

Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Electric drives and Traction



**OPTIMIZATION OF ENERGY CONSUMPTION FOR ELECTRIC VEHICLE  
DRIVING CYCLE**

by

*TOMÁŠ KACETL*

A master thesis submitted to  
the Faculty of Electrical Engineering, Czech Technical University in Prague,  
in partial fulfilment of the requirements for the degree of Master.

Master degree study programme: Electrical Machines, Apparatus and Drives

Prague, May 2016

**Supervisor:**

Ing. TOMÁŠ HAUBERT  
Department of Electric Drives and Traction  
Faculty of Electrical Engineering  
Czech Technical University in Prague  
Technická 2  
166 27 Prague 6  
Czech Republic

Copyright © 2016 TOMÁŠ KACETL



České vysoké učení technické v Praze  
Fakulta elektrotechnická

katedra elektrických pohonů a trakce

## ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Tomáš Kacetl**

Studijní program: Elektrotechnika, energetika a management  
Obor: Elektrické stroje, přístroje a pohony

Název tématu: **Optimalizace spotřeby energie jízdního cyklu elektrického vozidla**

Pokyny pro vypracování:

- 1) V prostředí MATLAB navrhnete software pro zadání trasy elektrického vozidla pomocí Google Maps. Software bude obsahovat GUI, kde bude možné zadat parametry vozidla a případně upravit zvolenou trasu. Software navrhne vhodné rychlostní omezení podél trasy respektující zatáčky a nejvyšší povolenou rychlost.
- 2) Dále navrhnete a implementujete optimalizační algoritmus minimalizující spotřebu elektrické energie na zadané trase při navrženém rychlostním omezení.
- 3) V prostředí MATLAB/Simulink vytvořte real-time model pro vizualizaci navrženého jízdního profilu s možností rozšíření pro výzkumné pracoviště VTP Rostoky.

Seznam odborné literatury:

- [1] JAZAR, Reza N. Vehicle dynamics: theory and application. 2nd ed. New York: Springer, 2014,
- [2] CHAPMAN, S. J.: Electric machinery fundamentals. 5th ed. New York: McGraw-Hill, 2012
- [3] HUTCHISON, D.: Learning and intelligent optimization. 2012

Vedoucí: Ing. Tomáš Haubert

Platnost zadání: do konce letního semestru 2016/2017



Ing. Jan Baúer, Ph.D.  
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.  
děkan

V Praze dne 1. 4. 2016



---

# Anotace

Cílem této práce je vypočítat optimální rozložení rychlosti vozidla podél zadané trasy, takzvaný rychlostní profil, který v žádném bodě trasy neporušuje podmínky dané limity pohonné jednotky a rychlostním omezením stanoveným dopravním značením a maximálním odstředivým zrychlením při průjezdu zatáčkami. Práce obsahuje model vozidla respektující síly působící proti jeho pohybu a energii zmařenou v pohonné jednotce. Tento model je pro účely optimalizace dále zjednodušen, při zachování uspokojivé přesnosti. Nakonec je zaveden heuristický optimalizační algoritmus, který je inspirován v mnoha známých technikách a sestaven tak, aby co nejlépe vyhověl charakteru optimalizační úlohy. Celý projekt je vyvinut v prostředí Matlab s použitím objektově-orientovaného programování pro jednoduché rozšíření při navazujících projektech. Kromě zmíněných modelů a algoritmů, projekt navíc obsahuje grafické rozhraní pro zadání trasy v implementovaných Google mapách nebo import trasy ze souboru. Spolu s parametry vozidla, zadanou trasou a dílčími výsledky je celý projekt uložen ve vlastním souboru. Výsledky optimalizace mohou být použity pro simulaci v reálném čase, jejíž provozní veličiny jsou přímo v uživatelské aplikaci vizualizovány a po skončení simulace mohou být vyexportovány.

**Keywords:**

elektrické vozidlo, trasa, rychlostní profil, optimalizace, modelování, dynamika vozidla.



