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

Optimal Formation of Heavy Duty Vehicle Platoons

Dennis Edblom

Master of Science Thesis in Electrical Engineering

Optimal Formation of Heavy Duty Vehicle Platoons:

Dennis Edblom

LiTH-ISY-EX–20/5310–SE

Supervisor: **PhD Student Viktor Leek**
ISY, Linköpings universitet

Examiner: **Associate Professor Jan Åslund**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Platooning has the potential to significantly reduce fuel consumption, but with heavy duty vehicles scattered on roads driving alone, there is a need for coordination. One solution is for a vehicle to increase its speed to catch up and platoon with a preceding vehicle. This could reduce the fuel consumption of a mission, but it could also increase it if too much fuel is spent catching up. By finding the fuel consumption of catching up and platooning and comparing it to driving alone the decision of whether or not to catch up can be made.

This thesis proposes a fuel-optimal algorithm based on a look-ahead controller taking future road topography into account to find the optimal trajectory and merge point when catching up to a preceding vehicle. By weighting time against fuel in the objective function, the addition of a state to keep track of time can be avoided and thus the algorithm can remain low in complexity, making it suitable for dynamic programming (DP). The DP algorithm is iterated in a forward fashion keeping track of the time-to-come for each state until it catches up to the preceding vehicle, then the platooning is simulated with a constant time gap, making it easy and fast to simulate. The algorithm is tested on real-world road topography data where it showed that taking road topography into account when choosing the merge point can have a significant fuel reduction.

Acknowledgments

This thesis was done at Linköping University. I would like to thank Jan Åslund for great insight. And I would like to especially thank Viktor Leek for giving me great support and many opportunities over the years.

Linköping, May 2020
Dennis Edblom

Contents

Notation

Förkortningar

Förkortning	Betydelse
HDV	Heavy-duty vehicle
LAC	Look-ahead control
ALAC	Adaptive look-ahead control
CLAC	Cooperative look-ahead control
DP	Dynamic Programming
CC	Cruise control
ACC	Adaptive cruise control

1 Introduction

Transportation of goods has become a vital part of today's society. Where 76.6% of the inland freight transportation in the EU is done by road [7]. This accounts for 35% of the transport-related carbon dioxide (CO2) emissions which is 7% of the total energy-related emissions in the world [1]. There is an increasing focus on the environmental impact of heavy duty vehicles (HDV) by the worlds governments. The EU has put forwards CO2 emission reduction targets for HDVs which ensures that new HDVs will have a reduction of 30% in CO2 emissions by 2030 [6]. Fuel prices account for around 30% of the total operational cost of HDVs [13]. This gives environmental, economical, and political reasons to increase the efficiency of HDVs.

Platooning is one solution where HDVs drive close behind each other to reduce the air drag of the following vehicles, reducing the fuel consumption up to 7% [2]. Another solution is look-ahead control (LAC), where a HDV uses future road topography to find an optimal trajectory and reduce fuel consumption, [9].

1.1 Motivation

Platooning has the potential to significantly reduce fuel consumption, but with HDVs scattered on roads driving alone, there is a need for coordination. One solution is for a vehicle to catch up to another and platoon with it. This has the potential of reducing the fuel consumption, but it also has the potential to increase it if too much fuel is used to catch up. With LAC being able to reduces fuel consumption by taking future road topography into account, could it be used to see if it is fuel-efficient to drive faster to catch up, and how to do it optimally.

1.2 Purpose

The purpose of this thesis has been to create an algorithm that takes road topography into account to find the optimal way to catch up to a preceding vehicle and platoon with it. This algorithm could be used to decide if it is fuel-efficient to catch up or not.

1.3 Questions

The results expect to give answer to these questions.

- When is it fuel-efficient to catch up to another vehicle and platoon?
- What is the optimal way to catch up to another vehicle?
- Is there a need to take road topography into account?

1.4 Thesis Outline

This thesis is divided into 9 chapters. Chapter 2 covers related research in the subject field of the thesis. Chapter 3 describes the models used in the proposed algorithm. Chapter 4 covers the catch-up problem and the catch-up algorithm used to solve the problem. The algorithm is then validated in chapter 5. In chapter 6 the method for finding the optimal solution is described and the algorithm is extended to work in parallel. Chapter 7 covers numerical results from different missions, where one mission is on real-world road topography data. Chapter 8 discusses the results and method used. And finally, chapter 9 contains the conclusions of this thesis and future work.

1.5 Delimitations

The algorithm proposed in the thesis is not validated with real-life experiments. In the driveline model the components are considered stiff and ideal. Brakes are not considered when catching up since it is not optimal, but they are used when platooning. Traffic is not regarded in the thesis. The trajectory of the preceding vehicle is the same as the one the following vehicle would have used if it kept driving alone. The aim of the thesis has been to study the algorithm, so the software written will not be feasible for on-board use in a real environment.

2

Related research

Related research in the field of this thesis is focuses on two fields, look-ahead controller and platooning.

2.1 LAC

A look-ahead controller (LAC) uses the HDV's position together with future road topography to minimize the fuel consumption of a given mission. In Hellström et al. [9], the authors introduced a LAC based on dynamic programming (DP) that minimizes fuel consumption without increasing the travel time. The proposed algorithm finds the optimal trajectory for the wanted mission time without introducing an extra state to keep track of time, by weighting time in the objective function. This made it possible to find the fuel-optimal solution for the wanted mission time. The authors used kinetic energy as a state instead of velocity, and showed that it helped reduce interpolation errors and oscillations of the solution. By using kinetic energy and not having to introduce an extra state the algorithm is computationally efficient. The efficiency with the robustness of the algorithm means that it is appropriate to use on-board a HDV in real-world applications. The same approach was used in Alam [2] for a single HDV but the author also looked into other approaches more suitable for use when platooning, see Section 2.2.1. The LAC in Hellström [9] was evaluated and it achieved a fuel consumption of around 3.5% lower than when using a cruise controller (CC).

2.2 Platooning

There are many different problems to be solved for platooning. Here they have been divided into two subgroups, one for platoon control strategies where the

vehicles are already in a platoon, and one for platoon formation strategies.

2.2.1 Control strategies

In Alam [2] a longitudinal model for platooning is proposed, where the model is a modified version of the single HDV model proposed in the same thesis. The difference is that the air drag scales with the distance between the preceding and following vehicles. The scaling of the air drag is determined from empirical measurements.

In Alam [2], the author looked at LAC when platooning, and found that LAC for a single HDV is not practical when platooning. The author then looked into adaptive look-ahead control (ALAC) and cooperative look-ahead control (CLAC) compared to the commercially available adaptive cruise control (ACC). ACC is similar to a normal cruise controller (CC) but it takes into account the relative distance and velocity to the preceding vehicle, which has been considered as a means to enable platooning in Hendrick et al. [8]. ALAC assumes that the preceding vehicle is unwilling to change its velocity strategy, so the optimization problem becomes how should the following vehicle optimally adapt to the preceding vehicle. In CLAC both vehicles are willing to change their velocity strategy to minimize the total fuel consumption. It was found that both ALAC and CLAC has fuel saving potential and that CLAC obtains the maximum fuel reduction. For a steep uphill segment, a fuel reduction of 0.7% can be obtained with the CLAC compared to ACC. And for a steep downhill segment, a fuel reduction of up 14% can be obtained compared to ACC.

2.2.2 Platoon formation strategies

In Johansson and Mårtensson [10], the authors looked at how different profit distribution models affects vehicles choice of platooning. They considered the case of all vehicles having a common origin and destination with different departure times, where the vehicles can choose to depart at other times to get the benefits from platooning. The vehicles were thought of as being owned by different transportation companies, and thus wanting to maximize their own profit. The profit is described as a utility function with the benefits of platooning and the cost of deviating from the default departure time. They tested four different distribution models. Even out, where the following vehicle gave the leading one a pre-defined compensation. Score system, where the vehicles had an assigned score which decides the leading vehicle. Market where there are buyers and sellers trying to maximize their gain. And finally cooperative where everyone cooperated. The authors found that the cooperative distribution model had the best utility gain while the even out model was close second. This suggests that if the profit is shared in a fair way, then competing companies can obtain a profit close to the cooperative solution, by acting selfishly.

In the article *When is it Fuel Efficient for a Heavy Duty Vehicle to Catch up With a Platoon?* by Kuo-Yun Liang, Jonas Mårtensson and Karl Henrik Johansson [12], they looked at just that. The authors created a formula based on a flat road with

no vehicle accelerations, to calculate if it is more fuel efficient to catch up and platoon or to keep driving alone. They created a break-even ratio which says when it is as fuel efficient to catch up as it is to continue alone. The ratio takes into account the speed increase that the HDV has, the air drag reduction when platooning, the distance between the vehicles, and the distance to the goal. They also created a platooning incentive factor, which calculated normalized air drag for the whole mission. With the factor, the authors also found the most beneficial speed increase to catch up with, given the air drag reduction. This finding is very useful since it gives the theoretically most beneficial speed increase when catching up, on a flat road with no vehicle accelerations.

3

Modeling

This chapter goes through the different models in this thesis. The HDV dynamics are modeled with a longitudinal model, and a powertrain model. The powertrain model consists of an engine model, a driveline model and a gear shift model. Then the effects of platooning are modeled with a platoon model.

3.1 Longitudinal model

A HDV driving on a road can be modeled in one dimension along the road, a so-called longitudinal model, see Figure 3.1. The forces acting upon the HDV are presented in Table 3.1, with the parameters shown in table 3.2. The states for the model are velocity v, gear g and the traveled distance s and the controls are fueling u_f, gear u_g and braking u_b.

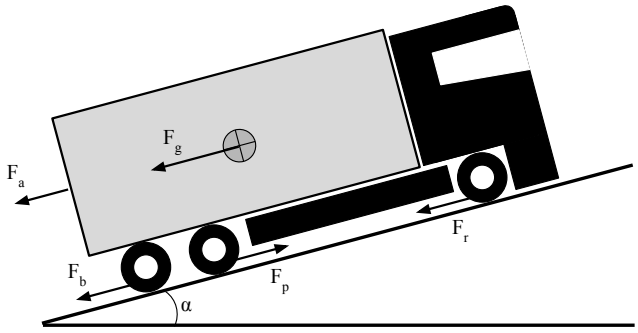

Figure 3.1: *Illustration of forces acting upon the HDV when moving in one dimension.*

Table 3.1: *Longitudinal forces acting upon the HDV.*

Variable	Description	Expression
$F_p(v, g, u_f)$	Propulsion force	See Sec. 3.3
$F_a(v)$	Air drag	$\frac{1}{2} c_d A_f \rho_a v^2$
$F_r(s)$	Rolling resistance	$m g_0 c_r \cos \alpha(s)$
$F_g(s)$	Gravitational force	$m g_0 \sin \alpha(s)$
$F_b(u_b)$	Braking force	$m g_0 \mu u_b$

Table 3.2: *Parameters of the longitudinal model.*

Parameter	Description	Unit
c_d	Air drag coefficient	[-]
A_f	HDV frontal area	m^2
ρ_a	Air density	$[\mathrm{kg/m}^3]$
r_c	Rolling resistance coefficient	[-]
m	HDV mass	[kg]
g_0	Gravitation coefficient	$[\mathrm{m/s}^2]$
α	Road slope	[degrees]
μ	Traction coefficient	[-]

Newton's second law of motion gives the motion equation

$$m\frac{dv}{dt} = F_p(u_f) - F_a(v) - F_r(s) - F_g(s) - F_b(s, u_b) \tag{3.1}$$

With F_r, F_g and F_b depending on the road slope which depends on the position. Formulating the motion equation into spatial coordinates

$$mv\frac{dv}{ds} = F_p(v, g, u_f) - F_d(s, v, u_b) \tag{3.2}$$

With $F_d = F_a + F_r + F_g + F_b$. The propulsion force is generated by the engine, which is further explained in Section 3.3.

3.1.1 Kinetic Energy formulation

The model can be expressed in terms of energy, as suggested in Hellström [9] where it is shown that formulating the problem in terms of energy reduces oscillations and interpolation errors when performing numerical optimization. Then the states for the model become kinetic energy E_k, gear g, and the traveled distance s.
Introducing kinetic energy

$$E_k = \frac{1}{2} m v^2 \tag{3.3}$$

The relation

$$\frac{dv}{dt} = v\frac{dv}{ds} = \frac{1}{2}\frac{d}{ds}v^2 \tag{3.4}$$

can be expressed in terms of energy

$$\frac{dE_k}{ds} = mv\frac{dv}{ds} \tag{3.5}$$

Now the motion equation in (3.2) can be expressed in terms of energy

$$\frac{dE_k}{ds} = F_p(\sqrt{2E_k/m}, g, u_f) - F_d(\sqrt{2E_k/m}, s, u_b) \tag{3.6}$$

3.2 Powertrain components

The powertrain consists of models for the engine, clutch, transmission, propeller shaft, final drive, drive shaft and wheels. See Figure 3.2 for an illustration and Table 3.3 for the variables of the powertrain.

