Fuel Optimized Predictive Following in Low Speed Conditions

Master's thesis
performed at **Vehicular Systems**

by
Johan Jonsson

Reg nr: LiTH-ISY-EX-3436-2003

28th June 2003

Fuel Optimized Predictive Following in Low Speed Conditions

Master's thesis

performed at **Vehicular Systems**,
Dept. of Electrical Engineering
at **Linköpings universitet**

by **Johan Jonsson**

Reg nr: LiTH-ISY-EX-3436-2003

Supervisor: **Zandra Jansson**
DaimlerChrysler
Michael Back
DaimlerChrysler
Mattias Krysander
Linköpings universitet

Examiner: **Professor Lars Nielsen**
Linköpings universitet

Linköping, 28th June 2003

Avdelning, Institution
Division, Department

Vehicular Systems,
Dept. of Electrical Engineering
581 83 Linköping

Datum
Date

28th June 2003

Språk
Language
☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category
☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN
—

ISRN
LITH-ISY-EX-3436-2003

Serietitel och serienummer **ISSN**
Title of series, numbering —

URL för elektronisk version
http://www.vehicular.isy.liu.se
http://www.ep.liu.se/exjobb/isy/2003/3436/

Titel Bränsleoptimerad prediktiv följning i låga hastigheter

Title Fuel Optimized Predictive Following in Low Speed Conditions

Författare Johan Jonsson
Author

Sammanfattning
Abstract

The situation when driving in dense traffic and at low speeds is called Stop and Go. A controller for automatic following of the car in front could under these conditions reduce the driver's workload and keep a safety distance to the preceding vehicle through different choices of gear and engine torque. The aim of this thesis is to develop such a controller, with an additional focus on lowering the fuel consumption. With help of GPS, 3D-maps and sensors information about the slope of the road and the preceding vehicle can be obtained. Using this information the controller is able to predict future possible control actions and an optimization algorithm can then find the best inputs with respect to some criteria. The control method used is Model Predictive Control (MPC) and as the name indicate a model of the control object is required for the prediction. To find the optimal sequence of inputs, the optimization method Dynamic Programming choose the one which lead to the lowest fuel consumption and satisfactory following. Simulations have been made using a reference trajectory which was measured in a real traffic jam. The simulations show that it is possible to follow the preceding vehicle in a good way and at the same time reduce the fuel consumption with approximately 3 %.

Nyckelord
Keywords Stop and Go, MPC, Dynamic programming, Fuel consumption

Abstract

The situation when driving in dense traffic and at low speeds is called Stop and Go. A controller for automatic following of the car in front could under these conditions reduce the driver's workload and keep a safety distance to the preceding vehicle through different choices of gear and engine torque. The aim of this thesis is to develop such a controller, with an additional focus on lowering the fuel consumption. With help of GPS, 3D-maps and sensors information about the slope of the road and the preceding vehicle can be obtained. Using this information the controller is able to predict future possible control actions and an optimization algorithm can then find the best inputs with respect to some criteria. The control method used is Model Predictive Control (MPC) and as the name indicate a model of the control object is required for the prediction. To find the optimal sequence of inputs, the optimization method Dynamic Programming choose the one which lead to the lowest fuel consumption and satisfactory following. Simulations have been made using a reference trajectory which was measured in a real traffic jam. The simulations show that it is possible to follow the preceding vehicle in a good way and at the same time reduce the fuel consumption with approximately 3 %.

Keywords: Stop and Go, MPC, Dynamic programming, Fuel consumption

Preface

This thesis has been carried out at the DaimlerChrysler department for powertrain control, REM/EP, in Esslingen am Neckar, Germany.

Thesis Outline

Chapter 2 The Vehicle Model to be Controlled The model used for the simulations is presented. Basic mathematical relations are described as well as simplifications that have been made.

Chapter 3 MPC Theory The control method used in this thesis is Model Predictive Control. Here a brief theoretical description for both linear and nonlinear MPC are given.

Chapter 4 Dynamic Programming When using MPC an optimization problem has to be solved. In this thesis this has been made with help of the Dynamic Programming method.

Chapter 4 Control and Optimization In this chapter the control optimization algorithm for the special case of Stop and Go is presented.

Chapter 5 Simulations and Results Simulations have been made using a reference trajectory which has been obtained in a real traffic jam. The results from these simulations are presented.

Chapter 6 Conclusions The conclusions that have been drawn from the simulations are presented here.

Chapter 7 Further Work A discussion of possible improvements for the controller can be found in this chapter

Acknowledgment

I am deeply indebted to my supervisors Zandra Jansson and Michael Back for all help and support during the work with this thesis.

Thanks also to Mattias Krysander for giving me feedback on the report. I would also like to show appreciation toward the members of DaimlerChrysler department REM/EP who made my stay in Germany pleasant.

Contents

Chapter 1

Introduction

The aim of this thesis is to make a controller for *Stop and Go* (S&G) driving conditions based on the theory of Model Predictive Control with Dynamic Programming and to investigate if there are any possibilities to save fuel. The main idea of a S&G controller is to reduce the driver´s workload in dense traffic by automatically keeping a proper distance to the car ahead. The old cruise control systems, with the task only to keep a predefined velocity, are becoming less meaningful as the traffic density increases. For that reason many automotive manufactures have produced different kinds of intelligent cruise control (ICC) systems that can not only maintain the pre-selected velocity but also have the capability to keep a safe distance to the car ahead. With help of radar and sensor systems the distance to the preceding vehicle can be obtained and the controller will make necessary braking and acceleration actions. As these systems are mainly produced for use on roads where the speed is high, it is important to have quite a large safety distance between the following car and the preceding car. A S&G system, on the other hand, can be used in situations with dense traffic where one can not keep a large distance to the preceding vehicle. Another difference of implementing a controller for S&G compared to ICC is that the torque converter of the transmission is assumed to be fixed for ICC and that is not the case for the S&G controller.

Chapter 2

The Vehicle Model to be Controlled

The car model used for the simulations is a Simulink model of a Mercedes-Benz S-class with a combustion engine. Later on the model will be used to predict future states when using the control method Model Predictive Control. The model is fed with three inputs namely required engine torque M_{engine}, gear i and the slope of the road γ. "Required" means the torque needed from the engine to propel the vehicle at certain speed. After applying the engine torque to the model it is transferred to the wheels via different subsystems. These subsystems will be presented in more detail further on in this chapter. As output from the model the velocity of the vehicle is obtained. The model is based on Newton's second law, $F = ma$, when describing the dynamics and on functions that transfer the required engine torque to the torque acting at the wheels. When a system modeled includes rotating parts like the wheels in this case, Newton's law for rotating movements can be used to express the transfer of torque from one part of the vehicle driveline to another. It relates the two variables torque M [Nm] and angle velocity ω [rad/s] to each other according to

$$M(t) = J \cdot \frac{d}{dt}\,\omega(t) \tag{2.1}$$

where J is constant and denotes the moment of inertia $[Nm/s^2]$. We can therefore write

$$\omega(t) = \frac{1}{J}\int_0^t M(\tau)\,d\tau \tag{2.2}$$

When no physical equation for describing a certain relationship in the model is available, maps generated on testbeds have been used. The maps consist of the input and output values represented by vectors.

When evaluating the output, the input signal or signals are compared with the corresponding input vector. If the input matches a value in the input vector, the output will simply be the corresponding value in the output vector. Otherwise, in case of no match, interpolation is used.

2.1 Model Dynamics

Assume that a vehicle with total mass m is driving on a road with an incline according to Figure 2.1. The total force $F_{tot} = m\dot{v}$ acting on the vehicle is the sum of the drive force generated from the engine F_{df} and of different resistance forces. According to Newton's second law the time derivative for the velocity can be expressed as

$$\dot{v} = \frac{1}{m}F_{tot} = \frac{1}{m}\left(F_{df} - F_{res}\right) \tag{2.3}$$

where F_{res} is the sum of the resistance forces, which are:

- Air resistance, which is a function of the square of the velocity:

$$F_{air} = \frac{1}{2}\rho_{air}A_f c_w v^2 \tag{2.4}$$

  Here ρ_{air} is the density of the air, A_f the maximum vehicle cross area and c_w the air drag coefficient resistance. This coefficient depends on the shape of the moving object and, in this case, describes how big impact the design of the car body has.

- Rolling resistance is mass dependent as well as affected by the slope of the road:

$$F_{rollingres} = f(v) \cdot mg\cos(\gamma) \tag{2.5}$$

  The function $f(v)$ is the rolling coefficient. In this model a map is used instead of explicit function for obtaining this coefficient.