---

# Abstract

The goal of the thesis The optimization of the drive cycle consumption of the electrical vehicle is to calculate the optimal distribution of the vehicle speed along a given track, the so called speed profile, which does not exceeds constraints subjected to the powertrain limits, speed limits in respect to the traffic signs and comfortable centrifugal acceleration when taking corners. The thesis consists of modeling the electrical vehicle in respect to the motion resistance forces applied in opposite directions to the motion, and the energy dissipation in the powertrain. This model is further modified for the purpose of optimization algorithm to keep satisfactory precision. Finally, the custom assembled algorithm inspired in existing heuristic methods is introduced, fitting the character of the objective function subjected to the given constraints. The whole project is developed in Matlab environment using object-oriented programming to allow easy extensions. Besides mentioned calculations, the application uses the graphical user interface to define the track in the implemented google maps, or it imports the track from a file. The created project is saved in the project file with all parameters and achieved results. The results of the optimization can be validated on the dynamic model running in the windows real-time simulations with direct visualization to the GUI.

**Keywords:**

Electrical vehicle, track, speed profile, optimization, modeling, vehicle dynamics.



---

# Acknowledgements

I would like to express my gratitude to my master thesis supervisor, Ing. Tomáš Haubert. He has been a constant source of encouragement and insight during my research and helped me with numerous problems. I would like to thank to Ing. Petr Kočárník Ph.D. and Ing. Tomáš Haniš Ph.D. for their expert consultations and a lot of precious advices. Finally, my greatest thanks go to my family members, for their infinite patience and care.

*“A man of genius makes no mistakes. His errors are volitional and are the portals of discovery.”*

---

— James Joys





---

# Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 27. května

.....



---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Statement . . . . .	1
1.3	Goals of the Master Thesis . . . . .	1
1.4	Structure of the Master Thesis . . . . .	1
<b>2</b>	<b>Track</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Source of data . . . . .	3
2.3	Speed limits . . . . .	4
2.3.1	Geometrical circle approximation . . . . .	6
2.3.2	Taubin-Raphson approximation . . . . .	7
2.3.3	Taubin-Raphson implementation . . . . .	8
2.3.4	Velocity calculation . . . . .	10
<b>3</b>	<b>Vehicle dynamics</b>	<b>13</b>
3.1	Introduction . . . . .	13
3.2	Applied forces . . . . .	13
3.2.1	Gravitation force . . . . .	13
3.2.2	Drag force . . . . .	14
3.2.3	Rolling resistance . . . . .	15
3.2.4	Acceleration force . . . . .	16
3.3	Forward vehicle model . . . . .	16
3.4	Powertrain modeling . . . . .	18
<b>4</b>	<b>Energy model</b>	<b>21</b>
4.1	Introduction . . . . .	21
4.2	Velocity of the vehicle . . . . .	21
4.3	Energy of section . . . . .	21
4.3.1	Kinetic energy . . . . .	22
4.3.2	Gravitational energy . . . . .	22
4.3.3	Drag resistance energy . . . . .	23
4.3.4	Roll resistance energy . . . . .	23

4.3.5	Total energy . . . . .	24
4.4	Powertrain simulation . . . . .	24
4.5	Time calculation . . . . .	25
4.6	Numerical methods . . . . .	26
4.6.1	Newton-Raphson method . . . . .	26
<b>5</b>	<b>Optimization</b>	<b>29</b>
5.1	Introduction . . . . .	29
5.2	Optimization problem . . . . .	29
5.3	Heuristic algorithm . . . . .	30
5.3.1	Introduction . . . . .	30
5.3.2	General scheme . . . . .	31
5.3.3	Constraints validation . . . . .	33
5.3.4	Local minimum prevention . . . . .	33
5.4	Reference drive . . . . .	35
5.4.1	Control algorithm . . . . .	35
<b>6</b>	<b>Programming</b>	<b>39</b>
6.1	Introduction . . . . .	39
6.2	Computer parts . . . . .	39
6.3	Graphical user interface . . . . .	40
6.3.1	Properties . . . . .	41
6.3.2	Methods . . . . .	42
6.4	Data gathering . . . . .	43
6.4.1	Google maps . . . . .	43
6.4.2	Transferring data . . . . .	44
<b>7</b>	<b>Application</b>	<b>45</b>
7.1	Introduction . . . . .	45
7.2	Project wizard . . . . .	45
7.3	Project viewer . . . . .	46
7.4	Optimization tool . . . . .	47
<b>8</b>	<b>Results</b>	<b>49</b>
8.1	Model comparison . . . . .	49
8.2	Test drives . . . . .	50
8.2.1	Vehicle . . . . .	50
8.2.2	Flat drive . . . . .	51
8.2.3	Hill drive . . . . .	53
8.2.4	Reverse drive . . . . .	55
8.2.5	Long drive . . . . .	57
8.2.6	Arrival time influence . . . . .	59
8.3	Conclusion . . . . .	61
<b>9</b>	<b>Conclusion</b>	<b>63</b>
9.1	Summary . . . . .	63
9.2	Contributions of the Master Thesis . . . . .	64

