CHALMERS
UNIVERSITY OF TECHNOLOGY

Predictive Longitudinal Control of Heavy-Duty Vehicles Using a Novel Genetic Algorithm and Road Topography Data

Master's thesis in Complex Adaptive Systems

Fredrik Hoxell

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016:08

Predictive Longitudinal Control of Heavy-Duty Vehicles Using a Novel Genetic Algorithm and Road Topography Data

FREDRIK HOXELL

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

Predictive Longitudinal Control of Heavy-Duty Vehicles Using a Novel Genetic Algorithm and Road Topography Data
FREDRIK HOXELL

Supervisor: Pär Degerman, Scania, REIV
Examiner: Bengt Jacobson, Applied Mechanics, Chalmers university of technology

Master's Thesis 2016:08
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Scania R 450 4X2 Highline. Photo by Dan Boman, 2014.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2016

Predictive Longitudinal Control of Heavy-Duty Vehicles Using a Novel Genetic Algorithm and Road Topography Data

FREDRIK HOXELL

Department of Applied Mechanics
Chalmers University of Technology

Abstract

Fuel costs account for approximately one third of the total cost of haulage contractors. This makes it very lucrative from both the contractors' and hence Scanias' perspective to reduce the vehicles' fuel consumption. With limited power-to-mass ratio of heavy-duty vehicles, anticipatory control is crucial for fuel- and time-efficient manoeuvring. Solutions addressing this problem are already in production, but with ever-increasing system complexity the usefulness of conventional mathematical methods is suffering. As an alternative approach, this thesis is aimed at investigating the applicability of a real-time genetic algorithm (GA) to the domain of longitudinal control of heavy-duty vehicles for fuel-saving adaption to road topography data. Known to be computationally heavy, an as lightweight as possible algorithm is developed and aimed at optimising the engine torque by model predictive control. The final algorithm uses a vehicle prediction model of fuel-consumption data including a gear prediction model. Validated through simulation this novel approach displays a clear improvement over a similar MPC-controller utilising a QP-solver and a cost function similar to that of the GA.

Keywords: Adaptive, Look-ahead, Cruise Control, Genetic Algorithm, Quadratic Programming, Heavy-Duty Vehicles, Model Predictive Control

To Karin, for all your love and support

Acknowledgements

First I would like to express my deepest gratitude to Pär Degerman, my supervisor at Scania. You have offered great help and support throughout the project, but also have you believed in my work and given me great freedom to choose my own path and make this thesis truly mine.

I would also like to thank Professor Bengt Jacobson of Chalmers university of technology for helping make this project reality. It has been a pleasure to have you as my examiner and I am truly grateful for all your help and effort throughout this thesis.

A special thanks also goes out to the people of REIV. You have made my time as a thesis student memorable and with your ever-encouraging words and interest in my project, you have made me feel welcome and appreciated. Furthermore, I would like to thank Ulrica Hysing of REI for truly exceptional support and Christoffer Norén of REVD for inspiration and guidance as well as practical assistance.

Finally, this section cannot be completed without mentioning my family and friends, who have both managed to put up with me for all these years and been such a great part in bringing me where I am today.

Fredrik Hoxell, Södertälje, june 2016

Contents

Nomenclature

Acronym	Meaning
ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance System(s)
DP	Dynamic Programming
EA	Evolutionary Algorithm
ECU	Electronic Control Unit
GA	Genetic Algorithm
GMS	Gear Management System
HDV	Heavy-Duty Vehicle
LACC	Look-Ahead Cruise Control
LP	Linear Programming
MPC	Model Predictive Control
OEM	Original Equipment Manufacturer
QP	Quadratic Programming
SUS	Stochastic Universal Sampling

1

Introduction

Scania has a central role in the development of safer and more sustainable commercial transports. Today Scania offers driver assistance solutions such as Advanced Emergency Braking and Look-Ahead Cruise Control (LACC), while research is conducted in areas such as platooning and autonomous driving in traffic jams.

The conducted research indicates that new technological solutions have the potential to lower fuel consumption by 15% (e.g. platooning) and, for autonomous driving in traffic jams, this figure could be as high as 18% [1, 2]. In addition to improved fuel economy and thus reduced environmental impact, vehicles capable of switching into a mode of autonomous driving could increase the efficiency of the driver and reduces the risk of human errors.

There will be some time before fully automated vehicles reach the market, and currently there is a continuous transition happening in which the vehicles are step- or functionality-wise augmented as subsystems are being automated and in many cases more interconnected. One such system is cruise control, which for many years has been a widely implemented driver assistance system that aims to keep a constant cruise speed. This system, however, is challenged by the more recent adaptive cruise control (ACC). In cars, this generally means adaption to the speed of the vehicle ahead while keeping a safe distance [3]. For heavy-duty vehicles on the other hand, the limited motor power and potentially heavy load pronouncedly limits the speed and acceleration of the vehicle, making it highly desirable to add the ability to plan ahead in time and use road gradient information to utilise gravity and predict demanding ascents, streamlining the conversion between potential and kinetic energy. This becomes even more profound in the case of platooning of heterogeneous vehicles [4].

To this end, previous work has been conducted in the field of LACC (e.g. [5, 6]). In both papers the proposed method is dynamic programming for solving the optimisation problem with respect to time and fuel consumption. In [5] it is shown that the developed algorithm is able to run on an embedded system rated at 200MHz and with 32Mb of RAM. However, none of these methods are implemented in Scania vehicles. Instead, Scania Active Prediction (see [7]) is the system that is currently offered to customers; a look-ahead cruise control that is based on other methods. This system has proved to improve the fuel efficiency of heavy-duty vehicles (HDVs), thus potentially implying that there may be even more to gain by increasing its level of adaptivity and using control signals with different characteristics.

1.1 Purpose

Due to the effects of limited power-to-mass ratio of HDVs on the dynamics of such vehicles, the fuel efficiency can be improved by optimising the engine control with respect to fuel consumption by using information about upcoming road topography, typically 1-10 kilometers into the future.

The arising optimal control problem has been solved with a range of techniques, but in vehicular applications most traditional methods fail due to their need of processing power and memory, which in general cannot be met by electronic control units (ECUs) currently in production. Furthermore, these methods' reliance on mathematically stringency often require simple models and/or approximations to be made.

Whereas mathematical optimisation techniques, and especially dynamic programming, have been applied, there has been an upsurge in the application of evolutionary algorithms [8]. Research has been conducted within the field of evolutionary algorithms (EAs) for path planning ([9, 10]), but little or no research has been aimed at investigating the applicability of the algorithms to longitudinal control when restricted by efficiency and time constraints.

The main purpose of the thesis is to enter this previously unexplored field by investigating if genetic algorithms (GAs) can successfully be applied to a control problem of this nature. The problem may on a higher level be described as adapting the driving style to the road topography so that fuel consumption will be minimised without compromising the time efficiency. Although the investigated solution is applied to a problem that is already addressed in production software, the ultimate purpose it not simply to replace the existing solutions, but to investigate what potential lies in the application of genetic algorithms for longitudinal HDV control.

1.2 Specification of the purpose

The main objective of this thesis is to propose a genetic algorithm based controller for on-line fuel consumption optimisation via engine control in the HDV industry. An attempt is made to bring inspiration from genetic algorithms and soft computing into the field of on-line optimal control.

The relevant parameters describing the vehicle states are known to the algorithm, as are the vehicle model required to predict the vehicle's longitudinal dynamics and fuel consumption. The objective of the algorithm (also referred to as the *solver*) is to optimise the engine torque output with respect to time and fuel-consumption, subject to a set of constraints and reference values used to ensure driver comfort and speed limits among others.

1.2.1 Delimitations

For future automated trucks to be able to offer at least the same fuel efficiency as that of experienced drivers, the cruise control system must be able to adapt to the driver profiles of the vehicles within some distance from the ego-vehicle; both in the case of platooning but also in normal driving mode. However, this adaption to surrounding traffic does not fall within the scope of the project. The algorithm will thus not take potential fuel-savings associated with trailing other vehicles into account. Therefore, it is assumed that the vehicle travels on a highway or rural road with non-dense traffic, implying that interference from surrounding vehicles is at a minimum. Furthermore, the algorithm should neither take into account the curvature of the road nor lane changes or overtakings.

To fully optimise the speed profile of the vehicle with respect to efficiency and time, there is an imminent need to gain control over the gearbox, engine and brakes, amongst others. This is hampered by the current architecture of the communication- and control systems of Scania vehicles. Therefore, also the restriction that the planner to be developed is limited to controlling the engine is included.

1.3 Method

In the initial phase of the thesis, an in-depth literature study was conducted. Previous work within the field of look-ahead control and the closely related field of trajectory planning was studied to identify the strengths and weaknesses of various approaches. This study was supplemented by discussions with professionals within the area and the general direction of the project and solution could be decided.

As for the main part of the project, a simulation- and evaluation environment was developed along with the control algorithm. The modules were created as independent of each other as possible to facilitate the porting of the algorithm to different environments[1]. The purpose of the simulation module was to serve as a rapid-prototyping environment during the algorithm development.

The development of the algorithm and the framework was divided into cycles. Each cycle delivered working software but, more significantly, the various modules were evolved as more Scania-internal data information were made available in the later cycles.

As the algorithm approached its final form it was tuned and tested in a more extensive simulation environment including both theoretical formulae and in part also vehicle data collected from measurements. In its final form, the algorithm was also evaluated using this framework.

[1] e.g. Simulink models or StateFlow charts

1.4 Report outline

1. (**Introduction**)
2. **Background and previous work** - In this section some of the ideas from previous studies, upon which parts of this project are based, are presented. This thesis being a novel approach, a range of studies and applications are presented in an attempt to convey the core ideas of stochastic optimisation methods and what they can add to the field of (classical) optimisation.
3. **Heavy-duty vehicle prediction model** - Having the controller to be developed rely on state predictions of heavy-duty vehicles, this chapter is aimed at developing the required prediction models. The longitudinal dynamics of heavy-duty vehicles are addressed and presented along with motivated approximations.
4. **Model predictive control** - The core principles of the controller are presented with reference to the extensively employed method of model predictive control. A simplified version of the problem solved by the final algorithm is formulated in terms of two common classical optimisation methods; linear and quadratic programming.
5. **Genetic algorithms** - The main algorithm of this thesis is presented from the bottom up. A range of operators are presented along with reference to findings in previous studies, leading up to the final form of the genetic algorithm used in the controller.
6. **Hybrid algorithm** - As this thesis makes use of multiple solvers, the final solver is termed *hybrid algorithm*. In this section the structure of this hybridisation is presented.
7. **Algorithm evaluation** - Here the method of algorithm evaluation is addressed. It explains how the results were generated and includes an abstracted illustration of the simulation model developed in this thesis.
8. **Results** - Results generated through simulations are presented. This section contains results aiming to evaluate the fuel-saving potential of the algorithms, but also are results regarding computational time and algorithm predictability presented.
9. **Discussion**
10. **Conclusions**
11. **Future work**

2

Background and previous work

Optimisation is a field that has an almost infinite number of applications, spanning a tremendously wide range of scientific fields. The first section addresses this area from a point of view that serves as one of the main sources of inspiration for the contents of this project and moves on to the human addition to optimisation, which is concretised by applications in the automotive industry. Finally, important scientific results and issues are presented, which have served as motivation and/or inspiration for the choices that have been made in this thesis.

2.1 Evolutionary optimality and the human addition

The problem of optimisation is an ancient issue. Indeed, these types of problems have even been an integral part in the evolution. The concept of *survival of the fittest* may in many senses be translated to *survival of the most optimal*. Not only has evolution acted as a force of optimisation, but there are also obvious signs that animals can perform some kinds of optimisation (e.g. learn a policy) to maximise the return[1] of moving from one state to another [2]. Unlike many methods of optimisation that are widely used today, a very central part of the optimisation found in nature is *adaption*.

A clear human addition to the field of solving optimisation problems is the highly systematic approach. The most widely adopted tool is of course mathematics. There is a vast set of strictly mathematical optimisation techniques employed to find some optimum of a mathematical function, possibly under a set of constraints. Their widespread use alone indicates that the mathematical treatment of optimisation problems has certainly been fruitful. A prerequisite of the purely mathematical models is that the problem must be defined in terms of mathematics as well. In the case of systems, a mathematical model is often desirable since it enables the use of a wide range of methods of mathematical analysis. This is thoroughly exemplified by the almost countless number of studies performed within mathematical optimi-

[1] "Maximisation of return" could mean, for example, minimising the effort of moving from one point to another, or maximising the amount of food found while foraging.

[2] The terms "policy", "return" and "states" are taken from the field of reinforcement learning. Within this field, a policy is equivalent to a decision-making rule [11].

sation, the huge amount of literature on the subject, and not to mention today's implementation of Active Prediction. Evidently, the human addition to the field of optimisation is quite distinguished from that of nature, but both methods have their strengths and share the characteristic of performing a directed search.

2.2 Optimality in the vehicle industry

Optimality can mean different things and can vary widely depending on constraints. A common meaning of optimality is maximum efficiency (e.g. energy efficiency, cost efficiency, or time efficiency). Typically prominent actors are vehicle OEMs, but they are by no means the only ones. In the case of vehicles, there are various approaches to the problem of improving efficiency. Restricted to fuel efficiency, there are coarsely put two groups of measures; (1) improve the efficiency of the vehicle (e.g. minimise energy losses in the engine, reduce drag, reduce friction) and (2) improve the operation of the vehicle. The latter has a rather wide span, but a relevant part for this thesis is that of Advanced Driver Assistance Systems (ADAS). Although these kinds of systems have not fully penetrated the market and often are considered as premium-options, much research effort is being put into developing new systems. Examples are adaptive cruise control, lane-keeping assist, Advanced Emergency Braking and automatic parking. These systems aim to improve traffic safety, improve efficiency, relieve the driver, and/or improve the driving experience. A possible and certainly sought outcome for the future is that these systems will be able to fully replace the driver.

Some of the systems are intended to take over some of the driver's tasks or improve the awareness of the driver. However, a second set of systems is aimed at purely enhancing the driving in ways that even the most experienced drivers could not. Examples of such systems are map-enhanced or map-enabled ADAS, where map data is utilised when available or is a strict necessity for the function of the system, respectively. The system could then adapt to a particularly demanding part of the road topography even before the driver is aware of that specific road segment [3] [12].

2.3 Dynamic programming and the curse of dimensionality

Professor Richard Bellman is the father of dynamic programming. In the time period 1948 - 1952 he formed the foundation of a theory that is still used extensively today in various optimisation problems [13]. In short, the idea is to trade time complexity of algorithms for increased memory complexity. This is done by subdividing a problem into smaller parts, called stages, solving them one at a time. After a one-stage solution has been found, the next stage is included in the optimisation problem,

[3] Of course, a driver familiar with the road can also prepare for this demanding segment, but that is a special case, especially for transportation over long distances.

and so the problem is solved as a sequence of one-stage optimisation problems. Three main characteristics of a dynamic programming problem are that it should lend itself to division into stages, have states, and require recursive optimisation. The stages are required in order to subdivide the problem, while the states should contain the necessary information about the implications of the current decision for the future actions. Lastly, a prerequisite for applying dynamic programming is that the optimal policy satisfies the *principle of optimality*, which may be stated as:

> *Any optimal policy satisfies the condition that regardless of the current state and decision, the remaining decisions must yield an optimal policy with respect to the state that is reached as a consequence of the current decision.*

In general, the application of dynamic programming to a problem requires much thought and ingenuity in order to define the problem on the appropriate form. A very intuitive example, on the other hand, is that of the shortest path problem, or the closely related problem of finding the fastest path during rush hour [14, 15]. In those cases the stage may be represented by the number of blocks you are from your goal, while the state is represented by what intersection the traveller is at.

At a more concrete level, dynamic programming has for example been employed in optimisation of hybrid powertrains in [16]. As a characteristic of dynamic programming, the authors focus on the optimisation of the driving cycle of vehicles equipped with more than one power source, in this case a hybrid electrical vehicle. Other applications of dynamical programming are the problem of dividing a paragraph into lines of approximately equal length as discussed in [17], inferring batting conditions in cricket [18], and, what has been the subject of many theses and research projects, longitudinal control of heavy-duty vehicles (see for example [4, 5, 19, 20]).

Focusing on the latter example of applications, dynamic programming proved to be conceptually fruitful, albeit not fit for real-time on-board operation in all cases. In the one case where it was, lots of effort was put into researching suitable approximations and shortcuts in the algorithm, requiring extensive knowledge of the optimisation problem. Similarly, in order to keep the memory requirements within reasonable limits, the authors have made conscious decisions in designing the algorithms. The latter is a consequence of what often is referred to as *the curse of dimensionality*, meaning that an inherent property of dynamic programming is that the memory requirements grow out of hands very quickly when there are more than only a few state variables and the problem is of moderate size[4].

[4] An exact upper limit on the number of state variables and problem size for dynamic programming to be useful is very difficult to define since it is highly dependent on the resources allocated for the computations, but also because many workarounds have been developed, which are not necessarily universally applicable.

2.4 Genetic algorithms

A general trend in mathematical optimisation is that as the problem to be solved increases in complexity, they soon become unmanageable. Introduced by John Henry Holland in the 1960's, genetic algorithms are examples of soft-computing techniques that have been used in a wide range of applications to take over where conventional methods fall short [21]. As for dynamic programming, the field of applications is very wide. Also, when applied to real-world problems it is not unusual that the fitness function is non-differentiable and/or discontinuous and is cluttered with local minima and noise. In such cases it has been shown that conventional methods such as sequential quadratic programming and simulated annealing are outperformed by GAs [22]. To this end, it was also shown in a research project at Shenyang Institute of Automation that by utilising a GA for path planning, a snake-like robot could not only determine the shortest path, but also would it limit the influence of motor errors [23].

There are various methods to handle errors such as those of the motors in the snake-like robot. However, errors introduces extra complexity and in coupled systems with many components it may be very difficult or even impossible to manage these errors. Figure 2.1 shows two graphs. The solid line represents the actual system complexity that must be handled to fully master systems, whereas the dashed line represents our ability to handle system complexity. Evidently, there is a gap between these two lines and more importantly the system complexity increases faster than we develop methods to handle this complexity, leaving us increasingly incapable of mastering systems. In [24] this gap is referred to as the *catastrophe gap*. The main implications of this increasing gap are that the available methods are limited in terms of handling complexity and that our understanding of the holistic perspective of systems is fragmented. This, in turn, fuels the argument that employing self-adapting systems is a promising approach. In particular, genetic algorithms can handle complex optimisations problem without requiring the developers to have deep insight in the system to be optimised [25].

Of relevance for this thesis are mainly the applications of this algorithm family to trajectory planning and research of how to increase convergence speed and thus decrease run-time and resource requirements. As for the first point, the authors of [9] successfully applied a genetic algorithm to the path-planning problem facing a UAV travelling in the presence of other aircraft with uncertain trajectories. In the study it was found that the UAV managed to plan and execute a smooth trajectory while maintaining a good fuel economy and avoiding aggressive manoeuvres when yielding for other aircraft. No matter the vehicle, this avoidance of aggressive manoeuvring is often desired and the implications of the genetic algorithm carry over also to domains other than aerial vehicles. Quite similar to the studied UAV path planning it has also been shown that a GA employing variable-length chromosomes and tailored operators effectively and efficiently manages to plan a nice trajectory on a two-dimensional plane containing both static and dynamic obstacles [26].

Since GAs in general are computationally demanding (see Appendix B for run-time

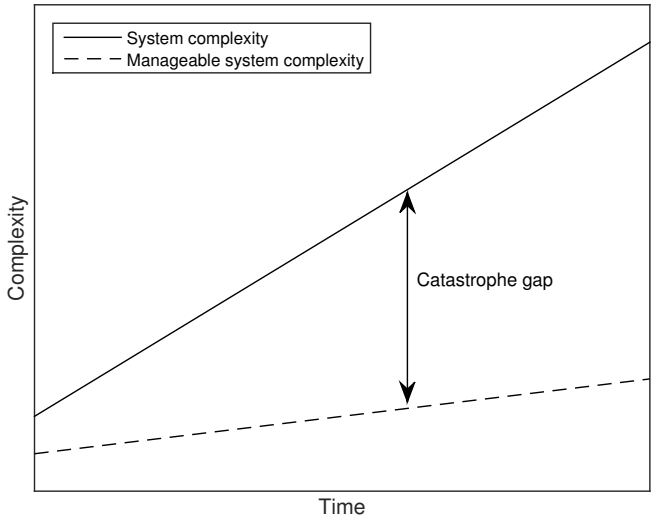

Figure 2.1: A graphical illustration of actual system complexity and the human ability to handle complex systems through history. What should be specifically noted is the ever-growing gap separating the lines. Finally, it should be remarked that the axes are left blank as the graph is only a conceptual illustration.

estimation), their initial field of application was static optimisation problems. According to the authors of [25], it was in the late 80's or early 90's that GAs were first considered interesting for application to optimal control problems. Thus, this means that the applications have matured over a period of just under 30 years. Also, as there has been a constant increase in accessibility of computational power over time, new areas of applications have emerged naturally. As a result, genetic algorithms are no longer restricted to static problems and are extensively covered in literature. For example, in [22] the authors consider GAs as viable and intelligent solvers for computationally expensive problems and, serving as one of many examples, the authors of [27] dive into the field of multi-objective optimisation from the perspective of GAs. Although this thesis does not include multi-objective optimisation in the strict meaning, it is certainly of relevance for vehicle control.