- Gravitational force:

$$F_{incline} = mg\sin(\gamma) \tag{2.6}$$

- Braking force:

$$F_{brake} \tag{2.7}$$

  In a braking situation the engine will brake as much as possible. If the required braking torque exceed the maximum braking torque for the engine, the torque caused by the brakes will be the difference between this maximum and the desired braking torque.

The drive force F_{df} correspond to the drive torque which has been transmitted from the engine via the driveline to the wheels. To express this torque it is necessary to have a good model of the driveline and it's parts. This will be explained in the next section.

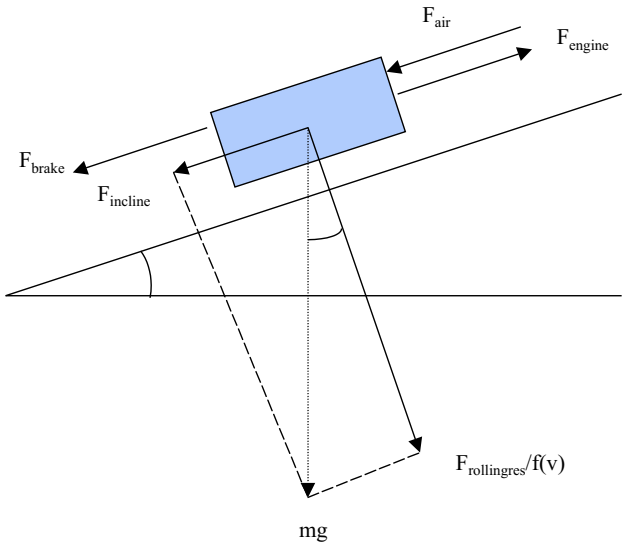

Figure 2.1: Forces acting on the vehicle

2.2 Driveline

When modeling a driveline for a vehicle the main parts to be modeled are engine, clutch, transmission, propeller shaft, final drive, drive shafts and wheels. Figure 2.2 shows where the different parts are located on the driveline. The model used in this thesis contains some simplifications which mean that all parts in Figure 2.2 are either not modeled or they have been assumed to behave in a way that differs from reality. To read more about driveline modeling see [6].

2.2.1 Basic Driveline Relations

Consider the situation in Figure 2.3 describing how torque and moment of inertia are related when using only a stiff transmission between engine and wheels. The transmission has a fixed conversion ratio

$$i = \frac{\omega_{engine}}{\omega_{wheel}} \tag{2.8}$$

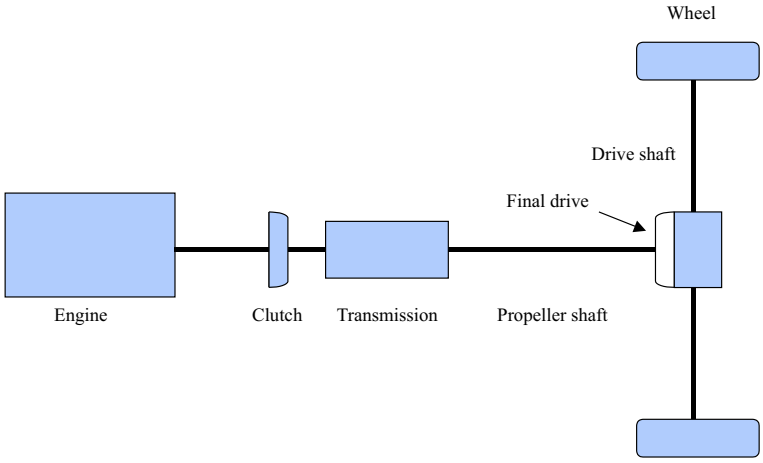

Figure 2.2: Driveline for a vehicle [6]

and an efficiency factor η_i. By considering the conversation law of energy, the next two expression describe the relation between the torque and moments of inertia:

$$M'_{wheels}\,\omega_{engine} = \frac{1}{\eta_i} M_{wheels}\,\omega_{wheels} \Leftrightarrow$$

$$M_{wheels} = \eta_i i M'_{wheels} \tag{2.9}$$

$$\frac{J'_{wheels}\,\dot{\omega}_{engine}}{2}\,\eta_i = \frac{J_{wheels}\,\dot{\omega}_{wheels}}{2} \Leftrightarrow$$

$$J_{wheels} = \eta_i\, i^2 J'_{wheels} \tag{2.10}$$

The expressions (2.9) and (2.10) play an important role when modeling the torque flow through the different parts of the model. Figure 2.4 shows the vehicle driveline used in the model with respective torque and angle velocity labels.

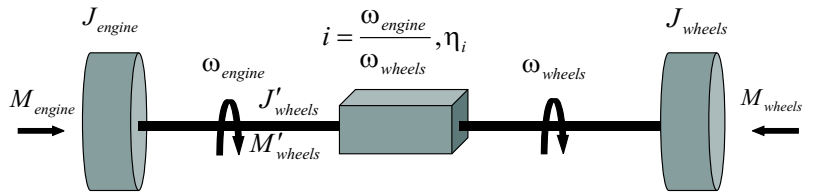

Figure 2.3: Transmission

2.2.2 Model Subsystems

As mentioned before the driveline model used here is different from the driveline shown in Figure 2.2. First of all there is no clutch. The small amount of time that the clutch is actually engaged does not effect the total torque flow significantly and can therefore be neglected. Instead an automatic gear box with fixed gear ratios is used. All wheels are also assumed to be one single wheel, placed in the middle of the vehicle, with a mass equal to the sum of all wheels. For that reason the drive shaft is not considered here. Furthermore the propeller shaft and final drive is represented by a differential gear. Another simplification that has been made is the neglection of possible torsional effects.

As mentioned in the beginning of this chapter, one of the system inputs is the required engine torque which will be transferred via different subsystems. These subsystems, the parts of which this model consist, are:

- Combustion engine. When the model is used for the simulations further on in this thesis the input will just be the required engine torque. It is also possible to feed this block with the angle of the gas pedal instead of directly applying the torque. In that case the combustion engine block calculates the required engine torque via the current rotating speed of the engine and the throttle angle. Since these relations are strongly nonlinear, two maps are used. The first map describes the throttle angle α from the angle of the gas pedal α_{pedal}, i.e.

  $$\alpha = f_1(\alpha_{pedal}) \tag{2.11}$$

  Another map calculates the engine torque M_{engine} by taking the throttle angle and the current rotation speed as inputs, i.e.

  $$M_{engine} = f_2(\alpha, \omega_{engine}) \tag{2.12}$$

  To avoid unrealistic large engine torques caused by the interpolation or unrealistic inputs, two maps calculates the upper and

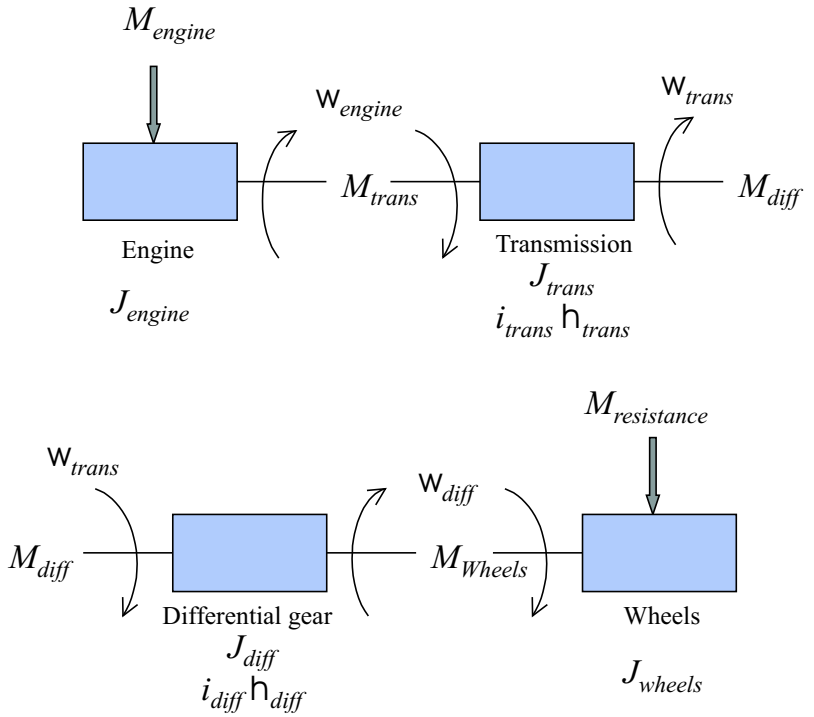

Figure 2.4: The subsystems used in the model with torque and angle velocity labels

lower bound for the engine torque in respect to the current rotation speed, i.e

$$f_{min}(\omega_{engine}) \leq M_{engine} \leq f_{max}(\omega_{engine}) \tag{2.13}$$

These limit values are chosen if the requested torque is out of the bounds.

