the Hamiltonian that is presented in the following equation, where P_{fuel} is the fuel power, $P_{battery}$ the battery power and s the equivalence factor. The equations for the different power sources depend on the design of the powertrain, i.e. the hybrid configuration and components. The specific equations that are used for the machine in this thesis are presented in the ECMS implementation in Chapter 4.

$$H = P_{fuel} + s \cdot P_{battery} \tag{3.24}$$

The strategy compares the fuel power to the electric power to find the optimal power distribution, which in the case of a series hybrid is distribution of the generator and battery power. In ECMS an equivalence factor is introduced in the cost function to convert the battery power to an equivalent fuel power since these are not directly comparable otherwise. The equivalence factor is a dimensionless scaling of the adjoint variable λ. When SOC is used as state the equivalence factor is defined as

$$s = -\lambda \frac{q_{LHV}}{U \cdot Q_0} \tag{3.25}$$

where q_{LHV} is the lower heating value of the fuel, U is the voltage of the battery and Q_0 the capacity of the battery. There are as many equivalence factors as there are states. This means that for a series hybrid that uses engine speed and SOC as states, there are two equivalence factors. Note that in the case of a series hybrid the equivalence factor for engine speed is neglected, i.e. set to zero, in the cost function that is used in this thesis. This means that there is no added cost for changing the engine speed.

The equivalence factor is needed to attain an optimal solution and is crucial to achieve charge sustaining results. For a known drive cycle the equivalence factor that results in charge sustenance can be obtained by systematic optimization. There can be different approaches to select the optimal equivalence factor. One approach is to use a value for discharging scenarios and another for charging. However, in this thesis the equivalence factor is assumed to be equal for both charging and discharging.

3.2.3.1 Adaptivity

The simplest version of ECMS is to assume that a constant equivalence factor is suitable for the total drive cycle, unfortunately this is not always true. Plenty of research has been made on implementing adaptivity to the strategy by continuously updating the equivalence factor. A strategy called A-ECMS is presented in [21], where no a priori knowledge of the drive cycle is needed. The result is an optimal solution close to the DP solution, i.e. the globally optimal solution. However, this study and most other studies have the focus on not knowing the drive cycle in advance, which is not a problem in this thesis. The adaptivity that is needed and should be implemented in this thesis is to make sure that a stable or charge sustaining result is attained despite uncertainties. The method for adaptivity depends on if it is desirable to improve the efficiency or the stability.

Improving Efficiency

There are different approaches to implement adaptivity for efficiency improvement. As discussed in [22], there are three main approaches. The first approach is to use past drive cycle information which is what pattern recognition does. The equivalence factor can be pre-computed offline for a suitable amount of representative cases. The pre-computed equivalence factors can be used by for example a neural network that systematically identifies the drive cycle and applies the correct value.

The next approach is to use both past and present information. This is done by adjusting the equivalence factor until it represents the current drive cycle. This is possible by making a function with the equivalence factor that represents the fuel and electric energy at current state. A problem with this approach is that a charge sustaining result cannot be obtained since no future prediction is made.

The third approach uses past, present and future information. The future information is retrieved by predicting and estimating the drive cycle and by this approach a stable and charge sustaining result can be attained.

Another approach of interest, as presented in [10], is to predict the energy demand based on route prediction by pattern recognition. A reference SOC profile can then be tracked with a feed forward PI-controller. This approach can also improve stability, under the assumption that the reference SOC profile is stable. Since the energy demand is not predicted in this thesis, this approach is not used.

Improving Stability

Stability can be improved by implementing an adaptive method that results in a robust control strategy. One method that results in a stable battery usage is to implement a PI-controller that carefully helps the SOC to reach a desired value. This measure could advantageously be used in combination with another adaptive approach with focus on improving the efficiency. Though, this approach can result in increased fuel consumption since the controller limits the battery usage even if not desired.

By adjusting the cost function according to equation (3.26) the stability can be improved as well. The term added to the cost function depends on the SOC which makes it possible to penalize based on the deviations in SOC. If a suitable additive term is chosen the results can be a stable SOC profile that does not exceed limitations.

$$H = P_{fuel} + s \cdot P_{battery} + f(SOC) \tag{3.26}$$

The cost function can also be adjusted by multiplying the battery power with a factor, as presented in [19]. Depending on the SOC deviation from a goal SOC this multiplying factor compensates the battery part of the equation so that discharging or charging occurs when necessary. This adjustment would result in the following cost function.

$$H = P_{fuel} + s \cdot P_{battery} \cdot f(SOC) \tag{3.27}$$

4 Implementation

The relevant and possible approaches of implementing an adaptive energy management strategy are described in this chapter. The simulation environment that is used is also presented.

4.1 Simulation Environment

This section describes the implementation and design of the simulation environment, which is implemented in Matlab and Simulink. The main objective of the simulation environment is to test the implemented strategy and to compute the corresponding fuel consumption and SOC profile. The simulation environment is represented by the flowchart in Figure 4.1. The simulation environment uses drive cycle information, such as velocity and power demand, as input to the strategy that finds the optimal power distribution. A simple powertrain model is used to calculate fuel consumption and SOC profile. The SOC is used as feedback to the strategy for stability.

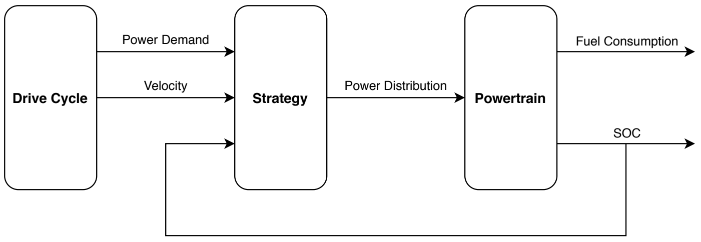


Figure 4.1: *A flowchart of the simulation environment that is used to test implemented strategies.*

Logged machine data is used as input to the simulation model of the powertrain and to the strategy. The simulation model of the powertrain consists of an internal combustion engine, a generator and a battery model.

The strategy part of the simulation environment is illustrated in Figure 4.2. The pattern recognition block uses characteristics from the drive cycle data to identify the current operating cycle, velocity is used as characteristics which is discussed further in section 4.3.2. The block called optimal equivalence factor selector uses the cycle prediction to select an optimal equivalence factor, which is then forwarded to the ECMS block that calculates the optimal power distribution. Training the neural networks and numerically calculating the equivalence factor are offline computations that are executed as well, which is illustrated by dotted lines in the figure.

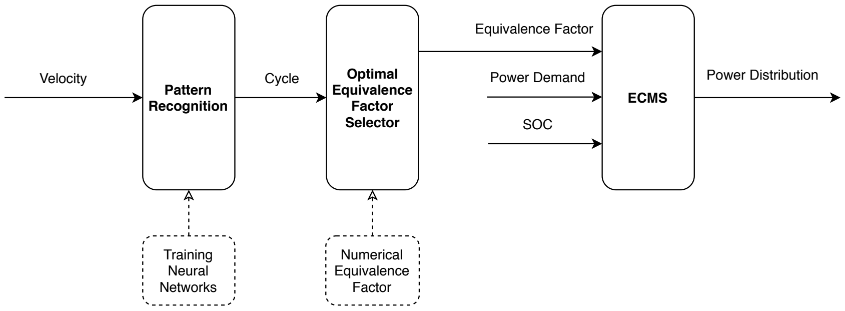


Figure 4.2: *A flowchart of the strategy block in the simulation environment, which consists of pattern recognition, equivalence factor selector and ECMS. Dotted lines indicate offline computations.*

4.1.1 Quasistatic Simulation

The system is modeled quasistatic instead of dynamic to minimize the computational load. A quasistatic simulation is performed by assuming the system to be piece-wise constant during each time step. This type of simulation calculates the power demand from the drive cycle and continues backwards through the powertrain to determine the fuel consumption and SOC profile, see Figure 4.3.

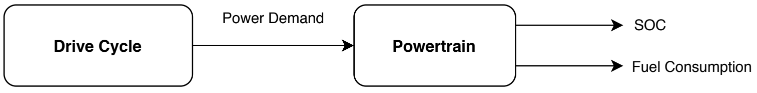


Figure 4.3: *A flowchart of a quasistatic simulation, where fuel consumption and SOC profile are determined by backwards calculations from the drive cycle and powertrain.*

4.2 Data Processing

The pre-processing of the cycle data consists of dividing the raw data into clean cycles, meaning that there are no additional standstills, transportations etc. If the raw data is too noisy it is necessary to filter and remove the noise when using it as input to the pattern recognition algorithms. However, this is not necessary for the data that is used in this thesis.

The processed data can then be used to create test data sets for testing the implemented methods. The test data is combined and configured in different ways for different tests. This is to make the tests representative and testable in a reasonable time frame. All test data cycles used in this thesis are described in Table 4.1.

Table 4.1: *Description of all the test data.*

Test Data	Length	Description of Cycle
SLC	52 s	A representative SLC
LaC	110 s	A representative LaC
SLC All	20 min	All SLC combined
LaC All	15 min	All LaC combined
Main Cycle 1	35 min	SLC All and LaC All combined
Main Cycle 2	8 h	12 SLC All and 17 LaC All combined

4.3 Pattern Recognition

Four different pattern recognition algorithms are developed for cycle detection; three neural networks and one rule-based approach. Firstly, the cycle characteristics must be chosen in order to use pattern recognition, since it is used as input. Training data that represent the cycles must be collected and processed as well, which is used by the pattern recognition algorithm in order to learn to identify the correct output.

4.3.1 Training Data

The training data used for the different pattern recognition algorithms is derived from logged machine data from many variations of the two cycles. The algorithms are tested on unseen data, which is data that has not been used for training. The unseen data is obtained by splitting the logged machine data into two sets; training and testing set. The split is that about 60% of the data is used for training and 40% for testing. This split is important in order to see that the algorithms are well generalized. The training data set is then divided into random sections with fixed time length in order to increase the amount of unique data.

4.3.2 Cycle Characteristics

The cycle characteristics that are used as input to the different pattern recognition algorithms are chosen based on comparisons of available data that is collected from the machine during operation. The data that differentiates the most between the two different cycles is chosen as characteristics.

When analyzing the parameters it can be stated that velocity differentiates the most when comparing the two different operating cycles SLC and LaC. Power and force could be potential characteristics, but in comparison to velocity the difference between the two cycles is much smaller. Thus, power and force characteristics are not used. The chosen cycle characteristics are described in the following equation, where v is the velocity of the machine in the longitudinal direction. The standard deviation and variance of the velocity is used as well as its maximum and mean.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} max(v) \\ mean(v) \\ std(v) \\ var(v) \end{bmatrix} \tag{4.1}$$

If other parameters are chosen it might be necessary to normalize them before using them as characteristics, this is to increase the performance of training. Normalization is needed if the amplitude varies a lot between different characteristics, e.g. if power and velocity is used.