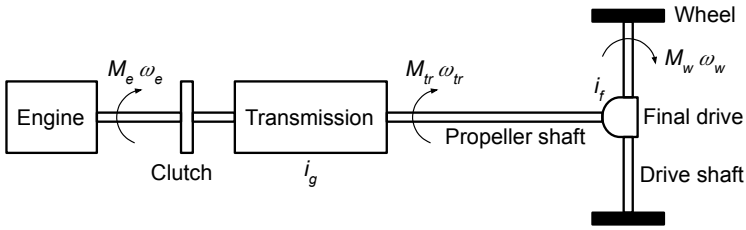

Figure 3.2: *Illustration of the powertrain.*

3.3 Engine model

The engine is modeled with Willans line [16], which is a simple yet powerful model. The model converts input energy into output work and losses, see Equation (3.7). The model gives the output torque M_e from the engine based on fueling u_f. It does this by using an energy converter W_e and an external loss term W_{loss} which can be fit to measured engine data.

$$M_e(u_f) = W_e u_f - w_{loss} \tag{3.7}$$

Table 3.3: Powertrain parameters and variables.

Parameter	Description	Unit
M_e	Torque produced by engine	[Nm]
M_{tr}	Torque from transmission	[Nm]
M_f	Torque from final drive	[Nm]
M_w	Torque on wheels	[-]
i_g	Gear ratio in transmission	[-]
i_f	Gear ration in final drive	[-]
i_{tot}	Total gear ratio	[-]
r_w	Wheel radius	[m]
ω_e	Engine rotational speed	[rad/s]
ω_{tr}	Transmission output rotational speed	[rad/s]
ω_w	Wheel rotational speed	[rad/s]
v	Vehicle speed	[m/s]
$\dot{m}$	Fuel mass flow	[m^3/s]
n_{cyl}	Number of cylinders	[-]
n_r	Revolutions per stroke	[-]

3.3.1 Fuel consumption

The fuel consumption is modeled as a mass flow

$$\frac{dm}{dt} = \frac{n_{cyl}}{2\pi n_r}\omega_e u_f = \frac{n_{cyl}}{2\pi n_r}\frac{i}{r_w} v u_f \tag{3.8}$$

Using the relation $v = \frac{ds}{dt}$

$$\frac{dm}{ds} = \frac{n_{cyl}}{2\pi n_r}\frac{i}{r_w} u_f \tag{3.9}$$

The parameters can be seen in Table 3.3.

3.4 Driveline model

The driveline consists of all the powertrain components except the engine. The variables can be seen in Table 3.3 It is modeled as stiff and ideal, meaning there is no torsion on the components and no losses between them. The wheels are always in traction, meaning there is no slip between them and the road. This makes it easy to model the relationships between the driveline components. It is also assumed that the clutch is engaged at all times and has no slip or losses. In reality, the driveline is not ideal and there will be some losses, these losses can be included in the loss term w_{loss} in the engine model (3.7). The rotational speed at

the wheels assuming that there is no slip can be derived from the velocity of the HDV

$$\omega_w = \frac{v}{r_w} \tag{3.10}$$

Then the rotational speed at the transmission can be calculated under the same assumption

$$\omega_{tr} = i_f \omega_w \tag{3.11}$$

With this and assuming that the clutch has no slip the engines rotational speed can be calculated

$$\omega_e = i_g \omega_{tr} \tag{3.12}$$

Where i_g is the gear ratio for the currently engaged gear g. Now the rotational speed of the engine can be expressed in the HDV's velocity

$$\omega_e = i_{tot} \frac{v}{r_w} \tag{3.13}$$

Where $i_{tot} = i_g i_f$.

With the engine speed known the torque from the engine can be calculated with Willans line (3.7). Then since the driveline is stiff with no slip the torque can be transferred through the transmission and final drive to the wheels

$$M_{tr} = i_g M_e \tag{3.14}$$

$$M_w = i_f M_{tr} \tag{3.15}$$

The Torque at the wheels can then be expressed with

$$M_w = i_{tot} M_e \tag{3.16}$$

Where $i_{tot} = i_g i_f$. Then the propulsive force can be found

$$F_p = \frac{M_w}{r_w} = \frac{i_{tot} M_e}{r_w} \tag{3.17}$$

Where M_e is given by Willans line (3.7).

3.5 Gear shift model

The transmission is of the manual type and gear shift is done with engine control. The idea is to control the engine to a state where there is no torque acting on the transmission and then disengaging the gear. Then controlling the engine to

a state where the input and output rotational speed of the transmission for the new gear is synchronized. During the gear shift, the HDV will have no engine propulsion and thus roll freely, which is also modeled.

Consider a gear shift from g_1 to g_2 then the current gear $g(t)$ becomes

$$g(t) = \begin{cases} g_1 & t < 0 \\ 0 & 0 \leq t \leq \tau \\ g_2 & t > \tau \end{cases} \tag{3.18}$$

τ is chosen to be constant for all possible gear shifts

$$\Delta T = \tau \tag{3.19}$$

Then with the initial speed v_0 we get the distance traveled when shifting gears

$$\Delta s = \int_0^\tau v(t)dt \tag{3.20}$$

Where $v(t)$ is given by solving the initial value problem given by (3.2) for $F_p = 0$ in the interval $t \in [0, \tau]$ and $v(0) = v_0$.

Then with $v(t)$ known

$$\Delta v = v(\tau) - v_0 \tag{3.21}$$

The torque of the engine, while the gear is disengaged, is dependent on the engine inertia

$$I_e \dot{\omega}_e = T_e \tag{3.22}$$

With the initial velocity v_0 and the final velocity v_1 known the initial engine speed ω_0 and the desired one ω_1 can be found with the help of (3.13) for the different gears g_1, g_2. The fuel needed for a gear shift can be seen as the fuel required to synchronize the rotational speed and the fuel required to overcome friction in the engine. To find the fuel consumption of synchronizing the engine speed, the rotational energy required is used with a proportionality constant γ [g/J] is used, as suggested in Hellstöm [9].

$$\Delta m = \gamma \frac{1}{2} I_e(\omega_1^2 - \omega_0^2) + m_{fric}(\omega_w, u_f, \tau) \tag{3.23}$$

3.6 Platoon model

The air drag is modeled in Section 3.1 but this does not take into account the effects of platooning. Here is a modified version that takes the distance between the preceding and following HDVs into account

$$F_a(v) = \frac{1}{2} c_D A_f \rho_a v^2 \tag{3.24}$$

Where $c_D = c_d * (1 - f_i(d)/100)$ and f_i is a non-linear function that scales the air drag coefficient c_d. The air drag is reduction is dependent on the distance d between the preceding and following vehicle and the position i in the platoon. In this thesis, a linearized model is used for the air drag reduction as suggested in Kemppainen [11].

$$f_1(d) = -0.9379d + 12.8966 \qquad 0 \leq d \leq 15 \tag{3.25}$$
$$f_2(d) = -0.4502d + 43.0046 \qquad 0 \leq d \leq 80 \tag{3.26}$$
$$f_3(d) = -0.4735d + 51.5027 \qquad 0 \leq d \leq 80 \tag{3.27}$$
$$f_i(d) = f_3(d) \qquad i \geq 4 \tag{3.28}$$

4

The catch-up problem

When deciding whether to catch up to a preceding vehicle and form a platoon or to continue as planned, the cost of the cases needs to be compared. There are many solutions to finding the cost of the vehicle continuing as planned. For instance, a cruise controller (CC) where the HDV tries to maintain a speed reference set by the driver, or a look-ahead controller (LAC) where the HDV takes into account the road topography and finds a fuel-optimal trajectory. This thesis will look at a LAC solution and take into account the road topography when catching up and when continuing as planned.

This chapter mainly looks at the problem of one HDV minimizing fuel consumption for a mission by catching up to a preceding vehicle and then forming a platoon, (the catch-up problem). The chapter also shows how the cost of the HDV continuing as planned can be found with the same algorithm. The catch-up problem is split into two parts, the first part where the HDV's are apart from each other and one has to catch up, and the second part where they platoon together to the end, see Figure 4.1. It is not evident how this is best done, is it better to drive faster and consume more fuel but catching up quicker or is it better to go slower but then platoon for shorter.

The proposed algorithm consists of two parts one for each part of the problem. The first part is handled by the catch-up algorithm and the second part is handled by the platooning algorithm.

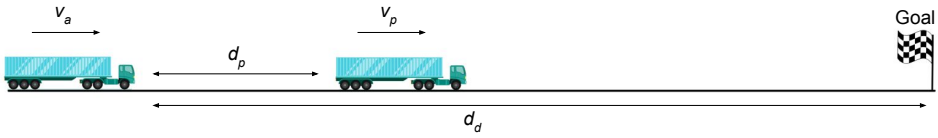

Figure 4.1: *Illustration of starting position for the catch up problem.*

4.1 Objective

The objective is to minimize the fuel M required for the whole mission, firstly catching up and then platooning to the goal. The problem considered is the one where the HDV reaches the goal at the same time as the platoon, resulting in a fixed end time T_0.
Giving us the objective function:

$$\begin{aligned} &\text{minimize} \quad M \\ &\text{s.t.} \quad T = T_0 \end{aligned} \tag{4.1}$$

To avoid increasing the complexity by adding a state for time and succumbing to the *Curse of dimensionality* [4] another objective function is used:

$$\text{minimize} \quad M + \beta T \tag{4.2}$$

This is an approach used in Monastyrsky and Golownykh [14]. This weights fuel against time with β as a tuning parameter. This objective function will be used for the catch-up algorithm.

4.1.1 Time Penalty Parameter β

Weighting fuel against time, the penalty parameter β is hard to get right. A high β will result in time being costly, therefore the algorithm will spend a bit more fuel to reduce time resulting in the mean velocity being higher. If β is lower then the mean velocity will be lower. Therefore β can be viewed as the parameter that sets the mean velocity when catching up. Since β sets the mean velocity it will also decide when the HDV catch up to the platoon, a higher β will result in us catching up quicker, while a lower one will result in it taking more time.

To calculate β the same approach as in Hellström [9] will be used. Consider a small step Δs then Equation (3.6) assuming no breaking becomes

$$\Delta E_k = (F_p - F_a(\hat{v}) - F_r(\hat{v}) - F_g)\Delta s \tag{4.3}$$

Then using a proportionality constant γ [g/J] which approximates the required fuel ΔM needed for an increase in energy ΔE_k

$$\Delta M \approx \gamma \Delta E_k \tag{4.4}$$

the equation can be rewritten

$$\Delta M = \gamma(\Delta E_k + (F_a(\hat{v}) - F_r(\hat{v}) - F_g)\Delta s) \tag{4.5}$$

Then for a mission of distance S with constant velocity $\hat{v}$ as the solution, using Equation (4.5) and (4.2) with $S = \hat{v}T_0$

$$J(\hat{v}) = \gamma(E_k(S) - E_k(0)) + \gamma(F_a(\hat{v}) - F_r(\hat{v}) - F_g)S + \beta S/\hat{v} \tag{4.6}$$

When the HDV has reached the wanted velocity $\hat{v}$ then $J(\hat{v})$ will be a stationary point and the derivative will be zero, resulting in

$$\beta = \gamma \hat{v}^2(F_a'(\hat{v}) + F_r'(\hat{v})) \tag{4.7}$$

which can be used to approximate β in terms of mean velocity $\hat{v}$.

With β being able to be expressed as a mean velocity a lower and a upper bound for β can be set. The lower bound being for the mean velocity that would result in the vehicle reaching the end without platooning at the fixed time T_0. And the upper bound being when the HDV drives as fast as possible in regards to the speed limit and vehicle dynamics until it reaches the platoon. Any higher β would not have any impact since the HDV is not able to go any faster.

4.2 Discretization

The states and controls are discretized, see Figure 4.2. The different states are position s, kinetic energy E_k and gear G which are discretized into s_k, Ek_i and G_j respectively. The distance s is the independent variable and it is discretized into N stages for the whole mission. The objective function for the whole mission, catching up and then platooning can be formulated as

$$\text{minimize} \quad J_0(x_0) + \sum_{k=1}^{N} \zeta_k(x_k, u_k) \tag{4.8}$$

which can then be divided into two parts

$$\text{minimize} \quad J_0(x_0) + \sum_{k=1}^{N_p} \zeta_k(x_k, u_k) + \sum_{k=N_p+1}^{N} \zeta_{p,k}(x_k, u_k) \tag{4.9}$$

Where $J_0(x_0)$ is the cost for the different initial states x_0 and is used to start the algorithm in the desired starting state by giving the undesired states a high cost. The two sums represent the cost for the different parts of the mission, catching up and platooning. $\zeta_k(x_k, u_k)$ and $\zeta_{p,k}(x_k, u_k)$ is the step cost for the catch-up and the platooning parts. The step cost represents the cost it takes to go from stage $k-1$ to k. The stage N_p that separates the two parts is the stage where the HDV has caught up to the preceding vehicle, and can then platoon (the merge point). This stage is found by running the catch-up algorithm in Section 4.3.