A consequence of more efficient computers is decreasing computer size as well as price drops. This opens up for implementing genetic algorithms in systems where price, size and/or weight are limiting factors (e.g. vehicles, airborne systems or systems in mass production). In an investigatory study the authors of [28] implemented a Nondominated Sorting Genetic Algorithm (NSGA-II) in a 180MHz microcontroller. Specifically, the authors conclude that the application of the developed algorithm to real-time vehicle control is successful and refers to the solution architecture as a viable option for ADAS implementations.

Summed up, genetic algorithms have been thoroughly studied and applied to dynamic optimisation problems of various kinds, most of which have no direct connection to longitudinal vehicle control. However, despite the problem formulations not being the same, the conceptual ideas of the previous studies form a firm foundation

to build upon when developing a real-time controller including computation-intense functions.

3

Heavy-duty vehicle prediction model

As the state of a heavy-duty vehicle at a specified position may depend on the state and control signals of the truck several kilometres back, the algorithm that is developed in this thesis relies on making predictions about future states and control signals. The details are left for chapters 4 and 6, but suffice to say that in order to predict the state of the vehicle, a model must be developed. Furthermore, as the fuel consumption is a direct measure of the success and usefulness of the algorithm, both the longitudinal dynamics and fuel consumption properties of the vehicle must be considered. This chapter is dedicated to developing these models. Specifically, in Section 3.1 a fuel consumption model with low online computational complexity is presented, while Section 3.3 proposes a realistic, yet simplified, propulsion model whose main characteristics are captured in a required simplification developed in Section 3.4. The chapter also presents real data for Scania engines, but all data has been considerably corrupted and scaled to unity to enforce company secrecy. The most fundamental data for this chapter is a 3D map of the fuelling as a function of engine speed and torque and is presented in figure 3.1.

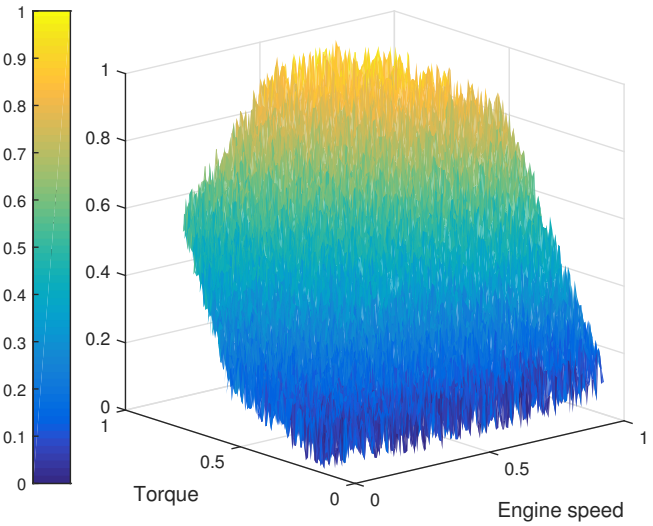

Figure 3.1: A typical map of the fuel flow as a function of torque and engine speed. Note that the data has been corrupted.

3.1 Fuel consumption

To describe the truck, a state vector $x = [v, s, G]$ is used, where the speed of the truck is denoted v, s is the distance from the reference point, and G is the engaged gear. The basic control signals of a basic propulsion system exposed to the driver or control system are throttle, brake and gear. However, in this thesis, gear selection is assumed inaccessible for the control system to be developed. The control signals that are available to the system are presented in table 3.1.

Table 3.1: Control signals available to the control system.

Variable	Signal	Unit
u_f	Fuelling	g/min
u_b	Brake	Nm

The engine output torque τ_e depends both on the fuelling and the engine speed. As found in [5], the dependence is almost linear:

$$\tau_e(\omega_e, u_f) = e_1\omega_e + e_2u_f + e_3, \tag{3.1}$$

where ω_e is the engine speed and u_f is the fuelling.

Although this may capture the coarse characteristics, it is seen from figure 3.2 that there are clear deviations. The graphs are generated by finding the coefficients $e_i,\ i = 1, 2...$, in

$$\tau_e(\omega_e, u_f) = e_1\omega_1 + e_2u_f + e_3u_f\omega_e + e_4u_f^2 + e_5\omega_e^2 + e_6\omega_e^3 + e_7 \tag{3.2}$$

that minimises the squared difference at the sampling points.

Even when including some of the $3rd$-order terms, the fitted function deviates notably at several points and increasingly so towards the endpoints of the interval of engine-speed values. With the restriction in computational power, more advanced functions are not considered and as for equation (3.2), it is deemed inadequate. It is instead replaced by a lookup table, which has an associated time complexity of $O(1)$ and can easily represent non-linear behaviour in the fuel flow map. The trade-off is instead that analytical approaches are obstructed. Based on this note, equation (3.2) is replaced by the mapping

$$\hat{\tau}_e(\omega_e, u_f) = map_\tau(\omega_e, u_f) \tag{3.3}$$

Similarly, when measuring fuel consumption the resulting data is structured as a discrete map. Figure 3.3 shows a typical fuel flow map as a function of engine speed. The map is generated by measurements of the fuel consumption at specific steady states with constant engine speed and torque. The map in figure 3.3 is upsampled by cubic interpolation between these steady-state measurements. The plot clearly visualises the maximum fuel flow for the different engine speeds.

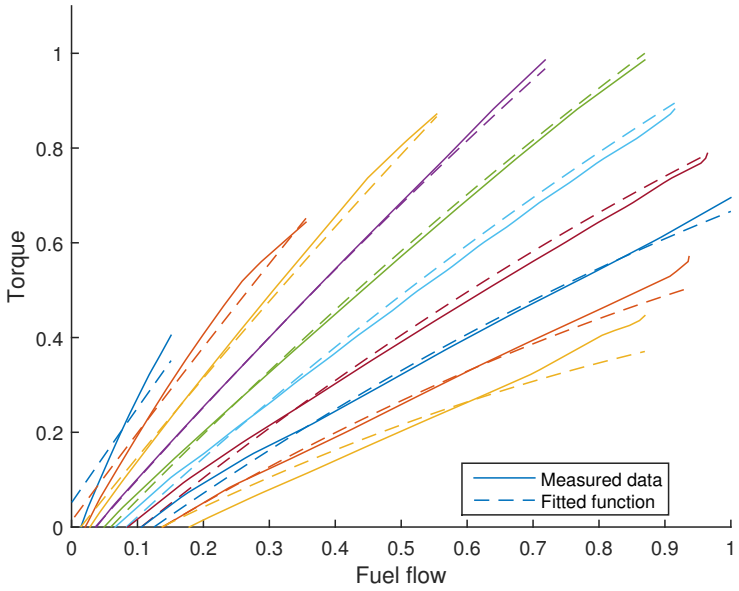

Figure 3.2: Shifted and normalised data from fitting the coefficients of equation (3.2) to the measured data. Each line colour corresponds to constant engine speed (increasing from up/left to down/right).

More important for the design of the algorithm in this thesis is the projection of the fuel-flow map onto the fuelling-torque plane. This projection is presented in figure 3.4. Two lines have been superimposed on the graph. Line A represents the maximum engine torque and line B represents the torque when the fuelling is zero and the engine thus is completely dragged. Evidently, the range of available torque output from the engine is a varying function in engine speed. As will be described in greater detail in the following chapters, the output from the algorithm is the recommended torque request, and the dynamic range must be handled somehow. This problem of varying torque range is addressed by letting the algorithm request any torque, but simply pulling any outliers back inside the valid interval at evaluation time.

3.1.1 Total fuel consumption

As the vehicle accelerates or decelerates, the engine speed changes. However, as the sampling interval is traversed in approximately one second, this change in engine speed is rather small over a single segment. Given this, the predicted fuel consumption is computed based on the mean value of the engine speed at the start and end of the segment to reduce the number of computations needed, provided that no gear shift occurs.

Per definition one has

$$\bar{v} = \frac{\Delta s}{t} \quad \Leftrightarrow \quad t = \frac{\Delta s}{\bar{v}}, \quad t > 0,$$

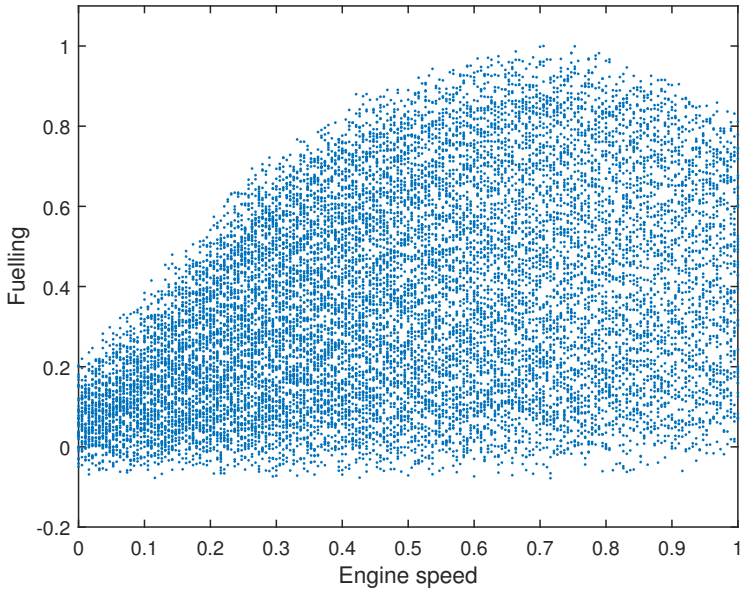

Figure 3.3: The fuelling as a function of engine speed, for a family of curves with constant torque. This scatter plot is a projection of figure 3.1 onto the fuelling-speed plane. Note that the data have been intentionally corrupted.

where Δs is the distance travelled in time t.

With constant acceleration, a, it follows that

$$\bar{v} = \frac{v_f + v_0}{2},$$

which is the mean value of the initial and final speed of the truck.

Thus, the time needed to travel over a segment of length Δs is

$$\tilde{t} = \frac{\Delta s}{(v_f + v_0)/2}.$$

Assuming nearly constant acceleration, $\tilde{t}$ is a good approximation of the time taken to travel a distance Δs.

Assuming that the fuel flow over each discrete segment may be modelled as constant and denoting the fuel flow at segment k by $\dot{m}_k$, the total fuel consumed when travelling over N intervals, each of length Δs, becomes

$$m_f = \sum_{k=1}^{N} \dot{m}_k \tilde{t}_k, \qquad [\text{g}]. \tag{3.4}$$

It is convenient to have a way of relating the fuel consumption in mass to the contained energy, since it then can be compared to the kinetic energy of the vehicle and the useful energy output or absorbed by its engine and brakes. This is done by

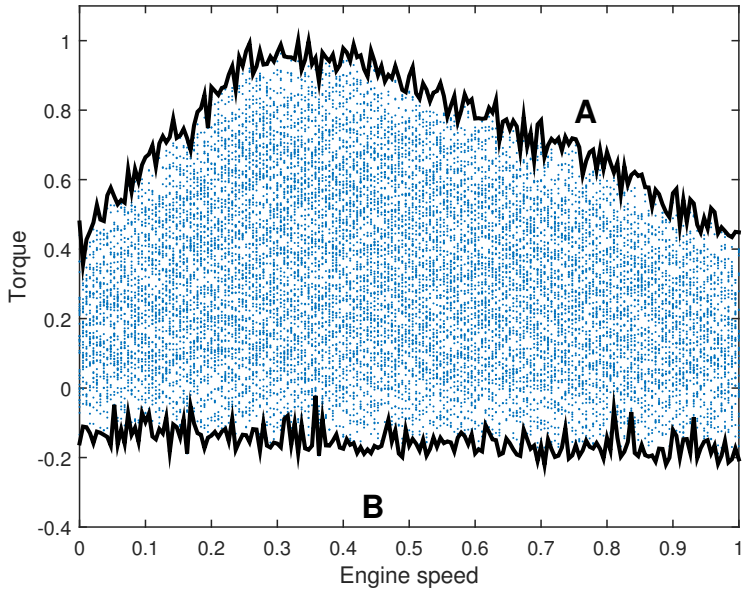

Figure 3.4: Interpolated scatter plot of the torque as a function of engine speed. Each point represents a constant fuel flow. The line labelled 'A' marks the maximum torque and the line labelled 'B' marks the torque when the engine is completely dragged (i.e. when the fuel flow is zero). The lines are very jagged due to the heavy corruption of the data to ensure company secrecy.

converting the energy content of consumed fuel to Joules. The value for the energy content used in this thesis is $c_f = 4.8 \cdot 10^7$ J/kg. The energy equivalent to the mass fuel consumption is

$$E_f = c_f \sum_{k=1}^{N} \dot{m}_k \tilde{t}_k, \qquad [J]. \tag{3.5}$$

3.2 Longitudinal dynamics

To capture the complete characteristics of a vehicle, it must be considered in all three dimensions. However, under the assumption that the road is well-behaved (smooth curves etc.), which in general is the case for highway-driving, the problem can be reduced to only encompass the longitudinal dimension.

The longitudinal component, a, of the instantaneous acceleration of a HDV is

$$a = \frac{1}{m}(F_w - F_r - F_d - F_g), \tag{3.6}$$

with m being the mass of the vehicle, F_w the longitudinal force from the ground acting on the wheels (i.e. propulsion and braking), F_d the air drag, F_g the longitudinal gravitational component, and F_r the rolling resistance. In fact, the rolling

resistance should be modelled as a torque if tire slippage is to be taken into consideration. Since it is assumed that the tires do *not* slip, the rolling resistance is in spite the previous remark modelled as a force in order to maintain consistency with the referenced theory.

A frequently used model for air drag is

$$F_d = \frac{\rho}{2} C_d A_v v^2, \tag{3.7}$$

where ρ is the density of air, C_d and A_v are the drag coefficient and frontal area of the vehicle, respectively, and v is the relative speed between the vehicle and the air [29].

The rolling resistance is given by

$$F_r = C_r F_N, \tag{3.8}$$

where C_r is the coefficient of rolling resistance and F_N is the normal force acting on the wheel under consideration [30]. For a truck travelling on a road of slope $\alpha(s)$, where s is the distance from some reference point, the normal force is given by

$$F_N = mg\cos\alpha(s). \tag{3.9}$$

The rolling resistance is highly dependent on various factors such as tire pressure, tire make, temperature and of course the road surface itself. Additionally, the rolling resistance is speed dependent; a dependence proposed in [31] to be on the form

$$C_r = C_{r,1} + C_{r,2} v^2. \tag{3.10}$$

$C_{r,1}$ and $C_{r,2}$ are constants related to the tire. In practice, $C_{r,2}$ is typically many orders of magnitude smaller than $C_{r,1}$ and can be both positive and negative, and can thus generally be dropped completely from the above equation. However, for completeness, it is kept throughout the calculations below.

When travelling on a road of slope α, the gravitational contribution to the longitudinal force is

$$F_g = mg\sin\alpha(s). \tag{3.11}$$

Inserting equations (3.7) - (3.11) in (3.6) then yields

$$ma = F_w - \frac{\rho}{2} C_d A_v v^2 - mg\sin\alpha(s) - (C_{r,1} + C_{r,2} v^2) mg\cos\alpha(s), \tag{3.12}$$

assuming that the truck is travelling forward at speed $v > 0$. A summary of the forces acting on the truck is given in table 3.2. The three rightmost terms are

straightforward since they depend only on the speed, v, and position, s,[1] of the vehicle. The force that the vehicle exerts on the road to propel itself, however, depends on both the state and characteristics of the drivetrain. A simplified version of this dependence is presented in Section 3.3.

Table 3.2: Summary of forces

Force	Designation	Equation
Gravitational	F_g	$mg\sin\alpha$
Normal	F_N	$mg\cos\alpha$
Rolling resitance	F_r	$(C_{r,1} + C_{r,2}v^2)F_N$
Air drag	F_d	$\frac{1}{2}\rho C_d A_v v^2$
Propelling force	F_w	See Section 3.4

3.3 Vehicle motion

According to Newton's second law, the rotational acceleration of the engine is given by

$$J_e\dot{\omega}_e = \tau_e - \tau_{out}, \tag{3.13}$$

where τ_e is the instantaneous torque generated by the engine, J_e is the moment of inertia of the engine and τ_{out} is the torque supplied to the clutch or torque converter.

The gear ratio separating the engine and the wheels consists of the gearbox transmission ratio, i_g, and the final drive ratio, i_f. The total transmission ratio depends on the gear, G, and is given by

$$i(G) = i_g i_f. \tag{3.14}$$

The relation between the engine speed and the rotational speed of the driving wheels, ω_w, is

$$\omega_w = \frac{\omega_e}{i(G)}. \tag{3.15}$$

The torque transferred to the clutch, T_{out}, decreases gradually due to energy losses as it is transferred through the driveline components. This decrease is modelled by an efficiency, η. The effective torque appearing at the driving wheels is thus

$$\tau_{e,w} = \eta \cdot i(G) \cdot \tau_{out}. \tag{3.16}$$

The governing equation for the propulsion is thus

[1] henceforth the dependence on distance is assumed self-evident and thus omitted for brevity.

$$J_d\dot{\omega}_w = \tau_{e,w} - F_w R_w - \tau_b = \eta i(G)\tau_{out} - F_w R_w, \tag{3.17}$$

where τ_b is the brake torque, R_w is the wheel radius and J_d is the inertia of the drive train and wheels together.

Under the condition of no slip, the acceleration of the truck is related to the angular acceleration of the wheel according to

$$a = R_w\dot{\omega}_w. \tag{3.18}$$

As explicitly done in appendix A, combining equations (3.13) - (3.18) and solving for a yields

$$a = \frac{R_w}{J_d + mR_w^2 + J_e\eta i^2(G)}[i(G)\eta\tau_e - \tau_b - R_w(F_d + F_r + F_g)] \tag{3.19}$$

3.4 Simplified prediction model

The above model is indeed a simplification of the complex workings of an engine and driveline, but still it contains parts that are very specific to what engine and driveline components the vehicle is endowed with. In a simplified model, a proportion η_s of the torque from the engine is transferred to the wheels, where η_s represents the internal losses in the driveline components and moments of inertia of the powertrain constituents. In reality, a constant efficiency η_s cannot replace the characteristics of equation (3.19), but recalling that only highways and rural roads are considered, the vehicle will operate in a narrow(er) operating space which increases the validity of this assumption. The effect of the assumption is that the net force acting on the wheels becomes

$$F_w = \frac{\eta_s\tau_e i(G)}{R_w} - \frac{\tau_b}{R_w}. \tag{3.20}$$

Equation (3.12) then becomes

$$ma = \frac{\eta_s\tau_e i(G)}{R_w} - \frac{\tau_b}{R_w} - \frac{\rho}{2}C_d A_v v^2 - mg\sin\alpha - (C_{r,1} + C_{r,2}v^2)mg\cos\alpha. \tag{3.21}$$

The calculations below are simplified by the substitution $T = mv^2/2$, where T then is the kinetic energy of the truck. Furthermore, since the road is discretised into segments of length Δs, typically in the vicinity of 20 m, and τ_e, τ_b, α and $i(G)$ are assumed constant over each interval, it is possible to collect (piecewise) constant terms in (3.21) according to

$$ma = c_2 - c_1 T, \tag{3.22}$$

where

$$c_1 = \frac{\rho C_d A_v}{m} + 2C_{r,2} g \cos\alpha, \qquad c_2 = \frac{\eta \tau_e i(G)}{R_w} - \frac{\tau_b}{R_w} - C_{r,1} mg \cos\alpha - mg \sin\alpha.$$

A result of basic dynamics is the relation

$$v \cdot dv = a \cdot ds.$$

Under the substitution $T = mv^2/2$, this turns into

$$dT = ma \cdot ds.$$

Together with equation (3.22) this yields

$$dT = (c_2 - c_1 T) ds.$$

Solving this separable differential equation results in

$$\frac{|T - \frac{c_2}{c_1}|}{|T_0 - \frac{c_2}{c_1}|} = e^{-c_1 \Delta s}.$$

In the special case of $T_0 = c_2/c_1$, the resulting force is zero, which means that T will not change (i.e. $T = T_0$). If $T_0 < c_2/c_1$, there is initially a resultant force propelling the vehicle. T can only approach the equilibrium $T = c_2/c_1$ from below, but never exceed it. In the opposite case, $T_0 > c_2/c_1$, T can only approach the equilibrium from above. This observation indicates that the sign of the expressions within the bars will always have the same sign, and it follows that

$$T = (T_0 - \frac{c_2(\tau_e, \tau_b, s)}{c_1(s)}) e^{-c_1(s)\Delta s} + \frac{c_2(\tau_e, \tau_b, s)}{c_1(s)}, \tag{3.23}$$

where the variables' dependencies have been re-included for clarity.

Recall that the above result was derived under the assumption that the truck was travelling forward. When a truck travels on a highway or rural road under normal conditions and with the cruise control active, this is a valid assumption. However, the exclusion of the sign-dependence of the rolling resistance on the vehicle speed causes the rolling resistance to appear as a force *always* acting in the backward direction. If the HDV comes to a halt between two sampling points, equation (3.23) will become negative, which is physically impossible. Thus, it must appear natural to the algorithm to not assume that the vehicle will in fact reach the end of every segment, in which case the solution should be discarded as it is deficient.