4.3.3 Rule-Based

The rule-based pattern recognition algorithm is implemented as described in the following equation, where y is the prediction, x_2 the mean velocity and τ the threshold.

$$y = \begin{cases} 0 & \text{if } x_2 < \tau \\ 1 & \text{if } x_2 \geq \tau \end{cases} \tag{4.2}$$

The predictions are discrete, where 0 represents SLC and 1 represents LaC. The threshold is determined empirically by looking at the general mean velocity for the two different cycles.

4.3.4 Multilayer Perceptron

Two different configurations of MLP are implemented and described in this section. The main differences are the number of hidden layers used in the network, where the first configuration has one hidden layer and the other one has two. Both configurations give predictions y_1 and y_2 as the probability of the cycle being a SLC or LaC respectively. The output can only take on values between 0 and 1 due to the sigmoid activation function. The output can be interpreted as 1 meaning that there are 100% probability of the output being of that class and

0 that there are 0% probability. All hidden and output nodes in both networks use the sigmoid function as activation function. The predictions are discretized according to the following equation when probabilistic outputs are not needed.

$$y = \begin{cases} 0.5 & \text{if } y_1 < 0.5 \text{ and } y_2 < 0.5 \text{ or } y_1 = y_2 \\ 0 & \text{if } y_1 > y_2 \\ 1 & \text{if } y_1 < y_2 \end{cases} \tag{4.3}$$

In the discretized prediction 0.5 represents the scenario of when the network is uncertain of the operating cycle. The output 0 corresponds to SLC, which means that the SLC probability is higher than the LaC probability. The output is 1 if the prediction is LaC, which means that the LaC probability is higher than the SLC probability.

One Hidden Layer

An illustration of the implemented MLP network with one hidden layer (MLP 1L) can be seen in Figure 4.4. This network has 4 nodes in the hidden layer and 2 nodes in the output layer.

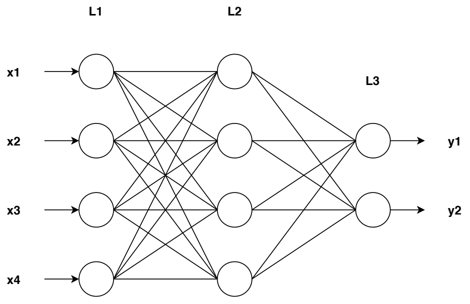


Figure 4.4: *An illustration of the implemented MLP neural network with one hidden layer.*

Two Hidden Layers

An illustration of the implemented MLP network with two hidden layers (MLP 2L) can be seen in Figure 4.5. This network has 4 nodes in the first hidden layer, 2 nodes in the second hidden layer and 2 nodes in the output.

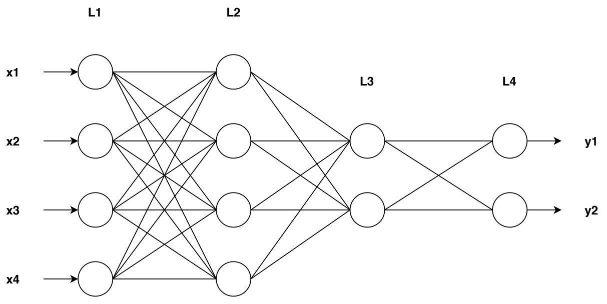


Figure 4.5: *An illustration of the implemented MLP neural network with two hidden layers.*

Initialization

All initial values of weights and biases in both MLP implementations are randomly assigned values between 0 and 1 from a set seed, this is to improve learning and to get variations in the weights. There is otherwise a risk of getting the same values on different weights, which is not preferable. The set seed is to make testing of different configuration possible without any disturbances due to the initial weights.

Training

Both MLP networks use the same training data and are trained using the same method, which is backpropagation. The training data originally consists of a set number of SLC and LaC cycles. Each type of cycle is divided into a set number of random segments with a specified time length, as described in the second approach in Section 3.1.3. This results in an increased amount of training cases compared to the original set of cycles. The network is then trained on this data for 500 epochs with a learning rate α of 0.001. These hyperparameters are chosen based on empirical testing.

4.3.5 Learning Vector Quantization

The implemented LVQ network can be seen in Figure 4.6. The competition layer uses the euclidean distance as comparison between the inputs. The LVQ network returns discrete predictions, where 0 and 1 represent SLC and LaC respectively. This network has 4 nodes in each layer and 1 output node consisting of two classes.

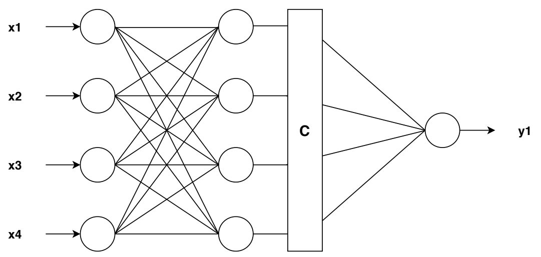


Figure 4.6: An illustration of the implemented LVQ neural network.

Initialization

The initial weights are set to the mean of each input characteristic according to the following equation, where x is the characteristics and $i \in \{1, 2, 3, 4\}$ since four different characteristics are used. This initialization is needed to ensure that the global optima is obtained instead of a local optima. The Kohonen rule converges when using this initialization for this LVQ implementation, meaning that the Bees algorithm is not necessary. For the general case, it is not guaranteed that this initialization works.

$$\begin{cases} w(1,i) & = mean(x_i) \text{ for all SLC training data} \\ w(2,i) & = mean(x_i) \text{ for all LaC training data} \end{cases} \tag{4.4}$$

Training

The LVQ network is trained using the training data and the Kohonen rule. The training data originally consists of a set number of SLC and LaC cycles. Each type of cycle is divided into a set number of random segments with a specified time length, as described in the second approach in Section 3.1.3. This results in an increased amount of training cases compared to the original set of cycles. Training on this data is done with a decreasing learning rate α from 0.5 to 0.001 with steps of 0.001, which results in 500 epochs.

4.4 Control Strategy

Equivalent Consumption Minimization Strategy is the control strategy that is developed and intended for use as optimal control strategy in the real-time online application. Dynamic Programming is used to measure its performance since DP is a global optimal control strategy. This means that two control strategies are implemented in total.

4.4.1 Dynamic Programming

For the series hybrid machine there are two states; engine speed and SOC. This makes the DP problem a multi variable problem. The solver and algorithm still work as described in section 3.2.1, however the problem is solved in two dimensions since there are two states. The cost function that is minimized in DP is set as the fuel consumption and is penalized so that the following limitations in the components of the powertrain are never exceeded.

- Maximum battery power
- Maximum engine torque
- Maximum engine power
- Negative generator power

The maximum engine torque for each engine speed is represented by a look up table inside the engine model. The limiting component in the engine and generator set is the engine, therefore no limitations are needed for the generator. Negative generator power is not allowed since it is not desired to engine brake when the system can use regenerative braking to recuperate the energy instead.

The fuel consumption that is used as cost function is obtained from the engine model, which requires knowledge of the engine power and engine speed. Since engine speed is a state, only engine power needs to be calculated. To calculate the engine power, the battery power must be calculated first. The battery power is calculated using the same model that is presented in equation (2.6). In discrete time the battery power is

$$P_{battery} = U \cdot I_{battery} = -U \cdot \frac{Q_{final} - Q_{start}}{h} \tag{4.5}$$

Where Q_{start} and Q_{final} are the battery capacity at start and final state of the time step h. These are calculated as described below

$$Q_{start} = SOC_{start} \cdot Q_0 \tag{4.6}$$

$$Q_{final} = SOC_{final} \cdot Q_0 \tag{4.7}$$

The generator power needs to be calculated as well to determine the engine power. The power demand should be the sum of the power delivered from generator and battery, which is easily seen if the series hybrid configuration in Figure 2.2 is studied. Since the power demand for the operating cycle is known and the battery power is calculated as above, the generator power can be calculated as the following equation.

$$P_{gen} = P_{demand} - P_{battery} \tag{4.8}$$

The engine power can be calculated using the generator power since the generator efficiency is the only differentiating factor between the two powers.

$$P_{ICE} = \frac{P_{gen}}{\eta_{em}} \tag{4.9}$$

4.4.2 Equivalent Consumption Minimization Strategy

The ECMS strategy requires two states since the machine is a series hybrid. The states used for ECMS are engine speed and SOC. The state grid sizes are of importance since the resolution determines the quality of the solution. The output from ECMS is the optimal power distribution, i.e. the generator power and battery power. The cost function used for ECMS is the Hamiltonian that was introduced in equation (3.24), since the equivalence factor for the engine speed is set to zero. When the equivalence factor for engine speed is zero it means that changing the engine speed is cost free.

For the Hamiltonian, fuel power and battery power are required and must be derived for the specific machine. The battery power, generator power and engine power can be calculated as in DP, see equations (4.5), (4.8) and (4.9). With the engine power and engine speed it is possible to use the engine model that provides the corresponding fuel consumption for the specific operating point. The fuel power can be calculated as follows given the fuel consumption.

$$P_{fuel} = \dot{m}_{fuel} \cdot q_{LHV} \tag{4.10}$$

The cost function is penalized so that the limits of the components in the powertrain are never exceeded, as listed in the previous section about DP. The optimal power distribution is obtained as a result of minimizing the cost function and finding the optimal operating point for the engine. It is possible to get multiple minimum points and when this occurs the point that has the minimum engine usage is selected.

4.4.3 Numerical Computation of Equivalence Factor

A method to find the optimal equivalence factor numerically is needed to ease the process of comparing multiple cycles. The equivalence factor is considered optimal when its value results in charge sustenance, which is important for a hybrid electric machine. The optimal equivalence factor is obtained by using the bisection method until the start and final SOC values differ as little as desired.

The numerical method to obtain the optimal equivalence factor is executed by simulating the system and systematically guessing the equivalence factor. The numerical method for calculating the optimal equivalence factor is described in Figure 4.7. Start values for the upper and lower bound are set in the *initial guess* block. The mean of the upper and lower bound is set as current guess in the *current guess* block. The current guess value is used to run a simulation of the system. The current guess is updated depending on the terminal SOC value according to the figure.

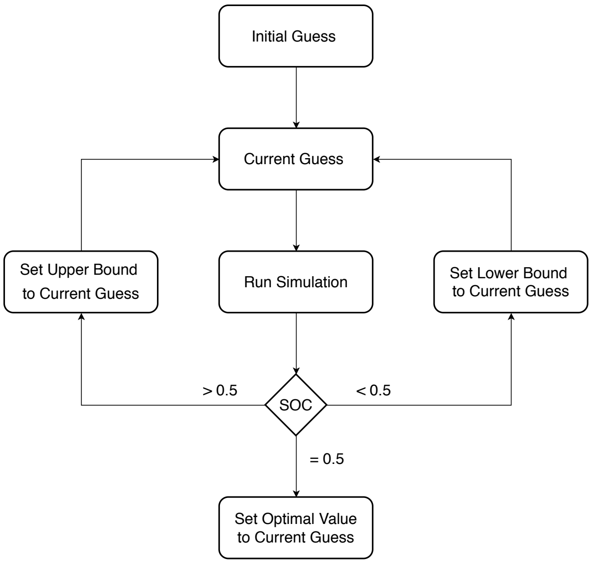

Initial Guess
Current Guess
Set Upper Bound to Current Guess
Run Simulation
Set Lower Bound to Current Guess
> 0.5
SOC
< 0.5
= 0.5
Set Optimal Value to Current Guess

Figure 4.7: *Flowchart of the bisection method that is used to obtain the optimal equivalence factor.*

This method is an effective way of calculating an approximation of the optimal equivalence factor that gives the system charge sustaining behavior. The main downside with this method is that it is quite computationally demanding, due to it having to run each cycle multiple of times. The numerical equivalence factor solver is implemented so that it automatically runs the method for each new drive cycle and saves the results together with the corresponding label of the data. All the labeled equivalence factors can then be used by the optimal equivalence factor selector that is presented in section 4.5.2.1.