- Transmission. For every gear the transmission has a transmission ratio i_{trans} and an efficiency coefficient $\eta_{i_{trans}}$, which convert torque and moment of inertia from the engine according to equation (2.9) and (2.10).

- Differential gear. The differential gear is between the transmission and the wheels. In the differential gear block calculations for torque and moment of inertia at the wheels take place. They are derived in the same manner as for the gear box with the exception that the transmission ratio i_{diff} and the efficiency coefficient $\eta_{i_{diff}}$, unlike the transmission, have constant values and are not dependent on the gear.

- Wheels. In this block a new angle velocity is obtained by using (2.2). The torque is the one coming from the differential gear where braking and resistance torque have been subtracted, i.e.

$$M_{wheels} = M_{diff} - M_{brake} - M_{resistance} \tag{2.14}$$

The total moment of inertia is the sum of the incoming moment of inertia from the differential gear block and the total moment of inertia caused by the wheels. Since all of the four wheels are considered as one the contribution in moment of inertia from the wheels is given by

$$J = J_4 + m_{tot} r^2 \tag{2.15}$$

The constants J_4 and m_{tot} is the moment of inertia for all the four wheels and the total mass for the vehicle respectively.

With the wheel radius of the value r_{wheel}, equation (2.3) can be expressed as

$$\begin{aligned} \dot{v} &= \frac{1}{m} F_{tot} = \frac{1}{m} \Bigg(\frac{1}{r_{wheel}} \left(M_{diff} - M_{brake} \right) - \\ &\quad \frac{1}{2} \rho_{air} A_f c_w v^2 - f(v) mg \cos(\gamma) - mg \sin(\gamma) \Bigg) \end{aligned} \tag{2.16}$$

where M_{diff} is calculated as described above.

Chapter 3

MPC Theory

Model Predictive Control (MPC) is a control technique that has become very popular in the last 10 to 20 years. Several predictive control techniques were presented more or less simultaneously under different names [2]. An early introduction of a predictive control technique was made by the oil company Shell in the late 70's called Dynamic Matrix Method (DMM). Using MPC requires big computational effort and has, for that reason, been especially used in chemical industries where one can find systems with large time constants. There are several reasons why MPC has become popular in the industry. Most important point is that MPC can handle constraints as for example limited control variables and safety limits on the outputs explicitly. Furthermore MPC is easy to explain for those who do not have much knowledge of automatic control theory. For a small introduction to MPC see for example [4] or [5]. A more complete theory can be found in [2].

3.1 Introduction to Model Predictive Control

MPC can typically be described in the following steps:

1. At time t, predict future outputs $\hat{\bar{y}}(t+k|t)$, $k = 1, \ldots, N$. These outputs will depend on future control signals $\bar{u}(t+j), j = 0, 1, \ldots, M$ and on measurement obtained at time t.

2. Formulate some criteria J based on these variables and optimize with respect to the control signals.

3. Feed the model with $\bar{u}(t)$.

4. $t := t + 1$

5. Go back to 1.

The values M and N are called *control* and *prediction horizons* where $M \leq N$. The prediction horizon should be chosen in such a way that it covers a settling time of the system, so that the system has reached a more stable behavior. However, the prediction horizon should at the same time not be chosen too large since it affects the computational efforts. If there were no disturbances and no mismatches between model and controlled system, a prediction over an infinite horizon in time $t = 0$ would be enough. Then it would be possible to just apply the control vector found in time $t = 0$ for all times $t \geq 0$. In reality disturbances exist, as well as a possibility that the model does not totally describes the controlled object, which makes it necessary to recalculate the prediction and optimal control action in the next time step.

3.2 Linear Model Predictive Control

When explaining the theory of MPC it makes no difference if the system has one or more inputs and outputs, therefore a multi-variable system is assumed. The model of the object to be controlled is described with a time discrete linear state-space system:

$$\bar{x}(k+1) = A\bar{x}(k) + B\bar{u}(k) \tag{3.1a}$$

$$\bar{y}(k) = C\bar{x}(k) \tag{3.1b}$$

where

$$\bar{u}(k) = \begin{bmatrix} u_1(k) \\ \vdots \\ u_m(k) \end{bmatrix}, \ \bar{y}(k) = \begin{bmatrix} y_1(k) \\ \vdots \\ y_p(k) \end{bmatrix}, \ \bar{x}(k) = \begin{bmatrix} x_1(k) \\ \vdots \\ x_n(k) \end{bmatrix} \tag{3.2}$$

and A, B and C are matrices of proper sizes. The cost function J that has to be minimized can be chosen in many ways depending on the purpose of the controller. The minimization is often done under constraints and then especially in form of limited control variables, $u_{min} \leq u_i \leq u_{max}$. A very common situation is that one has to follow a predefined reference trajectory, $\bar{r}$, and that the criteria punish deviations in the output signal from this trajectory. In this case the criteria can look like this[1]:

$$\min_{\bar{u}_{min} \leq \bar{u} \leq \bar{u}_{max}} J = \sum_{j=0}^{N-1} \|\bar{y}(k+j+1) - \bar{r}(k+j+1)\|_{Q_1}^2 + \|\bar{u}(k+j)\|_{Q_2}^2 \tag{3.3}$$

[1] $\|\bar{x}\|_Q^2 = \bar{x}^T Q \bar{x}$

In (3.3) the control signal has been added to the criteria which is common to prevent the control signals to get unrealistic values. The number N is the *prediction horizon* and Q_1 and Q_2 are weighting matrices and semi-positive definite. The weighting matrices and the prediction horizon are tuning parameters that have to be chosen by the user. If, for example, it is desirable to control the system to follow the reference fast, and it is not so important to use small control inputs, then the elements in Q_1 should be chosen big and in Q_2 small.

3.2.1 Prediction of Future States

When using MPC without taking disturbances and model mismatches into account, the prediction for future states is derived in an explicit way. If the model is used for a two-step prediction the state can be obtained by using the model recursively. The predicted state is then expressed as

$$\bar{x}(k+2) = A\bar{x}(k+1)+B\bar{u}(k+1) = A^2\bar{x}(k)+AB\bar{u}(k)+B\bar{u}(k+1) \quad (3.4)$$

Now this process can be repeated to predict all the N future states. It is practical to write these predictions in a compact form like

$$X = H\bar{x}(k) + SU \quad (3.5)$$

where U is the future control signals and X the states that these generates, i.e.

$$U = \begin{bmatrix} \bar{u}(k) \\ \bar{u}(k+1) \\ \vdots \\ \bar{u}(k+N-1) \end{bmatrix}, \; X = \begin{bmatrix} \bar{x}(k+1) \\ \bar{x}(k+2) \\ \vdots \\ \bar{x}(k+N) \end{bmatrix} \quad (3.6)$$

and where

$$H = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \; S = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \quad (3.7)$$

3.2.2 Analytical Solution

It can be interesting to look at an example where no constraints exist even if one of the greatest advantages in using MPC is that it can handle constraints. The interesting aspect of this special case is that an analytical solution can be found. With an analytic solution faster calculations and better analysis are possible. First, introduce a vector

R for the future reference values and a matrix $\mathcal{C}$ which transforms future states into future outputs.

$$R = \begin{bmatrix} \bar{r}(k+1) \\ \bar{r}(k+2) \\ \vdots \\ \bar{r}(k+N) \end{bmatrix}, Y = \begin{bmatrix} C\bar{x}(k+1) \\ C\bar{x}(k+2) \\ \vdots \\ C\bar{x}(k+N) \end{bmatrix} = \begin{bmatrix} C & & & \\ & C & & \\ & & \ddots & \\ & & & C \end{bmatrix} X = \mathcal{C}X \tag{3.8}$$