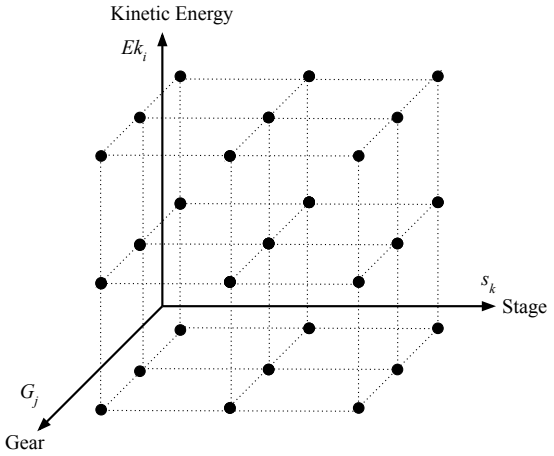

Figure 4.2: *Illustration of the discretized 3D state space.*

4.3 Catch Up Algorithm

To solve the catch-up problem, a modified version of the LAC algorithm proposed in Hellström [9] is used. The algorithm in Hellström is a dynamic programming (DP) algorithm, it starts at the end state and iterates backward calculating the cost-to-go for each state until the starting state is reached. Then the lowest cost-to-go from the starting state to the end state is found, i.e. the optimal solution. This is a so-called DP backward algorithm since it starts at the end state and iterates backward.

Dynamic programming backwards

How it works is that the state space is discretized as seen in Figure 4.2, that is in 3 dimensions one along the distance traveled where each point is called a stage, one for kinetic energy and one for the available gears.

Considering only one gear, first the cost-to-go at the end state is defined. This is to make sure the algorithm goes to a desired state since it will take the path with the lowest cost. The cost can either be used to determine a specific state or it can be a function that weights fuel against kinetic energy, which is useful in the case of a receding horizon since energy gained becomes fuel saved when running the algorithm for the next horizon.

With the cost at the end stage s_N known, the algorithm can calculate the cost-to-go $J(s_{N-1}, E_k)$ for each discretized state E_k at the previous stage s_{N-1}. This is done by having the algorithm try a discretized set of the possible controls $uf_k \in U_f$. Then the cost-to-come $J(s_{N-1}, E_k)$ for the state E_k is the sum of the step cost $\zeta(uf_k)$ and the cost-to-come for the state the step ends up in $J^*(uf_k)$

$$J(s_n, E_k) = \min_{uf_k \in U_f} \{\zeta(uf_k) + J^*(uf_k)\} \tag{4.10}$$

This is then iterated backward until the starting stage s_0 is reached, and the optimal solution is found. The objective function is the one from (4.2), where β weighting fuel against time, is used to set the mean velocity of the mission as done in Section 4.1.1.

To consider more than one gear, another dimension is added with one level per gear so that all states are evaluated for all possible gears. Then at each state, a gear shift is evaluated to see if it is more efficient to switch gears or not. This means that an increase from 1 gear to 4 gears will result in a 4 times larger state space. Meaning that 4 times as many states need to be evaluated, and in the worst-case scenario it will take 4 times longer to find the solution.

This works well in the case of a look-ahead controller, but not as well for the catch-up problem. This is because the algorithm starts at the end and iterates backward. In the catch-up problem, the endpoint would either be at the end of the mission or the merge point where the HDV has caught up to the preceding vehicle.

Starting at the end of the mission

If the algorithm were to start at the end of the mission, the HDVs would start within the platoon. Then the problem would become, when should the algorithm leave the platoon to get back the starting point. The starting point also has to satisfy the initial conditions, i.e. the distance between the HDVs at the beginning and the starting velocity. To solve this, another state for either the distance or the time between the vehicles would have to be added. This would guarantee that the solution found is the optimal one, but it would also greatly increase the complexity and succumb to the *Curse of dimensionality* [4].

Imagine a mission 30km long and the HDVs starting 1km apart, then what should the discretization of the distance between the HDVs be? Let us say its 10m between each discretization point, then that would have to add 100 levels to the state space. This would result in the algorithm having to evaluate 100 times as many states. So even if smart solutions to reduce the size of the state space was used, resulting in only a 30 times larger state space. Then it would still take 30 times longer to calculate than if another state was not added.

Starting at the merge point

To avoid adding another state, the algorithm from Hellström [9] could be used by starting at the merge point to find the optimal solution for the catch-up part, while the platooning part could be simulated. But since the optimal merge point is unknown, the algorithm would have to guess where it is and then find the β that corresponds to the trajectory that satisfies the initial conditions. It would be hard finding the correct β with varying road topography, and then to guarantee optimality different merge points would have to be tested.

Dynamic programming forwards

The proposed solution is to run the DP algorithm in a forwards manner, calculating a cost-to-come for each state until the preceding vehicle is found. Using β to set the mean catch-up velocity and thus decide where the platoon is found. This solves the problem of guessing the merging point and then finding the correct β that satisfies the initial conditions. The algorithm will still have to try different solutions to guarantee optimality, which is discussed in Section 6.3.

The proposed algorithm can be seen below.

Algorithm 1 Catch-up Algorithm

```
1:  J_N(x) = J̃_N(x) for all x ∈ X_n
2:  For k = 1, 2, ..., N
3:     For x ∈ X_k
4:        J_k(x) = min_{u∈U_k}{ζ_k(x, u) + J_{k−1}(x_{k−1})}
5:        if x ∈ X_{k,traj} AND T(x) ∈ [T(X_{k,traj}), X_{k,traj} + ΔT]
6:           N_platoon = k
7:           break;
8:        end if
9:     end For
10: end For
```

The algorithm goes through each stage k in a forwards manner and calculates the cost-to-come $J_k(x)$ for each state. The principle of the algorithm is that the cost-to-come $J_k(x_k)$ is known for $k \leq n$, then the cost-to-come for $J_k(x_k)$ for $k = n + 1$ can be calculated as a function of $J_k(x_k)$, $k \leq n$. The cost for the first stage $J_0(x_0)$ is used to start the HDV in its starting position, by setting the cost of that state to zero while the cost of the other states is high. This forces the algorithm to choose a trajectory from the actual starting state of the HDV.

Introducing the discretized states from Section 4.2, for a given kinetic energy Ek_i and a gear number G_j at position s_k, the cost-to-come is $J_k(x) = J(s_k, Ek_i, G_j)$. The cost-to-come is first calculated under the assumption that there is no gear shift $J_{cg}(s_{k+1}, Ek_i, G_j)$. Then the cost-to-come is calculated for gear shifts $J_{gs}(s_{k+1}, Ek_i, G_j)$. Finally, the cost-to-come is given by the one of the two with the lowest cost

$$J(s_{k+1}, Ek_i, G_j) = \min\{J_{cg}(s_{k+1}, Ek_i, G_j), J_{gs}(s_{k+1}, Ek_i, G_j)\} \tag{4.11}$$

4.4 Cost-to-come, constant gear

Here the cost-to-come in the case of constant gear G is shown.

The step cost is

$$\zeta_{cg}(u_k) = \Delta M + \beta \Delta T \tag{4.12}$$

Where the fuel ΔM is expressed with (3.9)

$$\Delta M = \int_{s_{n-1}}^{s_n} \frac{dm}{ds} ds \approx \frac{n_{cyl}}{2\pi n_r} \frac{i}{r_w} u_f \Delta s \tag{4.13}$$

And the time it takes ΔT is

$$\Delta T = \int_{s_{n-1}}^{s_n} \frac{ds}{v(s)} \approx \frac{\Delta s}{\frac{v(s_n)+v(s_{n-1})}{2}} \tag{4.14}$$

The velocity $v(s_n)$ is given by the current state $Ek(s_n)$ with $v = \sqrt{2E_k/m}$, and $v(s_{n-1})$ is given by the previous state $Ek(s_{n-1})$. The previous state is approximated with Euler backward

$$Ek(s_{n-1}) \approx Ek(s_n) - \frac{dE_k(s_n)}{ds} \Delta s \tag{4.15}$$

Where $\frac{dE_k}{ds}$ is given by Equation (3.6)

$$Ek(s_{n-1}) = Ek(s_n) - [F_p(v, g, u_f) - F_d(v, v, u_b)]\Delta s \tag{4.16}$$

Where $v = \sqrt{2E_k/m}$, and F_p is given by Willans line (3.7) in combination with Equation (3.17). The previous state $Ek(s_{n-1})$ can be in between the discretized states, see Figure 4.3. If $Ek_{i-1} \leq Ek(s_{n-1}) \leq Ek_i$ then the cost J^* is given by linear interpolation between $J(s_{n-1}, Ek_{i-1}, G_j)$ and $J(s_{n-1}, Ek_{i-1}, G_j)$.

The cost-to-come at position s_n and kinetic energy Ek_i is then obtained by calculating the cost-to-come for a discretized set of controls $u_f \in U_f$ and then choosing the one that minimizes the sum of the step cost and the cost-to-come at position s_{n-1}

$$J_{cg}(s_n, E_k, g) = \min_{uf_k \in U_f} \{\zeta_{cg}(uf_k) + J^*(uf_k)\} \tag{4.17}$$

4.5 Cost-to-come, gear shift

Shown here is the cost-to-come in the case of a gear shift from G to $G' \neq G$. The gear shift is done similarly as in Section 3.5.

The step cost for a gear shift is

$$\zeta_{gs}(G) = \Delta M + \beta \Delta T \tag{4.18}$$

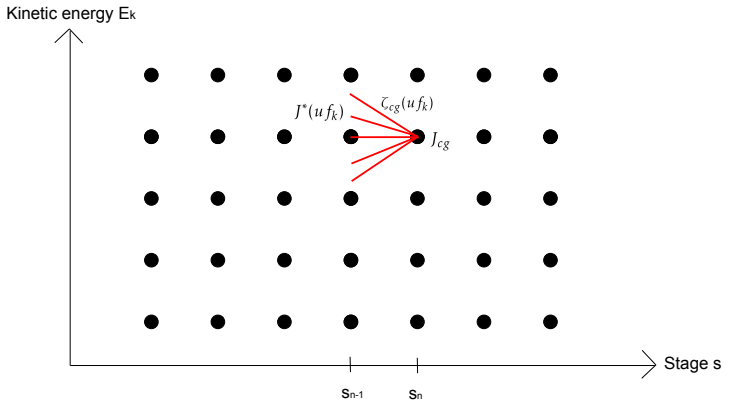

Figure 4.3: *Cost to go for constant gear.*

The time it takes to perform a gearshift is considered constant

$$\Delta T = \tau \tag{4.19}$$

Equation (3.23) gives the fuel it takes ΔM to perform a gear shift

$$\Delta M = \gamma \frac{1}{2} I_e[\omega(v(s_n), G)^2 - \omega(v(s'), G')^2] + m_{fric}(\omega_w, u_f, \tau) \tag{4.20}$$

Where m_{fric} is the fuel required to overcome friction in the engine. The rotational speed ω in the engine is given by the velocity v of the vehicle and the chosen gear G, see Equation (3.13). The velocity at the current position $v(s_n)$ is known with $v = \sqrt{2E_k/m}$ and the kinetic energy state $Ek(s_n)$. For a gear shift, the time ΔT is known but the distance Δs is not, so the approach taken to find the previous state for a constant gear (4.15) cannot be used. Instead, Euler backward on the velocity is used

$$v(s') \approx v(s_n) - \frac{dv(s_n)}{dt}\Delta T \tag{4.21}$$

Now the velocity $v(s')$ is known but the position s' is not. $\frac{dv(s_n)}{dt}$ is given by Equation 3.1 with $F_p = 0$ since the engine is disengaged during a gear shift. Now the distance it takes to perform a gear shift can be found

$$\Delta s = \int_{v(s')}^{v(s_n)} v(t)dt \approx \Delta T \frac{v(s') + v(s_n)}{2} \tag{4.22}$$

Then s' can be found

$$s' = s_n - \Delta s \tag{4.23}$$

However s' can fall in between different stages $s_{k-1} < s' < s_k$. Then the approach taken is to calculate the cost-to-come under the assumption of no gear shift from s' to s_{k-1}, see Figure 4.4. The step cost with constant gear $\zeta_{cg}(u_k)$ for the distance $s' - s_{k-1}$ is calculated the same way as in Section 4.4.