4

Model predictive control

Model Predictive Control (MPC) is, as the name suggests, an advanced control method where predictions of future states are made based on a model of the system. MPC is not an algorithm itself, but an umbrella term for control strategies that seek to optimise a process by finding the control signal sequence that minimises a cost function. Since it was first proposed as a control strategy already in the late 1970's, MPC has evolved and is today applied in a variety of control situations. MPC is synonymously termed Receding Horizon Control, which stems from the fact that, at each sampling point, a prediction about a finite future is made. Thus, the horizon of the prediction is pushed forward a step Δs at every sampling point, where Δs is the sampling interval. The total look-ahead is therefore $S = N\Delta s$, where N is referred to as the *prediction horizon*. In the prediction of signals and states at sampling point $q \in \mathbb{N}^0$, an MPC controller can (but is not obliged to) take into consideration all states and signals preceding that point [32]. A typical MPC problem, which is also the form used in this thesis, could take the form

$$\begin{aligned} &\arg\min_{u(x)} \sum_{k=q}^{N+q-1} f(x,u,r), \qquad q \in \mathbb{N}^0 \\ &\quad \text{s.t.} \\ &\qquad u(k) \in \mathcal{U} \ \forall k, \\ &\qquad x(k) \in \mathcal{X} \ \forall k, \\ &\qquad x(k+1) = f_s(x,u) \end{aligned} \tag{4.1}$$

with

$f(x,u,r)$ = Cost function
$f_s(x,u)$ = System model. Gives the next state, given previous states and signals
$r(k)$ = Reference signal(s) over the kth interval
$u(k)$ = Control signal(s) over the kth interval
$x(k)$ = System state at the kth sampling point
$\mathcal{U}$ = Set of possible signals ($\dim(\mathcal{U}) = \{\text{number of signals}\}$)
$\mathcal{X}$ = Set of possible states ($\dim(\mathcal{X}) = \{\text{number of state variables}\}$)

In terms of the controller developed in this thesis it follows that $x = [v, s, G]$ and $u = [\tau_e, \tau_b]$. As regards the computation of the next state, $x(k+1)$, the QP-solver uses the rather crude forward Euler method due to its associated simplicity, low computational cost and the limiting mathematical requirements put on the formulation in order to turn it into a QP-problem, but also due to the fact that this method is indeed still widely used today. The use of the forward Euler method is further justified by the fact that the vehicle model used by the QP-solver is simplified compared to the prediction model used by the GA. Increasing the accuracy of the QP-solver, and thus generally the computational complexity, will not necessarily result in appreciable gain. Compared to the QP-controller, the GA puts much less emphasis on the mathematical formulation, enabling it to employ more advanced methods to predict the value of $x(k+1)$. As a result, the forward Euler method is in this specific case replaced by the analytical expression in equation (3.23).

Different solvers have different performance with respect to various parameters, but typically must the cost function not be too complex for the problem to be tractable in terms of complexity, memory consumption and computational effort [33].

Algorithmically, the concept of MPC can be summarised as in algorithm 4.1.

Algorithm 4.1 Simple MPC algorithm

```
1: p ← FormulateProblem()                    ▷ On the form required by solver
2: q ← q_0
3: while True do
4:     x(q) ← MeasureState()
5:     u_temp ← Solve(p, x, u)               ▷ Returns signals for next N steps
6:     u(q) = u_temp(0)                      ▷ 0-indexing
7:     Send u(q) to the system
8:     q ← q + Δq
9: end while
```

As can be seen from algorithm 4.1, when the solution to the minimisation problem is found, only the very first element of the proposed sequence of control signals is actually sent to the system. A potential advantage of this approach is that if the state of the system can be quickly and accurately determined and the algorithm completes sufficiently fast, the errors of a simplified system model will not directly affect what control signals will be proposed in the future as the algorithm continuously updates the state estimation and predictions at each sampling. On the other hand, if the accuracy and speed of predicting states is better than measuring them at each sampling point, or if the algorithm is very slow compared to the system dynamics, then the algorithm could be adjusted to accept a greater part of the proposed signals. However, lots of research has been put into developing algorithms to improve the solution speed for MPC controllers in systems with fast dynamics (see for example [34, 35, 36]), which has resulted in a very wide range of optimisation methods for MPC, both addressing problems with fast dynamics, limited computational power and/or complicated system models.

4.1 Minimising engine energy output by model predictive control

The engine efficiency is a function of the working point. As a result, minimising energy output of the engine is not equivalent to minimising fuel consumption. However, for normal driving, there is a correlation between fuel consumption and engine energy output, and using either one as the cost function will lead to a solution that is at or close to the minimum in fuel consumption.

As a discretised problem, the control actions are assumed constant over an interval Δs. It then follows that the energy output from the engine over the kth interval is

$$E_{e,k} = \frac{\tau_{e,k} i_{g,k} i_f}{R_w} \Delta s. \tag{4.2}$$

Similarly, the brake energy is

$$E_{b,k} = \frac{\tau_{b,k}}{R_w} \Delta s. \tag{4.3}$$

The core of the problem is the minimisation of the cost function

$$\begin{aligned} J &= \sum_{k=1}^{N} E_{e,k} - c_T \sum_{k=1}^{N} T_k \qquad (4.4) \\ \text{s.t.} \\ &\frac{1}{2} m v_{min}^2 \leq T_k \leq \frac{1}{2} m v_{max}^2, \\ &0 \leq E_{e,k} \leq E_{e,max}, \\ &0 \leq E_{b,k} \leq E_{b,max}, \\ &T_{k+1} = T_k + E_{e,k} - E_{b,k} - E_{env,k}, \end{aligned}$$

where E_{env} denotes the environmental forces (i.e. gravity, roll resistance, and air drag). The inclusion of kinetic energy term is explained by the fact that the optimal strategy when only considering engine output energy is simply to give no gas at all, which obviously conflicts with the desire of the driver to maintain speed and arrive at the destination. Furthermore, energy can indeed be absorbed by the engine by letting it be dragged. However, while less or no fuel at all is consumed while dragging the engine, letting the engine energy output be negative leads to solutions where the engine brake is used in inappropriate situations, which explains the second constraint. This may be better understood by noting that applying the engine brake will in fact not recover energy, but simply avoid consuming fuel. Therefore, the terms in the first sum of equation (4.4) should not be allowed to decrease the value of the cost function by being negative.

The above problem formulation may be readily stated on the standard form of a linear programming problem,

$$\begin{aligned} &\arg\min_{\mathbf{y}} \mathbf{p}^\intercal \mathbf{y} \\ &\text{s.t.} \\ &\quad A\mathbf{y} = \mathbf{b} \\ &\quad C\mathbf{y} \leq \mathbf{d}, \end{aligned}$$

where $\mathbf{y}$ is the vector of variables, $\mathbf{p}$, $\mathbf{b}$ and, $\mathbf{d}$ are known vectors and A and C are known matrices.

As pointed out in [37], when employing the cost function in (4.4) the vehicle speed usually tends towards the edges of the allowed speed range in static driving (i.e. constant slope). This is an undesired behaviour, since under static conditions the cruise control system should track the reference speed provided by the driver. Another essential factor to take into account is the driver comfort, which would be compromised by excessive changes in engine torque. In addition to driver comfort, smooth driving reduces mechanical wear as well as fuel consumption [37].

A natural way to include these factors are to penalise deviations from the reference speed as well as changes in torque, thus introducing the following costs in the cost function:

$$c_T \sum_{k=1}^{N} \left(T_k - \frac{1}{2} m v_d^2 \right)^2 + c_s \sum_{k=1}^{N} \left(E_{e,k} - E_{e,k-1} \right)^2,$$

yielding the final cost function

$$J = \sum_{k=1}^{N} E_{e,k} + c_T \sum_{k=1}^{N} \left(T_k - \frac{1}{2} m v_{d,k}^2 \right)^2 + c_s \sum_{k=1}^{N} \left(E_{e,k} - E_{e,k-1} \right)^2, \tag{4.5}$$

where $v_{d,k}$ is the desired speed at segment k, and c_T and c_s are non-negative parameters defining the importance of tracking the reference speed and smooth driving, respectively.

With the new additions, the problem turns into a quadratic programming (QP) problem, whose general form is

$$\begin{aligned} &\arg\min_{\mathbf{y}} (\frac{1}{2}) \mathbf{y}^\intercal H \mathbf{y} + \mathbf{p}^\intercal \mathbf{y} \\ &\text{s.t.} \\ &\quad A\mathbf{y} = \mathbf{b} \\ &\quad C\mathbf{y} \leq \mathbf{d}, \end{aligned} \tag{4.6}$$

where H is a known matrix.

Rewritten on this form, the stated MPC problem becomes a convex QP problem [37]. Both LPs and QPs have been thoroughly studied and hence there are many robust high-speed solvers concerned with the task of solving these kinds of problems. As reported in [38] the solution to the current problem represented on the form (4.6) may be found in less than 1/5 of a millisecond [1].

4.1.1 Constant-speed correction

In steady state, the third term in equation (4.5) is identically zero, making term 1 and 2 the only competing terms. Lower speed requires less (propulsion) energy from the engine since the air drag and rolling resistance decreases, although the latter only decreases very marginally. At the same time, as the speed is lowered below the set speed, the second term grows as a consequence of the square. In simulations conducted both in this thesis and in [38] it is observed that close to the set speed, v_d, the magintude of the derivative of the second term is greater than that of the first term, causing the steady-state speed to be slightly below the set speed. The reason for this is formally clarified by a steady-state analysis. In steady state, for which $\alpha = 0$ is assumed, the engine only has to balance the rolling resistance and air drag. Thus,

$$E_e = F_r \Delta s + F_d \Delta s = (C_{r,1} + \frac{2C_{r,2}T}{m}) mg \cos(0) \Delta s + \rho C d A_v \frac{T}{m} \Delta s. \tag{4.7}$$

Furthermore, the cost function in (4.5) reduces to

$$\begin{aligned} J_{steady} &= N E_e + c_T N (T - T_d)^2 \\ &= N \Delta s ((C_{r,1} + \frac{2C_{r,2}T}{m}) mg + \rho C_d A_v \frac{T}{m}) + c_T N (T^2 - 2TT_d + T_d^2), \end{aligned} \tag{4.8}$$

where T and T_d are the kinetic energies corresponding to the desired speed and the corrected desired speed, respectively.

Differentiating equation (4.8) with respect to kinetic energy yields

$$\frac{\partial J_{steady}}{\partial T} = 2NC_{r,2} g \Delta s + \frac{N \rho C_d A_v}{m} \Delta s + 2N c_T T - 2N c_T T_d. \tag{4.9}$$

To ensure that the steady-state speed is not different from the desired speed, we require that the minimum of the reduced cost function in equation (4.8) coincides with the steady state. Thus, it is required that

$$\frac{\partial J_{steady}}{\partial T} = 0,$$

[1] The result is based on a discretisation of $\Delta s = 25\,\text{m}$ and $N = 20$. The solver (custom-generated by CVXGEN) was run on a computer equipped with Intel Core i5 (2.67 GHz).

which yields

$$T_d = \frac{\Delta s C_{r,2} g}{c_T} + \Delta s \frac{\rho C_d A_v}{2 c_T m} + T.$$

Finally, solving for v_d gives the final expression

$$v_d = \sqrt{\frac{2\Delta s C_{r,2} g}{c_T m} + \frac{\Delta s \rho C_d A_v}{c_T m^2} + \frac{2T}{m}}. \tag{4.10}$$

To ensure this is a minimum, the derivative in equation (4.9) is differentiated once more with respect to T:

$$\frac{\partial^2 J_{steady}}{\partial T^2} = 2N c_T > 0,$$

thus confirming that if the desired speed is defined according to equation (4.10), then the minimum of the cost function in (4.5) during steady-state driving on flat ground will be such that the actual speed and uncorrected value of the desired speed coincide.

5

Genetic algorithms

Genetic Algorithms constitute a subgroup of evolutionary algorithms. EA is a collection term for a family of stochastic optimisation algorithms. They are termed *evolutionary* due to their property of resembling the evolution found in nature. The various kinds of EAs are based on different evolutionary concepts, and in the case of genetic algorithms it is, of course, genes that serve as main inspiration. However, it should be emphasised that the actual biological process that serves as inspiration for genetic algorithms is many times more complex than the resulting optimisation method [39].

5.1 The biological process in short

Evolution is the continuous development of living organisms over time. The process is very slow and the final result of evolution is an accumulation of the changes up the branches of ancestors. The evolutionary progress thus relies on changes to persist through generations. To that end, the information must somehow be stored. Additionally, it must also be passed on to the offspring. More specifically, in Darwinian theory of evolution one is talking about a heritage in behavioural and/or physical traits [40]. In nature, this is realised by the genome - the complete DNA-set of an organism. The units of the DNA that codes for a specific protein or set of proteins are called genes. Focusing on the human species, there are between 20,000 and 25,000 genes amassed in 23 chromosome pairs.

The information that is stored in the DNA is not directly accessible by the part of the human cell that builds the proteins from the instructions contained in the genes. That is, the useful information is *encoded* and must be decoded before it can be used. In the cell, the decoding is performed by an enzyme, RNA polymerase, in a process that outputs the *messenger ribonucleic acid* (mRNA). The information that is transcribed in the mRNA-molecule is then used in the ribosomes so as to synthesise the proteins that in turn form the individual [41].

This is only a very brief description of the biology behind the synthesis of proteins from DNA information, but it suffices for the purpose of developing the basics of GAs. In addition to the theory concerned with the workings of biological processes, other scientific theories are often used as inspiration, such as the Mendelian theory of inheritance, but also non-biological ideas such as simulated annealing or, more

generally, statistical physics.

5.2 Algorithm design

There are many ways to take inspiration from genetics when building an algorithm. In this section, the fundamental building blocks of a GA are presented, upon which the design choices of this thesis are based. In addition to this approach of bringing forth a part of the theory underpinning GAs, the choices made in designing the controller algorithm are presented.

5.2.1 Constituents

The main part of a genetic algorithm is the genes. Like in the biological case, the genes hold the smallest parts of (useful) information. Some authors like to define a gene as the smallest constituent of a chromosome. In that case, the internal structure of the gene is very simple; each gene may only hold a single unit (e.g. a number, an operator, or an object). In the binary case, a gene is then the equivalent of a *bit* as defined in the computer context. All by themselves they would not convey much information, but grouped into chromosomes or parts of a chromosome they hold useful information that can be decoded and interpreted in the system or process to be optimised. Typically, in a multivariate function optimisation problem, each variable could correspond to some contiguous fixed-length sequence of genes in the chromosome. Thus, a problem of n variables where each variable is represented by m_g genes would then form a chromosome consisting of $n \cdot m_g$ genes. Although this definition of a gene as the smallest block of a chromosome is convenient in some cases, it fails to capture the information about what is the smallest structure needed to represent useful information. For example, in the multivariate optimisation problem, it is apparently possible to represent each variable as a given number of elements in the chromosome, why it appears natural to define a gene such that there is a one-to-one correspondence between the variables and the genes. The trade-off is evidently that when using the latter definition the internal structure of the gene must be provided to fully specify it. In this thesis, the latter definition is used unless explicitly otherwise stated.

Unlike in human beings, a single chromosome in the GA contains all the information about the individual and the terms 'chromosome' and 'individual' are therefore used interchangeably for simplicity. An illustration of a shorter binary chromosome is presented in figure 5.1.

The GA considered here employs multiple individuals which are then collectively referred to as a population. As will be clarified as the operators are presented, employing multiple chromosomes is a prerequisite for the algorithm, but also does this open up for diversity in the population. In this context, diversity implies exploration of the search space. Exploration means that the algorithm more efficiently sweeps the search space, which in turn improves the odds of finding the global optimum.

The overall structure of these constituents is illustrated in figure 5.2.

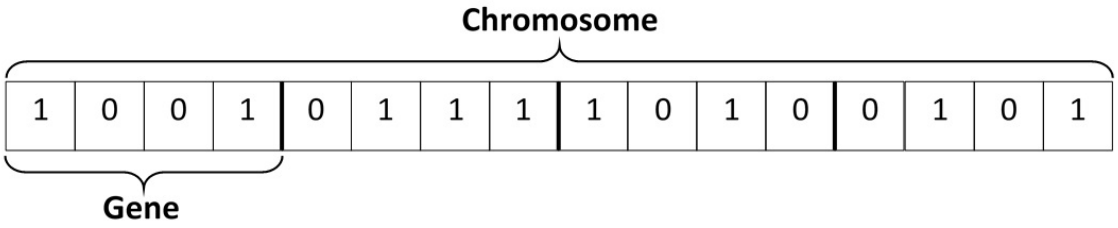

Figure 5.1: Illustration of a binary chromosome. The chromosome consists of four genes à four elements, with each element holding either a '1' or a '0'.

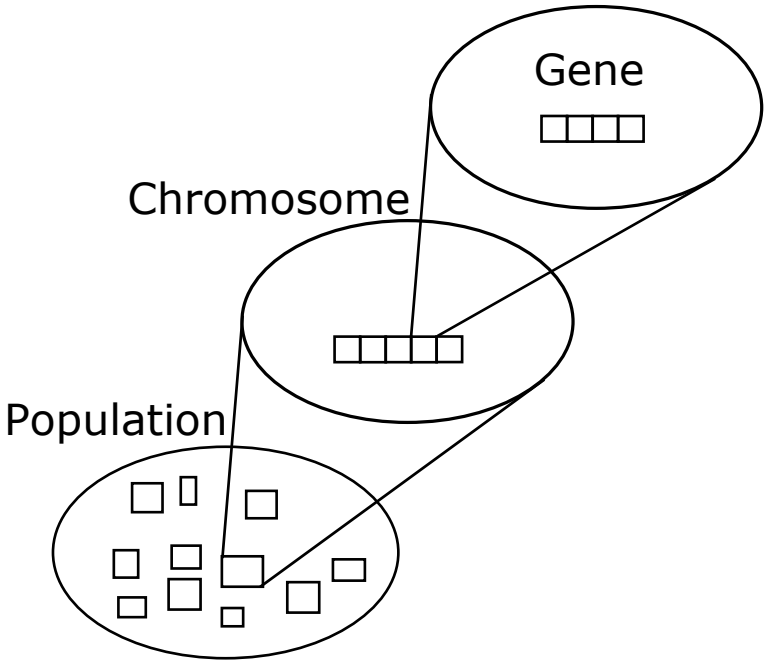

Figure 5.2: The internal structure of each individual in the population. At the lowest level there is a gene containing a number of elements that each can store an object. What kind of object is stored depends on the encoding.

5.2.2 Operators

There are many different operators that can be included in a GA. In fact, one of the difficulties in optimisation using GAs is the wide range of parameters and operators to choose from. Due to this fact, in order to successfully apply this family of algorithms it takes some thought to reduce the computational effort needed to arrive at the solution as well as improve the odds of arriving at the global optimum within the allocated time. A downside to this type of algorithms is thus that they generally do not carry over between different optimisation problems without being modified. However, the generality is simply traded for a higher level of adaptivity in the case when the algorithm is applied to the problem(s) it is designed for.

5.2.2.1 Initiation of population

In the most basic case, a population of size m_p consisting of chromosomes of length n is initiated by generating m_p strings with n random elements each. The distribution used to generate the population can be chosen in various ways, based on heuristic or mathematical ideas about the location of the optimum in the search space. Also, there is room for hybridisation (i.e. mixing optimisation methods). Given the current best solution as found by a different method, it could for example be given a spot in the initial population while the rest of the population is randomly generated. In the context of hybrid methods, the converse is also true; the best solution, as found by a GA, could be fed to a mathematical solver that might have trouble converging to the global optimum unless the initial point is sufficiently close.

As genetic algorithms mainly are inspired by both the Darwinian theory of evolution and the Mendelian concept of propagation and mixing of genetic material, the findings in [42] that the initial population and thus the initial genetic content of the population strongly influences the performance of the algorithm are indeed intuitive. As pointed out in [43], completely random initialisation does not guarantee a spread of the individuals in the solution space. In the extreme case the individuals may all be initialised in a small region, depriving the population of initial diversity. A state of low diversity is not inescapable as the algorithm family contains many stochastic operators, but typically the loss of initial diversity decreases the chances of finding the global optimum within the allocated time interval.

The problem of initial diversity is actively assessed in the algorithm developed here. As presented in [44], this may for example be done by computing the generalised Hamming distance between the individuals. However, this is both inconvenient and increases the computational complexity, why the initialisation in this thesis is done in a process simplified so as to decrease the cpu requirements. At the core, the initialisation operator relies on the assumption that the initial solver gives a solution that is not too far from the global optimum with respect to the genetic algorithm. The validity of this assumption is of course highly dependent on how different the solvers and utilised vehicle models are. As will be seen, this assumption is indeed justified by the results.

With this assumption the required initial diversity may be reduced since the main traits of the optimal solution with respect to the second solver are comparable to those generated by the pre-solver. The implication is that components of a chromosome that are very different compared to the corresponding components of the pre-solver solution are likely to be of poor quality. Therefore, instead of randomly initialising the population, the population is initialised by generating a complete population consisting solely of copies of the warm start solutions. At least one of each warm start solution is kept unaltered, while the rest undergoes the same mutation process as that used in the main loop of the algorithm. However, to impose appreciable diversity, the mutation rate is significantly higher in the initialisation phase than in the main loop.

5.2.2.2 Encoding and decoding

Encoding is the process of representing the search space of the optimisation problem in the *coding space*. Bringing this back to the biological domain, the search space may be thought of as the phenotype and the coding space as the genotype[1].