With help of (3.5) the criteria in (3.3) can now be written in a vector based form as

$$J = (\mathcal{C}(H\bar{x}(k) + SU) - R)^T \mathcal{Q}_1 (\mathcal{C}(H\bar{x}(k) + SU) - R) + U^T \mathcal{Q}_2 U \tag{3.9}$$

where $\mathcal{Q}_1$ and $\mathcal{Q}_2$ have been defined as

$$\mathcal{Q}_1 = \begin{bmatrix} Q_1 & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_1 \end{bmatrix} \mathcal{Q}_2 = \begin{bmatrix} Q_2 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_2 \end{bmatrix} \tag{3.10}$$

In this case, were no constrains exist, the criteria will be minimized by the U for which the gradient of (3.9) is zero, i.e.

$$2S^T\mathcal{C}^T\mathcal{Q}_1(\mathcal{C}(H\bar{x}(k) + SU) - R) + 2\mathcal{Q}_2 U = 0 \tag{3.11}$$

The resulting U is:

$$U = -(S^T\mathcal{C}^T\mathcal{Q}_1\mathcal{C}S + \mathcal{Q}_2)^{-1} S^T\mathcal{C}^T\mathcal{Q}_1(\mathcal{C}H\bar{x}(k) - R) \tag{3.12}$$

The control signal, $u(t)$, is the first m rows in U, i.e.

$$\bar{u}(k) = [I\ 0\ \dots\ 0]U \tag{3.13}$$

3.3 Nonlinear Model Predictive Control

While the theory for linear Model Predictive Control has been examined and advanced much the last decades, the theory for nonlinear Model Predictive Control (NMPC) has not been paid as much attention. The interest for NMPC is steadily increasing because the industrial processes are running under tighter conditions and more constraints as safety and environmental restrictions are added. To meet these demands, nonlinearities often have to be taken into explicit account in the controller which justifies the use of NMPC. Nonlinear predictive control can be seen as an extension of linear Model Predictive Control.

An introduction to NMPC can be found in [3]. Analogous to (3.1) a general NMPC problem can be formulated as:

$$\begin{aligned} \min_{\bar{u}} J &= \min_{\bar{u}} \sum_{k=1}^{M} L(\bar{x}(k), \bar{u}(k)) \\ \bar{x}(k+1) &= f(\bar{x}(k), \bar{u}(k)), \quad \bar{x}(0) = \bar{x}_0 \\ \bar{u}(k+j) &\in \mathcal{U}, j = 0, \ldots, M-1 \\ \bar{u}(k+j) &= \bar{u}(k+M), j \geq M \\ \bar{x}(k) &\in \mathcal{X}, \forall k \geq 0 \end{aligned} \tag{3.14}$$

Here both the criteria and the system are nonlinear, which will lead to a nonlinear controller where L is describing a nonlinear function. The control signals after the control horizon, M, have been set to $\bar{u}(k+M)$. The problem with the nonlinear controller is that it can be very hard to find an analytical solution and in most cases this does not exist. A numerical solution is then required but can demand great computational time, often longer than what is acceptable for on-line applications.

One method to decrease the computational time is to linearize the system around different working points. The linearized system can then be used to obtain linear controllers which can control the nonlinear plant in each working point.

Chapter 4

Dynamic Programming Theory

Dynamic programming was first developed by R. Bellman in the 50's [1]. As the name indicates Dynamic Programming is a dynamic optimization method which means that instead of obtaining a constant value as optimum the method gives us an optimal *trajectory*.

4.1 Bellman's Optimization Principle

Assume that the optimization problem is to find a set of control signals $\{\bar{u}(0), \ldots, \bar{u}(p-1)\}$ that brings a system, $\bar{x}(k+1) = f(\bar{x}(k), \bar{u}(k))$, from state $\bar{x}(0) = \bar{x}_0$ to $\bar{x}(p) = \bar{x}_p$ and at the same time minimizes a cost criteria

$$J = \sum_{k=0}^{p-1} L(\bar{x}(k), \bar{u}(k)) \tag{4.1}$$

L describes the cost of going from present state $\bar{x}(k)$ to the next state by applying $\bar{u}(k)$. The optimization principle from Bellman says that if the total state trajectory from $\bar{x}(0)$ to $\bar{x}(p)$ is optimal, then every part of the trajectory has to be optimal. That means that if the system in time k_1 is in the state $\bar{x}(k_1) = \bar{x}_{k_1}$ the problem to be solved is to find the inputs $\{\bar{u}(k_1), \bar{u}(k_1+1), \ldots, \bar{u}(p-1)\}$ that minimize the sum of the costs

$$J_1 = \sum_{k_1}^{p-1} L(\bar{x}(k), \bar{u}(k)) \tag{4.2}$$

Note that it is not necessary to have the cost function L in (4.2) time independent, but for the simplicity this has been assumed here. In Figure 4.1 the optimization principle for a continues system has been

reproduced. The optimal trajectory from $\bar{x}(t_0)$ to $\bar{x}(t_p)$ has been divided into two trajectories, one from state $\bar{x}(t_0)$ to an arbitrary state $\bar{x}(t_k)$ and another from $\bar{x}(t_k)$ to $\bar{x}(t_p)$. In Figure 4.1 these are denoted as $\bar{x}_1^*(t)$ and $\bar{x}_2^*(t)$, respectively. The fact that the total trajectory was optimal implicates that the rest trajectory $\bar{x}_2^*(t)$ has to be optimal too. If not, another way, $\bar{x}_2^{'}(t)$ in Figure 4.1 would exist, which together with $\bar{x}_1^*(t)$ would be an optimal trajectory. Since the trajectory $\bar{x}_1^*(t)$ together with $\bar{x}_2^*(t)$ has already been stated as optimal that can not be the case. The principle is valid even in the discrete case.

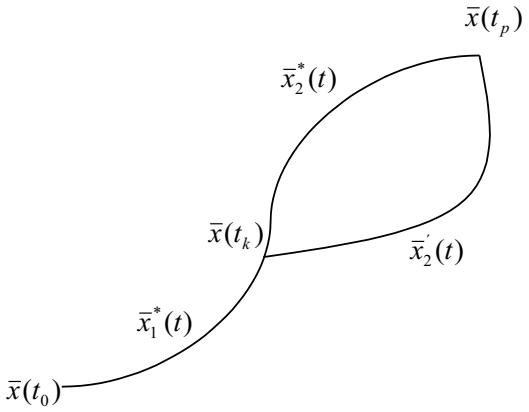

Figure 4.1: Bellman's optimization principle

Now let the minimal cost for going from state $\bar{x}(k_1)$ to $\bar{x}(p)$ be represented by a remaining cost function, $R(\bar{x}(k_1))$. This function gives the minimum value of (4.2), i.e.

$$R(\bar{x}(k_1)) \equiv J_1^* = \min_{\{\bar{u}(k_1),\ldots,\bar{u}(p-1)\}} J_1 = \min_{\{\bar{u}(k_1),\ldots,\bar{u}(p-1)\}} \left\{ \sum_{k_1}^{p-1} L(\bar{x}(k), \bar{u}(k)) \right\} \tag{4.3}$$

As the remaining cost in the end state has to be defined beforehand and is therefore known, it is possible to set up a recursive formula for calculating all the remaining costs. In the end state the remaining cost function is

$$R(\bar{x}(p)) = \phi(\bar{x}(p)) \tag{4.4}$$

therefore the remaining cost in time $p-1$ can be derived as

$$R(\bar{x}(p-1)) = \min_{\bar{u}_{p-1}} \{ L(\bar{x}(p-1), \bar{u}(p-1)) + \phi(\bar{x}(p)) \} \tag{4.5}$$

Insert (4.4) into (4.5) and using $\bar{x}(p) = f(\bar{x}(p-1), \bar{u}(p-1))$

$$R(\bar{x}(p-1)) = \min_{\bar{u}_{p-1}} \{ L(\bar{x}(p-1), \bar{u}(p-1)) + R(\bar{x}(p)) \} \tag{4.6}$$

Note that the minimization in (4.6) only is dependent on $\bar{u}(p-1)$ and not on future control signals which is a consequence of Bellman's optimization principle. It is now possible, from (4.6), to express the remaining cost for an arbitrary state $\bar{x}(k)$, i.e.

$$R(\bar{x}(k)) = \min_{\bar{u}_k}\{L(\bar{x}(k), \bar{u}(k)) + R(\bar{x}(k+1))\} \tag{4.7}$$

The remaining cost can now be calculated recursively according to (4.7) which is known as Bellman's equation.