4.4.4 Compensation for Variations in Terminal States

To be able to compare the performance of different strategies it is necessary to be able to calculate a total fuel consumption. For a hybrid machine the variations in terminal SOC must be taken into consideration in order to calculate a comparable fuel consumption. To compensate for variations in terminal SOC a method described in [19] is implemented. The main idea of this method is to run the baseline ECMS with small perturbations on the equivalence factor that causes variations in the terminal SOC and fuel consumption. The least square method can then be used to fit a line to the data according to the following equation, where m_{fuel} is the actual fuel consumption and θ_1 and θ_2 are the model parameters for the line.

$$m_{fuel} = \theta_1 + \theta_2(SOC_{final} - SOC_{start}) \tag{4.11}$$

When the parameters are fitted, the compensated fuel consumption $m_{fuel,comp}$ can be calculated according to the following equation which is a rewritten version of the previous equation.

$$m_{fuel,comp} = m_{fuel} - \theta_2(SOC_{final} - SOC_{start}) \tag{4.12}$$

The parameter θ_2 can be physically interpreted as the following equation, where U is the system voltage, q_{LHV} the lower heating value of the fuel, η_{genset} the combined efficiency of the generator and internal combustion engine during the cycle.

$$\theta_2 = \frac{UQ_0}{q_{LHV} \cdot mean(\eta_{genset})} \tag{4.13}$$

The total genset efficiency is calculated as follows, depending on the direction of the power demand and battery power.

$$\eta_{genset} = \begin{cases} \eta_{discharge} & \text{if } P_{demand} \geq 0, P_{battery} \geq 0 \\ \eta_{charge} & \text{if } P_{demand} > 0, P_{battery} < 0 \\ \eta_{regen} & \text{else} \end{cases} \tag{4.14}$$

The efficiency of the system when discharging the battery is calculated according to the following equation.

$$\eta_{discharge} = \frac{P_{demand}}{P_{battery} + P_{fuel}} \tag{4.15}$$

The efficiency of the system when charging the battery is calculated according to the following equation.

$$\eta_{charge} = \frac{P_{demand} + |P_{battery}|}{P_{fuel}} \tag{4.16}$$

The efficiency of the system when charging the battery through regenerative braking is calculated according to the following equation.

$$\eta_{regen} = \frac{|P_{battery}|}{P_{fuel} + |P_{demand}|} \tag{4.17}$$

4.5 Adaptive Strategy

Two sorts of adaptive strategies are implemented. One that improves the stability and assures that the SOC does not exceed set boundaries. The other method is used to update the equivalence factor in ECMS to represent the current operating cycle.

4.5.1 Stability

For stability two different adaptive approaches are implemented. Both approaches are based on adjustments of the cost function that is used in ECMS. The first approach is to use an additive term and the second one is to use a multiplicative factor. Both are functions that penalize depending on the SOC deviation. The stability in SOC is needed to ensure that the deviation that can be caused by non-perfect equivalence factors is not accumulated. If the deviation is accumulated the limits of battery can be reached, causing failure in the machine and shortening the battery life.

4.5.1.1 Additive Penalty

An approach to achieve stability in SOC is by adding a term to the cost function that is activated when going outside a predetermined interval. The interval used in this thesis is between 45% and 55%, which means that within this interval no additive penalty is added to the cost. This cost free interval is used to allow more battery usage compared to having no interval and a penalty that is constantly active. This interval depends on the size of the battery and how important charge sustenance is. The interval used in this thesis is chosen based on empirical testing. It is desired that the additive term increases drastically, preferably asymptotically, when approaching the maximum and minimum SOC limits to ensure that those are never exceeded.

A tangent function is used as the additive term to achieve the asymptotic behaviour. It is possible to design the tangent function to be asymptotic when approaching the SOC boundaries. This means that the value of the function reaches infinity when getting close to the admissible limits which makes it impossible to exceed the limits. The tangent function that is implemented has the design presented in equation (4.18). This function is illustrated with a principle figure in Figure 4.8 and as can be seen it is centered at 0.5 and is asymptotic at the limits 0.2 and 0.8 since this tangent function has the period 0.6. It is centered at 0.5 since it is desired that SOC is charge sustaining at this value. The constant k determines the inclination of the function and is chosen so that the penalty increases desirably. How different constants affect the inclination is illustrated in the figure of the tangent function.

$$f(SOC) = k \cdot \tan\left(\frac{\pi \cdot SOC}{0.6} - \frac{\pi}{2 \cdot 0.6}\right) \tag{4.18}$$

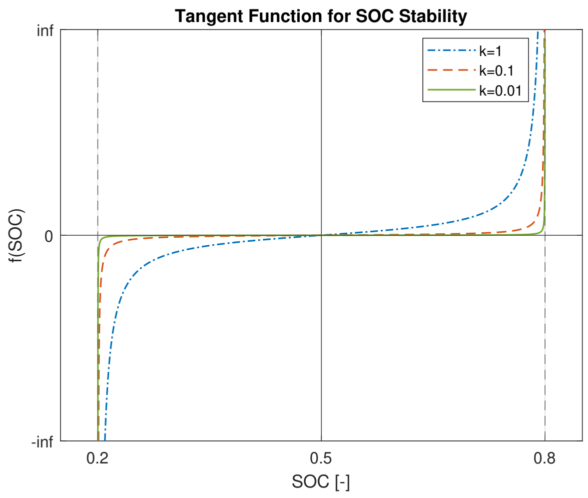
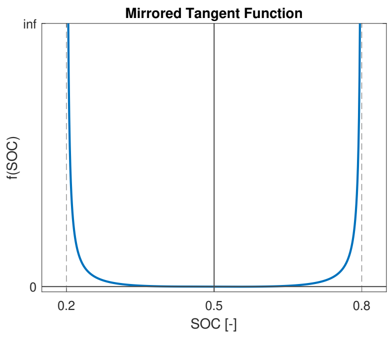


Figure 4.8: *The tangent function presented in equation* (4.18) *for different constants k, which is used as an additive SOC term in the cost function.*

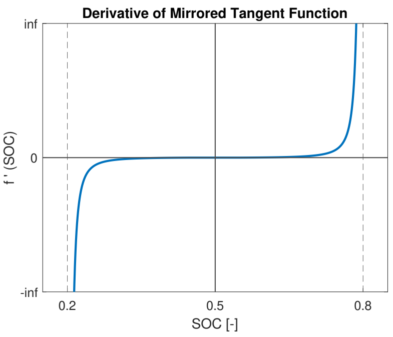
Since the term is additive it is desired that the term increases asymptotically towards positive infinity when approaching both the upper and lower limits. To obtain this behaviour the tangent function is mirrored around 0.5. This means that the tangent function is negative for SOC values below 0.5 and positive above, resulting in an additive term that is described as follows

$$f(SOC) = \begin{cases} -k \cdot \tan\left(\frac{\pi \cdot SOC}{0.6} - \frac{\pi}{2 \cdot 0.6}\right) & \text{for } 0.2 < SOC < 0.5 \\ k \cdot \tan\left(\frac{\pi \cdot SOC}{0.6} - \frac{\pi}{2 \cdot 0.6}\right) & \text{for } 0.5 < SOC < 0.8 \end{cases} \tag{4.19}$$

To assure smooth transitions when the term is activated with the predetermined interval, the term is compensated so that the function value begins at 0 when the penalty is activated at 45% and 55%. The mirrored and compensated tangent function and its derivative are presented in Figure 4.9 for the constant $k = 0.01$. This constant is used as an example to generate the figures.



(a) The mirrored tangent function that is used as the additive term.



(b) The derivative of the mirrored tangent function.

Figure 4.9: *The mirrored tangent function and its derivative, both of which are used as an additive penalty in the ECMS cost function to achieve stability in SOC.*

The adjoint equation, presented as equation (3.23), is no longer zero, since a term that depends on the state SOC is added to the Hamiltonian. This leads to an equivalence factor that changes with time. To update the equivalence factor a simple approximation of an integral is used, presented in the following equation.

$$s(t+1) = s(t) + h \cdot \dot{s}(t) = s(t) - h \cdot \frac{q_{LHV}}{U \cdot Q_0} \dot{\lambda}(t) \tag{4.20}$$

The equivalence factor only consists of one variable, the adjoint variable. To update the equivalence factor it is only necessary to determine how the adjoint variable changes with time, which can be done by solving the adjoint equation as follows.

$$\dot{\lambda} = -\frac{\partial\left(P_{fuel} + s \cdot P_{battery} + f(SOC)\right)}{\partial SOC} = -\frac{\partial f(SOC)}{\partial SOC} \tag{4.21}$$

The derivative of the adjoint variable only depends on the added SOC term, which results in the following expression for the derivative.

$$\dot{\lambda} = -\frac{\partial\left(k \cdot \tan(\frac{\pi \cdot SOC}{0.6} - \frac{\pi}{2 \cdot 0.6})\right)}{\partial SOC} = -\frac{k \cdot \pi}{0.6} sec^2\left(\frac{\pi}{2 \cdot 0.6} - \frac{\pi \cdot SOC}{0.6}\right) \tag{4.22}$$

The updated equivalence factor is needed since the ratio between running cost and the state equations changes when a term is added to the running cost. With an added term in the cost function the battery becomes proportionally cheaper, which means that the equivalence factor must be updated in order to retain the comparability between fuel and battery power.

4.5.1.2 Multiplicative Penalty

The cost function can also be adjusted with a multiplicative factor to achieve SOC stability. The factor that is used is presented in [19], it is a penalty function that penalizes based on deviations in SOC according to equation (4.23), where SOC_{max} and SOC_{min} are the SOC boundaries and SOC_{goal} is the preferred value of terminal SOC. The function is presented in Figure 4.10 for different exponents a.

$$f(SOC) = 1 - \left(\frac{SOC - SOC_{goal}}{\frac{1}{2}(SOC_{max} - SOC_{min})} \right)^a \tag{4.23}$$

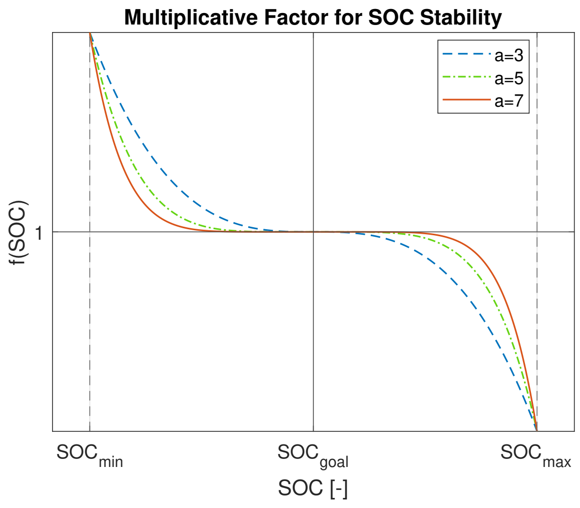


Figure 4.10: *The multiplicative factor function for different exponents a, which is used as the multiplicative penalty for SOC stability.*

4.5.2 Adaptive Equivalence Factor

The pattern recognition algorithm predicts the type of operating cycle, based on the prediction the optimal value for that type of cycle is selected by the optimal equivalence factor selector. The selected optimal equivalence factor is then used in ECMS, which makes the equivalence factor piece-wise constant, i.e. the value is constant as long as the prediction is the same.