Between these two spaces, the encoding and decoding procedures act as mappings. Depending on the encoding scheme, this mapping between search space and coding space is not necessarily bijective. An example of a mapping that may not be bijective is tree encoding, as commonly used in genetic programming. Non-redundancy is generally desirable in GAs and this non-bijectivity breaks this rule-of-thumb, but the use of these kinds of mappings have been found fruitful in certain applications and are therefore still used despite this downside [46].

5.2.2.2.1 Binary encoding

One of the most widely employed encoding is the binary encoding. Binary encoding was presented in figure 5.1. If a binary encoded gene consists of n elements, it is capable of representing 2^n different values. In the case of binary encoding, the search space must be bounded somehow. In the continuous case it means that there is an upper and lower bound for each variable, while a discrete problem requires a bounded set [46]. Given a range, $[a, b]$ and using binary encoding, this range can only be divided into $2^n - 1$ intervals. The average resolution offered by binary encoding is then $(b - a)/(2^n - 1)$.

The decoding function can be chosen arbitrarily. For example, inspired by the binary system, a binary gene g_i $(i = 0, ..., n-1)$ representing the range $[a, b]$ may be decoded according to

$$x = a + \frac{\sum_{i=0}^{n-1} 2^i g_i}{2^n - 1} \cdot (b - a). \tag{5.1}$$

Evidently, the resolution can easily be controlled by choosing the length of the genes.

Some advantages of this approach are clear already at this point (e.g. exact representation of integers, easy to control resolution etc.), but it also opens up for the use of Gray Codes, amongst others [39]. However, there are also obvious downsides to binary encoding, one of which is its inherent property of encoding error, which may be reduced on the expense of increased chromosome length and thus increased search space dimensions and computational complexity.

5.2.2.2.2 Value encoding

Just like the operators, there is a vast amount of encoding schemes that can be used. In fact, the encoding schemes may be infinitely customised to suit the problem. This

[1] Phenotype is the visible traits of the individuals as caused by the genetic information, also referred to as the genotype [45].

does rather well illustrate the often needed and witnessed ingenuity in contexts of alternative and adaptive algorithms. Although very many encoding schemes fall within the field of binary encoding, these schemes are not universally applicable and, even if they are, they may not be the best suited. Suitable encoding is crucial for the success of genetic algorithms [47].

A competitor to binary encoding that manages to overcome some of the associated shortcomings is value encoding. Instead of encoding the information in binary format and employing mappings between the search space and coding space, this method involves representing something connected to the optimisation problem in real values. The most obvious reason to use value encoding is in cases where it is not possible to represent the problem in binary format or where the encoding error associated with binary encoding becomes too large. However, in addition to these fundamental reasons, Michalewicz found that the utilisation of value encoding is making its way into the domain of genetic algorithm as the main findings in [48] are that real-value encoding has the potential to outperform binary encoding.

Furthermore, it should be noted that value encoding does not mean that the genes must hold a number, but it could be any object. What might be considered a drawback of this method is that it often is necessary to tailor the operators to the specific nature of the problem [46]. This affects the generality and the typical ease-of-use, but there is a direct gain in computational speed as encoding and decoding processes often are less demanding and in the extreme case, the search space is directly represented in the coding space and no transformations are needed.

The algorithm is intended to control the engine torque output and with the aim of making the computational footprint small in the developed application, value encoding is used. As presented in figure 3.4, there are natural upper and lower bounds on the engine torque output. The bounds are functions of engine speed and are therefore difficult to directly include in the algorithm as the engine speed is highly dependent on the previous control signals sent to the powertrain. This problem is assessed by having the genes represent any torque-values but pulling outliers back inside the valid interval at evaluation time.

5.2.2.3 Evaluation

The purpose of the evaluation is to assign a *fitness value* to each individual based on their phenotype. Since GAs are deeply inspired by natural selection, the fitness value of a solution is a very central part for the progression of the algorithm. The fitness function thus has fundamental influence on the success of the algorithm. To achieve a good result, the fitness function should assign high fitness values to individuals with desirable traits while undesirable characteristics should be penalised. In view of conventional mathematical optimisation methods, this process is the equivalence of formulating the problem in mathematical terms. However, there is a fundamental difference. As many mathematical models rely on that the mathematical formulation of the problem fulfills certain criteria, GAs do only put very loose restrictions on the problem formulation. GAs do not even require the problem to be expressed

mathematically. The main point is that the evaluation could be performed in any manner as long as it enables a fitness value or rank to be attributed to each individual [39, 46].

For the purposes of longitudinal control of an HDV, numerical models are readily available and a mathematical formulation of the problem is indeed convenient. The formulation may be formed in an infinite number of ways and on forms that can be tailored to a specific problem. For the case at hand an approach of modest model complexity is chosen. Behind this lies the reasoning that generality decreases with increasing model complexity, that there is a correlation between simplicity and robustness, and that the model should not be too expensive in terms of computational complexity. In addition to these remarks, a consequence of only making small changes to the truck model used by standard solvers (e.g. quadratic programming) is that it is easier to identify any improvements that can be attributed to the developed solver.

With (at best) a correlation between engine energy output and fuel consumption, the cost function given by equation (4.5) cannot be used to find the optimal sequence of control signals with respect to fuel flow. Noting that the energy contained in the consumed fuel and the energy output are comparable and only differ by a factor typically in the range 2-3 due to engine efficiency, equation (4.5) may be modified according to

$$J = \delta_f \sum_{k=1}^{N} E_{fuel,k} + c_T \sum_{k=1}^{N} \left(T_k - \frac{1}{2} m v_{d,k}^2 \right)^2 + c_s \sum_{k=1}^{N} \left(E_{e,k} - E_{e,k-1} \right)^2 , \tag{5.2}$$

where $E_{e,k}$ has been replaced by the fuel energy content, $E_{fuel,k}$ along with a correction factor δ_f to account for the engine efficiency.

Thus, while the genetic algorithm has the potential to employ very complex cost functions, the extension of the cost function is in this case rather subtle. Importantly, however, it adds the ability to evaluate a proposed solution based on fuel consumption instead of engine output energy.

5.2.2.4 Selection

The purpose of the selection process is to select a number of individuals from the population and let them transfer their genes to the next generation. If sexual reproduction is used, individuals are typically selected pairwise and are then allowed to mate. The general approach is presented in figure 5.3.

In the strict literal sense of *survival of the fittest*, the individual with the highest fitness value would get to procreate. However, it is possible and, in general, preferred to implement schemes that do not blindly select the best individuals but also consider the individuals with lower fitness values. Thus, the fitness values may merely be used as indicators or recommendations of specific individuals. To give an idea of the magnitude of the influence of the fitness value, the term *selection pressure* is introduced. High selection pressure implies strong reliance on fitness value, while

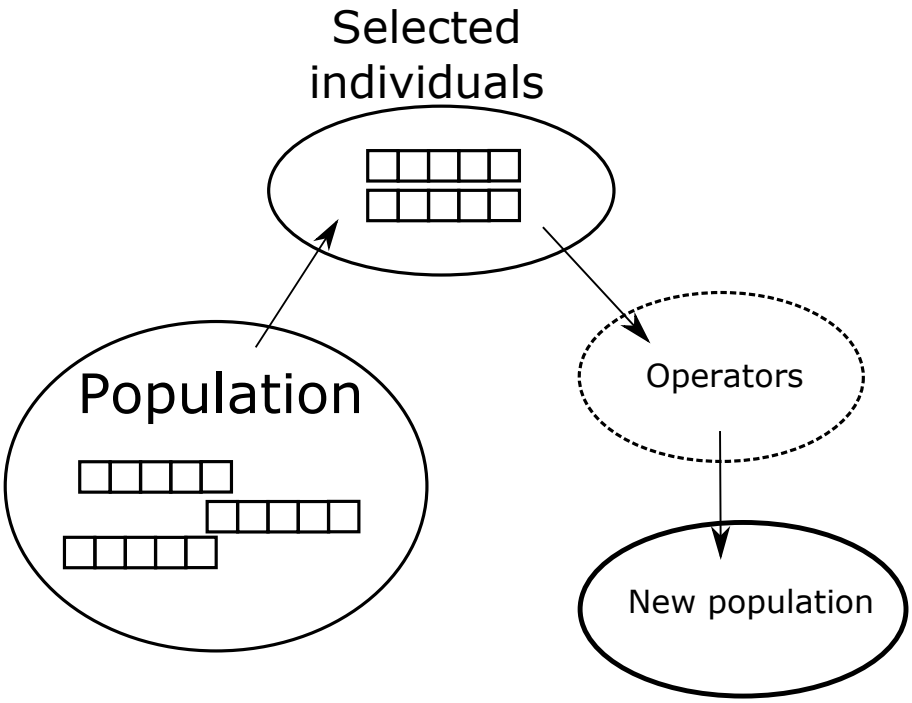

Figure 5.3: From the population, a given number of individuals are selected in the selection process. These individuals are passed on to the other operators and finally placed in the new population.

low pressure indicates a more arbitrary selection with regards to fitness. The orthogonality of high and low selection pressure leads to different characteristics; high pressure leads to faster convergence at the expense of the odds of finding the global optimum. Low pressure, on the other hand, may lead to slower convergence, but it is also associated with a better chance of finding the global optimum. Put differently: high pressure promotes exploitation while low pressure promotes exploration [46].

5.2.2.4.1 Roulette wheel selection

A simple scheme for selection is the so called *roulette wheel selection*. The name is derived from the casino game, but a more accurate name would maybe be *wheel-of-fortune selection* after the American TV show. In the standard case, the probability of an individual being selected is proportional to the individual's fitness. If individual i ($i = 1, 2, ..., m_p$) has fitness f_i, then the probability of individual j being selected is

$$p_j = \frac{f_j}{\sum_{i=1}^{m_p} f_i}, \qquad j = 1, 2, ..., m_p. \tag{5.3}$$

Since probabilities must be non-negative, the method, as stated here, requires non-negative fitness values.

To implement this version, the cumulative probability, θ_j, is used:

$$\theta_j = \frac{\sum_{i=1}^{j} f_i}{\sum_{i=1}^{m_p} f_i}, \qquad j = 1, 2, ..., m_p \tag{5.4}$$

The selection is performed by drawing a random number, $r \in [0,\ 1]$, and the selected individual is the first one fulfilling

$$\theta_j > r.$$

An example of this process is illustrated in figure 5.4.

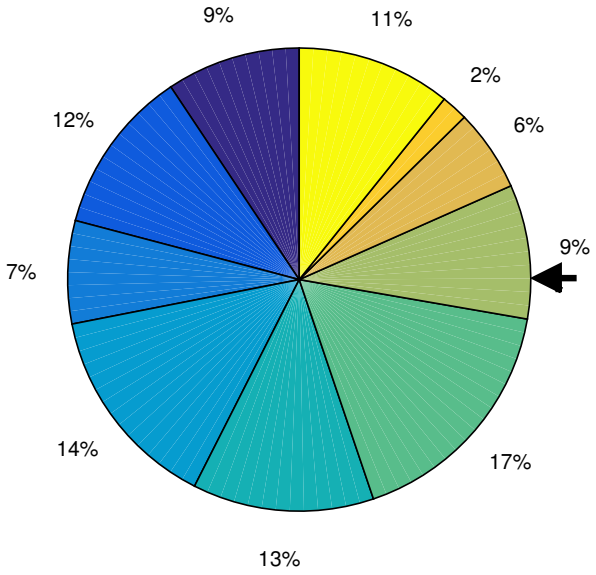

Figure 5.4: Roulette wheel selection for a population consisting of 10 individuals. If the fitness values are normalised, the illustrated case corresponds to $r = 0.25$. Counted clockwise, the fourth individual is selected.

5.2.2.4.2 Tournament selection

Along with roulette wheel selection, tournament selection is the most widely employed selection operator [39]. While roulette wheel selection is inspired by the game rather than nature, tournament selection is directly inspired by a selection process in nature. In a natural tournament, there is always a risk of various factors leading to the superior individual loosing and consequently allowing the inferior creature to transfer its genes to the next generation. This cause of diversity and, algorithmically speaking, exploration of the search space, is captured by the tournament selection operator. In this scheme two or more individuals are randomly selected from the population. Out of these individuals, the best one is selected with probability p. The process is recursively applied until an individual has been selected or only a single individual remains and thus is automatically selected.

5.2.2.4.3 Boltzmann selection

One the one hand, roulette wheel selection and tournament selection are based on intuition and nature's counterpart, respectively. On the other hand, they don't take into account the evolution of population over time and appropriately adjust the selection (i.e. their selection pressure is constant). A well-known heuristic approach to finding an optimum in a search space is to start out on a coarse scale and then successively zoom in on the interesting regions. This is a phenomenon witnessed in statistical physics rather than biology, and the work and ideas of Ludwig Boltzmann within the field of statistical physics have been a major source of inspiration [49]. The algorithm is inspired by the annealing process of solids, which involves heating a metal to specific temperature for a specified amount of time and then slowly cooling it in a controlled way [50]. Ideally, at the maximum temperature, the metal atoms are randomly located in the liquid phase. If, additionally, the cooling is sufficiently slow, the result of the annealing process is a solid in which the particles have arranged themselves in the low energy ground states [51]. The direct connection to this theory is the simulated annealing algorithm, but it also carries over to the selection process of GAs [46]. In that case, equation (5.3), the probability of selecting individual j ($j = 1, 2, ..., m_p$) with fitness f_j in roulette selection, is replaced by

$$p_j = \frac{e^{f_j/T'}}{\sum_{i=1}^{m_p} e^{f_i/T'}}, \tag{5.5}$$

where T' is the equivalence of temperature in a annealing process [39].

Equation (5.5) is merely an example of a Boltzmann inspired selection scheme. In [39] a second selection process derived from statistical physics is presented, but it is based on tournament selection instead. Yet another approach to the same kind of selection is found in [46]. The latter also proposes a logarithmically decreasing[2] temperature:

$$T' = T'_0(1-\alpha)^k, \qquad k = 1 + 100\frac{n_{gen}}{G},$$

where T'_0 is the initial temperature, n_{gen} is the current generation number, G is the maximum number of generations and α is control parameter in the interval $[0, 1]$. Although these rule of thumbs exist, experimenting is generally required for good results [39].

5.2.2.4.4 Stochastic universal sampling

In view of the performance of the genetic algorithm, [53] introduces three measures:

[2] A logarithmically decreasing function is a function whose value decreases to zero more slowly than any nonzero polynomial [52]

- *Bias* - The absolute difference between the expected value[3] and the actual value. Optimal (zero) bias is thus achieved when the selection algorithm perfectly respects the expected value.
- *spread* - The range of number of times that an individual may be selected in the selection process.
- *efficiency* - The complexity of the algorithm (e.g. time complexity).

In the situation of optimal bias and minimum spread, the actual value (number of offspring) for individual i is thus restricted to the set

$$\{\lfloor e_i \rfloor,\ \lceil e_i \rceil\},$$

where e_i is the expected value.

From this it follow that a selection algorithm should have minimal spread and zero bias, and be efficient. An algorithm with these properties is stochastic universal sampling (SUS) [54]. The efficiency is in the order of m_p, the population size.

Conceptually, the algorithm is very similar to roulette wheel selection. However, instead of repeating the selection process m_p times, all m_p individuals are selected at once and not independently. The selection process starts by normalising the fitness values to sum to 1. Next, a pointer is placed at random in the interval $[0, 1/m_p]$. Subsequent pointers are then placed a distance $1/m_p$ apart, as illustrated in figure 5.5. Intuitively this may be thought of as placing a comb with equidistant teeth in figure 5.5, where the position of the first tooth is chosen at random in the interval $[0, 1/m_p]$.

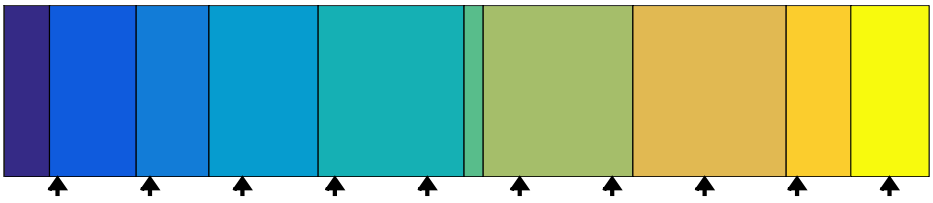

Figure 5.5: Illustration of stochastic universal sampling. The size of each segment corresponds via some predefined rule to the fitness of the corresponding individual. The total length of the segments is 1, and all pointers are therefore separated by an interval equal to $1/m_p$, where m_p in this case is 10. The first (leftmost) pointer was randomly selected in the interval $[0, 1/m_p]$ and in this case it was placed at 0.0572.

In figure 5.5 two individuals are sampled twice. Programmatically, when these individuals are extracted form the population as in figure 5.3 the simplest method is to extract them in order and thus place any multiple samplings next to each other. Depending on the implementation of the crossover process presented next, this adjacency may lead to crossover between the very same individual. The result is

[3]The expected value of an individual is defined as the average number of offspring that it should receive.

that since crossover between identical chromosomes in many schemes simply clones the parents, there is no net result. To avoid this situation, a shuffling algorithm is applied to the selected population.

5.2.2.5 Fitness transformation

In the algorithm, the direct fitness value is computed according to equation (5.2). As a result of having each gene code for the torque for a given road segment, if a single gene is altered while keeping the rest fixed, all future states of the vehicle are affected by this change. The result is that one "bad" gene can cause the whole chromosome to appear as a solution far from the optimum. This has the potential to decrease the fitness value considerably, leading to a loss of valuable information. In view of this issue, it is necessary to either lower the selection pressure directly or transform the fitness in order not to lose good solutions disguised by a set of poor genes.

Since SUS is used as sampling technique, and this method offers optimal bias and minimal spread, it is straightforward to control the expected value, e_i, and more importantly, the expected value of the elite.

Instead of normalising the fitness values as described in section 5.2.2.4.4, the fitness values can be left unchanged and the pointer interval will then be of length

$$i_p = \frac{\sum_{i=1}^{m_p} f_i}{N_s},$$

where N_s denotes the number of individuals to be selected.

The expected value of individual j is then

$$e_j = \frac{f_j}{i_p} = \frac{f_j}{\sum_{i=1}^{m_p} f_i / N_s}.$$

A common way of transforming the fitness is to employ fitness ranking, which in its basic form means that one, in a population of m_p individuals, assigns a fitness value of m_p to the best individual, $m_p - 1$ to the next best and so on. However, when employing SUS and selecting as many individuals as there are in the population, it follows that

$$e_{best} = \frac{m_p}{\sum_{i=1}^{m_p} f_i / m_p} = \frac{m_p^2}{\sum_{i=1}^{m_p} i} = \frac{m_p^2}{(m_p+1)m_p/2} = \frac{2m_p}{m_p+1}.$$

Thus,

$$\lim_{m_p \to \infty} e_{best} = 2,$$

indicating that the best individual will copied two times into the new generation in the limit as m_p tends to infinity.

To control the expected values, the algorithm instead employs two successive fitness transformations. First it applies fitness ranking to effectively decrease the selection pressure. To assess the problematic tendency of copying the best individual twice into the next generation, the second transformation takes the form

$$\hat{f}_j = f_{j,rank}^{1/7}.$$

For $m_p = 50$ and $m_p = 100$ with $N_s = 50$, this yields the graphs illustrated in figure 5.6. As can be seen from the figure, this transformation guarantees that a bit more than half of the best individuals are guaranteed to be selected (i.e. $e_j \geq 1$) if $m_p = N_s = 50$. Also, it should be noted that this approach does not completely prohibit the algorithm from carrying over two copies of the same individual to the next generation as the expected value lies between 1 and 2, but it does decrease the probability and in the obvious way it is possible to decrease this probability even further by choosing a different function for the second fitness transformation.

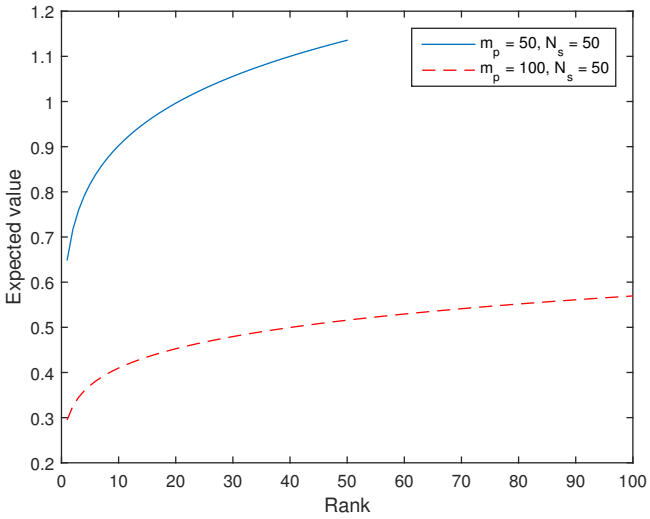

Figure 5.6: Expected value after transformation of fitness ranking values. The solid line represents the case where equally many individuals as there are in the population are to be selected, while the dashed line illustrates the case where only half of the individuals in the population are to be selected. In both cases the absolute number of individuals to be selected is the same.