Now the cost-to-come for a gear shift J_{gs} is found by determining the fuel and gear that minimizes the sum of the two step costs and cost-to-come $J^*(u_k, G')$

$$J_{gs}(s_n, Ek_i, G_j) = \min\{\zeta_{gs}(G') + \zeta_{cg}(u_k, G') + J^*(u_k, G')\} \tag{4.24}$$

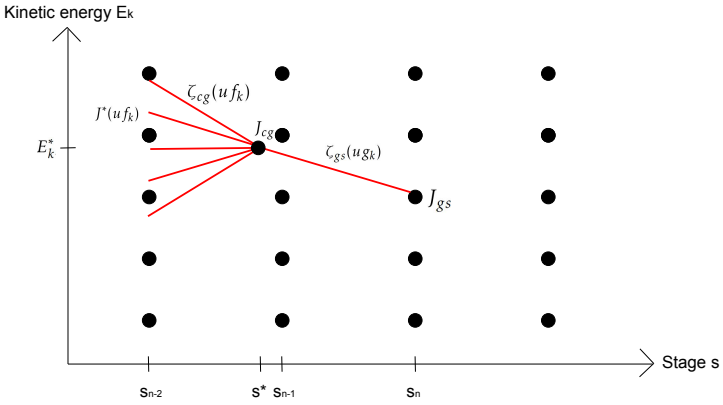

Figure 4.4: *Cost to go for a gear shift.*

4.6 Finding the platoon in time

For the HDV to have caught up to the preceding vehicle, they have to be at the same place at the same time. The algorithm has a state for position s but the time it takes to get there is unknown. This is solved by keeping track of the time-to-come for each state. First, the cost-to-come is calculated according to Equation (4.11), then the time-to-come is calculated for the chosen control. If the chosen control is of constant gear then the time-to-come is calculated with

$$T(s_{k+1}, Ek_i, G_j) = \zeta_T(u_k) + T^*(u_k) \tag{4.25}$$

Where $\zeta_T(u_k)$ is the step time given by (4.14), and $T^*(u_k)$ is the time-to-come for the previous state, for the chosen control u_k. Whereas before the previous

state $Ek(s_{n-1})$ can be in between the discretized states. Then the time-to-come for that state $T^*(u_k)$ is linearly interpolated between the two states.

If the chosen control is a gear shift, then the time-to-come is calculated with

$$T(s_{k+1}, Ek_i, G_j) = \zeta_{T,gs}(G') + \zeta_{T,cg}(u_k, G') + T^*(u_k, G') \tag{4.26}$$

Where $\zeta_{T,gs}(G')$ is the step time for the gear shift (4.19), $\zeta_{T,cg}(u_k, G')$ is the step time to go to the previous stage from where the gear shift ended (4.14). Then $T^*(u_k, G')$ is the time-to-come for the previous state for the control and the gear G', which is linearly interpolated as before.

It is desired that the HDV has the same velocity as the preceding vehicle at the merge point. Since the HDV needs to have a higher velocity than the preceding vehicle to catch up, then it can't have a lower velocity at the merge point since then there would be an earlier point where it was even closer to the preceding vehicle. If the HDV has a higher velocity when it has caught up then it has spent more energy than if it would have the same velocity. Then the states where it is needed to check if the HDV has caught up or not can be found on the trajectory of the preceding vehicle. To gain accuracy the preceding vehicle's trajectory is discretized so that the algorithm can check if it has caught up at the same point for the same velocity.

4.7 Calculating trajectory

The trajectory for the mission is separated into two stages, one for the catch-up part and one for the platooning part. The platooning part is given by the preceding vehicle's trajectory, see Section 4.8. How to find the trajectory of the catch-up part is shown below.

After running the catch-up algorithm the cost-to-come for each state is known, but the trajectory to go from the starting position to the merge point, where the HDV caught up to the preceding vehicle is not known. To find the trajectory a modified version of the catch-up algorithm is used. It starts at the merge point that was found and then it calculated the cost-to-come as in (4.11) and chooses the control with the lowest cost. Then the previous state from that control is found and the cost-to-come is calculated again. This is iterated in a backward fashion until the trajectory from the end to the start is found. The algorithm can be seen below, with $x_{k-1} = F(x, u)$.

Algorithm 2 Calculate trajectory

1: **For** $k = N_{platoon}, N_{platoon} - 1, ..., 1$
2: $\quad J_k(x) = \min_{u \in U_k} \{\zeta_k(x, u) + J_{k-1}(F_k(x, u))\}$
3: **end For**

4.8 Platooning algorithm

When platooning the HDV follows behind with a constant time gap, thus the HDV will have the same velocity profile as the preceding vehicle. Since the velocity profile is known the platooning part can be simulated.

With the velocity profile known the kinetic energy Ek can be calculated for each position s, then to find the needed control the relation

$$\frac{dEk}{ds} \approx \frac{Ek(s_{n+1}) - Ek(s_n)}{\Delta s} \tag{4.27}$$

is used. Now the needed propulsive force can be calculated with Equation (3.6)

$$F_p(\sqrt{2E_k/m}, g, u_f) = \frac{dE_k}{ds} + F_d(\sqrt{2E_k/m}, s, u_b) \tag{4.28}$$

Where $F_d = F_a + F_r + F_g + F_b$ and F_a is given by the platooning model in Section 3.6. The distance d between the HDV's is found with the current velocity v and the time gap t_{gap} between them

$$d = vt_{gap} \tag{4.29}$$

Then u_f is found with the help of Willans line (3.7) and the relation (3.17) for each gear G_j. Then the fuel required can be calculated for each gear

$$\Delta M = \int_{s_{n-1}}^{s_n} \frac{dm}{ds} ds \approx \frac{n_{cyl}}{2\pi n_r} \frac{i_{tot}(G_j)}{r_w} u_f(G_j) \Delta s \tag{4.30}$$

The control and gear with the lowest fuel consumption are then chosen, thus enabling instant gear switching. This can be done here but not in the catch-up algorithm since the platooning algorithm simulates the cost of the preceding vehicle's trajectory. While the catch-up algorithm optimizes the trajectory to find the lowest cost, making it able to exploit weaknesses in the model.

4.9 Normal mission

The cost of the catch-up part can now be found but the cost for continuing as planned is still unknown. The suggested method of finding the cost is to use the catch-up algorithm but running the algorithm for the whole mission instead of stopping if it has caught up, and then finding the trajectory. Then β needs to be chosen to have the wanted mean velocity.

5

Validating the catch-up algorithm

This chapter will validate the catch-up algorithm and the platooning algorithm.

5.1 Validating β

Validating the β calculation for the wanted mean velocity. β is calculated with Equation (4.7). To test if the calculated β gave the correct velocity, the algorithm was used without platooning on a flat road 2000m long, where the HDV started and ended in with the wanted velocity. In Figure 5.1 the wanted mean velocity for β and the measured one can be found. It can be seen that they correspond well.

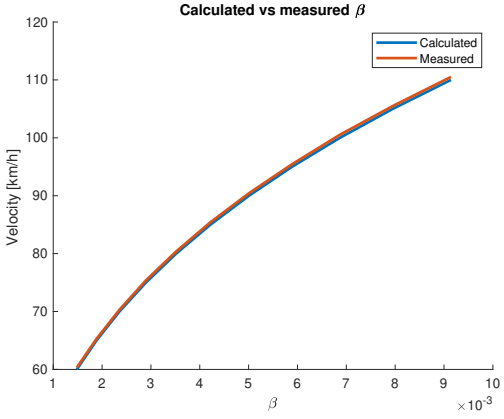

Figure 5.1: *Figure of the wanted and actual mean velocity given β.*

5.2 Validating time-to-come

Validating the time-to-come from Section 4.6 was done by comparing the interpolated time-to-come to the actual time it takes to run the optimal trajectory. The algorithm was used on missions of different lengths on a road with varying topography, and with several β corresponding to mean velocities between 80-100 km/h. Then the time-to-come was compared to the time it takes to follow the optimal trajectory, see Figure 5.2. The interpolation errors are quite small and there is no clear trend line. The errors are deemed sufficiently small.

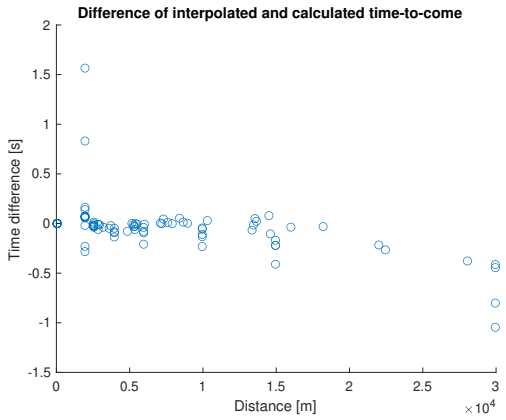

Figure 5.2: *Figure of difference between the interpolated time-to-come and the time-to-come from the calculated trajectory.*

5.3 Validating the Catch up algorithm

To validate the catch-up algorithm, two test missions with known optimal solutions were used. The algorithm was run without platooning for both tests. The tests are done such that they start and end with the same velocity, and β was tuned to achieve the desired mean velocity. The parameters used for both tests can be seen in Table 5.1.

Validation test 1

The first test is to maintain a constant velocity during a mission, which was shown by J. Chang and K. Morlok [5] to be the optimal solution. This holds under the condition that the HDV is capable to maintain a constant velocity during the entire mission. The test was done on a 10km long flat road, see Figure 5.3. The HDV maintains a constant velocity during the mission, which is the optimal solution.

Table 5.1: *Parameters used when validating the catch-up algorithm.*

Description	Parameter	value	Unit
N. of stage disc. points	s	200	[m]
N. of kin. energy disc. points	Ek	100	[J]
Allowed gears	g	12-14	[-]
N. of fueling disc. points	u_f	280	[mg/cycle]
Max velocity	v_{max}	100	[km/h]
Min velocity	v_{min}	60	[km/h]
Desired velocity	v_{des}	80	[km/h]
Time penalty parameter	β	0.00362	[-]
Time penalty parameter	β	0.00362	[-]

Validation test 2

The second test is the same one as the one used by R. Ohlsén and E. Sten [15]. The test is constructed with two uphills and two downhills. Where the first pair was constructed such that the HDV wouldn't be able to maintain a constant velocity while being able to do so for the second pair. The results can be found in Figure 5.4. It can be seen that the HDV accelerates before the first uphill and then loses some velocity while going up. Around 2km there is a dip in the fueling, this is because of the engine reaching the max torque, see figure 5.5. Then for the downhill, the behavior is similar but the inverse. As shown by Fröberg et al. [3], this behavior is optimal when maintaining a desired mean velocity. In the second pair of hills, the HDV is able to maintain the desired mean velocity which is the optimal solution as discussed for earlier.

5.4 Validating the platooning algorithm

To test that the platooning algorithm behaves as expected, two tests were used. Firstly the preceding vehicle's trajectory was created by running test 2 in the previous section. Then with the preceding vehicle's trajectory being known, the platooning algorithm was used for the whole mission. The HDV parameters were the same for both the preceding and following vehicle. Then two time gaps t_{gap} for the platooning algorithm were chosen, one to test the algorithm with reduced air drag $t_{gap} = 2$s and one to test the algorithm without reduced air drag $t_{gap} = 100$s.

When platooning with a constant time gap, the two vehicles will have the same trajectory. With a time gap large enough for there to be no air drag reduction $t_{gap} = 100$s and the same HDV parameters for both vehicles, both vehicles will experience the same forces for every point on the trajectory. Thus the following vehicle should behave the same as the preceding one, see Figure 5.6. It can be seen that the vehicles behave the same, and the fuel consumption is the same for both, see Table 5.2.

Then the algorithm was tested with a time gap small enough for the air drag to be reduced $t_{gap} = 2$s. This resulted in a lower fuel consumption as expected, see Table 5.2.