4.5.2.1 Optimal Equivalence Factor Selector

The optimal equivalence factor selector outputs the corresponding value for the predicted type of cycle. The simplest method is to have a static equivalence factor selector as presented in equation (4.24), where s_{SLC} and s_{LaC} are the mean equivalence factor values of all the data of the receptive cycles and s_{ALL} is the mean of the medians of the data for the two cycles. The mean of the medians is used to ensure that there are no bias towards the cycle that consists of more data. Otherwise, the mean would be closer to the mean of the cycle with the larger data set.

$$s = \begin{cases} s_{SLC} & \text{if } cycle = 0 \\ s_{All} & \text{if } cycle = 0.5 \\ s_{LaC} & \text{if } cycle = 1 \end{cases} \tag{4.24}$$

Another method that can be used when a MLP network is used as pattern recognition, is to interpolate between the different equivalence factors using the probability outputs. This can be done with an interpolating equivalence factor selector using the following equation, where y_1 and y_2 are the probability outputs of the MLP network and can only take on values from 0 to 1.

$$s = \frac{(s_{LaC} - s_{SLC})}{2} \cdot (y_2 - y_1) + \frac{(s_{LaC} + s_{SLC})}{2} \tag{4.25}$$

The last term in the equation is the mean of all the equivalence factors and is the output when the algorithm is uncertain of what cycle it is undergoing. This means that there can be a bias towards the cycle that consists of more data.

5 Results and Discussion

This chapter describes and discusses the results of the different implementations. Firstly, pattern recognition and control strategies are tested separately to see how well they perform individually. After that the two methods are combined into the adaptive equivalence factor method and tested to analyze the combined performance.

5.1 Pattern Recognition

In this section all the performance tests and results of the pattern recognition algorithms are described, presented and discussed. Both static data with fixed length and dynamic data through a buffer are used for testing. These tests show how well the different algorithms perform.

5.1.1 Static Testing

In the static identification tests fixed segments of data are used to test the performance. The segments used for static testing are derived the same way as the data the algorithms are trained on. Each test consists of 100 segments. These tests show how well the algorithms have learned from the training and generalized to new data.

5.1.1.1 Training Data

A test is performed on the training data that all the pattern recognition algorithms have learned from. In Table 5.1, the results of how well the algorithms have learned this particular data set are presented.

Table 5.1: *Results of static tests on training data.*

Method	Accuracy	Precision	Recall	F_1 Score
RB	98 %	100 %	98 %	0.9899
LVQ	99 %	100 %	99 %	0.9950
MLP 1L	100 %	100 %	100 %	1.0000
MLP 2L	100 %	100 %	100 %	1.0000

As can be seen, all of the algorithms have learned to detect the right cycle based on the training data. The best performance is achieved by both of the MLP networks, which is expected when the training is done using backpropagation.

5.1.1.2 Testing Data

The pattern recognition algorithms are tested on a new and unseen set of data, called testing data. The results are presented in Table 5.2 and determine how well the algorithms have generalized the data.

Table 5.2: *Results of static tests on testing data.*

Method	Accuracy	Precision	Recall	F_1 Score
RB	100 %	100 %	100 %	1.0000
LVQ	98 %	100 %	98 %	0.9899
MLP 1L	99 %	99 %	100 %	0.9950
MLP 2L	100 %	100 %	100 %	1.0000

As can be seen in the table all the algorithms have generalized to the testing data data satisfactorily. The 2 hidden layer MLP network has the best performance, which is expected since it has more parameters to fit and thus a smaller chance of overfitting. Overall in the static tests, both MLP networks have the best performance. However, all methods perform well and it is worth mentioning and pay attention to how well even the simplest method, RB, is performing. The reason for the better performance on the test data compared to the training data for RB is a coincidence that can be explained by the data sets, since the test data set is smaller than the training data set.

5.1.2 Dynamic Testing

In the dynamic identification tests a running buffer of data is used to test the performance of predictions in a simulation environment. The buffer length is the same for all tests and algorithms. These tests show how well the algorithms can predict on real-time data, which are tests that are closely related to the real world performance.

5.1.2.1 Repetitive Cycles

All pattern recognition algorithms are tested on repetitive cycles in the simulation environment. These tests show how well the algorithms are at making prediction of dynamic cycles. Both the quality and the speed of the predictions are of importance. The test data consists of sets of the same type of cycle, called SLC All and LaC All in Table 4.1, this is to test the consistency of the predictions. This type of test data is the closest approximation of how the machines are operated at an average work site. The results from these dynamic tests on repetitive cycles can be seen in Figures 5.1 and 5.2.

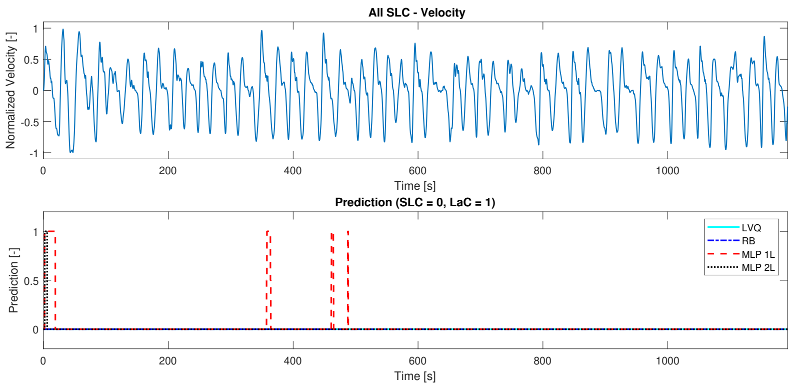


Figure 5.1: *Results of dynamic test on repetitive short loading cycles.*

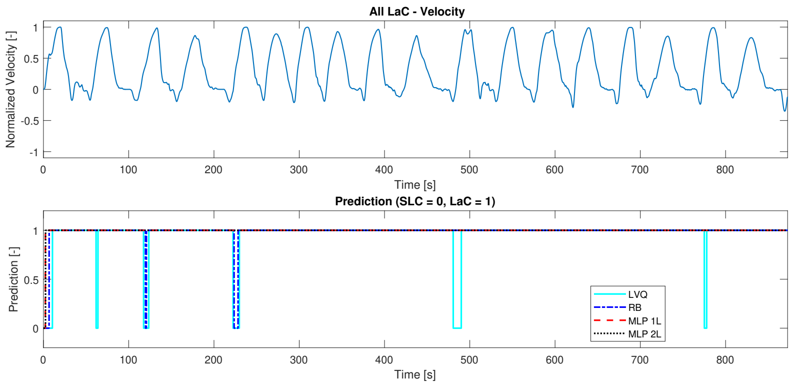


Figure 5.2: *Results of dynamic test on repetitive load and carry cycles.*

Ideally, all methods should be consistently 0 for the SLC All test and 1 for the LaC All test. It can be seen that LVQ, RB and MLP 1L lose the prediction sometimes, but the recovery back to the correct prediction is fast. LVQ and RB have problems during the slow part of the load and carry cycle, this is due to having too high bias towards the mean velocity signal. MLP 1L has instead problems in the short loading cycles, but it is not as clear why. The MLP 2L is almost perfect at detecting both repetitive cycles. In the load and carry cycle it can be seen that all predictions start at the prediction 0 which represent SLC, this is because the signal buffer is filled with zeros at the start of every cycle. This means that the mean velocity is very low in the beginning and thus all cycles will predict SLC, since that is the cycle with lower mean velocity. This is a reasonable approach due to the machine always standing still at the beginning of operation.

5.1.2.2 Time-Shifted Cycles

All pattern recognition algorithms are tested on time-shifted cycles in the simulation environment. These tests are examining the same performances as the previous tests but with time-shifted cycles. The tests also show how well the algorithms have generalized and how sensitive they are to time-shifts. Time-shifted cycles can be interpreted as testing with drivers with different amount of experience. The cycles that are denoted as fast and slow are time-shifted so that they are 10 % faster and slower in time respectively when compared to the original cycle. The results from the dynamic test on time-shifted cycles can be seen in Figures 5.3, 5.4, 5.5 and 5.6.

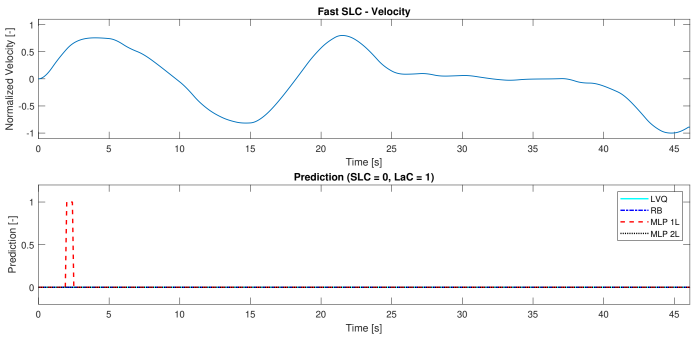


Figure 5.3: *Pattern recognition results of dynamic test on fast SLC data.*

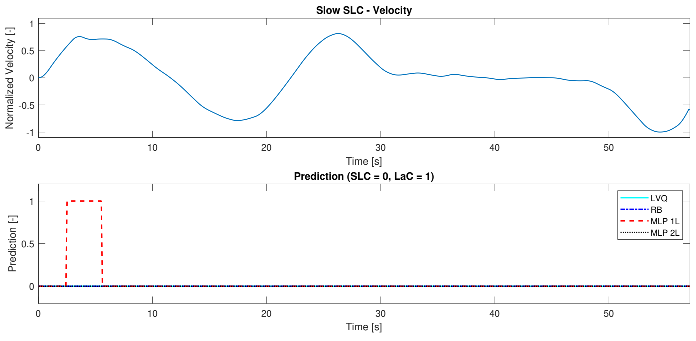


Figure 5.4: *Pattern recognition results of dynamic test on slow SLC data.*

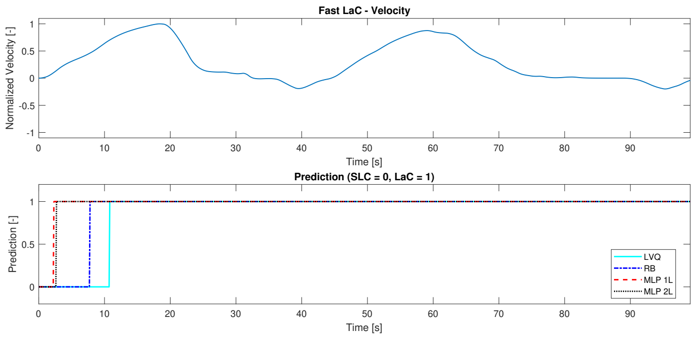


Figure 5.5: *Pattern recognition results of dynamic test on fast LaC data.*

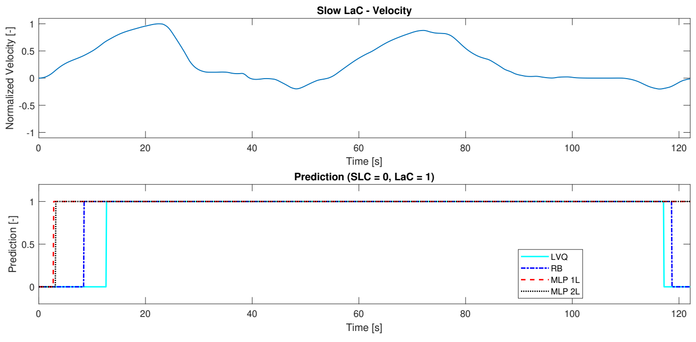


Figure 5.6: *Pattern recognition results of dynamic test on slow LaC data.*

The time-shifted tests on SLC show that all algorithms perform well on both the fast and the slow cycle, since almost all consistently predicts SLC, i.e. the output is 0. The MLP 1L can be seen making a wrong prediction in both tests. Fortunately, it is fast at correcting it.