5.2.2.6 Optimal crossover and mutation rates

The mutation and crossover rates are parameters whose values significantly affects the performance of a genetic algorithm [55]. Many articles are concerned with finding the optimal value for these parameters, but in general the findings rarely carry over between applications, making an algorithm relying on constant values

of these parameters fragile. A general conception is that there is no such thing as optimal rate, and in an attempt to assess this fragility and circumvent the need to explicitly set the rates the authors of [55] propose adaptive probabilities for both crossover and mutation. The proposed models are

$$p_c = \begin{cases} c_1 \frac{(f_{max} - f')}{f_{max} - \langle f \rangle}, & f' \geq \langle f \rangle \\ c_2, & f' < \langle f \rangle, \end{cases} \tag{5.6}$$

for crossover and

$$p_m = \begin{cases} c_3 \frac{(f_{max} - f)}{f_{max} - \langle f \rangle}, & f \geq \langle f \rangle \\ c_4, & f < \langle f \rangle, \end{cases} \tag{5.7}$$

for mutation. c_1, c_2, c_3 and c_4 denotes constants to be set, f_{max} and $\langle f \rangle$ denotes the maximum and average fitness values of the present generation, respectively, f' is the maximum fitness of the pair to cross and f is the fitness of the individual to mutate.

A GA is a directed search algorithm and typically the parameters p_m and p_c reflect the trade-off between the desire to have the algorithm being explorative or exploitive, ideally in that temporal order (i.e. first explore and then prioritise exploitation). To achieve this, a standard approach is to decrease the mutation and crossover rates with time (i.e. generation number). The stochasticity of the algorithm, however, makes the temporal development of the population unpredictable, which justifies the inclusion of population dependent mutation and crossover rates as in (5.6) and (5.7).

For both rates, there are default values for sub-average individuals. Now, focusing on the crossover rate, p_c, it can be seen that for pairs where the best individual has above-average fitness, the crossover rate decreases with increasing pair-wise maximum fitness and if the pair contains the best individual in the population, the rate is zero. Similarly for the mutation rate, there is a default rate for sub-average individuals, while for above-average chromosomes the rate is different and modelled by a decreasing function that goes to zero for the fittest one.

The zero-probability of the best individual being crossed prevents it from being destroyed, which is acceptable but not a requirement. However, both crossover and mutation rate must not be allowed to be zero for a single individual as this could lead to exponential growth and consequently an imminent risk of premature convergence. Based on this reasoning, the authors of [55] introduce a small default mutation rate of 0.005, acting as a minimum mutation rate for all individuals.

5.2.2.7 Crossover

Depending on the genetic algorithm, different crossover schemes must be used. The schemes that will be considered here are schemes where 2 parents give rise to 2 children and the chromosome length is preserved. Also, as the characteristics of the

crossover operation depends both on the encoding, fitness transformation, selection of individuals to cross, and the crossover operator itself amongst other, the adaptive mutation rate introduced above is dismissed for a constant crossover rate of 1. However, as the concept of competing generations is employed in the final selection of the next generation and stochastic uniform sampling is used as the mechanism for crossover selection, the effective crossover rate is less than 1. Furthermore, with the inclusion of competing generations, all genetic material from the previous generation is guaranteed to be present without any modifications when selecting individuals for the next generation. An important point to underline is that although the mutation rate is not adaptive, its characteristics are sought as a net effect in the design of the algorithm.

In the development process many crossover operators were evaluated. The most general forms of the evaluated operators are presented below, and in the final algorithm flat crossover is used as it proved to be best suited with regards to how the problem has been formulated in this thesis.

5.2.2.7.1 k-point crossover

The most fundamental crossover scheme meeting the requirements above is the k-point crossover. Recalling that each chromosome consists of n genes, a chromosome can be split at $n-1$ locations. The algorithm starts by drawing k unique random integers representing the crossover points. The two parent chromosomes are then split at these locations. Then every other segment is swapped, mixing the genes of the two parents. In many applications k is set to 1 or 2 and as found in [56], when compared to both uniform, flat and 2-point crossover, the 1-point crossover outperformed the others in the job shop scheduling problem. The job shop scheduling problem is clearly different from the problem of longitudinal vehicle control, but the results in [56] indicate that by increasing the number of crossover points, valuable schemas[4] may be destroyed and consequently make the algorithm perform worse.

5.2.2.7.2 Uniform crossover

k-point crossover is applicable for many different encoding schemes but, for value encoding, uniform or flat crossover are normally employed [57]. Much like k-point crossover, uniform crossover performs crossover on two individuals by traversing the chromosomes and swapping corresponding constituents between the individuals. However, uniform crossover differs in that each pair of corresponding genes of the two individuals are swapped with a certain probability. The result is that segments of varying length are swapped between the chromosomes, but unlike k-point crossover there is no predefined number of segments that are swapped. Instead the number of swapped segments is in the range $[0, n]$, where the cross ratio (i.e. the probability of swapping two genes) can be used to bias the number of swapped segments in either direction.

[4] A schema is a subsequence of a chromosome.

5.2.2.7.3 Flat crossover

Both k-point crossover and uniform crossover are underpinned by the theory of how genes are passed on from parents to offspring in humans amongst others. Formally speaking, the above crossover operators are concerned with the genotype of the parents and the generated offspring. Recalling the parallel drawn between the genotype and coding space, and phenotype and solution space, it can be said that the flat crossover operator targets the phenotype in cases where no encoding is used. Flat crossover can also be applied to encoded chromosomes, but in that case one should instead talk about genotype superposition as it does not operate directly on the phenotype.

In its simplest form, flat crossover generates the content, commonly referred to as allele, of gene number j in 2 children ($c1$ and $c2$) from 2 parents ($p1$ and $p2$) according to

$$g_j^{c1} = r_j g_j^{p1} + (1 - r_j) g_j^{p2},$$
$$g_j^{c2} = r_j g_j^{p2} + (1 - r_j) g_j^{p1},$$

where r_j is a random number in the range $[0, 1]$.

5.2.2.8 Mutation

As for this thesis, the dynamic mutation probability is considered of relevance, but even more so is the reasoning underpinning it. Specifically, the effective mutation effects should decrease as the evolution progresses. Evidently, since both the crossover and mutation operation potentially changes the fitness of the individuals, the fitness values must be updated more frequently at the expense of the time complexity of the algorithm. In view of this, the adaptive rate in equation (5.7) is rejected in favour of a combination of non-uniform mutation described next, and competing generations described in section 5.2.2.10. On top of this, a constant mutation rate is used. The rate is chosen in accordance with the optimal value derived in appendix B.

In spite of the remark earlier made how the stochasticity of the algorithm makes predictions of the state of the population at a given temporal point, a mutation operator based on temporal information is used. The operator, which is a modified version of the non-uniform mutation operator, is given by

$$g_i' = \begin{cases} g_i + f(t, l_{range}), & R = 1, \\ g_i - f(t, l_{range}), & R = 0, \end{cases} \tag{5.8}$$

where l_{range} is the absolute value of the maximum range that a gene can creep away from its current value in the mutation process and R is a random value drawn from the set $\{0, 1\}$. The function $f()$ is defined as

$$f(t, l_{range}) = l_{range}(1 - r^{(1 - \frac{n_{gen}}{G})^b}). \tag{5.9}$$

In the above equation r is drawn from the standard uniform distribution, n_{gen} denotes the current generation number, G is the maximum number of generations, and b is a parameter that has been set to 2 in this thesis.

In the search of optimal parameter values, the studies are often carried through with the genetic algorithm as the only solver. Consequently, this requires the algorithm to be well-tuned throughout the evolution process. With a vast search space and an interest in keeping its computational footprint low the algorithm is not suited to act as the only solver. Instead it is assumed that the GA will have access to a very qualified first guess of the optimum. The major consequence is that the algorithm should be exploitative rather than explorative. However, it should be noted that it is fundamentally required that some tendencies of exploration are kept.

By reducing the need for random exploration to "map out" the characteristics of the search space, the trade-off between exploration and exploitation encoded in the mutation rate may be biased to favour the exploitation. By letting l_{range} be some small value, typically some low multiple of 10, the mutations are effectively creep mutations drawn from an ever-narrowing distribution. An intuitive description of the underlying idea is that the initial and assumed qualified guess of the optimal control sequence is represented by a rubber band in the solution space. The purpose of the mutation is to pull each part of the rubber band towards the points in search space that will make the solution more optimal. Initially the algorithm can make large adjustments, but as the algorithm progresses the purpose of the mutations moves towards being to fine-tune the chromosomes.

5.2.2.9 Replacement

A general method of replacement is to replace the whole generation at once by deleting the old population and letting the offspring take its place. Another method is steady-state replacement, which involves replacing only a fraction of the population in each evaluation cycle [39]. One advantage of steady-state replacement is that, as often is the case in nature, the offspring is allowed to compete with the older generations. While the operators of a genetic algorithm rarely guarantee improvement of the operands, a desirable effect of this kind of selection is that poor individuals are given the chance of securing their place in the next generation while good individuals from the previous generation may also get to live on. In this thesis, a replacement operator somewhere in between these two is used. More specifically, the algorithm uses full replacement but with inter-generation competition, as explained next.

5.2.2.10 Competing generations

Genetic algorithms are stochastic inherently cannot guarantee that the overall fitness of a generation is equal or better than the previous generations. As for the maximum fitness there is the elitism operator, but it does not care about the generation as a whole. In nature there is often an overlap between generations, making it reasonable to introduce the concept of inter-generation competition. In terms

of genetic algorithms this means that in the final selection process of forming the next generation, both the proposed new individuals and the previous generation are allowed to compete. The net population to select individuals from is consequently doubled in size, effectively turning the solid graph in figure 5.6 into the dashed line. From this graph it should be noted that there is no longer a guarantee that even the best individual will be selected, making it necessary to include elitism.

5.2.2.11 Elitism

Thus far the stochasticity of the algorithm has been heavily emphasised as a fundamental property. It is indeed one of the most fundamental properties of the algorithm, but it also has the potential to disrupt the population in various ways. One such way is that it may destroy the best individual. A safeguard is to always keep the fittest individual in the population simply by ensuring that a copy of it is always transferred to the next generation. It should be noted that whenever the elitism operator is employed, there is a risk of the fittest individual taking over the entire population in case the collective effect of the operators is to promote the fittest individual very strongly. If promoted too strongly, chances are that the fittest individual is copied into the next generation multiple times, causing exponential growth.

5.2.3 The final algorithm

Through continuous reasoning and testing, the most suited operators of those presented above have been combined into the final algorithm. The main flow of the algorithm is presented in figure 5.7.

Also, so as to intuitively illustrate the operation of a genetic algorithm, a simplified case encompassing a 2-variable function optimisation is presented in figure 5.8.

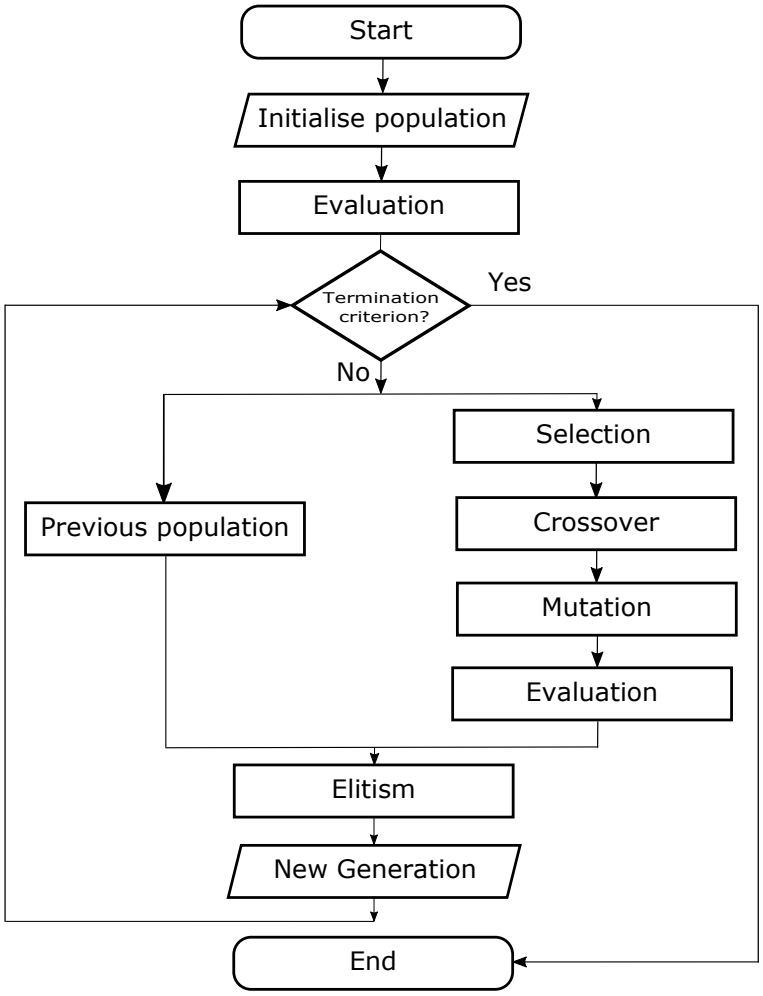

Figure 5.7: A flowchart illustrating the structure of the developed genetic algorithm.

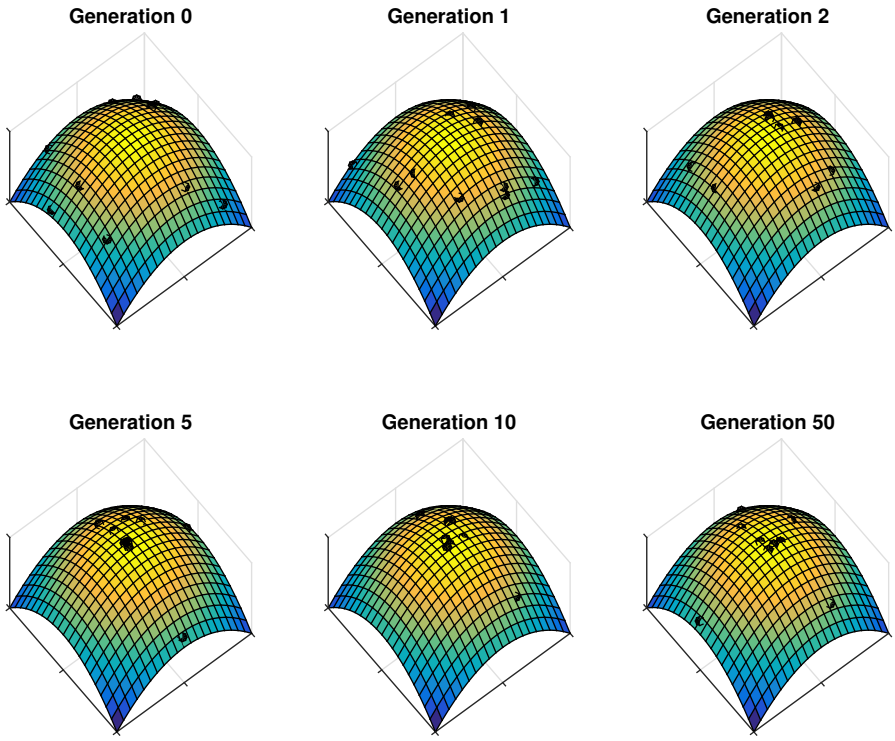

Figure 5.8: An example of the evolution of chromosomes in a simple function optimisation in two variables. The population encompasses 10 individual and each gene consists of 10 binary elements. The fitness function is defined as the function value and at generation 50 the maximum fitness had been achieved. The evolution towards higher fitness values is evident, but also should it be noted that although a near-optimal solution is found already in generation 5, diversity can still be observed in generation 50.

6

Hybrid algorithm

Since the gear cannot be controlled, it cannot be included in the optimisation problem on the form accepted by the above MPC algorithm. In general, the constraints corresponding to the gear selection are hard to incorporate in mathematical optimisation algorithms. Therefore, if there is a way to mix the determinism of LP/QP with the soft-computing advantages associated with GAs, an algorithm that is superior[1] to both algorithms alone can emerge.

6.1 Genetic algorithm with warm start

An important property of a real-time control algorithm is that it should always provide a feasible solution within some specified time, should it not manage to find the optimal one. While stochasticity is a fundamental property for the success of GAs, it also makes the algorithm unpredictable in the sense that it does not guarantee convergence within a given time window. The following sections are aimed at presenting a way of evading this problem and how information contained in previous solutions may be reused in order to improve the results without affecting the computational load.

6.1.1 Pre-solving and non-deterioration

A straight-forward solution to the problem of failure to converge within a specified time window is to apply a fast and deterministic solver to a simplified version of the problem. In view of the formulation of an MPC problem, equation (4.1), the cost function $f()$ can be chosen arbitrarily. Similarly, the system model $f_s()$ may also be chosen arbitrarily, but of course the choice directly affects the quality of the predictions made by the solver. A common way of simplifying a vehicle model is through linearisation techniques. Although model simplifications may be crucial and have been successfully applied to various applications, it must be emphasised that the exclusion of non-linearities in general will generate suboptimal control strategies that, depending on the context, may or may not be acceptable [58]. In the context of this thesis, the aim of this fast and deterministic solver is not to find the optimal

[1] Of course, superiority is highly context/application dependent!

control strategy, but to output a solution resembling the optimal strategy, serving as an initial guess for a second, more advanced solver.

In general, the stochasticity of the genetic algorithm introduces a risk of loosing useful information as the evolution progresses and the carriers of that information die out or the information is corrupted by the mutation and crossover operators. Thus, unless deliberately handled, the maximum fitness in a population can display a sudden drop. Previously explained, the elitism operator is a way to get around this problem, ensuring that an unmodified version of the fittest individual is always passed on to the next generation. Therefore, if elitism is employed, the maximum fitness of the population is a non-decreasing function in generation number. That is, even in the worst case scenario where the GA fails to find a solution with a better fitness value, the control signal that will be sent to the system is optimal with respect to the simplified formulation of the pre-solver, given that there exists a feasible solution.

6.1.2 Reusing previous solution information

In addition to warm starting with a different solver, the algorithm can be extended so that the most useful information emerging from previous evolutionary efforts remains in the population. Importantly, this implies that only during the very first iteration in a controller session a full evolution from the pre-solver solution to the optimal one is guaranteed to be required. As for subsequent iterations, if no assumption is made about how much the optimal solution changes between time steps, the only thing that can be said is that *at worst*[2] the algorithm will start over from the pre-solver solution and be forced to carry out a full evolution again.

However, as the algorithm is intended to run continuously with a look-ahead horizon of 50-100 steps, only a very small fraction of the road will change between adjacent steps[3]. Also, the state of the vehicle will not change much from one time step to the next during normal operation. Together these two observations implies that in most cases the optimisation problems for two adjacent time steps will be very similar, typically leading up to similar solutions for the problem at hand. This reasoning points in the direction that if previously found solutions are reused, the initial guesses of the GA could have potential to be very close to the actual optimal solution. Consequently, the overall quality of solutions would improve over time, but also would the algorithm converge more quickly on average, which brings the algorithm closer to real-time operation under hardware restrictions. If the number of iterations is kept constant, there is hence potential to successively increase the quality of the solutions. Based on the above points, an illustration of the resulting

[2] *worst* refers to the maximum difference between the fitness value of the GA- and pre-solver optimal solutions. It does not consider the distance between points in search space, given any metric.

[3] In the algorithm developed in this thesis, a variable sample length is employed. However, the way the sample length is chosen, it will not change significantly between adjacent time steps during normal cruising. Thus, the difference in road topography data between adjacent steps will in reality be only a few percent.

controller structure is presented in figure 6.1.

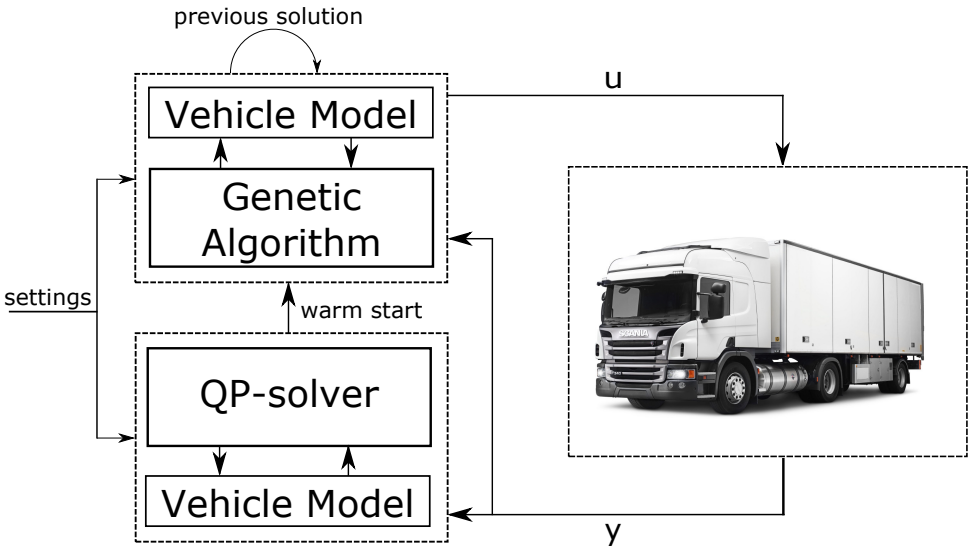

Figure 6.1: The structure of the developed controller. Although the genetic algorithm utilises a refined vehicle model as compared to the QP-solver, the models share the same parameters wherever applicable.

7

Algorithm evaluation

As extensive effort has been put into investigating and benchmarking various cruise controllers aimed at decreasing the fuel consumption of HDVs, there are lots of data available for comparison. However, in order to make a fair comparison the testing conditions should be as similar as possible between tests. In reality this is not achievable on actual roads, rendering it illogical to compare results collected at different occasions. Although real-world tests cannot be replaced, the above note favours testing through simulations. Based on this argument, the developed algorithm has been assessed through simulation.