4.2 Discretization and Quantization

The recursive equation (4.7) can be used with a time-discrete system, where the control and input variables have been discretized. Since the function for the remaining costs (4.7) is not an analytic expression the method for finding the minimum value has to be done numerically, for example, in time k, by applying all combinations of control inputs $\bar{u}^i(k),\ i = 1, \ldots, N_{\bar{u}}(k)$. These different inputs will bring the system into one of the in time $k+1$ discretized states $\bar{x}^j(k+1),\ j = 1, \ldots, N_{\bar{x}}(k+1)$. The quantities $N_{\bar{u}}(k)$ and $N_{\bar{x}}(k)$ are the number of discretized control signals and states in time k. With $\bar{x}^{i,j}(k+1)$ denoted as the state which has been reached from state $\bar{x}^i(k)$ by applying $\bar{u}^j(k)$, the remaining cost for going from state $\bar{x}^i(k)$ in time k is

$$J_i^j(k) = L(\bar{x}^i(k), \bar{u}^j(k)) + R(\bar{x}^{i,j}(k+1)) \tag{4.8}$$

If the system hasn't been discretized, $R(\bar{x}^i(k))$ is saved as the minimum value of (4.8), which is obtained when the optimal control input $\bar{u}^*(\bar{x}^i(k))$ is applied. Most technical systems are time-continuous and have to be discretized with a quantization step. The way this step length is chosen is a balance between computational time and accuracy for optimum. A problem that can occur when discretizing states is that one can reach states that don't agree with the states $\bar{x}^i(k)$. Therefore, in time k, $\tilde{R}^{i,j}(k+1)$ is used as an interpolation between the contiguous states' remaining costs.

When the backward calculation in (4.8) has been done, the remaining costs for every state are available and a forward calculation is necessary to find out the trajectory of optimal control values $\bar{u}^*(k)$. Starting in state $\bar{x}(0)$, all $\bar{u}^j(0)$ are applied to the system, which will then reach a new state, and then the resulting costs $L(\bar{x}(0), \bar{u}^j(0))$ are added to the interpolated remaining cost $\tilde{R}(\bar{x}(k+1))$. Optimal $\bar{u}^j(0)$ is the one that minimizes this sum. This means the $\bar{u}^*(k)$ can be calculated as follows

$$\bar{u}^*(k) = \arg\min_{\bar{u}(k)} \Big\{ L(\bar{x}(k), \bar{u}(k)) + \tilde{R}(\bar{x}(k+1)) \Big\} \tag{4.9}$$

Chapter 5

Control and Optimization

In the two previous chapters the theory of Model Predictive Control and Dynamic Programming was presented. In this chapter the theory will be applied to the special case of a Stop and Go controller. This controller and different control strategies will be presented.

5.1 The Controller

The controller structure to be used in this thesis is reproduced in Figure 5.1. The algorithm for the MPC controller is implemented in C code to be used in a Simulink S-function block. Inputs to the controller are the current position and the velocity of the preceding and the following car. Outputs from the control algorithm are the required engine torque and gear. Data from the preceding car is obtained from different telematic devices as sensors, GPS and 3D-maps. The sensors will detect the position and the velocity of the car in front and the GPS together with the 3D-maps deliver information about the slope of the road. In each time step the controller will use these inputs to obtain the optimal torque request and gear for an optimal driving strategy in respect to a criteria, which will be explained later on.

Figure 5.2 shows the following and preceding cars with the different labels which have been used during the optimization. The following car, which is to be controlled by the MPC controller, is represented here by a Simulink model of the vehicle as described in Chapter 2. This model is a nonlinear system and for that reason nonlinear Model Predictive Control has been used.

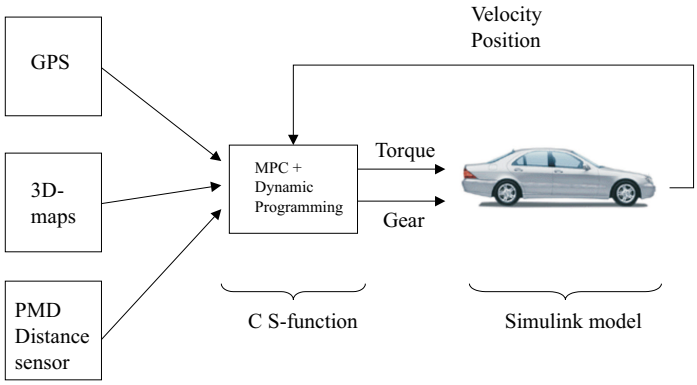

Figure 5.1: An overview of the controller structure

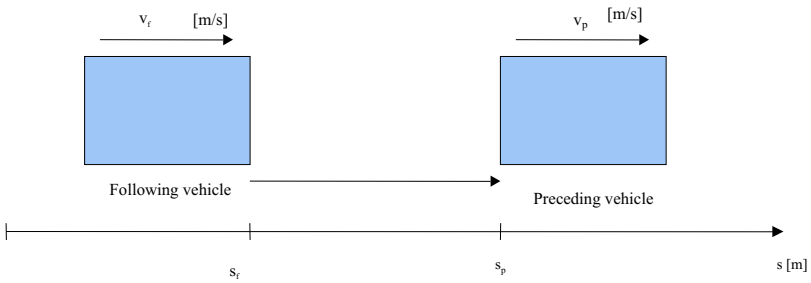

Figure 5.2: The control situation

5.2 Optimization and Control Strategies

5.2.1 Distance Base

The velocity of the controlled vehicle has been chosen as the only state, i.e. $\bar{x} = v$. The drawback of having the velocity as a single state is that no other quantities than the velocity can be controlled, but on the other hand it reduces the computational effort when using Dynamic Programming. The system can be described with a time-based nonlinear state space model

$$\dot{\bar{x}}(t) = f(\bar{x}(t), \bar{u}(t)) \tag{5.1a}$$

$$\bar{x}(t) = v(t) \tag{5.1b}$$

$$\bar{u}(t) = [M_e(t), g(t)]^T \tag{5.1c}$$

with inputs M_e, the requested engine torque, and gear g. The model (5.1) is a time-based model. It is not suitable to use a time-based model for this control purpose when the incline, given from the GPS and the 3D-map, isn't dependent on time but dependent on distance. Apart from that, a system based on distance instead of time is preferable for the controlling actions, which will be seen later on. Using the relation

$$\frac{dv}{ds} = \frac{dv}{dt} \cdot \frac{dt}{ds} = \frac{1}{v} f(v, M_e, g) \quad v \neq 0 \tag{5.2}$$

system (5.1) has been transformed into a distance dependent form. This is the form that from now on will be used to represent the system to be controlled, implemented in C and Simulink S-function.

5.2.2 Control Strategies

There are several things that have to be defined before starting the optimization according to the theory in chapter 4. A control strategy has to be set up as well as parameters for the optimization, which include:

- Defining the prediction and control horizon.
- Determine the velocity that the vehicle will reach in the last prediction step by defining the *Mayer function*.
- Defining the cost function.

The following strategy that has been used can be formulated as follows: *when the following car reaches the position where the preceding vehicle is at present time, it should have the same velocity as the preceding vehicle.*

That means, using the notation in Figure 5.2, when $s_f = s_p$ then $v = v_p$. The prediction horizon has been chosen to be s_p. The reason for this is that the mayer function can then be defined so that it punishes deviations in velocities from the preceding vehicle's. From now on s_f is assumed to be zero and the mayer function has been chosen as:

$$\phi(v(s_p)) = Q_1(v(s_p) - v_p(s_p))^2 \tag{5.3}$$

with a weighting constant Q_1. Since it is difficult to determine the required velocity for a specific time it would have been harder to formulate the mayer function (5.3) with a system depending on time instead of position. To define the cost function L in (3.14) one has to decide what the aim for the optimization is, i.e. what is going to be minimized? As the aim of this thesis is to find out if the MPC approach can reach good following and in the same time reduce the fuel consumption, the cost function has been chosen to be

$$L = f_{fuel}(n, M_e) + Q_2(v - v_p)^2 = f_{fuel}(C(g) \cdot v, M_e) + Q_2(v - v_p)^2 \tag{5.4}$$

The function giving the fuel consumption is in form of a map taking the rotation speed of the engine and the engine torque as inputs. The present rotation speed can be expressed in velocity by multiplying with

$$C = \frac{30 \cdot i_{diff} \cdot i_{trans}}{\pi \cdot r_{wheel}} \tag{5.5}$$

Apart from the fuel consumption a quadratic function has been included in the cost function. This term increases the cost when the velocity differs from the preceding vehicle's, and with the weight Q_2 one can decide what an acceptable difference in the velocities is.