It can be seen that all algorithms performs well on both the fast and the slow load and carry cycle as well. All predictions will start at 0 due to the signal buffer consisting of only zeroes at the beginning, this then lowers the mean velocity significantly at the start as mentioned earlier. Both LVQ and RB make mistakes at the unloading part at the end of the slow cycle, this is due to the mean velocity being very low.

5.1.2.3 Cycle Transitions

All pattern recognition algorithms are tested on cycle transitions in the simulation environment. These tests show how well and how fast the algorithms can react to a transition from one type of cycle to the other. It is of great importance that the algorithms are able to change its prediction fast and reliably in order to ensure high performance in the adaptive equivalence factor approach. The results from the dynamic test on cycle transitions can be seen in Figures 5.7 and 5.8.

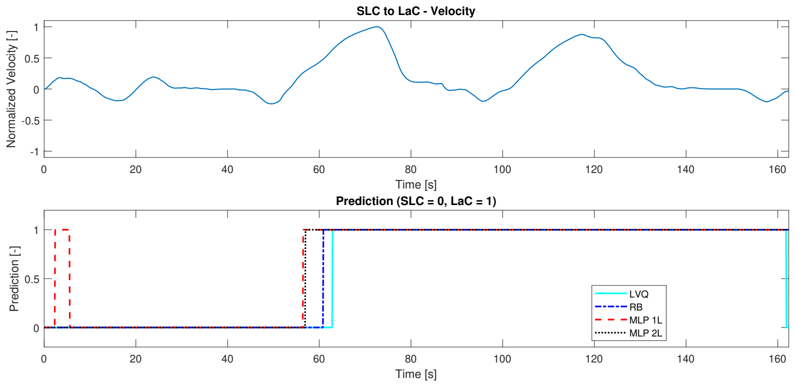


Figure 5.7: Pattern recognition results of dynamic test on cycle transitions from SLC to LaC. The cycle switch occurs at 52 seconds.

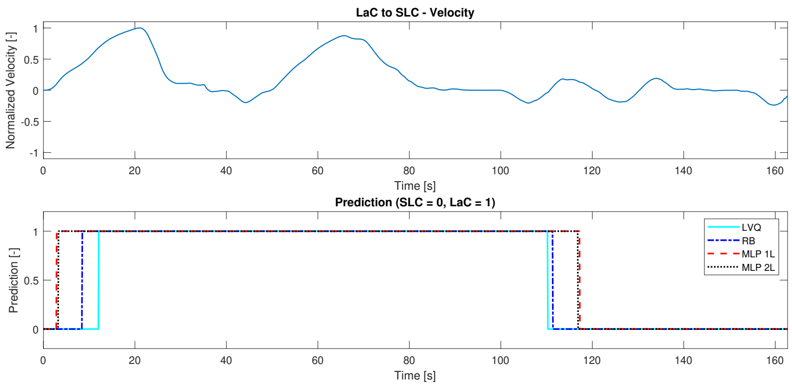


Figure 5.8: *Pattern recognition results of dynamic test on cycle transitions from LaC to SLC. The cycle switch occurs at 110 seconds.*

All algorithms perform well on both type of transitions. In the SLC to LaC test it can be seen that the MLP 1L makes a mistake at the beginning, but it recovers fast. Both MLP are the fastest at reacting to a transition from SLC to LaC, but LVQ and RB are faster at reacting to transitions from LaC to SLC.

5.1.3 Summary of Pattern Recognition Results

From the static tests it can be seen that all algorithms have learned the data and also that they generalized well to new data. From dynamic tests it can be seen that MLP 1L has some problems with SLC, while LVQ and RB have some problems with the LaC. These problems are small and might be negligible. The results depend on the number of parameters that are required for each algorithm, which are presented in Table 5.3. The number of parameters determines how complex the algorithm is and can affect the choice of algorithm depending on available computational power. However, the number of parameters is not the only factor to take into consideration if a new type of cycle would be implemented for identification. The pattern recognition method that would be chosen depends on the new cycle and how much it differentiates from the previous defined. It also depends on how much is known about the cycle and how much tuning that has to be re-done. This means that the Rule-based method that only consists of one parameter can be harder to use for a new cycle and the MLP 2L might be easier, it all depends on the new cycle.

Table 5.3: *Number of parameters for each pattern recognition algorithm.*

	RB	LVQ	MLP 1L	MLP 2L
Number of Parameters	1	8	30	36

A summary of the results from the pattern recognition tests can be seen in Table 5.4. The algorithms have been ranked from 1 to 4 for each test, where 1 is the best and 4 the worst performing algorithm. The ranking system is used to compare the algorithms to each other overall and to give an indication of the total performance.

Table 5.4: *Summary of all tests on the pattern recognition algorithm.*

	RB	**LVQ**	**MLP 1L**	**MLP 2L**
Training Data	4	3	1	1
Testing Data	3	4	2	1
Repetitive Cycles	1	4	3	1
Time-Shifted Cycles	2	3	4	1
Cycle Transitions	2	3	4	1
Total Score	12	17	14	5

From these tables it can be seen that MLP 2L is the best preforming algorithm but it is also the most complex. It has high quality of prediction and is also the fastest to predict. An interesting notation is that the RB method also works surprisingly well for being as simple as it is. It is also very consistent in its prediction, although very slow.

The data that is used for pattern recognition is directly logged from the machine and is not particularly noisy. Tests are performed with added noise to the signals to detect how sensitive the different algorithms are to noise. These tests showed that the two MLP networks had some small problems with detecting the cycle, which is likely caused by the drastic change in standard deviation and variance. The results of the noisy tests are presented in Appendix A. Noisy data can be counteracted with filtering before being used as input to the pattern recognition algorithms, this might lead to marginally slower predictions but with retained accuracy.

5.2 Control Strategy

In this section, the control strategy tests are described and the results are presented and discussed. The tests that are performed on the control strategy are based on validating the performance of ECMS, which is done by a comparison to the DP solution. The implementations of SOC stability are also tested by analyzing the SOC profiles and fuel consumption of ECMS with and without the stability implementations.

5.2.1 Performance

To validate the performance of the developed control strategy, the perfect ECMS solution is compared to the corresponding DP solution. Perfect ECMS means that no stability implementations are present and the optimal equivalence factor for the specific cycle is used, meaning that the result is as charge sustaining as possible. The comparison is based on a representative SLC and LaC cycle. The performance is analyzed by comparing the SOC profile and fuel consumption from the two control strategies. The SOC profiles obtained by the two strategies are presented in Figure 5.9 and Figure 5.10 for both the representative SLC and LaC cycle.

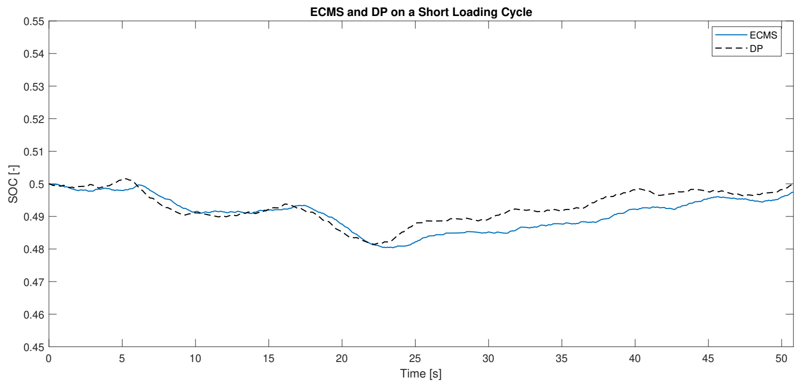


Figure 5.9: *The SOC profiles from DP and perfect ECMS for a SLC cycle that is used to evaluate the performance of ECMS.*

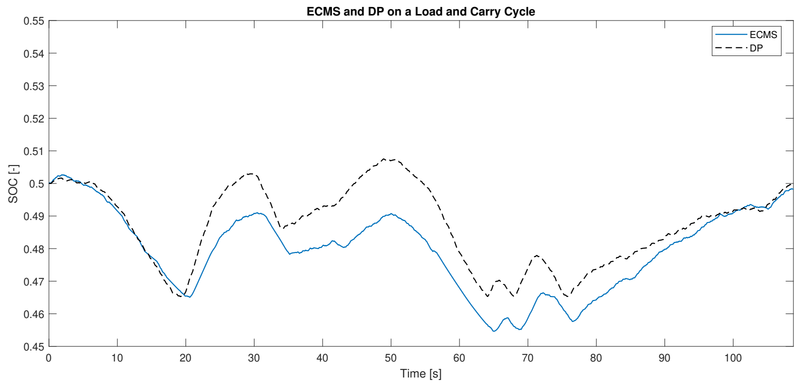


Figure 5.10: *The SOC profiles from DP and perfect ECMS for a LaC cycle that is used to evaluate the performance of ECMS.*

Based on the performance plots it can be stated that ECMS performs as desired. The solutions obtained from ECMS and DP are fairly similar and the behaviour is the same, which makes the ECMS solution satisfactory since it is close to the globally optimal DP solution. There is a small difference between the generated SOC profiles, which can partly be explained by the grid sizes. In this test the state grid sizes in ECMS and DP have the same resolution. However, a denser grid would most likely result in an even smaller difference in accordance to optimal control theory. The computational power is a limiting factor in this thesis, which make it not possible to use a denser grid than what is used in the presented results.