7.1 Simulation model

For the purpose of this thesis, a simulation model was developed in Simulink. A simplified scheme of this model is illustrated in figure 7.1. To improve the simulation results, some parts of the model are based on real data collected from tests involving the vehicle being simulated. In the figure the main components are included to illustrate the primary characteristics of the simulation. However, in the actual model the subsystems are highly interconnected and depend on a set of state parameters. Furthermore, as to not clutter the scheme with interconnections, only the ones required to emphasise the functionality of the model are included.

As can be seen from figure 7.1, the model consists of a main block that contains the powertrain, brakes and the physical model of the vehicle. This block is essentially an abstraction of the simulated vehicle. The internal combustion engine accepts a torque request from the controller developed in this thesis and computes the actual torque that appears at the clutch or torque converter, depending on transmission type.

The transmission includes a slightly different gear selection logic than that available to the genetic algorithm since it is of interest to test the performance of the algorithm under conditions where the actual gear selection software cannot be used in the prediction model for various reasons (e.g. it may be too heavy, too complex or not even available). Although a direct advantage of GAs is that more complex models can be employed, it is reasonable to argue that infinite model precision cannot be achieved for most systems, if not all, making it a necessity to be able to handle the arising discrepancies. By deliberately introducing differences between the

prediction and simulation models, this reasoning offers justification for the simplified gear selection logic.

The powertrain of an HDV is very complex and sophisticated controllers have been developed to control the various parts. In some of the previous studies involving look-ahead control of HDVs where the engine torque (indirectly) was one of the optimisation variables, the actual control of the engine was routed via either the standard cruise controller or even an interface to the driver (see for example [5, 37]). A major reason for this is that the look-ahead controller did not have to take engine oscillations and other undesired effects into account. The obvious drawback is the decreased ability to control the engine torque output with precision. The developed controller contains logic to enforce smooth driving, but it does not take into account the finer characteristics of the engine and powertrain. Despite this fact and the remark preceding it, the simulation model is based on direct control of the engine torque from the controller.

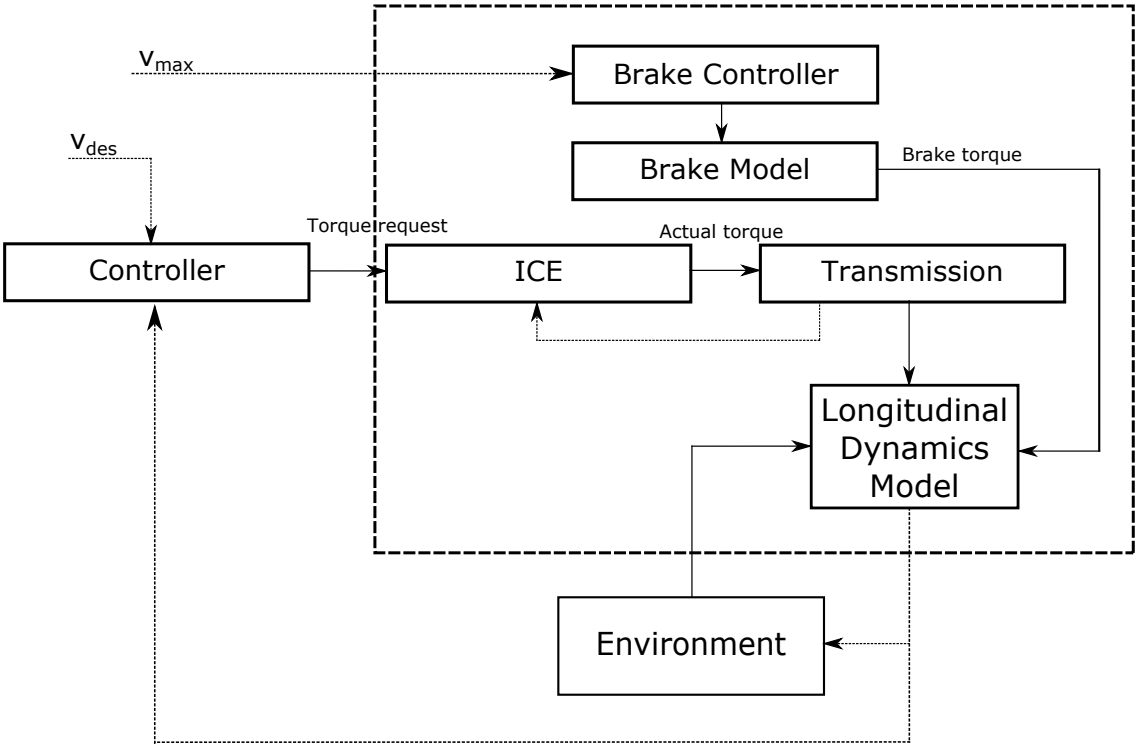

Figure 7.1: Simplified scheme of the simulation model developed for testing the algorithm. Solid lines indicate either a requested or actual torque, whereas dashed lines indicate interconnections of particular importance. v_{max} and v_{des} represent dynamic reference speeds and are externally provided to the controllers (e.g. from a driver).

8

Results

To generate the results presented below, the simulation model from section 7.1 was used. As the final algorithm relied on two different solvers, the output from the initial QP-solver alone was first considered. After that, the complete algorithm was assessed and compared to the QP-solver. The assessment was done with respect to short and representative road segments, long simulations with real road data, time, and algorithm predictability. The parameters and constants used in the simulations are presented in appendix C.

8.1 Evaluation of QP-solver

As described, the full version of the algorithm takes advantage of two different solvers by selectively applying them to the problem at different stages, taking advantage of their different strengths. Despite the fact that the initial QP-solver have been referred to as ”pre-solver”, it does in general output a solution that is optimal with respect to its prediction model. As no fair comparison can be made between simulation results and real-world tests, the most important part of the evaluation of the algorithm is to investigate the gain of applying the second solver (i.e. the GA). To do so, this section is dedicated to investigating the control signal as proposed by the QP-solver alone.

8.1.1 QP-solver performance for constant driving

Constant slope is equivalent to flat road with an additional constant force acting on the vehicle, and it is thus only necessary to consider a flat road in the case of constant driving. At each sample point, the QP-solver presents the predicted optimal control strategy and the predicted speed of the vehicle for the prediction horizon. For the case when both the initial and desired speed is 72 km/h, the predicted torque and speed profiles are those presented in figure 8.1

The solver displays the desired main traits of smoothness and unbiased reference speed tracking under static driving, but also should it be noted that there is the undesired torque and speed drop towards the end of the prediction interval. This drop is not a property of the QP-solver, but a direct consequence of the formulation of the cost function in equation (4.5). For the sake of comparison with literature,

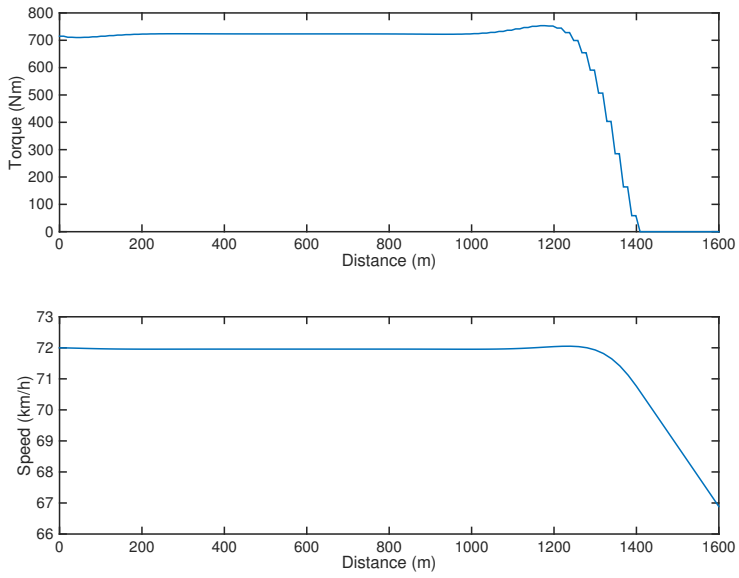

Figure 8.1: The torque and speed profile of the HDV for the case with $v_0 = 72\,\text{km/h}$. What should be noted is that the algorithm manages to keep the vehicle at a nearly constant speed very similar to the desired velocity. Furthermore, there is a noteworthy drop in torque, and thus speed, towards the end of the prediction horizon.

this undesired trait has *not* been rectified in the following results.

Figure 8.1 does not illustrate the trajectories actually taken by the vehicle, but merely the predicted engine torque output that minimises the cost function given by equation (4.5). The actual speed and torque trajectories are presented in figure 8.2 which displays a slightly different behaviour than the prediction as well as no drop in speed and torque towards the end of the travelled interval.

In conclusion, the simulated characteristics of the QP-solver are close to the prediction. Also, the figure displays clear tendencies of compensating for prediction- and simulation model differences as the algorithm initially increases the torque to compensate for the speed drop and then keeps torque and speed essentially constant.

8.1.2 QP-solver performance for varying road slope

In this section two more road profiles are considered to investigate the performance of the QP-solver alone. These road profiles are a crest (figure 8.3) and a dip (figure 8.4), where the latter has been generated by reflecting the crest in the horizontal axis. These two types of roads are chosen as they illustrate the main characteristics of the solutions output by the algorithm.

Figure 8.3 displays the characteristic behaviour of anticipatory driving; the vehicle increases its speed ahead of a demanding ascent that it will not be able to climb without loosing momentum. To save time it accelerates to the set speed as it arrives

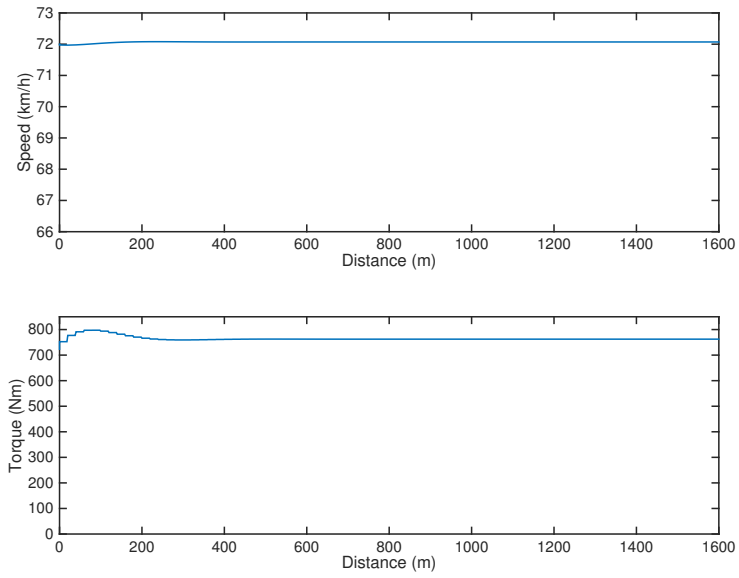

Figure 8.2: Actual speed- and torque trajectories. Still, the solver displays a smooth behaviour and good reference speed tracking. Initially there is a small unforeseen drop in the speed and after compensating for the lost speed a steady state torque is found.

at the top of the hill. Approximately halfway into the flat plateau, the speed is reduced as the algorithm predicts a large increase in speed when arriving at the descent. As the engine brake is not native to the QP-solver, it first enters what is known as eco-roll mode[1] and a bit into the descent the engine brake is engaged by transforming the requested foot brake torque to engine brake torque in the after-treatment of the solution output by the solver.

In figure 8.4 essentially the opposite situation as in figure 8.3 is presented. Initially the vehicle requests a mix of no torque and negative torque and remains in these modes a bit into the flat segment while it approaches the reference speed. Identifying the upcoming ascent the torque is then increased, triggering a downshift a few hundred meters ahead of the foot of the hill.

Essentially the algorithm behaves much the same way as should be expected based on engineering heuristics. However, it makes use of high-resolution control signals to very precisely control and predict the trajectory of the vehicle.

[1]Disengaging the engine, rolling on neutral gear.

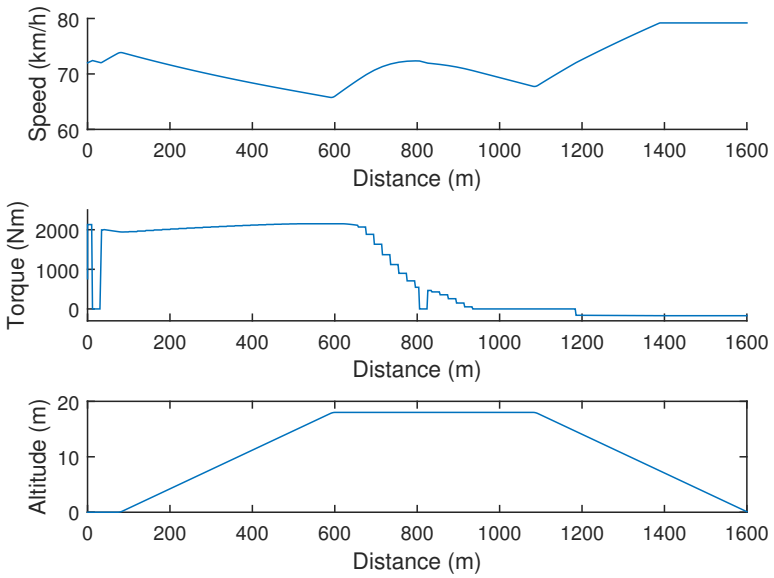

Figure 8.3: Simulated torque- and speed trajectories for a crest. The sudden drops in torque are due to gear shifts, which in this simulation are two. $v_{des} = 72\,\mathrm{km/h}$.

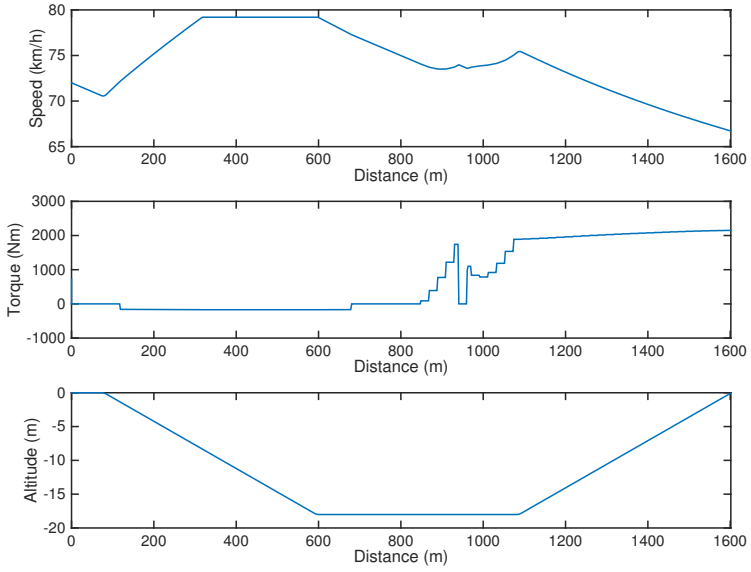

Figure 8.4: Simulated torque- and speed trajectories for a dip. The sudden drop in torque is due to gear shifts. $v_{des} = 72\,\mathrm{km/h}$.

8.2 Performance of hybrid algorithm

The performance of the algorithm must be judged both with respect to its ability to lower the fuel consumption, but also its local behaviour as it faces non-constant road slopes. The algorithm is assessed both by evaluating the local performance in the situations presented above, but also by using real road data in longer simulations.

8.2.1 Hybrid algorithm torque trajectory

The torque sequence predicted by the QP-solver alone was presented in the top panel of figure 8.1. Feeding this solution to the developed GA where the population size and number of generations have been set to 20 and 300, respectively, results in the solid line in figure 8.5. For convenience, the QP-solver's output is also included as a dashed line. The prediction is for perfectly flat ground and a preview horizon of 1600 m with sample points uniformly distributed. What should be noticed is that after applying the GA the resulting solution displays slow oscillations with an amplitude of less than 10 Nm. The oscillations are smooth and due to their small amplitude they would not be felt by the driver. Oscillations are generally undesirable, but as will be seen in the following sections, dynamic torque saves fuel even on flat ground as compared to when only the QP-controller is used. Thus, these modest oscillations are caused by the controller having information about how the working point of the engine affects the fuel consumption and actively taking this information into account when planning the trajectory.

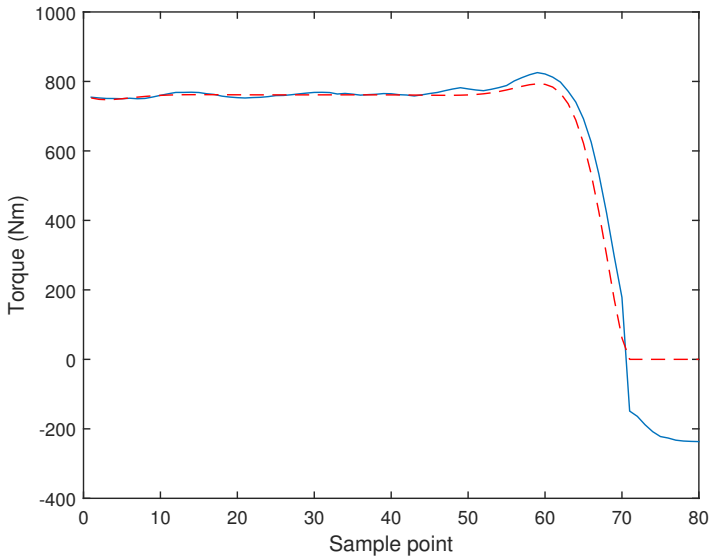

Figure 8.5: The optimal torque as predicted by the hybrid algorithm shown in solid. The warm start solution supplied by the QP-solver is shown as a dashed line.

8.2.2 Analysis of the behaviour of the hybrid algorithm for constant and varying road slope

For the sake of comparison, the three previously studied situations are presented (i.e. flat road, a crest, and a dip). All parameters are the same and the only addition to the algorithm is the inclusion of the GA on top of the QP-solver.

In figure 8.6 the case with flat ground is presented. The top panel describing the vehicle speed shows no noticeable deviations away from the desired speed. Furthermore, the inclusion of the GA has lead to the disappearance of the initial and very slight drop in speed witnessed for the QP-controller. Unlike the speed, the simulated torque output from the engine is non-constant. This dynamic behaviour is in view of figure 8.2 accredited to the genetic algorithm, indicating that in the case of static driving on flat ground, the algorithm does not enter a steady state in the strict meaning of the word. However, the torque variations are very small and happening very slowly, making them unnoticeable to the driver.

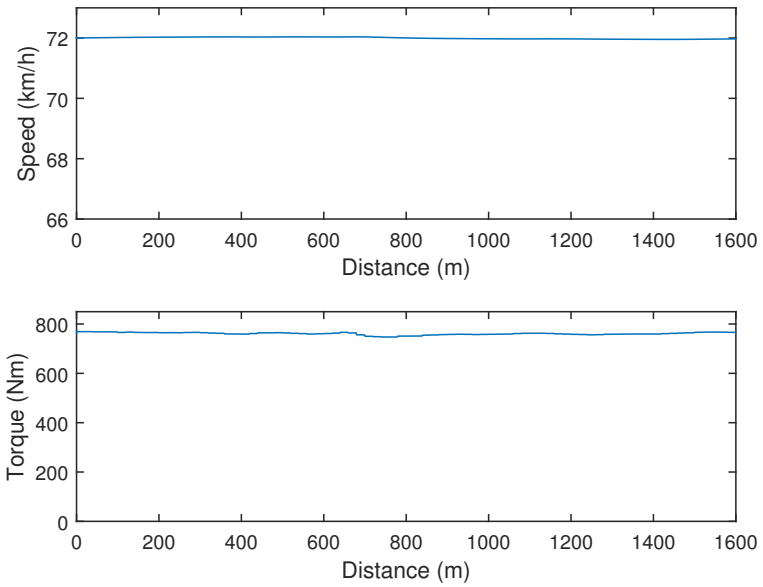

Figure 8.6: Simulated torque and vehicle speed when using the hybrid algorithm. From the bottom panel it is clear that the torque is dynamic and does not enter a steady state, unlike the simulation with the QP-solver on flat ground shown in figure 8.2. However, the weight of the truck and the small relative amplitude causes the speed to appear constant.

When faced with a non-constant slope as in figure 8.7 it can be seen how the vehicle first accelerates to enter the ascent with a kinetic energy reserve. The travelled hill is too steep and long for the engine to be able to maintain the set speed and the vehicle arrives at the crest with a somewhat lower speed. On the plateau the vehicle initially speeds up to attain set speed, but a bit before the downhill it smoothly reduces the torque, triggering the gearshift to happen a bit earlier than for the QP-controller, and applies the engine brake to save fuel. In contrast to the QP-solver,

the engine brake is native to the GA and the engine brake fully replaces the eco-roll mode proposed by the QP-solver for the same trip. The engine is then being fully dragged the rest of the way and after the maximum allowed speed (according to the algorithm) is reached, it is kept constant by applying the foot brake, which is not visualised. The net result is that no fuel is consumed during the second half of the interval.