Table 5.2: *Fuel consumption when platooning for different time gaps.*

Time gap	**Fuel consumption**
Preceding HDV	3.1507 L
100s time gap	3.1507 L
2s time gap	2.9614 L

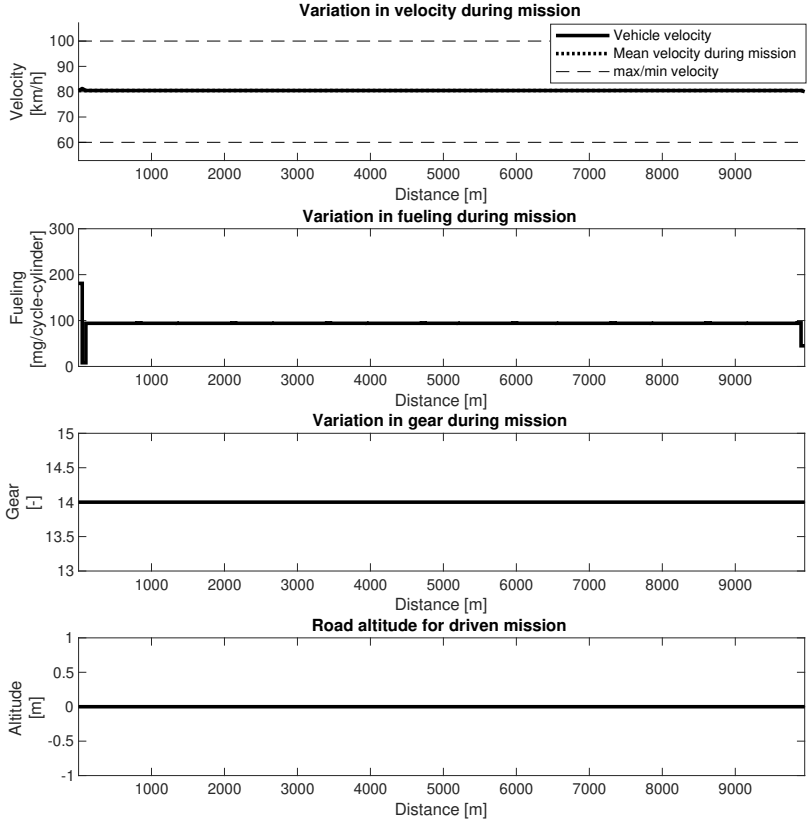

Figure 5.3: *Validation test 1.*

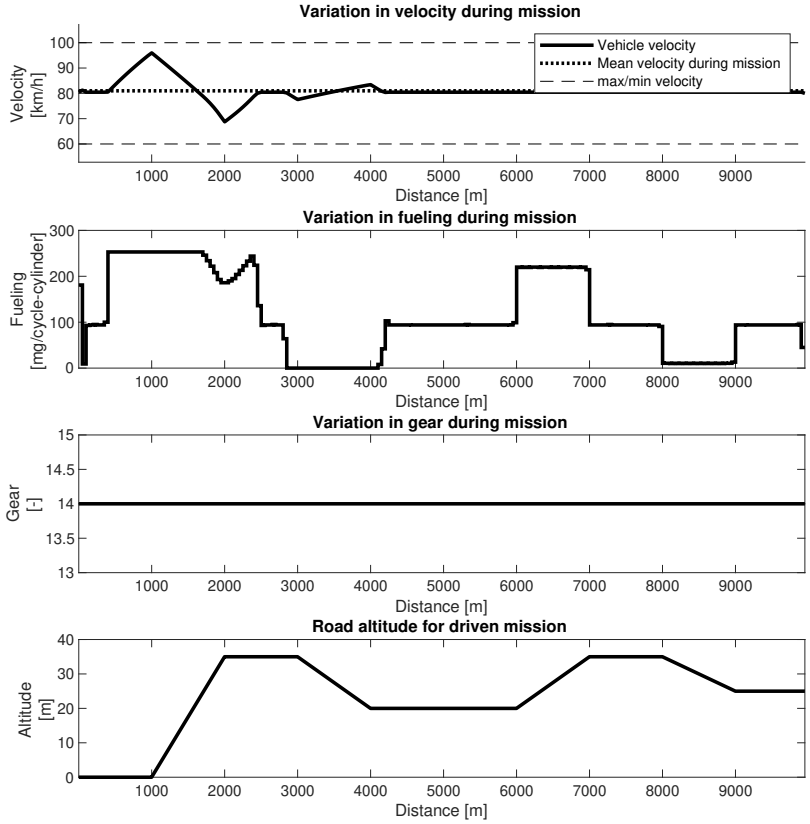

Figure 5.4: *Validation test 2.*

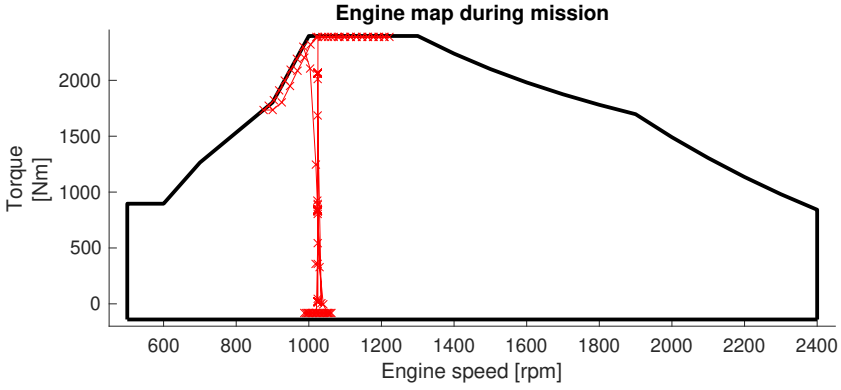

Figure 5.5: *Engine map for test 2.*

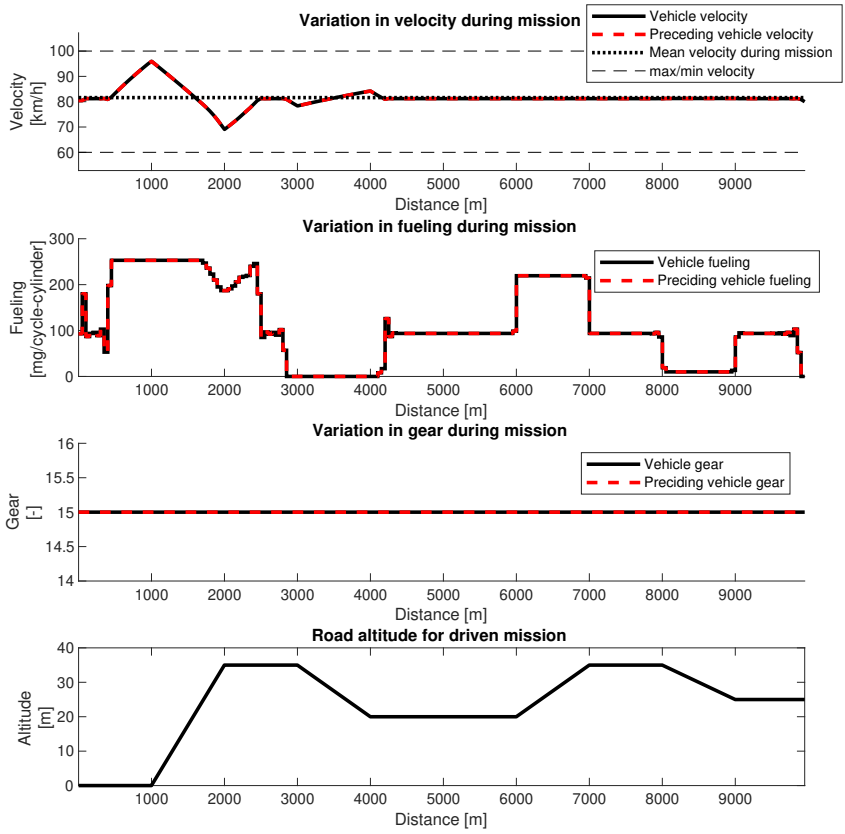

Figure 5.6: *Platooning test with no air drag reduction.*

6

Finding the optimal solution

The solution from Chapter 4 finds the optimal solution for the given β, i.e. a Pareto optimal solution. Then to find the fuel-optimal solution, the fuel-optimal β needs to be found. This chapter looks at how to find the fuel-optimal β, first by estimating an optimal β candidate and then running the algorithm for several β to find the fuel-optimal solution. The chapter will also look at how to update the algorithm to calculate the solution for several β at once.

The chapter will consider the catch-up problem, a HDV catches up to a preceding vehicle and platoons with it to the goal, see Figure 6.1. In the figure the initial distance to the end is d_d, the initial distance to the preceding vehicle is d_p, the velocity of the following vehicle is v_a and the velocity of the preceding vehicle is v_p.

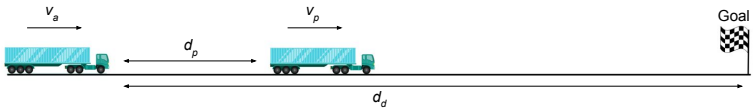

Figure 6.1: *The starting position for the catch-up problem.*

6.1 Optimal β candidate

In Liang [12] the authors looked at the problem of when is it fuel-efficient for a HDV to catch up with a platoon. They calculated the average normalized air drag during a mission given the air drag reduction when platooning ϕ and the velocity increase $r_v = v_a/v_p$ compared to the preceding vehicles velocity v_p. The calculation was done under the assumption that the road is flat and that the acceleration forces can be neglected when the mission is sufficiently long, i.e. the velocities

are constant. The authors introduced a platooning incentive factor, which is the amount of energy saved due to the reduction in air drag for the whole mission. The platooning incentive factor can be seen below

$$\kappa = 1 - \frac{d_p}{d_d}\frac{r_v}{r_v - 1}(r_v^2 - \phi) - \phi \tag{6.1}$$

Which contain

$$\frac{r_v}{r_v - 1}(r_v^2 - \phi) \tag{6.2}$$

This equation has one optimum for $r_v > 1$ given the air drag reduction ϕ, which can be seen in Figure 6.2. The air drag reduction when platooning ϕ is dependent on the distance between the vehicles s_{gap}, see Equation (3.24). Since platooning is done with a constant time gap t_{gap}, the distance between the vehicles s_{gap} depends on the velocity of the preceding vehicle v_p. The air drag reduction ϕ could be calculated at each stage of the mission since the preceding vehicle's trajectory is known, but it is deemed sufficient to use the mean velocity of the preceding vehicle $\hat{v}_p$

$$s_{gap} = \hat{v}_p t_{gap} \tag{6.3}$$

With this, the wanted catch-up velocity can be found and the optimal β candidate can be formulated with Equation (4.7).

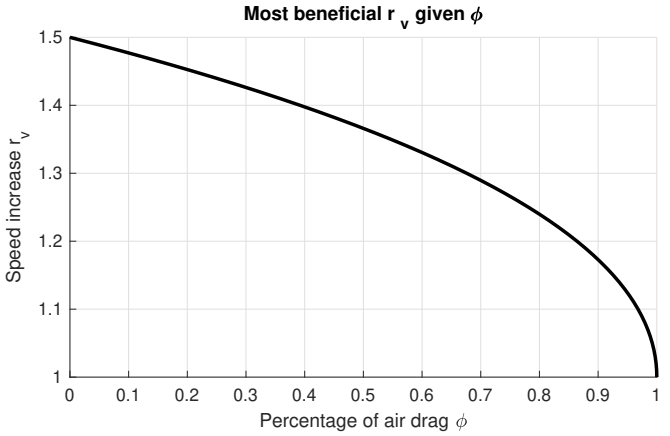

Figure 6.2: *Figure of the most beneficial r_v^* to catch up, given ϕ.*

Table 6.1: *Mission parameters when validating the optimal β candidate.*

Description	Parameter	value	Unit
Mission length	d_d	30	[km]
N. of stage disc. points	s	600	[m]
N. of kin. energy disc. points	Ek	120	[J]
Allowed gears	g	13-14	[-]
N. of fueling disc. points	u_f	280	[mg/cycle]
Max velocity	v_{max}	133	[km/h]
Max velocity	v_{min}	61	[km/h]
Preceding vehicle mean vel.	$\hat{v}_p$	80	[km/h]
Preceding vehicle start. dist.	d_p	1	[km]
Min. wanted catch-up vel.	$\hat{v}_{a,min}$	80	[km/h]
Max. wanted catch-up vel.	$\hat{v}_{a,min}$	110	[km/h]
Catch-up velocity step	h_v	1	[km/h]
N. of catch-up vel. used	N_v	30	[-]

6.1.1 Validating the optimal β candidate

There is no guarantee that the optimal β candidate found in Section 6.1 gives the fuel-optimal solution, since the algorithm takes into account the road topography and acceleration forces. To test the accuracy of the optimal β candidate, two Pareto fronts were created for two different missions. One mission on a flat road and one for a topographic road, the parameters used can be seen in Table 6.1. In Figure 6.3, the different road profiles for the two missions can be seen. The merge points, where the HDV has caught up to the preceding vehicle is marked in the figure.