5.3 The Control Algorithm

In order to use the model together with Dynamic Programming the system has to be discretized with a distance-step size Δs. The discretization method used in (5.6) is the Euler approximation and the approximated distance based state space system can then be written as

$$v(s + \Delta s) = v(s) + \frac{\Delta s}{v} f(v, M_e, g) \quad v \neq 0 \tag{5.6}$$

The MPC problem can now be defined according to (3.14). As the prediction horizon has been chosen to be s_p and the system is discretized

with the step size Δs, the optimization problem to solve is

$$\min_{\{\bar{u}(0),\ldots,\bar{u}(\lfloor \frac{s_p}{\Delta s} \rfloor \Delta s)\}} \left\{ \phi(v(s_p)) + \sum_{k=0}^{\lfloor \frac{s_p}{\Delta s} \rfloor} L(v(k \cdot \Delta s), M_e(k \cdot \Delta s), g(k \cdot \Delta s)) \right\} \tag{5.7a}$$

subject to

$$v(k+1) = v(k) + \frac{\Delta s}{v} f(v, M_e, g) \quad v \neq 0 \tag{5.7b}$$

$$M_e \in \mathcal{U}_e, \quad g \in \mathcal{U}_g \quad v \in \mathcal{X} \tag{5.7c}$$

The sum in the criteria J is called the *Lagrange function*. Unlike the Mayer function it affects the criteria in every time step, (see Chapter 4). The sets $\mathcal{U}_e$, $\mathcal{U}_g$ and $\mathcal{X}$ represent valid engine torques, gears and velocities. To obtain a solution of (5.7) numerically, the engine torque, which is limited to an upper and lower bound, has to be discretized with a step ΔM_e. The other control input is already discrete as the vehicle always uses one of five gears. The torque bounds are evaluated from the present velocity and gear as well as the possible gears. The gears that can be used are dependent on the current velocity. The sets can therefore be written as:

$$\mathcal{U}_e = \{M_{min}(v, g), M_{min}(v, g) + \Delta M_e, \ldots, M_{max}(v, g)\}$$

$$\mathcal{U}_g \subseteq \{g_1, g_2, g_3, g_4, g_5\}$$

Here $\mathcal{U}_g$ is dependent on the state, i.e. $\mathcal{U}_g$ contains the valid gears for the specific velocity.

The MPC algorithm to obtain the optimal control signals for solving (5.7) can be described in a few steps

1. Use the model to predict possible future velocities for every position step until the prediction horizon.
2. Apply the theory of Dynamic Programming to obtain optimal vectors containing torque requests and gears for every position step.
3. Take the first values in these vectors as inputs for the control object.

5.3.1 Prediction of Future Velocities

The first step in the MPC algorithm is to predict future *reachable* velocities, until the prediction horizon $s = s_p$, for the car according to the present velocity. This can be done with help of a model of the con-

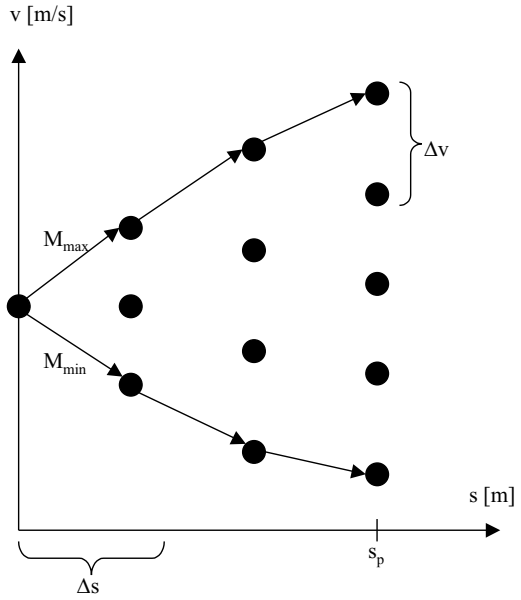

Figure 5.3: The prediction of possible future states until the prediction horizon

trol object. This model is implemented in C code and is equal to the Simulink model used as control object. Consequently, no disturbances or model mismatches have been taken into account. Now assume the situation in Figure 5.3 where the car has the initial velocity v_0. By applying maximum and minimum torque with a "suitable" gear maximum and minimum velocity in position Δs will be obtained. The gear is chosen as the one which, together with the extreme torque values, will bring the system into the maximum and minimum velocity in distance Δs. To find the maximum velocity in $s = 2\Delta s$, maximum torque is applied when the car has highest velocity in $s = \Delta s$, and analogous the minimum velocity is obtained when the minimum torque is applied from the lowest velocity. This process is then repeated for all position steps until the prediction horizon. Thus, all extreme values for the car's velocity until $s = s_p$ are known and therefore every velocity between the maximum and minimum velocities is also reachable. In Figure 5.3 all the states are reproduced with a velocity quantization Δv and the set of feasible velocities is

$$\mathcal{X}(k) = \{v_{min}(k) + \Delta v, \ldots, v_{max}(k)\} \tag{5.8}$$

5.3.2 Optimization

When all possible states have been obtained the next step is to find the optimal trajectory to position $s = s_p$ with help of the Dynamic Programming method. The optimization is performed according to the theory in Chapter 4 and the same terminology and notation is used. In the last step the remaining costs for all states are defined to be $R(v(s_p)) = \phi(v(s_p)) = Q_1(v(s_p) - v_p(s_p)^2)$ and for every $k = s_p - \Delta s, \ldots, 0$ the following calculations are done:

1. All combinations of admissible engine torques and gears $(M_e, g) \in \mathcal{U}_e \times \mathcal{U}_g$ are applied to the model (5.6) for all feasible velocities $v \in \mathcal{X}(k)$. Every combination of control inputs will bring the system into a new state in $k = s_p$.

2. Interpolate the remaining costs for every new state for which the input combinations give rise to.

3. Calculate the cost function (5.4) for all combinations and add to the corresponding interpolated remaining cost.

4. The combination that gives the total minimum cost are the optimal control inputs, and the total cost is saved as the remaining cost in $k = s_p - \Delta s$

This is then repeated for all k until $k = 0$, and all remaining costs for feasible velocities have been obtained. During the prediction the preceding car's velocity has been assumed to be constant. To obtain the optimal trajectory a forward calculation analogous to the backward one is necessary. A part of the trajectory is then applied to the control object.

Chapter 6

Simulations and Results

6.1 Reference Trajectory

To obtain a suitable reference trajectory for the preceding vehicle in a Stop and Go situation, measurements have been done in dense traffic during the rush hours on the highway A8 in Germany. Figure 6.1 shows a part of the obtained trajectory which was used for most of the simulations. This trajectory was chosen because it includes different driving actions typical for Stop and Go driving. That means periods of relatively high and low acceleration, stops, and periods when the car is driving at a more and less steady pace.

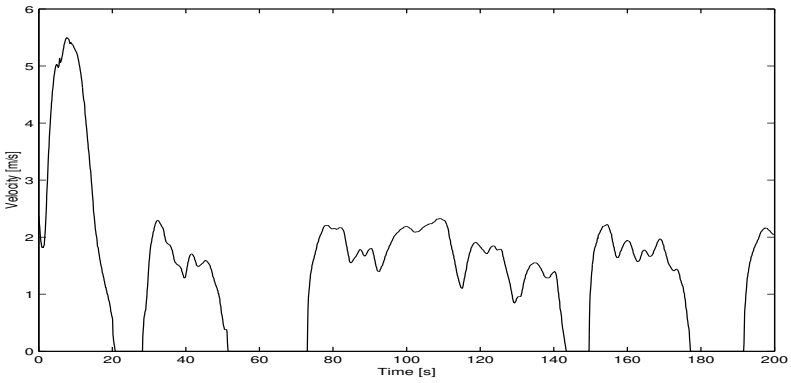

Figure 6.1: One part of the measurements made in a real Stop and Go situation

6.1.1 Comparing the Fuel Consumption

As the aim for the optimization is to save fuel it is necessary to know if the fuel consumption was reduced when using the Stop and Go algorithm. To do that one has to compare the consumed fuel when the fuel optimized control was used, to some reference consumption. The comparisons were made with help of a *quasi stationary* model. Instead of having the gear and engine torque as inputs to the model the quasi stationary model takes the velocity, acceleration, and the slope of the road as inputs. From these data the model calculates the engine torque and engine rotation speed that corresponds to the given inputs, i.e. the calculations are done "backwards" compared to the model which is used as control object. It is interesting to compare the fuel consumption with the one that the controlled car would have if the velocity was exactly the same as the preceding vehicle's using the optimal gear given from the optimization. In that case there is no chance of affecting the inputs for the controller and for that reason no possibilities of reducing the fuel consumption. When applying the velocity trajectory in Figure 6.1 the reference fuel consumption obtained is $31.61l/100km$. It is necessary to express the fuel consumption as depending on the distance, because when comparing different velocity trajectories the driven distance will be different in each case. Due to the fact that fuel is consumed even when the car has stopped, and is running on idle speed, it is important that the comparisons are made with the same trajectory.