The fuel consumption from the performance test is presented in Table 5.5 for the SLC cycle and in Table 5.6 for the LaC cycle. The fuel consumption is presented as the percentage of the fuel consumption of no battery mode, which means that the machine operates with only engine power and no battery power. Since ECMS is still used in the no battery mode it means that the optimal engine torque and engine speed is chosen. As can be seen in the table the DP solution results in the minimum fuel consumption, as expected. The perfect ECMS does not perform as well as DP, but minimizes the fuel consumption significantly. The presented fuel consumption of the perfect ECMS is compensated for the small deviation in terminal SOC.

Table 5.5: *Fuel consumption and terminal SOC for DP and perfect ECMS on a SLC cycle. The fuel consumption is specified as the percentage of the fuel consumption of the no battery mode. The fuel consumption of the perfect ECMS is compensated for the deviation in terminal SOC.*

Control Strategy	**Cycle**	**SOC** [%]	**Fuel Consumption** [%]
No battery mode	SLC	-	100.00
DP	SLC	50.00	95.50
Perfect ECMS	SLC	50.36	98.68

Table 5.6: *Fuel consumption and terminal SOC for DP and perfect ECMS on a LaC cycle. The fuel consumption is specified as the percentage of the fuel consumption of the no battery mode. The fuel consumption of the perfect ECMS is compensated for the deviation in terminal SOC.*

Control Strategy	**Cycle**	**SOC** [%]	**Fuel Consumption** [%]
No battery mode	LaC	-	100.00
DP	LaC	50.00	87.22
Perfect ECMS	LaC	49.99	87.69

How much room there is for improvement in ECMS by implementing adaptivity can be determined by studying the difference in fuel consumption between the DP and ECMS solution. The difference is 3.18% for the SLC cycle and 0.47% for the LaC cycle. This means that the implementation of pattern recognition to select the optimal equivalence factor can be expected to improve the fuel consumption by approximately 0.47% to 3.18%.

5.2.2 Stability

The fuel consumption and SOC profiles are used to validate and test the two different stability methods that are implemented in the control strategy. It is desired that the stability implementations do not affect the fuel consumption vigorously. How the stability implementations perform in a cycle can be analyzed by studying the SOC profiles. The terminal SOC is also of interest since it shows if the results become more or less charge sustaining. The tests also highlight if there is a difference between the two different implementations. The stability tests are performed on the SLC All and LaC All cycles that are presented in Table 4.1, the results are compared to the perfect ECMS in order to see the actual impact of the stability implementations. All stability tests are performed with optimal equivalence factor, which represents perfect cycle detection. The optimal equivalence factor in this case is not optimal for each and every specific cycle, instead the average equivalence factor for all SLC is used and for all LaC. This means that the equivalence factor is not perfect. The ECMS without penalty is called Baseline ECMS.

5.2.2.1 Additive Penalty

The additive penalty that is used for stability by penalizing deviations in SOC generates the SOC profile in Figures 5.11 and 5.12, for both SLC All and LaC All cycles.

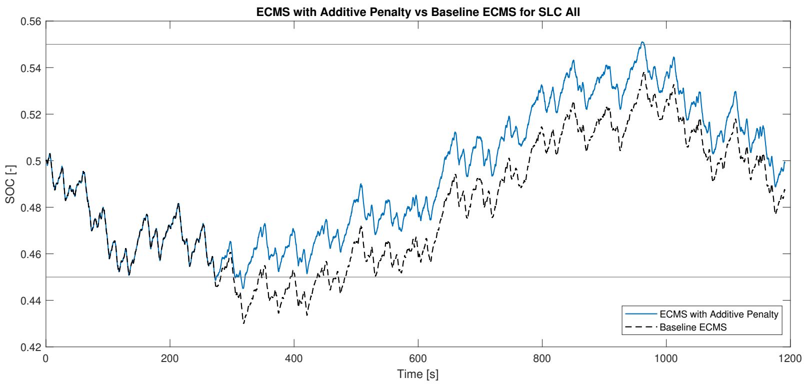


Figure 5.11: *ECMS with additive SOC deviation penalty plotted with baseline ECMS for all short loading cycles combined.*

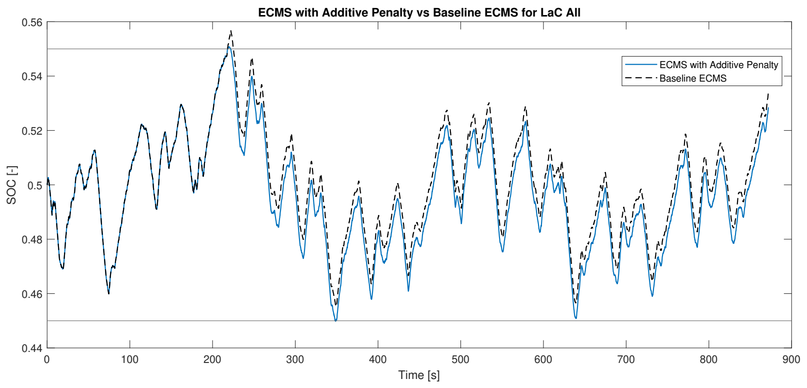


Figure 5.12: *ECMS with additive SOC deviation penalty plotted with baseline ECMS for all load and carry cycles combined.*

These tests are performed with a fairly strong penalty to demonstrate the penalty in action. This is extra clear in Figure 5.11, where the SOC profile does not exceed 45% or 55%, which is where the penalty is activated. Instead the SOC is kept in a stable region that results in a more charge sustainable terminal SOC. The same tendencies can be seen in Figure 5.12 as well. The terminal SOC and fuel consumption results are presented in Tables 5.7 and 5.8 and discussed further in the stability summary.

5.2.2.2 Multiplicative Penalty

The multiplicative penalty is also tested on the SLC All and LaC All cycles so that the SOC profile and fuel consumption can be analyzed. The results are compared to the baseline ECMS. The SOC profile comparisons are presented in Figures 5.13 and 5.14.

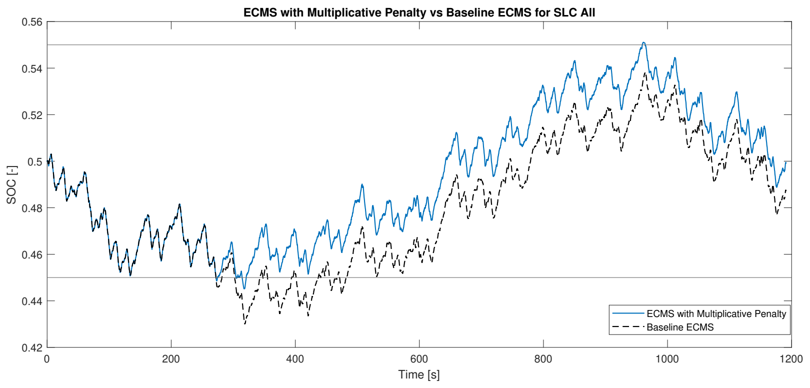


Figure 5.13: *ECMS with multiplicative SOC deviation penalty plotted with baseline ECMS for all short loading cycles combined.*

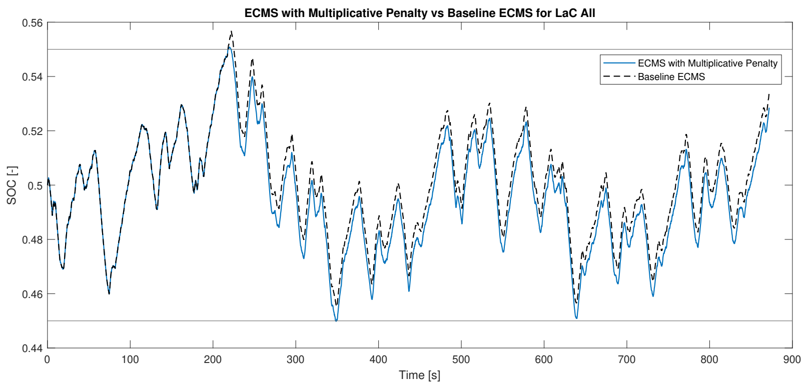


Figure 5.14: *ECMS with multiplicative SOC deviation penalty plotted with baseline ECMS for all load and carry cycles combined.*

The multiplicative penalty is fairly strong as well, as can be seen in the plots since the behaviour is similar to the additive penalty. When the penalty is implemented the SOC stays within the cost free interval since the penalty is strong, which is demonstrated at 45% and 55% in both figures. The terminal SOC and fuel consumption results are presented in Tables 5.7 and 5.8 and discussed further in the stability summary.

5.2.2.3 Summary of Control Strategy Stability Results

The fuel consumption and terminal SOC for all stability tests are presented in Table 5.7 for tests on SLC All and in Table 5.8 for LaC All. The fuel consumption is compensated for deviations in terminal SOC, according to the theory presented in Section 4.4.4. As can be seen the fuel consumption is mildly affected by the stability implementations, which is desirable.

Table 5.7: *Fuel consumption and terminal SOC for all control strategy stability tests on SLC All. The fuel consumption is specified as the percentage of the fuel consumption of the machine operating without battery usage, it is also compensated for deviations in terminal SOC.*

Control Strategy	**Cycle**	**SOC** [%]	**Fuel Consumption** [%]
No battery mode	SLC All	-	100.00
Baseline ECMS	SLC All	48.71	97.52
Additive ECMS	SLC All	49.89	97.38
Multiplicative ECMS	SLC All	49.93	97.35

Table 5.8: *Fuel consumption and terminal SOC for all control strategy stability tests on LaC All. The fuel consumption is specified as the percentage of the fuel consumption of the machine operating without battery usage, it is also compensated for deviations in terminal SOC.*

Control Strategy	**Cycle**	**SOC** [%]	**Fuel Consumption** [%]
No battery mode	LaC All	-	100.00
Baseline ECMS	LaC All	53.41	90.52
Additive ECMS	LaC All	52.85	89.32
Multiplicative ECMS	LaC All	52.85	89.32

It is not expected that the fuel consumption improves when a stability penalty is implemented, however that is the result of these tests. This can partly be explained by the non-perfect equivalence factor. Another reason that the fuel consumption is better with penalty can be the compensation. Since SLC All and LaC All are relatively short cycles, the compensation is a relatively big part of the total compensated fuel consumption. The compensation is based on the assumption that the calculated cycle efficiency is constant, which is not completely true. This assumption can therefore be part of the explanation of why the fuel consumption results in these tests are not completely reliable. However, the results indicate that the fuel consumption is not affected drastically by the penalty which is desired.

The SOC starts at 50% and for stability and charge sustenance it is desired that the terminal SOC is as close to the start value as possible. As can be seen in the result tables, the SOC becomes more stable with the stability implementations. Since the terminal SOC is closer to 50% with the penalty compared to no penalty it can be stated that the stability implementations result in better charge sustenance and improved stability. The stricter the penalty, the more stable SOC results. However, a softer penalty might be more favourable in the long run since it allows for more extensive battery usage which might be beneficial for the fuel consumption. Also, a more narrow penalty free interval would lead to even better charge sustenance if that is of importance.