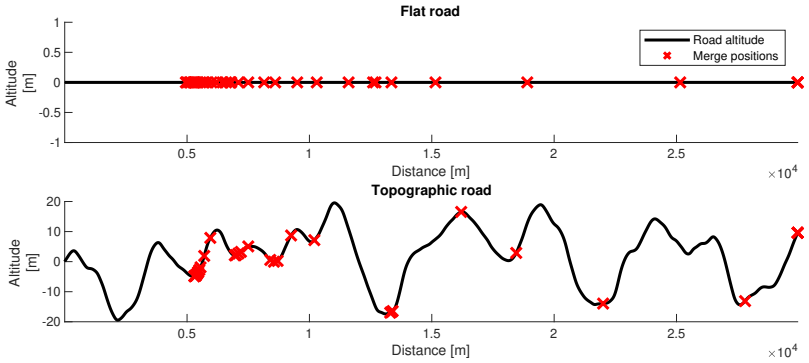

Figure 6.3: *Figure of the road profiles and the merge points.*

The Pareto optimal solution is calculated for each β corresponding to the wanted catch-up velocities. In Figures 6.4, 6.5, two Pareto fronts for the fuel consumption given β is shown for the two missions. The fuel consumption for

the nominal speed can be seen in the figures, which is for $\beta = 3.5 \cdot 10^{-3}$. Then as β increases the fuel consumption increases until $\beta = 4 \cdot 10^{-3}$, which is the minimum β to be able to catch up to the preceding vehicle. For $\beta > 4 \cdot 10^{-3}$ the HDV catches up and gets the benefits of platooning, thus the fuel consumption is reduced. Then the optimal β candidate and the found optimum can be seen in the figures. It can be seen that the road topography has a large effect on the fuel consumption. Whereas for the flat road the optimal β candidate is a very good option, while for the topographic road the optimal β candidate isn't as good of an option.

In Figures 6.6, 6.7 two Pareto fronts for the mean catch-up velocity is shown. Then in Figures 6.8, 6.9 the mean catch-up velocity compared to the wanted mean catch-up velocity, given β for the two missions is shown. It can be seen that the chosen β doesn't result in the wanted catch-up velocity. This is because the HDV doesn't start and end in the wanted velocity, and thus the mean velocity will be lower. And it's partly because of the influence of the road topography.

With this, it can be seen that the optimal β candidate is a good candidate but it doesn't guarantee the fuel-optimal solution.

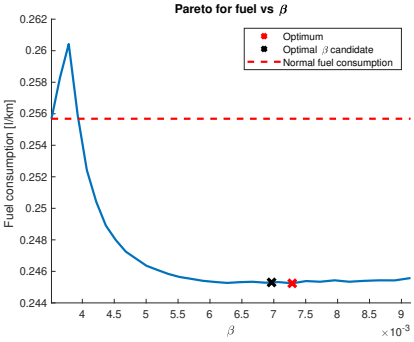

Figure 6.4: *Pareto of fuel consumption to β for a flat road.*

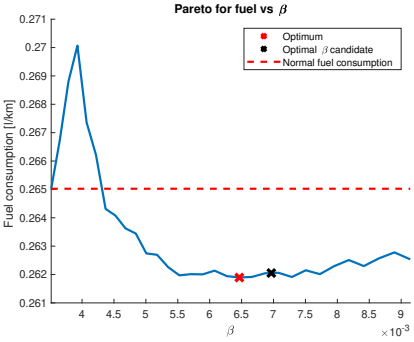

Figure 6.5: *Pareto of fuel consumption to β for a topographic road.*

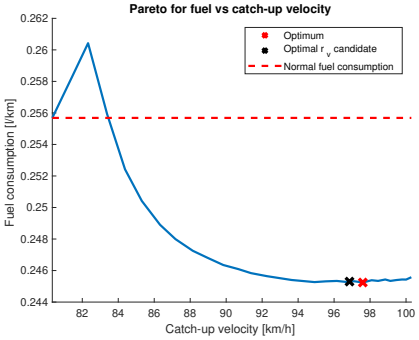

Figure 6.6: *Pareto of fuel consumption to catch-up velocity for a flat road.*

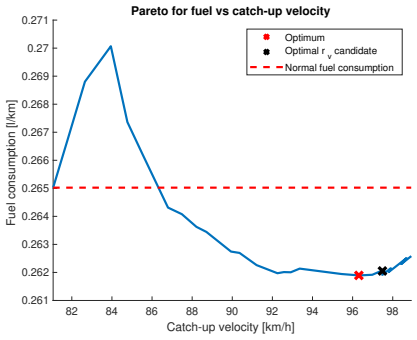

Figure 6.7: *Pareto of fuel consumption to catch-up velocity for a topographic road.*

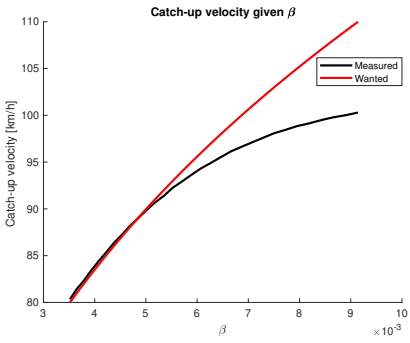

Figure 6.8: *Figure of the measured compared to the wanted catch-up velocity on a flat road, given β.*

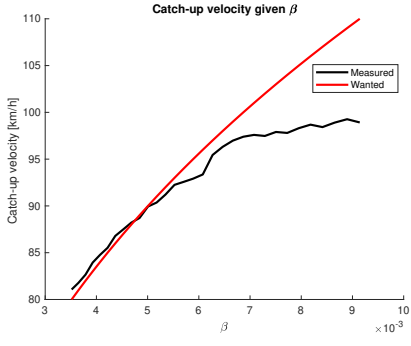

Figure 6.9: *Figure of the measured compared to the wanted catch-up velocity on a topographic road, given β.*

6.2 Parallel search

To find the fuel-optimal β, the solution will be calculated for several β. This could be done by running the algorithm several times for each β, which would take a lot of time. Instead, the algorithm can be run for several β in parallel, calculating the cost-to-come separately for each β. This can be done because some parts of the algorithm are independent of β and thus only needs to be calculated once.

The algorithm is iterated in the same way as before only that it calculates the cost-to-come for several β instead of one. The cost-to-come given by (4.11) is modified to calculate the cost to come for each β indexed b.

$$J_b(s_{k+1}, Ek_i, G_j) = \min\{J_{b,cg}(s_{k+1}, Ek_i, G_j), J_{b,gs}(s_{k+1}, Ek_i, G_j)\} \tag{6.4}$$

6.2.1 Parallel search for constant gear

The cost-to-come for a constant gear $J_{b,cg}(s_{k+1}, Ek_i, G_j)$ is done as before in Section 4.4 but for each β. Where the cost-to-come has to be calculated for each β and is given by (4.17)

$$J_{b,cg}(s_n, E_k, g) = \min_{uf_k \in U_f} \{\zeta_{b,cg}(uf_k) + J_b^*(uf_k)\} \tag{6.5}$$

Where J_b^* is interpolated as before in Section 4.4 for each given β. The step cost $\zeta_{b,cg}(uf_k)$ given by (4.12) has to be calculated for each β

$$\zeta_{b,cg}(u_k) = \Delta M + \beta_b \Delta T \tag{6.6}$$

Where two terms ΔM and ΔT are independent of the choice of β. This means that they only need to be calculated once and can be used for each choice of β to find the cost-to-come. ΔM and ΔT are given by Equations (4.13) and (4.14).

6.2.2 Parallel search for gear shift

The cost-to-come $J_{b,gs}(s_{k+1}, Ek_i, G_j)$ for a gear shift is done as in Section 4.5 but for each β. The cost-to-come has to be calculated for each β and is given by (4.24)

$$J_{b,gs}(s_n, Ek_i, G_j) = \min\{\zeta_{b,gs}(G') + \zeta_{b,cg}(u_k, G') + J_b^*(u_k, G')\} \tag{6.7}$$

Where J_b^* is interpolated as before in Section 4.5 for each β. The step cost $\zeta_{b,cg}(u_k, G')$ is given by (6.5) and the step cost $\zeta_{b,gs}(G')$ comes from (4.18)

$$\zeta_{b,gs}(G) = \Delta M + \beta_b \Delta T \tag{6.8}$$

Where the two terms ΔM and ΔT are independent of the choice of β. This means that they only need to be calculated once and can be used for every β to find the cost-to-come. ΔM and ΔT are given by Equations (4.20) and (4.19).

6.2.3 Parallel search, finding the platoon in time

To know when the HDV has caught up to the preceding vehicle, the time-to-come is computed for each β the same way as in Section 4.6. Then when the platoon is found for one β, the cost-to-come for that β doesn't have to be calculated anymore, thus saving computational time when calculating the cost-to-come for the other β. Then the algorithm is done either when the platoon is found for every β or the end is reached.

6.2.4 Parallel search, trajectory

The trajectory is calculated as before in Section 4.7, once for each β.

6.2.5 Parallel search, platooning

The platooning is done as before in Section 4.8, where it only needs to be calculated once from the earliest merge point and later merging points use the same trajectory starting from where they merged with it. This is because the merge point depends on β, since β affects the catch-up velocity. But the platooning trajectory is independent of β since platooning is done with a constant time gap and thus it will have the same trajectory as the preceding vehicle independent of β.

6.2.6 Parallel search, results

The results of the parallel search can be seen when comparing running the normal algorithm over and over for each β and computing every β at the same time, see Figure 6.10. It can be seen that computing the cost-to-come for every β at the same time reduces the computing time when using more than one β.

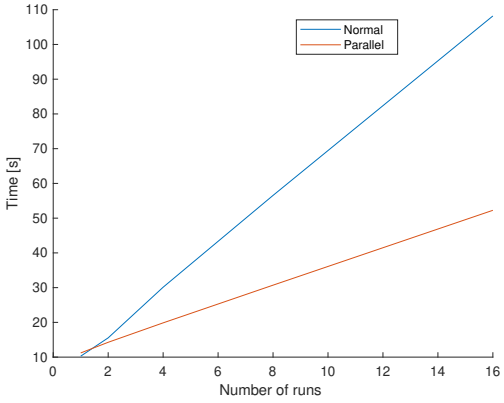

Figure 6.10: *Time it takes to run the algorithm once per β compared to running it for all β at once.*

6.3 Strategy for finding the optimal solution

When running the algorithm once per β, an interval halving approach for finding the fuel-optimal solution would be a good solution. But now that it is computationally beneficial to run the algorithm for several β at a time, another approach is used. The approach is to calculate the solution for some β in the surrounding to the optimal β candidate from Section 6.1, then fit a second degree polynomial to the calculated points and calculate the solution for β around the minimum of the fitted curve.

6.3.1 Search interval for β

This section will look at the search interval for β.

Consider the catch-up problem, a HDV catches up to a preceding vehicle and platoons with it to the goal. Then to be able to platoon, the mean catch-up velocity v_a needs to be high enough for the vehicle to catch up to the preceding vehicle before it gets to the finish line, giving the lower bound for the mean catch-up velocity

$$\underline{v_a} = \frac{d_d}{t_p(s_N)} \tag{6.9}$$

Where d_d is the initial distance to the goal at the start of the mission, see Figure 6.1, and $t_p(s_N)$ is the time the preceding vehicle is at the end stage s_N, i.e. the goal. The upper bound for the mean velocity $\overline{v_a}$ is given by either the speed limit or the vehicle dynamics. With this, the upper and lower bound for β can be found

$$\underline{\beta} = \beta(\underline{v_a}) \tag{6.10}$$

$$\overline{\beta} = \beta(\overline{v_a}) \tag{6.11}$$

Where $\beta(v)$ is given by Equation (4.7). Then the allowed interval for β is $\beta \in [\underline{\beta},\ \overline{\beta}]$.

After finding the allowed values for β the search interval to search for the optimal solution is defined. The search interval is in a surrounding to the optimal β candidate and is defined as

$$\beta \in [\underline{\beta_s},\ \overline{\beta_s}] \tag{6.12}$$

Where the lower bound for the search interval $\underline{\beta_s}$ is in between the optimal β candidate β_c and the lower bound $\underline{\beta}$

$$\underline{\beta_s} = \beta_c - \frac{\beta_c - \underline{\beta}}{2} \tag{6.13}$$

and the upper bound for the search interval $\overline{\beta_s}$ is in between the optimal β candidate β_c and the upper bound $\overline{\beta}$

$$\overline{\beta_s} = \beta_c + \frac{\overline{\beta} - \beta_c}{2} \tag{6.14}$$

In Figure 6.12 a Pareto front of the fuel consumption given β with the upper and lower bounds for both β and the search interval as well as the optimal β candidate can be seen. The mission was on a flat road $d_d = 30$km long and the preceding vehicle started $d_p = 1$km ahead. It can be seen that the search interval is on the part where the curve has flattened out.