6.2 Simulations

Before starting the simulations the user has to define different parameters depending on what behavior one requires from the controller. These are:

- Weighting constants Q_1 and Q_2
- Torque quantization ΔM_e
- Distance quantization Δs
- Velocity quantization Δv

Apart from these parameters the sample time for the simulation also has to be defined. A small sample time will contribute to a better following but, on the other hand, increase the computational effort. The discretization quantities ΔM_e, Δs and Δv should be chosen as small as possible to increase the accuracy in the simulations. In the simulations later on in this chapter the values of these quantities have been chosen smallest possible without getting to large simulation times.

First of all, it is not obvious that the quadratic term in the cost function (5.4) has to be included. Since the Mayer function (5.3) has been defined in a way that the velocity will reach the preceding one's in the last prediction step, this should be enough. In every time step the controlling actions that are applied to the car will try to adjust the velocity to the preceding vehicle's velocity. Figure 6.2 shows the results from a simulation where the quadratic term has been neglected, i.e. when $Q_2 = 0$. The problem that occurs is that the following vehicle

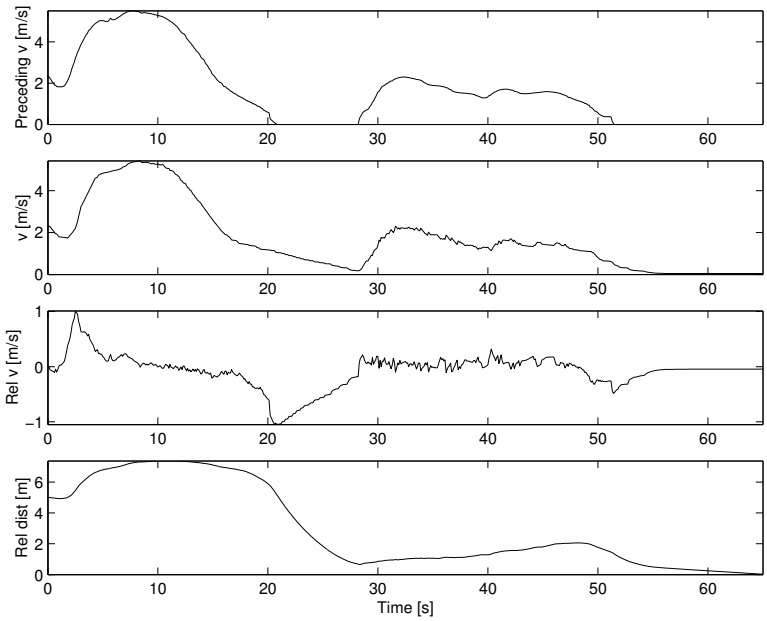

Figure 6.2: Simulation without taking the quadratic expression in the cost function into account, i.e. $Q_2 = 0$.

can not maintain a safety distance and crashes into the car ahead. Due to the fact that the velocity of the controlling vehicle is the only state there is no possibilities to control the distance between the cars. When $Q_2 = 0$ the controller will not adjust the speed during the prediction but only in the last prediction step.

To prevent the car from crashing into the car in front Q_2 has to be chosen to a value which is not zero. In Figure 6.3 Q_2 has been chosen to 0.001. When choosing $Q_2 \neq 0$ the controller will punish deviations, in the velocity from the preceding one, during the whole prediction and not only in the last prediction step. The result is a better following and the controlled car is able to keep a safety distance to the preceding vehicle. The simulation parameters used for the results seen in Figure

6.3 are:

- $Q_1 = 10000$, $Q_2 = 0.001$
- $M_e = 1\,Nm$
- $\Delta s = 0.3\,m$
- $\Delta v = 0.05\,m/s$

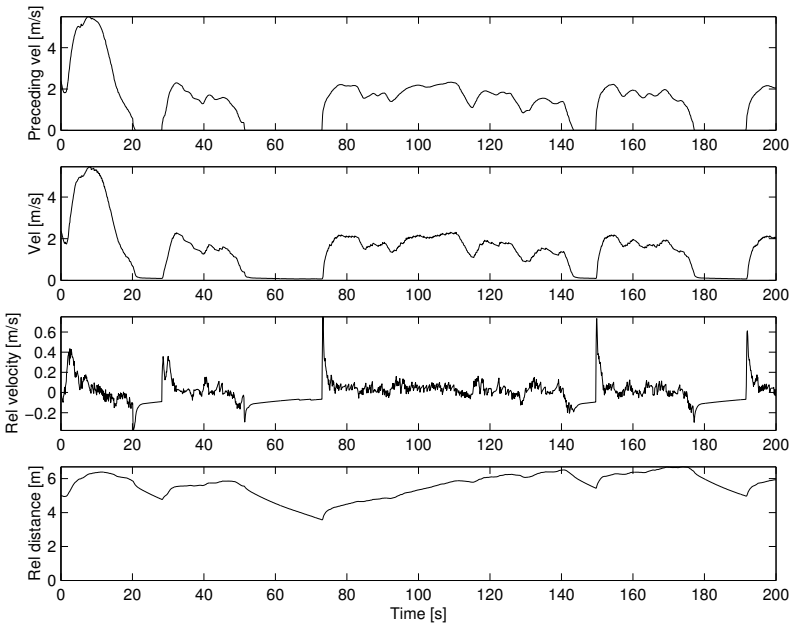

Figure 6.3: Simulation with high punishment on velocities that differ from the preceding vehicle's. $Q_1 = 10000$, $Q_2 = 0.001$

The fuel consumption for this simulation was $30.70l/100km$ and compared to the reference consumption this is a reduction of 2.9%.

A result of the fact that the simulation in Figure 6.3 has been done with a relative high punishment on velocities that differ from the preceding vehicle's ($Q_1 = 10000$) is that a rather good following has been achieved and the controlled car does not crash into the car in front. In the optimization problem (5.7) the velocity equal to zero is not defined. That means that the following car is not able to stop totally and is for that reason approaching the preceding car when it has stopped. This explains the "dipping behavior" of the relative velocity as seen in Figure 6.3. The consequences of a high punishment is that many of the predicted states can not be chosen by the controller. Therefore the

controller is not able to choose the inputs to the control object freely. The punishment can be chosen to be small, as seen in Figure 6.4. Here is $Q_1 = 1$ and the following is still acceptable. The difference to the simulation in Figure 6.3 is that the fuel consumption now have been reduced to $30.41l/100km$ or 3.8% less then the reference consumption.

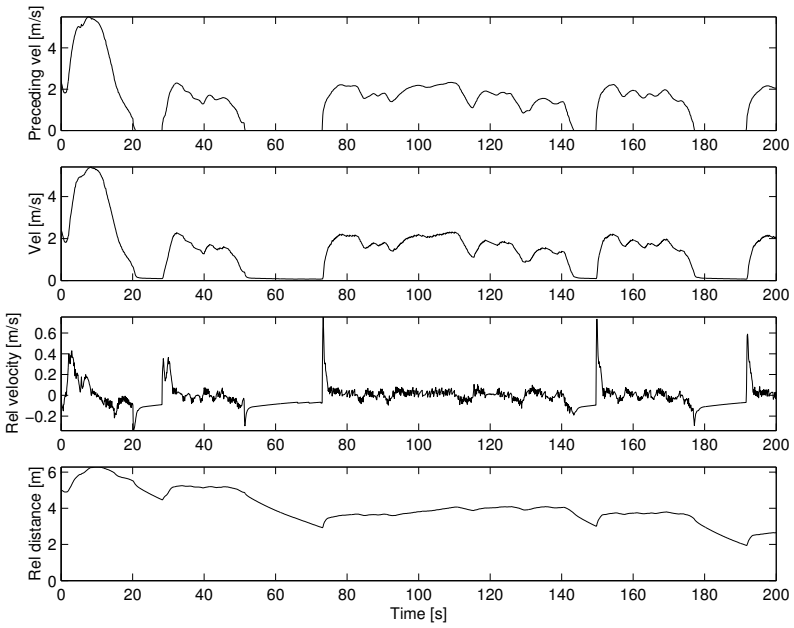

Figure 6.4: Simulation with low punishment on velocities that differ from the preceding vehicle's. $Q_1 = 1$, $Q_2 = 0.001$

How the matrices should be chosen is a balance between the fuel consumption and the ability to maintain an acceptable distance to the preceding vehicle. When trying to decrease the fuel consumption more by setting Q_1 to 0.01 and giving the controller even more freedom the behavior of the following car is not satisfactory. Figure 6.5 shows the simulation and instead of crashing into the car in front as in the simulation showed in Figure 6.2 this time a too big relative distance is obtained. The examples that have been presented here show the problem with choosing the optimal weighting constants for a specific driving situation.