5.3 Adaptive Equivalence Factor

Tests on the adaptive equivalence factor strategy are performed to determine the performance of the different combination of methods. The first set of tests determines how much improvement that can be excepted from the adaptivity implementations if the pattern recognition would be perfect. The second and third sets of tests determine the performance of A-ECMS, which is the adaptive energy management strategy that includes both ECMS and pattern recognition. All A-ECMS tests are performed on Main Cycle 1, which is SLC All and LaC All combined, as presented in Table 4.1.

5.3.1 Perfect Equivalence Factor

Perfect equivalence factors are used to determine what the maximum improvement of A-ECMS is. These tests indicate what improvements that can be reasonably expected, since it represents a perfect pattern recognition. The perfect equivalence factor, $s_{Perfect}$, means that the optimal value for each specific cycle is used, which should result in the most charge sustaining outcome possible. This can then be compared to the results of using equivalence factors that are optimal for the average SLC or LaC cycle, as well as the mean equivalence factor for all cycles. Both of the two different stability implementations in ECMS are used for each test to determine how the fuel consumption is affected. The results from the tests with different equivalence factors are presented in Table 5.9.

Table 5.9: *The terminal SOC and fuel consumption from when perfect equivalence factors are used together with the two different stability penalty implementations. The fuel consumption is specified as the percentage of the fuel consumption of the no battery mode and is compensated for deviations in terminal SOC.*

Equivalence Factor	**Penalty**	**SOC** [%]	**Fuel Consumption** [%]
No Battery Mode	None	-	100.00
s_{All}	Additive	42.09	93.56
s_{SLC}, s_{LaC}	Additive	50.20	93.12
$s_{Perfect}$	Additive	51.94	93.17
s_{All}	Multiplicative	48.01	93.24
s_{SLC}, s_{LaC}	Multiplicative	49.58	93.14
$s_{Perfect}$	Multiplicative	52.55	93.19

It can be seen in the table that the adaptive equivalence factor can reduce the fuel consumption by 0.4 percentage for the additive penalty and 0.1 for the multiplicative penalty. This means that the adaptivity is more important for the performance of the additive penalty. It can also be seen that the additive penalty is less aggressive on the terminal SOC compared to the multiplicative on s_{All}.

5.3.2 Static Equivalence Factor Selector

Tests are performed on A-ECMS with the different pattern recognition methods as well as the different stability implementations. The equivalence factor that is used by ECMS is determined by the static equivalence factor selector that uses the prediction from the cycle detection as input. The static equivalence factor selector operates as described in equation (4.24). Tests are also performed with no penalty implementation and no pattern recognition method, so that it can be seen how the fuel consumption is affected by adding adaptivity to ECMS. The equivalence factor used for the no penalty and no pattern recognition method is the average of all equivalence factor (called s_{All}). These tests determine how the fuel consumption is affected by the different methods. The test results of the different pattern recognition and stability implementations are presented in Table 5.10.

Table 5.10: *The terminal SOC and fuel consumption from when the static equivalence factor selector is used together with the two different stability penalty implementations.The fuel consumption is specified as the percentage of the fuel consumption of the no battery mode and is compensated for deviations in terminal SOC.*

ECMS	**PR Method**	**SOC** [%]	**Fuel Consumption** [%]
No Battery Mode	None	-	100.00
No Penalty	None	54.67	93.26
Additive Penalty	None	42.09	93.56
Additive Penalty	RB	50.12	93.12
Additive Penalty	LVQ	50.04	93.11
Additive Penalty	MLP 1L	50.20	93.12
Additive Penalty	MLP 2L	50.20	93.12
Multiplicative Penalty	None	48.01	93.24
Multiplicative Penalty	RB	49.58	93.14
Multiplicative Penalty	LVQ	49.58	93.14
Multiplicative Penalty	MLP 1L	49.58	93.14
Multiplicative Penalty	MLP 2L	49.58	93.14

It can be seen in the table that all used pattern recognition algorithms give a stable terminal SOC which is desirable. For the additive and multiplicative penalty it can be seen that all pattern recognition algorithms give as good results as the best ones from the previous test, this means that there is not much more performance that can be gained by improving the pattern recognition methods. The best performance is obtained when using an additive penalty together with LVQ. However, the differences between the different pattern recognition algorithms are negligible.

When comparing the no penalty test to the tests with penalty implementations it can be seen that a marginally better fuel consumption is obtained when using the multiplicative penalty. This is most likely caused by the compensation and the non-perfect equivalence factor. The penalty has great impact on the results if it is too strict, which is noticeable since the fuel consumption is almost the same regardless of the pattern recognition method that is used for the different penalties. In conclusion, it is more important to have a pattern recognition algorithm overall, than having the perfect one.

5.3.3 Interpolating Equivalence Factor Selector

When MLP is used, A-ECMS can use the cycle prediction to interpolate the equivalence factor. The equivalence factor is interpolated as defined in equation (4.25). A plot of how the interpolating equivalence factor performs during Main Cycle 1 is presented in Figure 5.15 for MLP 1L and Figure 5.16 for MLP 2L.

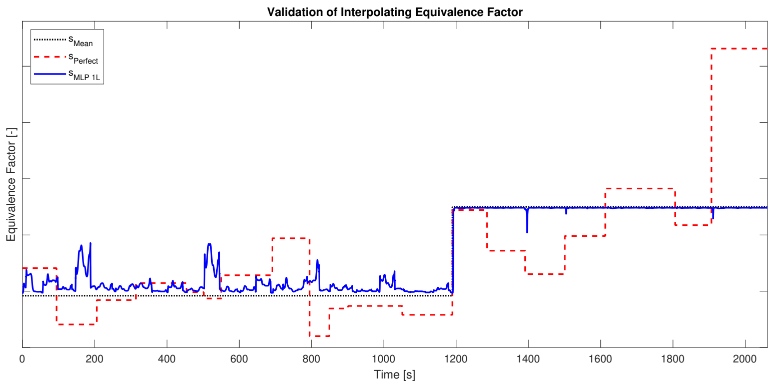


Figure 5.15: *The interpolated equivalence factor using MLP with one hidden layer compared to* $s_{perfect}$ and s_{mean}.

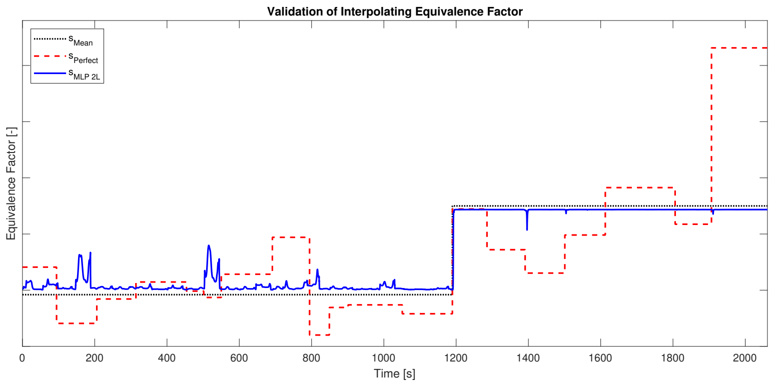


Figure 5.16: *The interpolated equivalence factor using MLP with two hidden layers compared to* $s_{perfect}$ and s_{mean}.

By comparing the interpolating equivalence factor to the perfect it can be determined how well it performs. Preferably, it should follow the behaviour of the perfect. As can be seen in the figures, this is not the case. The interpolating value is close to the mean equivalence factor which is good. However, at some points the value is interpolated towards the wrong cycle which most likely worsens the results. There is also a constant offset from the mean equivalence factor that contributes to worse results. This can be caused by a bias in the mean of all equivalence factors due to not being able to use the mean of the medians in this implementation. To conclude, it should not be beneficial to use an interpolating equivalence factor compared to a static. The main reason for this is that pattern recognition only uses the velocity as input, when there are more important factors that contribute to the value of the equivalence factor compared to the velocity.

The interpolating equivalence factor selector is tested together with the different stability implementations to determine the effect on the fuel consumption. The results from these tests are presented in Table 5.11.

Table 5.11: *The terminal SOC and fuel consumption from when the interpolating equivalence factor selector is used together with the two different stability penalty implementations. The fuel consumption is specified as the percentage of the fuel consumption of the no battery mode and is compensated for deviations in terminal SOC.*

ECMS	**PR Method**	**SOC** [%]	**Fuel Consumption** [%]
No Battery Mode	None	-	100.00
No Penalty	None	54.67	93.26
Additive Penalty	None	42.09	93.56
Additive Penalty	MLP 1L	50.20	93.12
Additive Penalty	MLP 2L	49.64	93.13
Multiplicative Penalty	None	48.01	93.24
Multiplicative Penalty	MLP 1L	49.55	93.14
Multiplicative Penalty	MLP 2L	49.16	93.16

It can be seen in the table that the interpolating equivalence factor selector still lowers the fuel consumption when compared to using no pattern recognition method at all. But the results are worse when compared to the static selector. This can be explained with the same reasoning that was presented in the paragraph above about the validation of the interpolating selector, that the equivalence factor is not as good as the static one. The interpolating selector is not to be recommended based on the validation of this implementation. However, it could be a powerful method if the neural network is designed to detect changes based on other characteristics than velocity.

5.3.4 Work Shift Simulation

The adaptive strategy is tested on a longer cycle that is more representative for an entire work shift. The cycle is 8 hours long and is the closest approximation available of a work shift. The cycle is not completely representative since only clean repetitive cycles are used, meaning that there are no breaks, transportation etc. This test is performed on Main Cycle 2, which consists of approximately 4 hours of SLC and 4 hours of LaC. The results from this test can be found in Table

5.12. Note that no test is performed with no penalty, since a cycle with this length the deviations in SOC accumulates over the limits, thus stopping the simulation.

Table 5.12: *The terminal SOC and fuel consumption from when the static equivalence factor selector is used together with the two different stability penalty implementations on Main Cycle 2. The fuel consumption is specified as the percentage of the fuel consumption of the no battery mode and is compensated for deviations in terminal SOC.*

ECMS	**PR Method**	**SOC** [%]	**Fuel Consumption** [%]
No Battery Mode	None	-	100.00
Additive Penalty	None	41.97	92.44
Additive Penalty	RB	50.10	92.41
Additive Penalty	LVQ	50.02	92.41
Additive Penalty	MLP 1L	50.18	92.41
Additive Penalty	MLP 2L	50.18	92.41
Multiplicative Penalty	None	47.98	92.44
Multiplicative Penalty	RB	49.51	92.41
Multiplicative Penalty	LVQ	49.51	92.41
Multiplicative Penalty	MLP 1L	49.55	92.41
Multiplicative Penalty	MLP 2L	49.55	92.41

The compensated fuel consumption is the same no matter the penalty or pattern recognition. The difference between having a pattern recognition method or not is minimal, which means that the pattern recognition performance is limited by the stability penalties. There are minor differences in the fuel consumption between the two penalty methods, which are negligible and cannot be seen due to the number of significant digits. The benefit of these tests is that the terminal SOC compensation factor has a minimal effect on the fuel consumption, due to the length of the cycle. The terminal SOC is close to 50%, which means that the strategy is charge sustaining. The main result of this test is that the fuel consumption can be improved by 7.59% with adaptive implementations compared to not utilizing the battery at all.