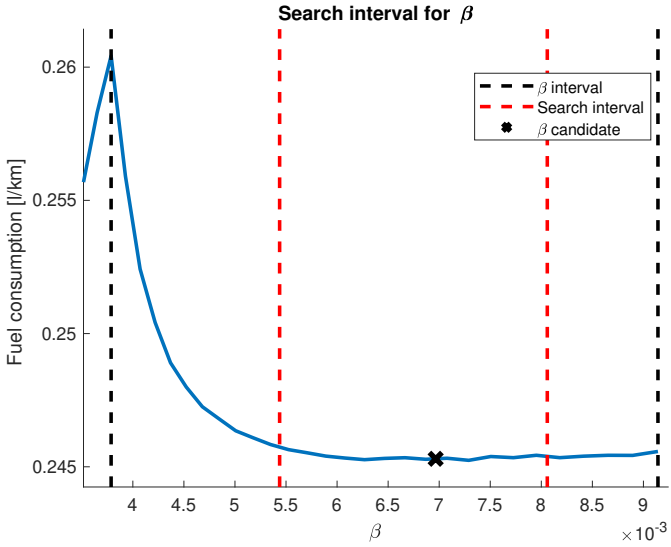

Figure 6.11: *Allowed β interval and the search interval for finding the optimal solution.*

6.3.2 Looking for the solution

The method used to find the fuel-optimal solution is outlined below.

Step 1

First step is to find the optimal β candidate from Section 6.1 and then define the search interval (6.12), see Figure 6.12.

Step 2

The second step is to run the algorithm for some β evenly spaced within the interval. Here 5 points will be used, see Figure 6.13.

Step 3

The third step is to fit a second degree polynomial to the points with the least squares method, see Figure 6.14.

Step 4

The fourth step is to search for the solution around the minimum of the fitted curve, see Figure 6.15.

Step 5

The fifth step is to choose the minimum point of all search points, see Figure 6.16.

6.3.3 Testing the solution on topographic road

There is not a large difference in the solutions when looking at a flat road, but for a topographic road the fuel consumption varies more with β. In Figures 6.17 to 6.21 the same steps are taken for a topographic mission. The mission is the same one as in Section 6.1.1, and the mission parameters can be seen in Table 6.1. It can be seen that the solution found a minimum point. This point is deemed good enough for the solution.

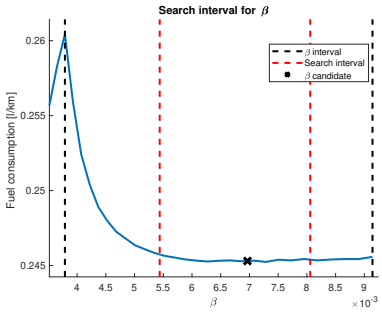

Figure 6.12: *Search interval.*

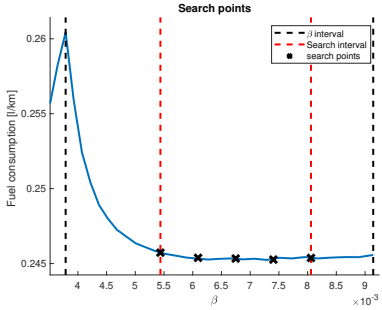

Figure 6.13: *Search points in the interval.*

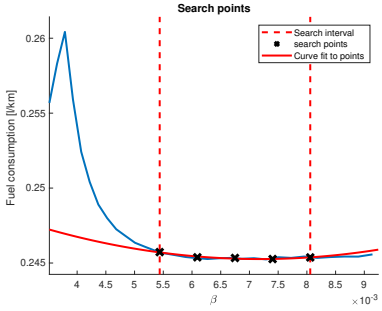

Figure 6.14: *Curve fitted to search points.*

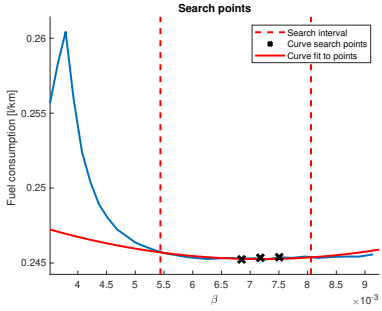

Figure 6.15: *Searching around minimum of the fitted curve.*

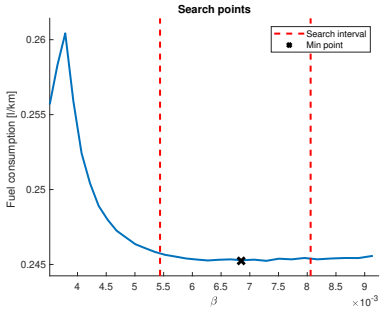

Figure 6.16: *Minimum of all points searched.*

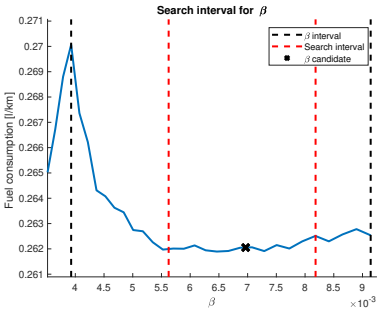

Figure 6.17: *Search interval.*

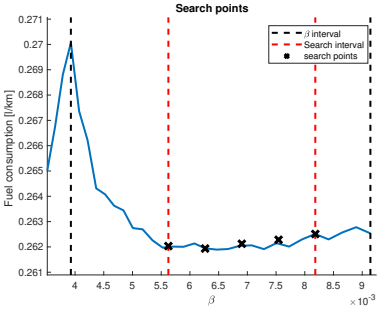

Figure 6.18: *Search points in the interval.*

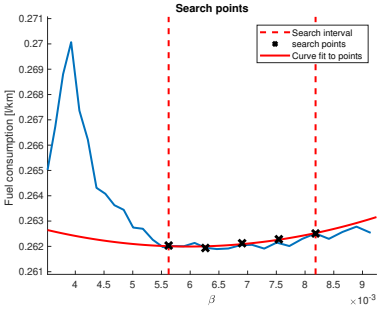

Figure 6.19: *Curve fitted to search points.*

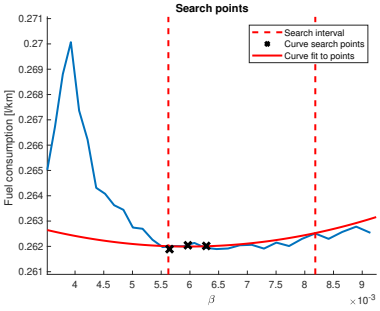

Figure 6.20: *Searching around minimum of the fitted curve.*

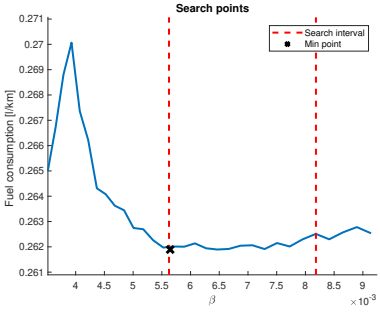

Figure 6.21: *Minimum of all points searched.*

7

Numerical results

Numerical results from different missions will be presented in this chapter.

7.1 Södertälje to Norrköping mission

A mission with topography data from the highway E4 between Södertälje and Norrköping. The mission is 100km long and the mission parameters can be seen in Table 7.1. This was done to show the fuel saving potential from taking the road topography into account.

The mission was done with two identical vehicles, where the trajectory of the preceding vehicle is is calculated with the algorithm without platooning for a mean velocity of 80km/h. The preceding vehicle was started 2700m ahead of the following one so that the optimal β candidate would catch up at a downhill segment, see Figure 7.1. This was done to show the fuel-saving potential in deviating from the optimal β candidate to benefit more from the road topography.

In Figure 7.2 the mission of the found optimum is shown. When comparing the two solutions it can be seen that the optimal β candidate has a higher mean catch-up velocity, which results in it catching up to the preceding vehicle in a downhill segment. While the optimal solution has a lower mean catch-up velocity, thus catching up to the preceding vehicle later and being able to use the downhill segment to catch-up. The results of the mission can be seen in Table 7.2. From the table, it can be seen that there is a fuel-saving potential of around 5.14% to be had when catching up and platooning, where around 0.17% comes from utilizing the strategy shown in Section 6.3.

Table 7.1: *Parameters for the Södertälje to Norrköping mission.*

Description	Parameter	value	Unit
N. of stage disc. points	s	2000	[m]
Stage step length	h_s	50	[m]
Mission length	s_N	100	[km]
Starting dist. to prec. veh.	d_p	2700	[m]
N. of kin. energy disc. points	Ek	120	[J]
Allowed gears	g	12-14	[-]
N. of fueling disc. points	u_f	280	[mg/cycle]
Max velocity	v_{max}	115	[km/h]
Min velocity	v_{min}	50	[km/h]
HDV mass	m_v	40	[Ton]
Platooning time gap	t_{gap}	0.5	[s]

Table 7.2: *Results for the Södertälje to Norrköping mission.*

Description	Results	Unit
Nominal fuel consumption	26.30783	[L]
Optimal fuel consumption	24.95398	[L]
β candidate fuel consumption	24.99957	[L]
Fuel consumption difference	0.04559	[L]
Optimal fuel savings	5.14620	[%]
β candidate fuel savings	4.97290	[%]
Fuel savings difference	0.17329	[%]