Summarizing the results from this section it has been seen that whether a good following or low fuel consumption is desirable it can be controlled via different weighting strategies. To achieve a good following where the fuel reduction is not as important as keeping a proper dis-

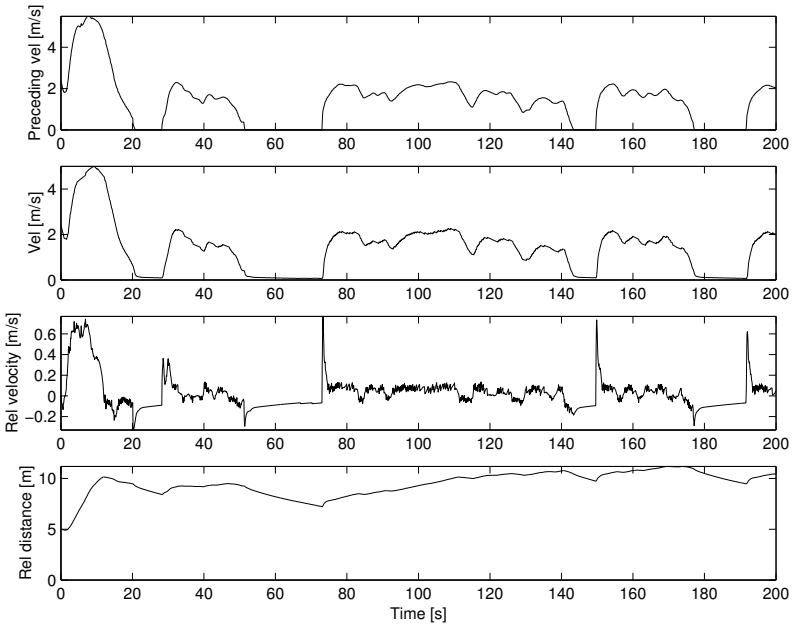

Figure 6.5: Simulation with very low punishment on velocities that differ from the preceding vehicle's. $Q_1 = 0.01$, $Q_2 = 0.001$

tance to the car in front, a high punishment on velocities that differ from the preceding one's can bee used as seen in Figure 6.3 ($Q_1 = 10000$, $Q_2 = 0.001$). On the other hand if the fuel reduction is more important than maintaining the distance it can be done with a small punishment instead, as for example in Figure 6.4 ($Q_1 = 1$, $Q_2 = 0.001$).

6.2.1 Simulation problems

During the work with the simulations different problems have occurred. These have mainly been numerical problems in the form of oscillations in the input signals. A consequence of this is that the input signals was very hard to analyze. The main reason for this oscillatory behaviour is the discretization of the input signals. The optimal input signals, which are given from the optimization algorithm, may not correspond exactly to any of the discretized values allowed as inputs. Thus the value of the signal will oscillate, always trying to be as close to the optimal (non-discretized) input as possible. Another problem is that the model that has been used is not optimal for the purpose discussed in this thesis and needs to be improved. This due to the model's inability to handle the case of zero velocity.

Chapter 7

Conclusions

The aim of this thesis was to obtain a controller for Stop and Go situations and investigate if there were any possibilities of saving fuel. Real driving situations have been used and for many cases the results are promising. The fuel consumption has been reduced which was the main goal of the S&G algorithm and in the same time an acceptable safety distance to the preceding vehicle is maintained. The numerical problems that occurred in the input signals to the model have made it difficult to evaluate how the signals are changing during the simulations and if it would be possible to apply this input sequences to a real engine. Due to the fact that the control algorithm was created using only a one state model it has been difficult to control the distance between the controlled and the preceding car. Therefore different optimization criteria have been tested. The first try, with punishment only in the last prediction step (Figure 6.2), was not enough for satisfactory following. When adding the quadratic term by choosing $Q_2 \neq 0$ in (5.4) this was improved. The controller was then able to adjust the velocity during the prediction and not only in the last prediction step and thus avoided crashing into the car in front. With help of the weighting constants it has been seen that the controller can be tuned in different ways depending on which purpose the controller has. Depending on whether a good following is required or if the fuel consumption has to be reduced as much as possible, the weighting constants are chosen accordingly, i.e. different punishment on the deviations of the velocity from the preceding vehicle's are used, see Chapter 6. As seen before the fuel consumption has been reduced with both high and low punishment. The mean reduction for these simulations have been around 3%.

Chapter 8

Further Work

There is still room for improvements of the controller actions and control strategies in the field of automated S&G driving. During the work with this thesis assumptions and restrictions have been made which have influenced the behavior of the following car (the control object). One of the first thing to do would be to try to minimize the numerical problems in the form of oscillations in the input signals. The model that has been used is also not optimal for the use in the simulations and may contributed to the numerical problems. These kind of problems are well known and have occurred in other applications similar to the one which has been described in this thesis. A specialized model would be required for best possible control results.

The discretization of the inputs often leads to problems during the optimization. When the optimal inputs are not valid due to the discretization the controller has to choose inputs that differ from the optimal one and oscillations occur. This problem would be one of the main points of work when continuing this project.

The model used consists of only one state, the velocity. This is of course a restriction in the controller and to add a state or states would lead to a controller with more possibilities to improve the behavior of the following car. Since adding states increases the computational effort one has to decide if it is really necessary to add an extra state or if the performance of the controller can be improved with help of other control strategies.

In this thesis the aim has been the reduction of the fuel consumption. This is of course not the only thing to be taken into account if a S&G controller will be a reality in future cars. The control actions must then also be comfortable and acceptable to the driver, which will lead to a much wider optimization problem.

References

[1] R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[2] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[3] F. Allgöwer R. Findeisen. An introduction to nonlinear model predictive control.

[4] L. Ljung T. Glad. *Reglerteori- Flervariabla och olinjära metoder*. Studentlitteratur, 1997.

[5] L. Ljung T. McKelvey A. Stenman J. Löfberg T. Glad, S. Gunnarsson. *Digital Styrning Kurskompendium*. 2002.

[6] L. Nielsen U. Kiencke. *Automotive Control Systems*. Springer, 2000.

Notation

Notation

Symbols

A_f	Maximum vehicle cross area
α	Throttle angle
c_r	Rolling friction coefficient
c_w	Air resistance constant
F_{air}	Air resistance
F_{brake}	Braking force
F_{df}	Drive force generated from the engine
$F_{incline}$	Force on vehicle caused by the incline of the road
F_{res}	Sum of the resistance forces acting on the vehicle
$F_{rollingres}$	Force on vehicle caused by the rolling resistance
F_{tot}	Total force acting on the vehicle
g	Gravitational acceleration or gear
i	Gear ratio
J	Moment of inertia
J_4	Moment of inertia for all wheels
m	Vehicle mass
M	Control horizon
M_{engine}	Engine torque
$M_{resistance}$	Resistance torque
M_{wheels}	Torque acting at the wheels
N	Prediction horizon
η_i	Efficiency factor of the gear
n_{engine}	rpm in engine
n_{wheel}	rpm of the wheels
Q	Weighting matrix
r_{wheel}	Wheel radius
ρ_{air}	Air density
s	Distance
s_p	Position for the preceding car
v	Velocity

v_p	Velocity for the preceding car
γ	Angle of incline
ω	Angle velocity
ω_{engine}	Angle velocity of the engine
ω_{wheels}	Angle velocity of the wheels

Abbreviations

MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
ICC	Intelligent Cruise Control
S&G	Stop and Go
RPM	Revolutions Per Minute
trans	Transmission
diff	Differential gear