5.3.5 Summary of Adaptive Equivalence Factor

The summary of the tests and results on adaptive equivalence factor is that it is profitable to implement adaptivity to the basic ECMS implementation. A stability implementation increases the fuel consumption slightly and when used together with a pattern recognition method it is possible to decrease the fuel consumption.

The best fuel consumption is achieved when using LVQ together with the additive penalty. When a multiplicative penalty is used the fuel consumption is consistently slightly higher compared to the additive. However, the difference is minor and almost negligible. It is more important to actually have a pattern recognition method rather than having a perfectly tuned one. The static selector should preferably be used, since the interpolating selector does not show any advantageous properties in this particular implementation.

6 Conclusion

The thesis is concluded by answering the problem statement based on the results. Repetitive cycles can be identified from current and past vehicle states using pattern recognition. Of the different pattern recognition algorithms tested in this thesis the MLP with two hidden layers is the best performing one, followed by the RB. MLP with two hidden layers is the most complex and versatile algorithm, while RB is the simplest and most limited.

The current drive cycle information can be used to manage the energy consumption if a control strategy in combination with pattern recognition is implemented. Adaptivity can be used to make use of the current drive cycle information by predicting the operating cycle and use the information to determine the optimal power distribution between fuel and battery power. The adaptivity consists of pattern recognition that is used for cycle detection and an optimal control strategy that includes stability. ECMS is used as control strategy since it is optimal and stability is achieved by implementing penalties based on SOC deviations in the cost function. The penalties can be either additive or multiplicative. Tests show that the difference between the two penalties is almost negligible, but the additive is slightly more beneficial in the fuel consumption aspect. Another benefit of the additive penalty is that it is designed to be asymptotic at the SOC limits, which is an assurance that the SOC is always stable and does not damage the battery. When using both the adaptive equivalence factor and a stability implementation, the fuel consumption can be improved by 7.59% compared to not utilizing the battery at all.

Two implementations for real-time online application are proposed. The first implementation is using RB for pattern recognition and the additive penalty in ECMS. This is the simplest implementation and requires the least amount of computational power. The downside to this implementation is that it is not very versatile for modifications. The second implementation is using MLP with two hidden layers for pattern recognition and the additive penalty in ECMS. This implementation does require more computational power but is much more versatile. The versatility is important when more complex problems with additional cycles or inputs are introduced. Both these implementations will result in a reduction of fuel consumption compared to a basic implementation of ECMS.

6.1 Future Work

With more time, the next step in this thesis project would be to test the different implementations in a dynamic simulation environment. This is to see how well they would perform in a dynamic system. These results will also tell if the internal models in ECMS are a good enough approximation of the real system. The last step of validating the developed strategy would be to test the implementations in a real machine to see if the simulation results will transfer to the real world.

An alternative approach that would be interesting is to optimize the equivalence factor with respect to minimizing the fuel consumption, instead of with respect to charge sustenance. The result would still be stable since the stability implementations would take care of the charge sustenance. With a narrow penalty free interval for the SOC the results would be stable and the fuel consumption could possible be improved.

During the thesis an interesting phenomenon occurred when having a very strict additive penalty. The optimal equivalence factor for charge sustenance was found by ECMS despite what the actual value of the equivalence factor was. This occurred when the penalty was activated and the Hamiltonian reached an equilibrium point. With this knowledge it might be possible to develop a completely different approach.

Appendix

A

Noisy Tests on Pattern Recognition

The tests performed on pattern recognition on a representative SLC and LaC cycle with added white noise to the velocity signal are presented in Figures A.1 and A.2. The same amplitude and frequency is used in both tests.

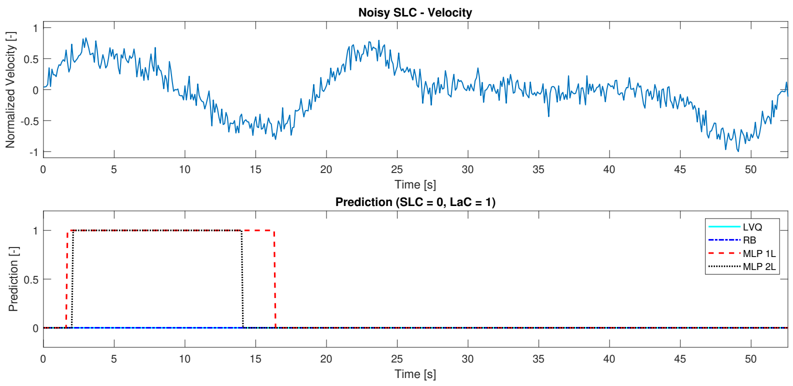


Figure A.1: Pattern recognition results of dynamic test on a noisy SLC.

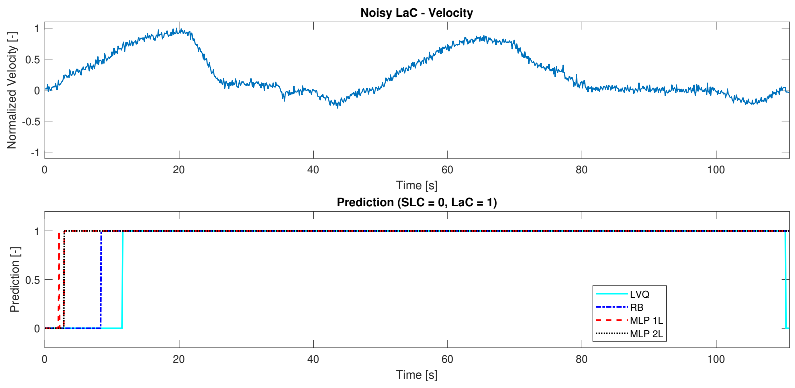


Figure A.2: *Pattern recognition results of dynamic test on a noisy LaC.*

Bibliography

[1] L. Eriksson and L. Nielsen. *Modeling and Control of Engines and Drivelines*. John Wiley & Sons, Incorporated, 2014. Cited on page 5.

[2] L. Guzzella and A Sciarretta. *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*. Springer, 3rd edition, 2013. Cited on pages 6, 7, 8, 22, 23, and 26.

[3] R. Filla. Optimizing the trajectory of a wheel loader working in short loading cycles. *The 13th Scandinavian International Conference on Fluid Power*, pages 307–317, 2013. Cited on pages 10 and 11.

[4] T. Nilsson, P. Nyberg, C. Sundström, E. Frisk, and M. Krysander. Robust driving pattern detection and identification with a wheel loader application. *International journal of vehicle systems modelling and testing*, pages 56–76, 2014. Cited on pages 10, 13, and 14.

[5] M.T. Hagan, H.B. Demuth, M.H. Beale, and O. De Jesús. *Neural Network Design*. Martin Hagan, 2nd edition, 2014. Cited on pages 14, 15, 16, 18, and 19.

[6] M. Kiani Deh Kiani, B. Ghobadian, T. Tavakoli, A.M. Nikbakht, and G. Najafi. Application of artificial neural networks for the prediction of performance and exhaust emissions in si engine using ethanol- gasoline blends. *Energy*, 35(1):65 – 69, 2010. Cited on page 14.

[7] K. Song, F. Li, X. Hu, L. He, W. Niu, S. Lu, and T. Zhang. Multi-mode energy management strategy for fuel cell electric vehicles based on driving pattern identification using learning vector quantization neural network algorithm. *Journal of Power Sources*, 389:230 – 239, 2018. Cited on pages 14, 18, and 20.

[8] J. Wang, Q. N. Wang, X. H. Zeng, P. Y. Wang, and J. N. Wang. Driving cycle recognition neural network algorithm based on the sliding time window for hybrid electric vehicles. *International Journal Of Automotive Technology*, 16:685 – 695, 2015. Cited on pages 14, 18, and 21.

[9] B. A. Hawickhorst, S. A. Zahorian, and R. Rajagopal. A comparison of three neural network architectures for automatic speech recognition. *Intelligent Engineering Systems Through Artificial Neural Networks, Fuzzy Logic and Evolutionary Programming*, 5:221 – 226, 1995. Cited on page 14.

[10] F. Tianheng, Y. Lin, G. Qing, H. Yanqing, Y. Ting, and Y. Bin. A supervisory control strategy for plug-in hybrid electric vehicles based on energy demand prediction and route preview. *IEEE Transactions on Vehicular Technology*, 64(5):1691–1700, May 2015. Cited on pages 14, 26, and 27.

[11] I. H. Witten and E. Frank. *Data mining: Practical Machine Learning Tools and Techniques*. Elsevier Science & Technology, 2005. Cited on pages 15 and 21.

[12] D. T. Pham, S. Otri, A. Ghanbarzadeh, and E. Koc. Application of the bees algorithm to the training of learning vector quantisation networks for control chart pattern recognition. *2nd International Conference on Information Communication Technologies*, pages 1624–1629, 2006. Cited on page 19.

[13] M. Enqvist, T. Glad, Gunnarsson S., P. Lindskog, L. Ljung, J. Löfberg, T. McKelvey, A. Stenman, and J.E. Strömberg. *Industriell reglerteknik: Kurskompendium*. Reglerteknik, ISY, Linköpings universitet, 2014. Cited on page 23.

[14] L. Fu, Ü. Özgüner, P. Tulpule, and V. Marano. Real-time energy management and sensitivity study for hybrid electric vehicles. In *Proceedings of the 2011 American Control Conference*, pages 2113–2118, 2011. Cited on page 23.

[15] G. Ripaccioli, D. Bernardini, S. Di Cairano, A. Bemporad, and I. V. Kolmanovsky. A stochastic model predictive control approach for series hybrid electric vehicle power management. *Proceedings of the 2010 American Control Conference*, pages 5844–5849, 2010. Cited on page 23.

[16] R. E. Bellman. *Dynamic programming*. Princeton Univ. Press, 1957. Cited on page 23.

[17] T. Nilsson, A. Fröberg, and J. Åslund. Using stochastic dynamic programming for look-ahead control of a wheel loader diesel electric transmission. *IFAC Proceedings Volumes*, 47(3):6630 – 6635, 2014. 19th IFAC World Congress. Cited on page 23.

[18] S. J. Moura, H. K. Fathy, D. S. Callaway, and J. L. Stein. A stochastic optimal control approach for power management in plug-in hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 19(3):545–555, 2011. Cited on page 23.

[19] S. Onori, L. Serrao, and G. Rizzoni. *Hybrid electric vehicles: Energy management strategies*. Springer, 2016. Cited on pages 25, 28, 38, and 43.

[20] E. Kural and B. A. Güvenç. Predictive-equivalent consumption minimization strategy for energy management of a parallel hybrid vehicle for optimal recuperation. *Journal of Polytechnic*, 18(3):113–124, 2015. Cited on page 26.

[21] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia. A-ecms: An adaptive algorithm for hybrid electric vehicle energy management. *European Journal of Control*, 11:509–524, 2005. Cited on page 27.

[22] A. Sciarretta and L. Guzzella. Control of hybrid electric vehicles. *IEEE Control Systems Magazine*, 27(2):60–70, April 2007. Cited on page 27.