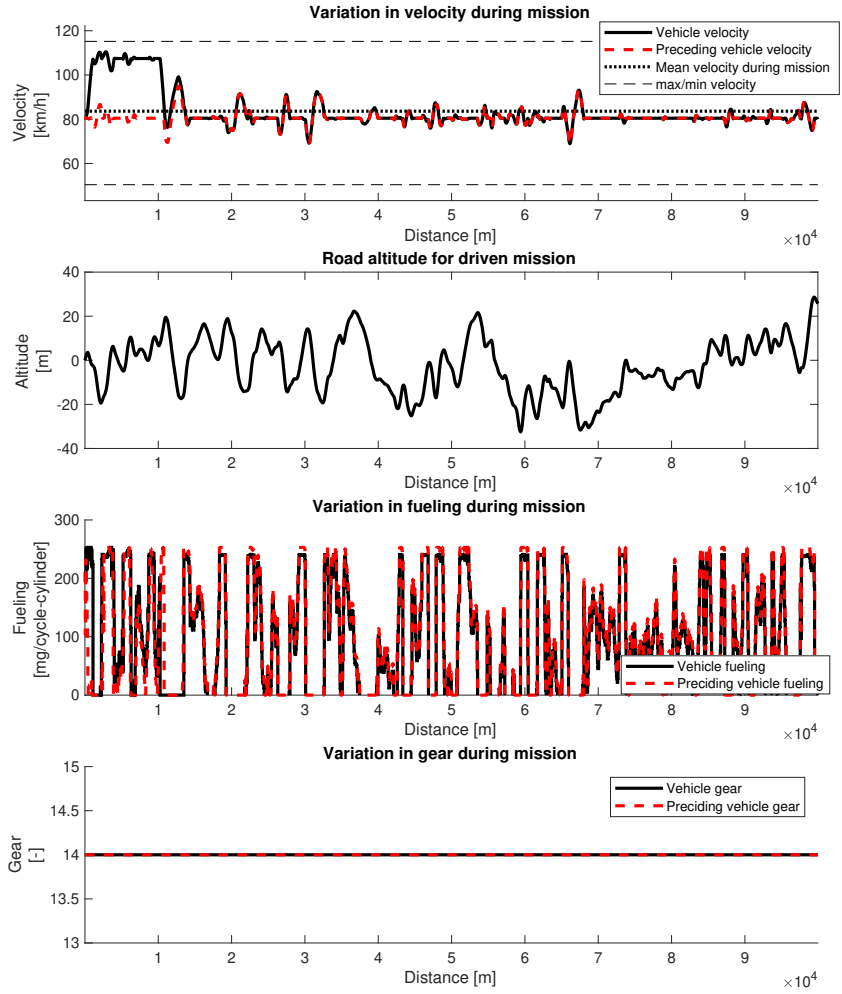

Figure 7.1: *Validation test 1.*

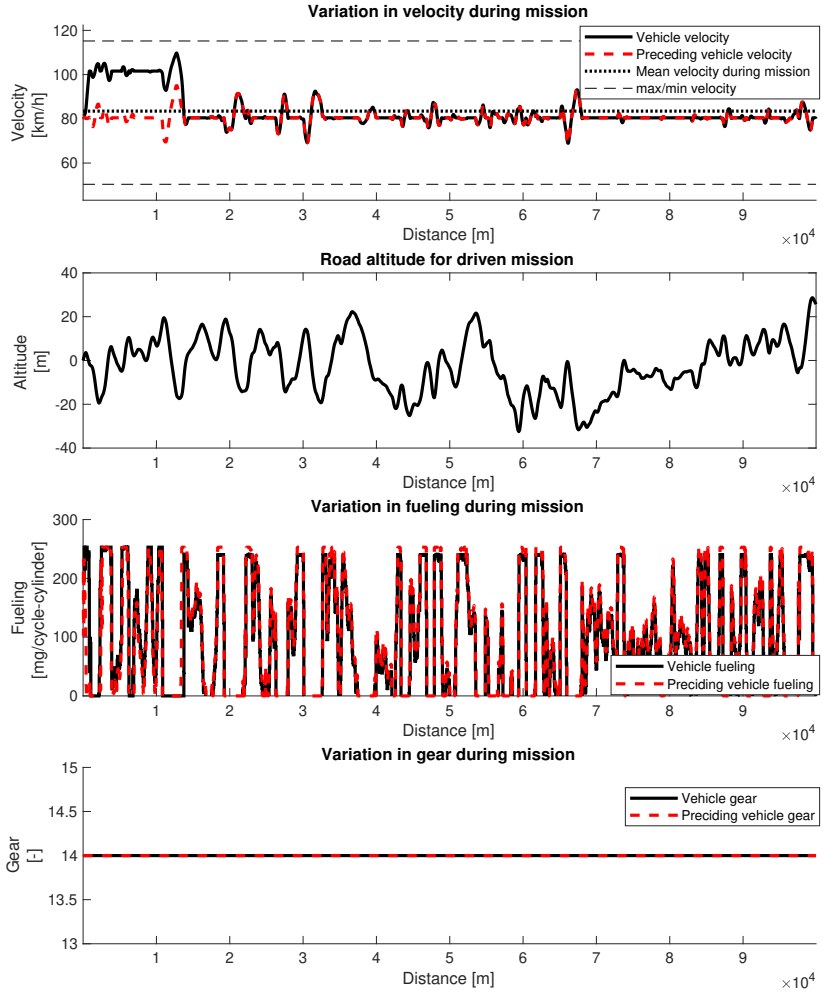

Figure 7.2: *Validation test 2.*

7.2 Missions with varying HDV mass

The algorithm was tested for two missions with varying HDV mass. One mission on a flat road and one for a topographic one. The mission parameters can be seen in Table 7.3. The results of the missions can be seen in table 7.4, 7.5. It can be seen that the fuel-saving potential is higher for lighter HDVs. It can also be seen that often there is no significant difference between the optimal β candidate and the optimal solution.

Table 7.3: *Mission parameters.*

Description	Parameter	value	Unit
N. of stage disc. points	s	600	[m]
Stage step length	h_s	50	[m]
Mission length	s_N	30	[km]
Starting dist. to prec. veh.	d_p	1	[km]
N. of kin. energy disc. points	Ek	120	[J]
Allowed gears	g	12-14	[-]
N. of fueling disc. points	u_f	280	[mg/cycle]
Max velocity	v_{max}	115	[km/h]
Min velocity	v_{min}	50	[km/h]
Platooning time gap	t_{gap}	2	[s]

Table 7.4: *Flat road 30 km*

Description	20 ton	40 ton	60 ton	Unit
Nominal fuel cons.	5.59029	7.65773	9.72530	[L]
Optimal fuel cons.	5.27174	7.34504	9.41943	[L]
β candidate fuel cons.	5.27236	7.34873	9.41983	[L]
Fuel cons. difference	0.00063	0.00369	0.00040	[L]
Optimal fuel savings	5.69828	4.08332	3.14515	[%]
β candidate fuel savings	5.68708	4.03512	3.14103	[%]
Fuel savings difference	0.01121	0.04820	0.00412	[%]

Table 7.5: *Topographic road 30km*

Description	**20 ton**	**40 ton**	**60 ton**	**Unit**
Nominal fuel cons.	5.72234	7.93203	10.17397	[L]
Optimal fuel cons.	5.54845	7.83668	10.12388	[L]
β candidate fuel cons.	5.54999	7.83861	10.12835	[L]
Fuel cons. difference	0.00154	0.00193	0.00447	[L]
Optimal fuel savings	3.03879	1.20203	0.49230	[%]
β candidate fuel savings	3.01196	1.17775	0.44841	[%]
Fuel savings difference	0.02683	0.02429	0.04389	[%]

8 Discussion

This chapter discusses the results and the method used in this thesis.

8.1 Results

The result of this thesis is an algorithm for finding how to optimally catch up to a preceding vehicle and platoon. By weighting fuel against time with β in the objective function, the introduction of an extra state was avoided, making the algorithm computationally fast. The algorithm has been shown to find the optimal trajectory with respect to the road topography, for two test cases. Then to find the fuel-optimal solution the corresponding β was found by first using an optimal β candidate and then searching for the solution in a interval around the candidate. This increases the computation time, but by utilizing that some parts of the algorithm are independent of β, it was possible to save time by running the algorithm for several β at once in parallel.

The algorithm was then tested on road topography data from the highway E4 between Södertälje and Norrköping. The test showed that there is fuel saving potential to be had from taking real-life road topography into account when catching up and platooning.

8.2 Method

The method for creating the algorithm was to start from the algorithm proposed in Hellström [9], and changing it to go in a forwards manner calculating a cost-to-come, instead of going backward calculating a cost-to-go. This created the catch-up part of the algorithm and it was validated on two test cases where the optimal solution was known. Then the algorithm was extended to keep track

of the time-to-come for each state, so that it could know when it has caught up to the preceding vehicle. Platooning with a constant time gap was added and validated, by having the HDV platoon with an identical vehicle but without the air drag reduction, which resulted in the HDV having the same controls and fuel consumption as the preceding vehicle.

When platooning with a constant time gap, the following vehicle will have the same velocity profile as the preceding vehicle, making the platooning part easy to simulate. The platooning part can change gears instantly, choosing the one with the lowest fuel consumption, while the catch-up part has a gear shift model. This could be done since the platooning part is a simulation so it will not abuse model flaws, while the catch-up part is optimized and could thus abuse the flaws in the model.

Platooning with a constant time gap has some drawbacks. One is that it requires the following vehicle to be able to follow the preceding vehicle's trajectory, which it might not always be able to. For instance in the case of a steep uphill segment where the following vehicle is heavier than the preceding one, it might not be able to achieve the same velocity. Another drawback is that the platooning is not as efficient as it could be if the vehicles cooperated. For instance in the case of a steep downhill segment with two identical vehicles, if the preceding vehicle is free rolling then the following vehicle will also free roll, but since it is affected by less air drag it will want to go faster. The constant time gap does not allow this and thus the following vehicle will have to break, thus wasting energy which could have been used to further reduce the fuel consumption. Platooning with a constant time gap has instead the benefit of more closely approximating the fuel consumption when platooning behind another HDV without cooperating with them.

With the algorithm able to catch up and platoon with a preceding vehicle, finding the Pareto-optimal solution for the given β, the task of finding the β which gives the fuel-optimal solution remained. This was done by first finding an optimal β candidate and then searching for the solution in an interval of the candidate. This could have been done with methods like interval halving, but after finding that there is performance to be gained from running the algorithm for more than one β at a time, another method was proposed. The method was to scatter some points around the optimal β candidate and approximate a curve and search around the minimum of the curve. This helped in finding a better solution than the optimal β candidate, but the efficiency of this strategy has not been compared against other strategies.

8.3 The results in a wider context

With increasing concern over emissions from transport and increasing fuel prices, there is an increasing need for fuel-saving solutions. Platooning is one such solution, whereby driving behind another vehicle the air drag acting upon the HDV is reduced, leading to less energy required for the mission.

On today's highways, there are many scattered HDVs working for different

transportation companies with different missions, making cooperation hard. The algorithm finds the fuel-optimal solution for when catching up and platoon, given that the preceding vehicle's trajectory can be estimated. By being computationally fast, the algorithm is able to be used on-board a HDV, and thus it could help enable platooning of scattered HDVs on highways.

9 Conclusions

The conclusion of this thesis will regard the questions given identified at the beginning of the work

- When is it fuel-efficient to catch up to another vehicle and platoon?
- What is the optimal way to catch up to another vehicle?
- Is there a need to take road topography into account?

The proposed algorithm is used to find the optimal fuel consumption when catching up as well as the nominal fuel consumption when continuing to drive alone. This is then used to see if it is fuel-efficient or not to catch up and platoon. This also answers the question of what the optimal way to catch up is.

The road topography was shown to play a role in Chapter 7.

9.1 Future work

There are many topics that could be investigated in future work. One is looking into how the algorithm could be utilized to slow down to let a following vehicle catch up.

It could also be looked into what the effects of the preceding vehicle's trajectory have on the choice of platooning or not. Can this be utilized to make better matches when planning a mission. And could different platooning strategies be used, for instance, if there are HDVs with different platooning technologies enabled could the fuel consumption be estimated for each.

The algorithm has only been used without cooperative, could it be extended to find the cooperative fuel-optimal solution.

Finally, it would be of interest to see if the algorithm could be used to find the optimal solution if there are more than one preceding vehicle with different speeds, if there is an optimal way of going between them.

Bibliography

[1] *The Future of Trucks. Implications for energy and the environment.* IEA, 2017. ISBN 9789264279452. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=cat00115a&AN=lkp.924802&lang=sv&site=eds-live&scope=site.

[2] Assad Alam. Fuel-efficient heavy-duty vehicle platooning. 2014. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsbas&AN=edsbas.78DE0F2D&lang=sv&site=eds-live&scope=site.

[3] Fröberg Anders, Hellström Erik, and Nielsen Lars. Explicit fuel optimal speed profiles for heavy trucks on a set of topographic road profiles., 2006. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsstp&AN=edsstp.2006.01.1071&lang=sv&site=eds-live&scope=site.

[4] Richard Ernest Bellman. *Adaptive control processes : a guided tour.* Princeton Univ. Press, 1961. ISBN 0691079013. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat00115a&AN=lkp.494&site=eds-live&scope=site.

[5] David J. Chang and Edward K. Morlok. Vehicle speed profiles to minimize work and fuel consumption. *Journal of Transportation Engineering*, 131(3): 173 – 182, 2005. ISSN 0733947X. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=16145767&site=eds-live&scope=site.

[6] European Comission. Clean mobility: Putting an end to polluting trucks. commission welcomes first-ever eu standards to reduce pollution from trucks. 2019. URL https://ec.europa.eu/commission/presscorner/detail/en/IP_19_1071.

[7] Eurostat (European Commission). Energy, transport and environment statistics. *EU publications*, page 76, 2019. doi: 10.2785/660147. URL https://ec.europa.eu/eurostat/en/web/products-statistical-books/-/KS-DK-19-001.

[8] J.K. Hedrick, D. McMahon, V. Narendran, and D. Swaroop. Longitudinal vehicle controller design for ivhs systems. *1991 American Control Conference, American Control Conference, 1991*, pages 3107 – 3112, 1991. ISSN 0-87942-565-2. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edseee&AN=edseee.4791980&lang=sv&site=eds-live&scope=site.

[9] Erik Hellström, Jan Åslund, and Lars Nielsen. Design of an efficient algorithm for fuel-optimal look-ahead control. *Control Engineering Practice*, 18(11):1318–1327, 2010. ISSN 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2009.12.008. URL http://www.sciencedirect.com/science/article/pii/S0967066109002366.

[10] Alexander Johansson and Jonas Martensson. Game theoretic models for profit-sharing in multi-fleet platoons. *2019 IEEE Intelligent Transportation Systems Conference (ITSC), Intelligent Transportation Systems Conference (ITSC), 2019 IEEE*, pages 3019 – 3024, 2019. ISSN 978-1-5386-7024-8. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edseee&AN=edseee.8917349&lang=sv&site=eds-live&scope=site.

[11] Josefin Kemppainen. Model predictive control for heavy duty vehicle platooning. Master's thesis, Linköping UniversityLinköping University, Department of Electrical Engineering, The Institute of Technology, 2012.

[12] Kuo-Yun Liang, Jonas Mårtensson, and Karl Henrik Johansson. When is it fuel efficient for a heavy duty vehicle to catch up with a platoon?. *IFAC Proceedings Volumes*, 46(21):738 – 743, 2013. ISSN 1474-6670. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselp&AN=S1474667016384622&site=eds-live&scope=site.

[13] Alan Mckinnon. Increasing fuel prices and market distortion in a domestic road haulage market: the case of the united kingdom. 03 2020. URL https://www.researchgate.net/publication/237450805_Increasing_fuel_prices_and_market_distortion_in_a_domestic_road_haulage_market_the_case_of_the_United_Kingdom.

[14] V. V. Monastyrsky and I. M. Golownykh. Rapid computation of optimal control for vehicles. *Transportation Research Part B: Methodological*, 27(3):219–

227, June 1993. URL https://ideas.repec.org/a/eee/transb/v27y1993i3p219-227.html.

[15] Rikard Ohlsén and Erik Sten. Optimal platooning of heavy-duty vehicles. 2018. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsbas&AN=edsbas.A846ECD1&lang=sv&site=eds-live&scope=site.

[16] S. J. Pachernegg. A closer look at the willans-line. In *SAE Technical Paper*. SAE International, 02 1969. doi: 10.4271/690182. URL https://doi.org/10.4271/690182.