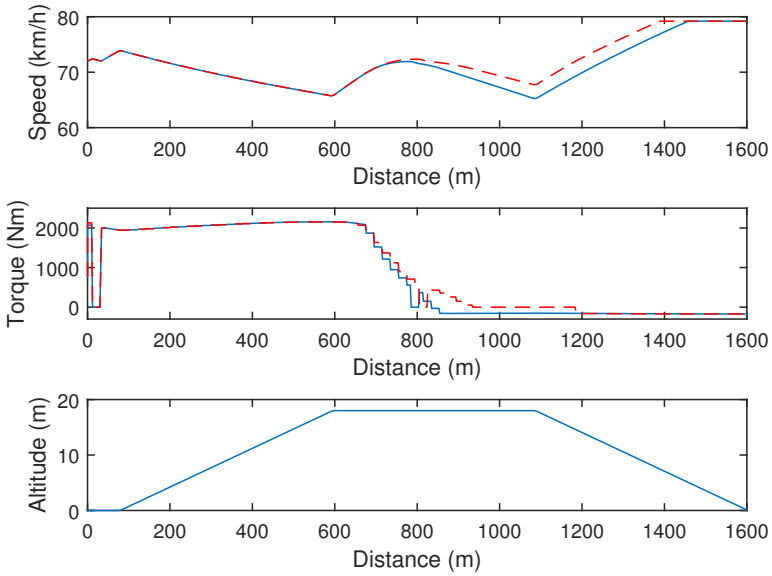

Figure 8.7: Simulated result of applying the hybrid algorithm are shown in solid. Figure 8.3 is superimposed as dashed lines. As the simulation starts only 100 m before the demanding ascent, the best strategy is to give full throttle, but as the hill has been climbed the two algorithms chooses different strategies. $v_{des} = 72\,\mathrm{km/h}$.

In a comparison between figure 8.8 and the corresponding figure 8.4 illustrating the case when only the QP-solver is employed, it should be noted that although the speed trajectories are very similar, the engine torque requested by the two versions of the algorithm differ. A noteworthy difference is that the GA manages to postpone the gear shift without any means of controlling the gear shifting logic, indicating that the inclusion of gear prediction affects the final behaviour of the vehicle and endows the algorithm with extended control capabilities, although only indirect ones. In the simulation this postponement is achieved by a more modest torque increase than the corresponding increase requested by the QP-algorithm, the trade-off being a marginal decrease in average speed.

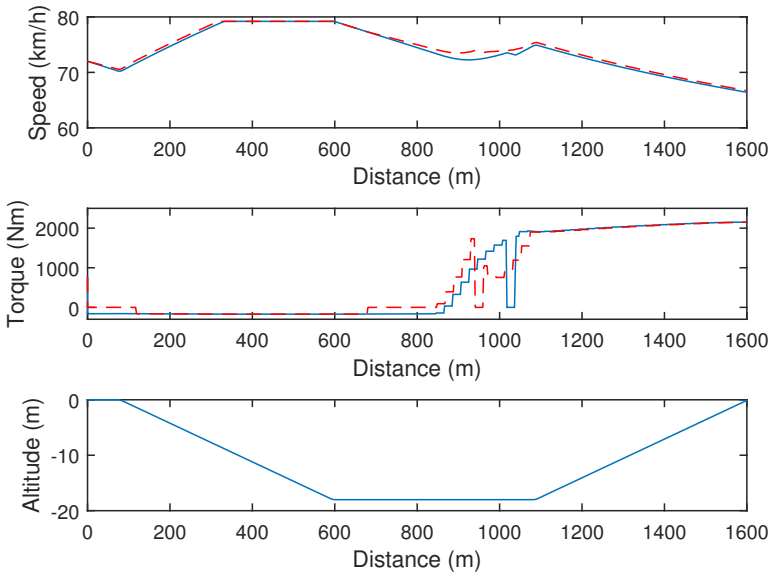

Figure 8.8: Simulated behaviour for the case when the hybrid algorithm is faced with a significant dip in the road profile. The solid line represents the hybrid algorithm while the corresponding output from the QP-solver has been superimposed as dashed lines. $v_{des} = 72\,\mathrm{km/h}$.

8.3 Numerical comparison for short-distance performance

To be successful in handling real driving situations, it is of great importance that the algorithm can handle the representative segments presented above well. There are many factors that determine the performance of the algorithm, some of which are driver comfort, fuel consumption, and travel time. The driver comfort has been addressed directly in the algorithm, but this section is exclusively concerned with the fuel consumption and travel time. To this end table 8.1 presents the fuel consumption and travel time for both algorithms faced with the above road topographies. For the stochastic GA, the simulations have been run 10 times each and then averages have been computed.

Table 8.1: Simulation results for the QP-solver (QP) and the hybrid algorithm (GA) for the three road profiles. Data is given as <fuel_consumption[l/100km]>/<average_speed[km/h]>.

Road profile	QP	GA	Difference (%)
Flat	31.7808/71.91	31.7647/71.89	-0.05/-0.023
Crest	47.3880/71.66	45.2624/70.99	-5.69/-1.21
Dip	39.0651/74.01	37.4764/73.45	-4.29/-0.43

For all three topographies the GA displays a reduced fuel consumption, ranging

from 0.05% to 5.69%, when compared to the QP-solver. In terms of travel time, the GA is a little bit slower on all segments. The biggest difference is for the crest, where the hybrid controller favours fuel savings over travel time particularly much.

The numerical values indicates that the addition of the GA saves fuel, but it should be clearly emphasised that these values are merely indicators as they are generated from artificial road segments and short travel distances. Furthermore, the results do not hold any information about whether the GA can reduce the fuel consumption more than toady's Active Prediction.

8.4 Large scale evaluation

In addition to evaluating the behaviour of the algorithms when faced with a specific local road topography, its overall performance over long distances was assessed. This evaluation was done by using recorded road data for the highway connecting Södertälje and Norrköping, measuring approximately 100km.

In figure 8.9 the complete simulations are shown in terms of speed, engine torque and altitude. As for the engine torque, the solutions are very similar. Both algorithms offer smooth torque changes to ensure driver comfort, but the different cost functions and prediction models used for the two optimisation algorithms have caused the GA-solution to deviate from the QP-solution used as starting point. In turn this has shifted some of the gear shifts, identified by the sudden drops in engine torque. The numerical values from the simulations are presented in table 8.2. From the table it is evident that the GA-controller saves fuel as compared to when only the QP-controller is used. As regards the speed, the table indicates that the mean speed is lower for the GA-controller than the QP-solver. But while this is true, it must be noted that the desired speed is set to 72 km/h and that the average speed of the GA-controller therefore is closer to the desired speed.

Table 8.2: Simulation results for the QP-solver (QP) and the hybrid algorithm (GA) for the Södertälje-Norrköping segment. The total distance simulated is 100km and the average is formed by simulating the trip 5 times.

Algorithm	Fuel consumption (l/100km)	Average speed (km/h)
QP	34.4974	72.39
GA	34.4404	72.11
Difference (%)	-1.63	-0.392

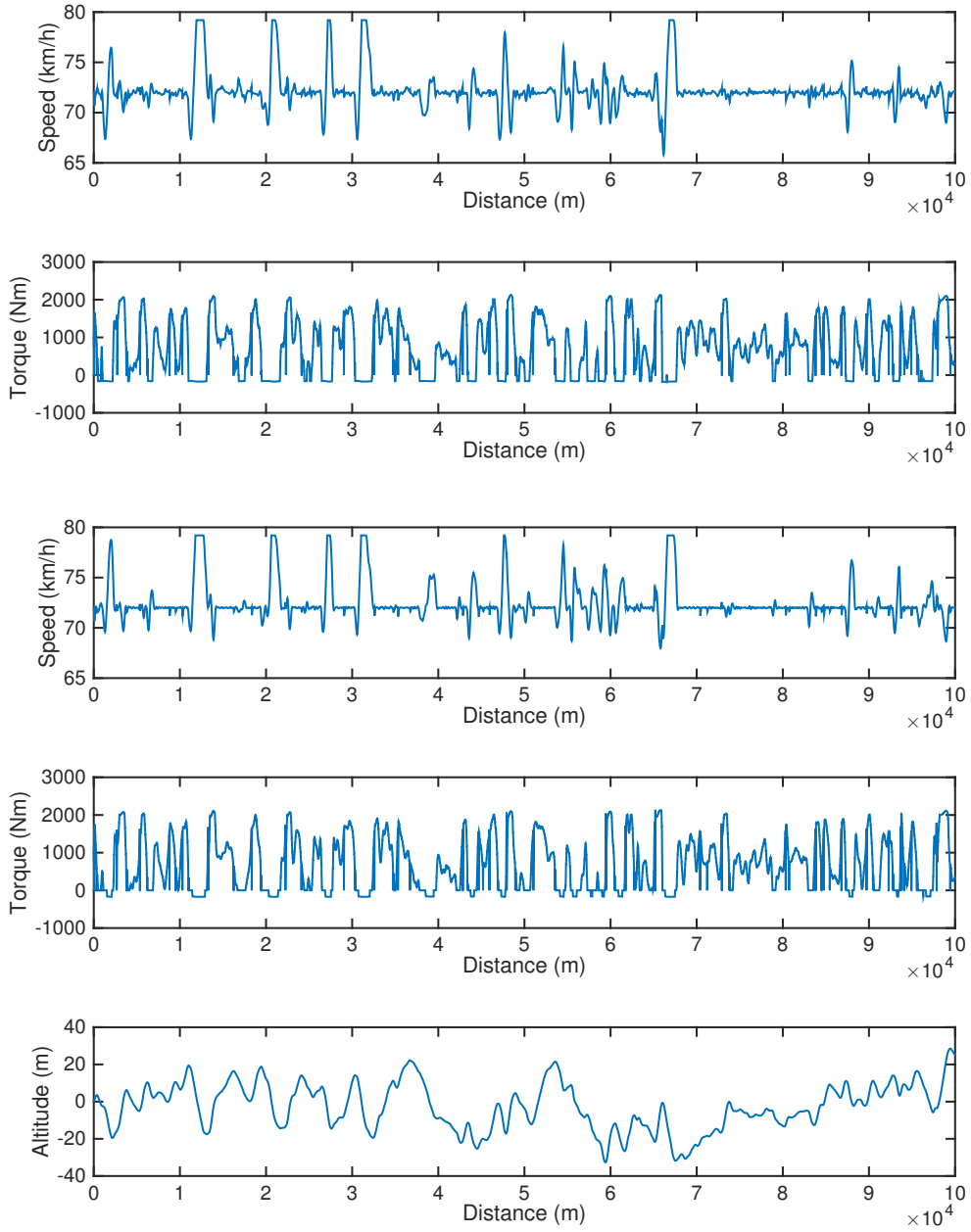

Figure 8.9: *Panel 1 and 2 (from the top)*: Simulation result for the Södertälje-Norrköping segment when using GA. There are clear deviations away from the set speed in the vicinity of demanding ascents/descents, but under more static conditions the vehicle closely tracks the reference speed. *Panel 3 and 4*: Simulation result for the Södertälje-Norrköping segment when using the QP-solver alone. Comparing panel 1 and 3, it can be seen that the strategies are similar but still notably different. While the GA effectively utilises the engine brake and engine efficiency information as well as a gear-prediction model, the QP shares the main traits with the GA but lacks the high-resolution finesse exhibited by the GA. *Panel 5*: Altitude of the travelled road.

8.5 Average performance of genetic algorithm

As a measure of the performance of the algorithm, it is also evaluated based on the spread of the results in fuel-speed-space. To increase the number of data points, the simulation was run 60 times, but only on the first 25 kilometers of the Södertälje-Norrköping segment. The results are presented in figure 8.10. The *averaged* average speed of the hybrid algorithm is 0.35% lower than that of the QP-controller, while the average fuel consumption is lowered by 1.73%. Evidently, the cluster generated from simulation with the genetic algorithm is different from the simulation with only the QP-solver active, both in terms of trip time and fuel consumption. This makes it more difficult to determine the exact effect of the inclusion of the GA on fuel consumption or average speed alone. However, while the desired speed for either one of the algorithms could have been adjusted in order to enforce similar average speeds, this option was discarded in favour of having the two algorithms use the same set of values for the parameters shared between them.

Furthermore, while the average speed is lower for the hybrid solver, it should again be noted that the utilisation of the hybrid solver leads to average speeds closer to the desired speed, but with all values slightly exceeding it, thus guaranteeing that no time is lost with respect to travelling at the set speed. From the scatter plot it should also be noted that the data from the GA-controller exhibits traits of predictability as the corresponding data points form a dense group with low variance.

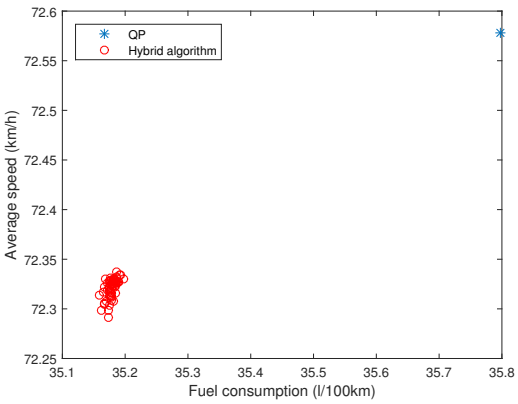

Figure 8.10: Scatter plot of the fuel consumption and average speed for the two algorithms running on the first 25 kilometers of the Södertälje-Norrköping segment.

8.6 Computational footprint

The major computational footprint is that of the genetic algorithm but also the QP-solver, which is not supported by MATLAB Coder and thus executed as an ordinary MATLAB function call, adds to the computational time. The computational time was measured with MATLAB's built-in `tic-toc` function on a computer with an Intel Core i7 (3.60GHz) processor. The graph in figure 8.11 represents the execution

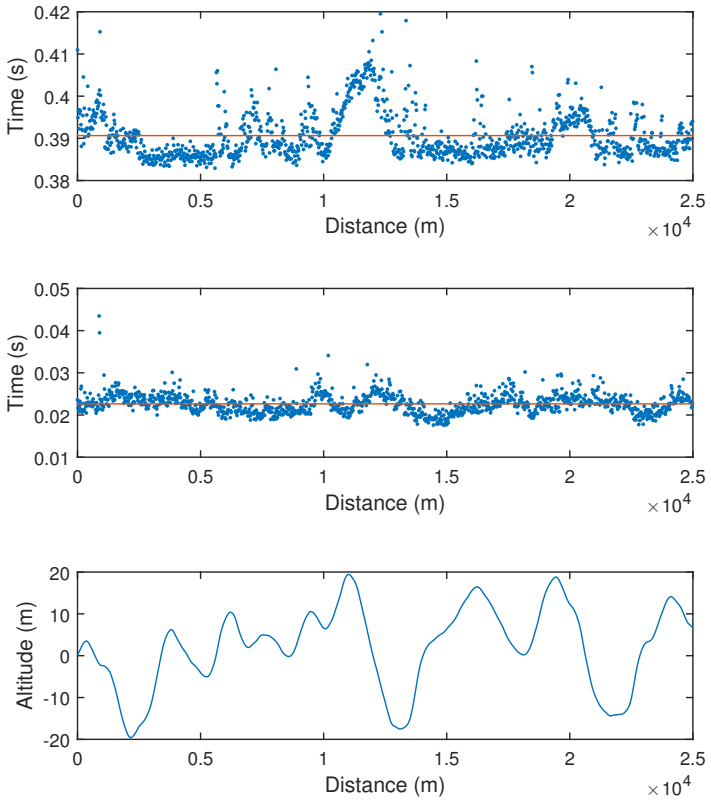

Figure 8.11: *Top panel*: Execution time of the GA for the first 25 kilometers on the Södertälje-Norrtälje segment. *Middle panel*: Execution time of the QP-solver for the first 25 kilometers on the Södertälje-Norrtälje segment. The red lines indicate the mean computation time. *Bottom panel*: Altitude of travelled road. Included to emphasise the computational times' dependence on topography.

time for the hybrid as well as QP-algorithm when simulated on the first 25km of the Södertälje-Norrköping segment. From the figures it can be seen that the execution times for both solvers have small local variances which indicates predictable computational time. Furthermore, it is evident that the average run-time of the GA is approximately 20 times as long as that of the QP-solver.

The small local variance of the computational times helps emphasise the computational times' dependence on local road topography. From the inclusion of the road topography in the bottom panel it becomes clear that the computational time of the GA increases with up to approximately 5%, or 0.02s, in the vicinity of the steepest descent. For the QP-solver, the relative increase is in the range 25%, but due to the shorter execution times the absolute difference is less than that of the GA.

9

Discussion

In this thesis there was the direct goal of developing an algorithm capable of decreasing the fuel consumption while ensuring driver comfort and without changing the trip time considerably. From the simulation results it is clear that the addition of the GA improves the fuel efficiency as compared to applying the QP-solver alone with maintained driver comfort. In figure 8.7 and 8.8, on the other hand, an additional and very important trait is manifested. The developed controller cannot control the gear directly, but nevertheless it is seen from the figures that it managed to influence the gear shifts (i.e. both postpone and move shifts forward as compared to the gear shifts observed for when only the QP-controller was employed). The algorithm achieved this with the only instructions being to reduce fuel consumption, drive smoothly and stay in the vicinity of the set speed; that is, without no instructions of trying to control the gear. Although being able to indirectly control the gear does not generalise to most vehicle control problems, the mere observation of this behaviour implies that the algorithm is able to draw conclusions that have not been included in the algorithm design. Importantly this characteristic endows the algorithm with an ability that loosely may be referred to as a kind of reasoning.

9.1 Decoupling of cost function, prediction model and solver

A result of what was described as *reasoning* in the previous section was an algorithm that required less strict definitions of the optimisation problem. For the algorithm to work, it only required access to a fitness function and it would stochastically work its way towards the optimum in a directed search. A direct gain of this was that the focus of the design process was moved away from *how* the goal should be reached to *what* the goal should be. This is important since the cost function, prediction model and solution method in general are highly coupled in conventional mathematical optimisation and the responsibility of managing this coupling and matching the problem formulation to the solver falls on the developers. Although the cost functions of the QP-solver and GA were intentionally made very similar, the above mentioned decoupling was clearly observed and taken advantage of.

9.2 Computational footprint

From the very beginning of the project GAs were known to be computationally demanding and subject to an inherent risk of premature convergence or failing to convergence. These characteristics were all observed in the development process and carefully taken into consideration and consciously addressed. However, as the algorithm shows signs of stochasticity in the final sequence of control signals, it must be concluded that either there are more than one global optima or that the algorithm fails to find the global optimum. Observations about the values of the fitness function during operation implies the latter, but it should be noted that this does not necessarily contradict the prior. It should also be taken into consideration that very few generations and individuals were used in relation to the size of the search space. Also, the vehicle prediction model is indeed a simplification and even if the global optimum with respect to this model were to be found, it would not make sense to claim that it is the true optimum.

Most effort was put into the task of adapting the algorithm to vehicle control. However, throughout the design of the algorithm there was a permeating thought of keeping the computational footprint low. Quite contradictory, but as a consequence of their fast prototyping and extensive simulation capabilities, MATLAB and Simulink were used as main tools, and the MATLAB coder was extensively used to improve the execution performance. Despite significant improvement in terms of computational speed, very much overhead is added by the coder, which makes it difficult to use the computational resources to the fullest. The main computer was equipped with a powerful Intel Core i7 processor rated at 3.60GHz, but the algorithm could also run effortlessly on a laptop with a 2.4GHz Intel Core i5. In the algorithm's current form, however, it is deemed too demanding for on-board operation. Despite this and in conjunction with the fact that the process of really optimising the algorithm was not given a part in the project, no statement about the suitability for on-board operation can be made.

9.3 Applicability to vehicle control

Disregarding the computational complexity, the algorithm shows potential to be used in look-ahead control. While maintaining all the main traits of the previously evaluated QP-solver (see [37, 38]), it manages to reduce the fuel consumption in the simulations even further. As found in [37], the fuel consumption for a 28000kg truck was lowered by 8.1% compared to a standard cruise controller; a number that thus possibly could be increased a bit more with the addition of the developed algorithm.

As predictive cruise controllers are not the only longitudinal control systems that require (or will require) on-board optimisation procedures, it is of relevance to assess this algorithm's applicability even for other systems. In general it is a difficult task to include constraints in a genetic algorithm, which makes the quite constraint-free predictive cruise controller well suited for prototyping. The indicators are that

it would be difficult for the current algorithm to handle a search space of higher dimension and greater complexity than that in this thesis, provided that nothing else is changed. This, of course, should be considered in conjunction with the assumption of very limited computational resources.

Admittedly, a great deal of the focus has been on the on-line optimisation capabilities of the algorithm. Simulations cannot replace actual in-vehicle tests and although the obtained results showed that the addition of the GA improved the performance of the look-ahead controller, it should not uncritically be taken as evidence of success for vehicle control. However, it does imply that the application of the GA was indeed successful, given the developed simulation model.

10

Conclusion

From the simulation results it is evident that the developed genetic algorithm leads to improved fuel efficiency without notably altering the trip time when compared to a conventional mathematical optimisation algorithm. A principal conclusion is that, given the simulation model, the hybrid genetic algorithm is an improvement over the QP-solver alone. As regards real-world implementation it can only be said that the algorithm displays potential of being successfully applied to real-time vehicle control. This conclusion can neither be rejected nor confirmed as regards the computational resources available on board Scania vehicles. What can be confirmed, however, is that the implementation as it is done in this thesis is too heavy for on-board real-time operation, but it must be emphasised that the current implementation offers much potential for optimisation.

Furthermore, with multiple objectives (i.e. smooth driving, fuel efficiency, and reference speed tracking) the algorithm was assessed from multiple perspectives. From the fuel efficiency point of view, the proposed algorithm is an improvement over the conventional QP-solver, even on flat ground where the QP-controller outputs a steady torque and closely follows the reference speed. With only a very slight decrease in fuel consumption on perfectly flat ground, it is concluded that the real gain of adding the GA-layer to the look-ahead controller is observed in dynamic slope situations. Although the end result is that the controller tends to increase the speed ahead of ascents and conversely decrease the speed before upcoming descents, very much like a skilled driver would, the inclusion of the algorithm in the loop introduces the crucial difference of being able to optimise the realisation of these strategies with high-resolution control signals. This behaviour was indeed displayed by the QP-solver alone. However, from the results of this thesis, it is concluded that there is potential for further improvements in terms of fuel consumption, as compared to the QP-solver, by introducing empirically collected engine data and a simplified gear-prediction model along with the genetic algorithm.

11

Future work

In this section a number of suggestions of future work are presented. The suggestions are both considered with potential improvements of the algorithm developed in this thesis and continuations of the conceptual idea of using a genetic algorithm for vehicle control.

11.1 Improving execution speed

- Addressing the current application and revisiting the execution times presented in figure 8.11, there is a set of measures available to decrease the computational burden, both in terms of computational time as a result of using a different language, and by rewriting the actual methods without changing their behaviour. A promising continuation would be to eliminate the overhead introduced by the MATLAB Coder and port the code to pure C/C++. This also means that the code can be written so as to utilise the embedded system in the optimal way.
- Using the built-in code profiler in MATLAB it was confirmed that the evaluation function accounts for far more of the execution time than all other functions. As the individuals are evaluated independently of each other the evaluation function offers great potential for improvement through parallelisation.
- A complementary approach to improving the execution speed is to reduce the actual complexity of the algorithm. As presented in the introductory chapters of this thesis, this has been addressed from many perspectives, one of which is to approximate the evaluation function. As the need to do so increases with increasing function complexity, the approximation method must in general be able to capture non-linearities and other complex traits of the evaluation function. The full story falls outside the scope of this thesis, but suffice to say that artificial neural networks constitute a group of methods that meet these needs and are widely applied today.

11.2 Improving and extending the algorithm

- As outlined in the beginning, investigating the applicability of the genetic algorithm to vehicle control was the main focus in this thesis. The idea of look-ahead control is nothing new, and under the assumption that there is no interference from surrounding traffic, the optimisation problem is simplified as the number of constraints significantly decreases and thus also the complexity of the problem. The results do imply that the addition of the GA improved fuel consumption, but to harvest more of the outlined potential more algorithmically demanding situations should be considered in further studies. It can not be said for sure, but there are indicators in this thesis that in order to apply the algorithm to a problem of greater complexity than LACC with sparse traffic the problem formulation should be revised so as to not increase the search space dimension above the current 80 dimensions.
- Real value encoding was chosen over binary encoding partly because it significantly reduces the chromosome length. However, with 80 values per chromosome, the search space is still of significant dimensionality with respect to the number of individuals and number of generations used in this thesis. Viewing the developed controller as a path planner in torque space, it is reasonable to borrow ideas from the field of pure path planning. For driver comfort it was claimed that the torque should display smooth transitions. This in turn opens up for the use of primitives[1] that code for the torque output over a longer distance than that of a single segment in the current algorithm. An example of how this could be done is to generate a set of primitives offline. In the algorithm, instead of coding for a single torque value, each gene codes for the type of primitive as well as its ”amplitude”. The final torque trajectory is then formed by placing the primitives one after the other.
- As the dynamics of HDVs are slow, the long look-ahead is crucial to even have the potential to optimise the trajectory, no matter the quality of the algorithm. Typically the error of the predicted state of the vehicle increases the further it is into the future due to error accumulation. This contradicts the use of high-resolution data at the far end of the prediction horizon. Also, the most important constraints in a vehicle-control problem of this type are likely to be local (e.g. avoiding other vehicles or driving as close as possible to the vehicle ahead). As a consequence, using high-resolution data and variables for the whole prediction horizon leads to increased computational demand for no gain. Originating from this observation, a promising approach is to develop methods that only optimise the control sequence as far into the future as is meaningful and approximate the cost associated with the far end of the look-ahead horizon.

[1] Primitives are the smallest parts that a solution (i.e. torque trajectory) consists of.

Bibliography

[1] Scania CV AB. *A forward step for platooning*. Apr. 2015. URL: http://www.scania.com/group/en/a-forward-step-for-platooning/.

[2] Scania CV AB. *Automatic driving in traffic jam situations*. Pressroom: Intelligent Vehicles. Oct. 2013. URL: http://scania.com/Images/Automatic_driving_in_traffic_jam_situations_tcm40-396486.pdf.

[3] Charles Desjardins and Brahim Chiab-draa. "Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach". In: *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS* 12.4 (2011), pp. 1248–1260.

[4] Assad Alam. "Fuel-Efficient Heavy-Duty Vehicle Platooning". PhD thesis. Royal Institute of Technology, 2011.

[5] Erik Hellström. "Look-ahead Control for Heavy vehicles". PhD thesis. Linköping University, 2010.

[6] Assad Alam, Jonas Mårtensson, and Karl H. Johansson. "Look-Ahead Cruise Control for Heavy Duty Vehicle Platooning". In: Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems. 2011.

[7] Scania CV AB. *Scania Active Prediction*. URL: http://www.scania.com/Images/P11Z01EN%20Active%20prediction_tcm40-285940.pdf.

[8] Nirwan Ansari and Edwin Hou. *Computational Intelligence for Optimization*. Springer US, 1997. DOI: 10.1007/978-1-4615-6331-0.

[9] *An Evolution Based Path Planning Algorithm for Autonomous Motion of UAV Through Uncertain environments*. Digital Avionics Systems Conference, 2002. Proceedings. 2002.

[10] Fangguo He, Huan Qi, and Qiong Fan. "An Evolutionary Algorithm for the Multi-objective Shortest Path Problem". In: *International Science Index, Mathematical and Computational Sciences* 1.1 (2007).

[11] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2012.

[12] Sinisa Durekovic and Nic Smith. "Architectures of Map-Supported ADAS". In: *IEEE Intelligent Vehicles Symposium*. 2011.

[13] E. Stanley Lee. "The history and development of dynamic programming". In: *control systems magazine* 4.4 (1984).

[14] Marko Čepin. *Assessment of Power System Reliability*. London: Springer, 2011. DOI: 0.1007/978-0-85729-688-7_17.

[15] S. P. Bradley, A. C. Hax, and T. L. Magnanti. *Applied Mathematical Programming*. Addison-Wesley, 1977. ISBN: 0-201-00464-x.

[16] Rui Wang and Srdjan M. Lukic. *Dynamic Programming Technique in Hybrid Electric Vehicle Optimization*.

[17] Donald E. Knuth and Michael F. Plass. "Breaking Paragraphs into Lines". In: *Software – Practice and Experience* 11.11 (1981).

[18] Scott Brooker and Seamus Hogan. *A method for Inferring Batting Conditions in ODI Cricket from Historical Data*. 2011.

[19] Erik Hellström. "Explicit use of road topography for model predictive cruise control in heavy trucks". PhD thesis. Linköpings universitet, 2005.

[20] George Jithin Babu. "Look-Ahead Platooning Through Guided Dynamic Programming". PhD thesis. Royal Institute of Technology, 2012.

[21] Melanie Mitchell. "Genetic Algorithms: An Overview". In: *Complexity* 1.1 (1995).

[22] L. Shi and K. Rasheed. "Computational Intelligence in Expensive Optimization Problems". In: ed. by Yoel Tenne and Chi-Keong Goh. Vol. 2. Berlin: Springer-Verlag, 2010. Chap. A survey of Fitness Approximation Methods in Evolutionary Algorithms. ISBN: 978-3-642-10701-6.

[23] Jinguo Liu et al. "Path Planning of a Snake-like Robot Based on Serpenoid Curve and Genetic Algorithms". In: *World Congress on Intelligent Control and Automation*. IEEE, June 2004.

[24] Harold "Bud" Lawson. *A Journey Through the Systems Landscape*. College Publications, 2010. ISBN: 978-1848900103.

[25] Zbigniew Michalewicz, Cezary Z. Janikow, and Jacek B. Krawczyk. "A Modified Genetic Algorithm for Optimal Control Problems". In: *Computers Math. Applic.* 23.12 (1992).

[26] Ahmed Elshamli, Hussein A. Abdullah, and Shawki Areibi. "Genetic Algorithm for Dynamic Path Planning". In: *IEEE* (May 2004).

[27] Carlos A. Coello Coello, Gary B. Lamont, and David A. van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd ed. New York: Springer, 2007.

[28] Martin Dendaluce et al. "Microcontroller Implementation of a Multi Objective Genetic Algorithm for Real-Time Intelligent Control". In: *Advances in Intelligent Systems and Computing*. 2014.

[29] Yunus A. Çengel and John M. Cimbala. *Fluid Mechanics Fundamentals and Applications*. 3rd ed. McGraw-Hill Higher Education, 2013. ISBN: 9780073380322.

[30] Tony Sandberg. "Heavy Truck Modeling for Fuel Consumptions Simulations and Measurements". PhD thesis. Linköping University, 2001.

[31] J. Y. Wong. *Theory of ground vehicles*. 2nd ed. John Wiley & Sons, 1993. ISBN: 0-471-52496-4.

[32] Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer-Verlag London, 2007. DOI: 10.1007/978-0-85729-398-5.

[33] Luigi del Re et al. *Automotive Model Predictive Control*. Springer-Verlag Berlin Heidelberg, 2010. DOI: 10.1007/978-1-84996-071-7.

[34] Hans Joachim Ferreau. "An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control". PhD thesis. Ruprecht-Karls-Universität, 2006.

[35] Tor A. Johansen. "Approximate explicit receding horizon control of constrained nonlinear systems". In: *Automata* 40.2 (Feb. 2004).

[36] Seyed Mostafa Safdarnejada and John D. Hedengren. "Initialization strategies for optimization of dynamic systems". In: *Computers and chemical engineering* 78 (July 2015), pp. 39–50.

[37] Michael Henzler, Michael Buchholz, and Klaus Dietmayer. "Online Velocity Trajectory Planning for Manual Energy Efficient Driving of Heavy Duty Vehicles Using Model Predictive Control". In: *International Conference on Intelligent Transportation Systems*. IEEE, Oct. 2014.

[38] Michael Henzler, Michael Buchholz, and Klaus Dietmayer. "Optimal Parameter Selection of a Model Predictive Control Algorithm for Energy Efficient Driving of heavy Duty Vehicles". In: *Intelligent Vehicles Symposium*. IEEE, June 2015.

[39] Mattias Wahde. *Biologically Inspired Optimization Methods - An Introduction*. WIT Press, 2008.

[40] James A. R. Marshall, John M. McNamara, and Alasdair I. Houston. "The state of Darwinian Theory". In: *Behavioral Ecology and Sociobiology* 65.3 (2010), pp. 417–420. DOI: http://dx.doi.org/10.1007/s00265-010-1121-y.

[41] National Human Genome Research Institute. *A Brief Guide to Genomics*. Online. URL: https://www.genome.gov/18016863 (visited on 12/04/2015).

[42] A. M. S. Zalzala and P. J. Fleming. *Genetic Algorithms in Engineering Systems*. The Institution of Engineering and Technology, 1991. ISBN: 9780852969021.

[43] Wei Gao. "Advances in Information Technology and Industry Applications". In: ed. by Dehuai Zeng. Berlin: Springer-Verlag, 2012. Chap. Study on New Improved Hybrid Genetic Algorithm. ISBN: 978-3-642-26001-8.

[44] Wei Gao. "An Improved Fast-convergent Genetic Algorithm". In: *International Conference on Robotics, Intelligent Systems and Signal Processing*. IEEE, Oct. 2003.

[45] U.S. National Library of Medicine. *Phenotype*. Online. URL: http://ghr.nlm.nih.gov/glossary=phenotype (visited on 12/07/2015).

[46] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer-Verlag, Berlin, 2008.

[47] Shaifali Aggarwal, Richa Garg, and Puneet Goswami. "A Review Paper on Different Encoding Schemes used in Genetic Algorithms". In: *International Journal on Advanced Research in Computer Science and Software Engineering* 4.1 (Jan. 2014).

[48] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, 1996. ISBN: 978-3-662-03315-9.

[49] Jos Uffinc. *Boltzmann's Work in Statistical Physics*. 2014. URL: http://plato.stanford.edu/entries/statphys-Boltzmann/#1.3.

[50] Jonathan Law and Richard Rennie. *A Dictionary of Physics*. 7th ed. Oxford University Press, 2015.

[51] P. J. M. van Laarhoven and E. H. L Aarts. *Simulated Annealing: Theory and Applications*. Springer-Science+ Business Media, B. V., 1987.

[52] Eric W. Weisstein. *Logarithmically Decreasing Function*. URL: http://mathworld.wolfram.com/LogarithmicallyDecreasingFunction.html (visited on 12/11/2015).

[53] James E. Baker. "Reducing Bias and Inefficiency in the Selection Algorithm". In: *Proc. of the 2nd Intl Conf on GA*. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 1987, pp. 14–21. ISBN: 0-8058-0158-8.

[54] K. F. Man, K. S. Tang, and S. Kwong. *Genetic algorithms: concepts and design*. Springer-Verlag London Berlin Heidelberg, 2001. DOI: 10.1007/978-1-4471-0577-0.

[55] M. Srinivas and L. M. Patnaik. "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms". In: *IIIE Transationcs on Systems, Man and Cybernetics* 24.4 (Apr. 1994).

[56] Jorge Magalhães-Mendes. "A Comparative Study of Crossover Operators for Geneteic Algorithms to Solve the Job Shop Scheduling Problem". In: *WSEAS TRANSACTIONS on COMPUTERS* 12.4 (Apr. 2013).

[57] R.R Sharapov. "Vision Systems: Segmentation and Pattern Recognition". In: ed. by Goro Obinata and Ashish Dutta. InTech, 2007. Chap. Genetic Algorithms: Basic Ideas, Variants and Analysis, pp. 407–422. ISBN: 978-3-902613-05-9.

[58] Maria Ivarsson et al. *Vehicle Control Using Preview Information*.

[59] C. W. Barnes. "Euler's constant and e". In: *American Mathematical Monthly* 96.7 (1984).

[60] C. Pomerance. *Math 105 notes*. URL: https://math.dartmouth.edu/~m105f13/m105f13notes1.pdf (visited on 01/02/2016).

A

Propulsion modelling

The equations governing the acceleration of the vehicle were in section 3.3 shown to be

$$J_e\dot{\omega}_e = \tau_e - \tau_{out}, \tag{A.1}$$

$$i(G) = i_g i_a, \tag{A.2}$$

$$\omega_w = \frac{\omega_e}{i(G)}, \tag{A.3}$$

$$J_d\dot{\omega}_w = \eta i(G)\tau_{out} - F_w R_w - \tau_b, \tag{A.4}$$

$$ma = F_w - F_d - F_r - F_g, \tag{A.5}$$

and

$$a = R_w\dot{\omega}_w. \tag{A.6}$$

Solving for $\dot{\omega}_w$ in (A.4) and inserting the result in (A.6) gives

$$a = \frac{R_w}{J_d}(\eta i(G)\tau_{out} - F_w R_w - \tau_b) \tag{A.7}$$

From (A.1) it follows that

$$\tau_{out} = \tau_e - J_e\dot{\omega}_e = \tau_e - J_e\dot{\omega}_w i(g) = \tau_e - \frac{a}{R_w}J_e i(G), \tag{A.8}$$

where the second and third equalities follow from (A.3) and (A.6), respectively. Inserting (A.5) and (A.8) in (A.7) yields

$$a = \frac{R_w}{J_d}[\eta i(G)(\tau_e - \frac{a}{R_w}J_e i(G)) - \tau_b - R_w(ma + F_d + F_r + F_g)].$$

Rearranging this equation results in the final expression:

$$a = \frac{R_w}{J_d + mR_w^2 + J_e \eta i^2(G)} \left[i(G)\eta\tau_e - \tau_b - R_w(F_d + F_r + F_g) \right]. \tag{A.9}$$

B

Genetic algorithm

It is an ample task to analyse complex GAs, and they may evidently be extensively tailored to suit the optimisation problem at hand, eliminating any hopes to derive general proofs. To this end, this appendix introduces a set of assumptions that trades the applicability to the algorithms used in this project for a manageable analysis. Thus, the following proofs are intended to serve as approximate indicators for algorithm settings and performance.

B.1 Run-time estimation of simplified GA

Consider a population consisting of a single binary chromosome. The reproduction is strictly limited to mutation and elitism is employed (i.e. the evolution of the fitness value is non-decreasing). Let n denote the number of genes, out of which m are 0s. Furthermore, assume that the fitness function is the one-max function [1]. Theoretically, each gene is independently mutated with some specified probability, p_m. For the mutated chromosome to have an associated fitness value that exceeds that of the parent, more 0s than 1s must be mutated. The probability of improvement, P_i, is approximately the probability of at least one 0 mutating while no 1 does, which mathematically can be expressed as

$$P_i(p_m, m) = (1 - p_m)^{n-m}(1 - (1 - p_m)^m). \tag{B.1}$$

The estimated time for an improvement is thus

$$E(K_m(p_m)) = \frac{1}{P_i(p_m, m)}. \tag{B.2}$$

As will be shown in B.2, the optimal mutation rate is $1/n$. However, a bit more generality may be introduced by defining the mutation rate to be a/n, with $a \ll n$.

Assuming that there is no bias in the initialisation of the chromosome, the expected number of 1s in the initial chromosome is $n/2$. Furthermore, according to (B.1), an

[1] The one-max function is the number of 1s in the chromosome. The aim of the (binary) GA is then to maximise the sum of the genes in the chromosome.

improvement is defined as any net increase in number of 1s. If each improvement is assumed to increase the number of 1s by 1, the expected run time becomes

$$E(K_{tot}) = \sum_{m=1}^{n/2} E(K_m(p_m)). \tag{B.3}$$

Now, equation (B.1) may be approximated by

$$P_i(p_m, m) \approx (1 - p_m)^{n-m} m p_m = (1 - \frac{a}{n})^{n-m} m \frac{a}{n}, \tag{B.4}$$

where the last term has been approximated by its first-order MacLaurin expansion along with the fact that $a \ll n$.

Using the definition

$$e^{-a} \equiv \lim_{n\to\infty} (1 - \frac{a}{n})^n,$$

equation (B.4) may be further approximated by

$$P_i(p_m, m) \approx e^{-a} m \frac{a}{n}, \tag{B.5}$$

for large values of n [59].

Inserted in (B.3) this yields

$$E(K_{tot}) \approx e^a \frac{n}{a} \sum_{m=1}^{n/2} \frac{1}{m}. \tag{B.6}$$

Finally, using the fact that

$$\sum_{m=1}^{n/2} \frac{1}{m} \approx \ln \frac{n}{2},$$

the final result is that the total estimated runtime of the algorithm is

$$E(K_{tot}) \approx e^a \frac{n}{a} \ln \frac{n}{2}, \tag{B.7}$$

[60].

B.2 Proof of optimal mutation rate

From (B.1), with the substitution $s = 1 - p_m$, it follows that

$$P_i(s,m) = s^{n-m}(1-s^m). \tag{B.8}$$

Differentiating this expression with respect to s results in

$$\begin{aligned}\frac{\partial}{\partial s}P_i(s,m) &= (n-m)s^{n-m-1}(1-s^m) + s^{n-m}(-m)s^{m-1} \\ &= s^{n-m-1}((n-m)(1-s^m) - ms^m) \\ &= s^{n-m-1}(n-m-ns^m).\end{aligned} \tag{B.9}$$

Equating this with zero yields

$$s^* = (1-\frac{m}{n})^{1/m}. \tag{B.10}$$

It therefore follows that

$$p_m^* = 1-(1-\frac{m}{n})^{1/m}. \tag{B.11}$$

Under the assumption that $m \ll n$, the MacLaurin expansion gives

$$p_m^* \approx \frac{1}{n}. \tag{B.12}$$

C

Parameters and constants

Table C.1: Parameters and constants used in the simulations

Description	Variable	Value
Vehicle:		
Vehicle mass	m	$40,000\,\mathrm{kg}$
Drag coefficient	C_d	0.6
Rolling resistance:		
	$C_{r,1}$	0.006
	$C_{r,2}$	$2.98 \cdot 10^{-6}\ \frac{1}{(\mathrm{m/s})^2}$
Wheel radius	R_w	$0.522\,\mathrm{m}$
Front area	A_v	$10\,\mathrm{m}^2$
Drivetrain efficiency	η_s	0.95
Final drive	i_f	3.08
Physical:		
Air density	ρ	$1.204\,\mathrm{kg/m}^3$
Gravitational acc.	g	$9.82\,\mathrm{m/s}^2$
Fuel density	$-$	$0.832\,\mathrm{kg/m}^3$
Control settings:		
Reference speed	v_d	$20\,\mathrm{m/s}$
Initial speed	v_0	$20\,\mathrm{m/s}$
Prediction horizon	N	80
Fuel correction factor	δ_f	1/3
Smooth driving coeff.	c_s	$3.3 \cdot 10^{-6}$
Speed tracking coeff.	c_T	$6.3 \cdot 10^{-8}$
Algorithm settings:		
Number of genes	n	80
Population size	m_p	20
Generations	G	300
Number of elites	$-$	1
Mutation probability	p_m	1/80
Crossover probability	p_c	1
Mutation range	l_{range}	$30\,\mathrm{Nm}$