9.3 Future Work . . . . .	64
<b>Bibliography</b>	<b>65</b>
<b>A Enclosed CD</b>	<b>67</b>
A.1 Content of the CD . . . . .	67

---

## List of Figures

2.1	Track discretization . . . . .	4
2.2	Centrifugal force applied on mass point in curve. . . . .	5
2.3	Spherical and cartesian coordinates. . . . .	5
2.4	Geometrical construction of the circle. . . . .	6
3.1	Gravitational force attraction between two objects . . . . .	14
3.2	Drag between the fluid and the vehicle in motion . . . . .	15
3.3	Resistance force applied on a rolling wheel . . . . .	15
3.4	Forward vehicle model . . . . .	17
3.5	Electric vehicle powertrain composition . . . . .	18
4.1	Speed profile introduction. . . . .	22
4.2	Newton-Raphson method principle . . . . .	27
5.1	Efficiency map . . . . .	30
5.2	Polynomial surface of the efficiency map . . . . .	30
5.3	Speed profile recalculated in respect to the time step. . . . .	31
5.4	Resultant speed profile of one iterative step of the algorithm . . . . .	33
5.5	Reduced grid . . . . .	34
5.6	Soften grid . . . . .	34
5.7	Brake test to validate velocity of the point (k+1) . . . . .	36
5.8	Inverse speed profile construction as speed limit violation occurred . . . . .	36
5.9	Example of reference control algorithm behavior. . . . .	37
6.1	Start up application window . . . . .	41
6.2	Web application for track design . . . . .	43
7.1	Implemented web application for track design. . . . .	45
7.2	Specification of the vehicle parameters . . . . .	46
7.3	Project viewer with running simulation. . . . .	47
7.4	Optimizer tool of the application. . . . .	47
8.1	Comparison of the energy and dynamical model of the vehicle . . . . .	50
8.2	Map of flat drive track. . . . .	51

---

8.3	Elevation profile of the flat track. . . . .	51
8.4	Suggested reference and optimal speed profile of the flat track drives. . . . .	52
8.5	Map of hill drive track. . . . .	53
8.6	Elevation profile of the hill track. . . . .	53
8.7	Suggested reference and optimal speed profile of the hill track drives. . . . .	54
8.8	Map of reverse drive track. . . . .	55
8.9	Elevation profile of the reverse track. . . . .	55
8.10	Suggested reference and optimal speed profile of the reverse track drives. . . . .	56
8.11	Map of long drive track. . . . .	57
8.12	Elevation profile of the long track. . . . .	57
8.13	Suggested reference and optimal speed profile of the long track drives. . . . .	58
8.14	Map of test track of arrival time influence test. . . . .	59
8.15	Elevation profile of the arrival time influence test track. . . . .	59
8.16	Example of reference and optimal speed profile for 140s arrival time. . . . .	60
8.17	Influence of different arrival times on consumed energy. . . . .	60

---

## List of Tables

2.1	Friction coefficients and maximum lateral acceleration. . . . .	10
8.1	Parameters of the vehicle and environment. . . . .	50
8.2	Overview of results of the flat track drive strategies. . . . .	52
8.3	Overview of results of the flat hill drive strategies. . . . .	54
8.4	Overview of results of the flat reverse drive strategies. . . . .	56
8.5	Overview of results of the long track drive strategies. . . . .	58



---

# Introduction

## 1.1 Motivation

The electrical vehicles is very progressive branch of industry in recent years, and the object of effort to improve the impact of automotor on the environment. The most significant problems of recent electric vehicles are high purchase price and short driving distance. Driving distance is proportional to the capacity of the battery pack, where all energy to run the electrical vehicle is stored in electrochemical cells. During the drive, the energy flows from battery through powertrain, where the electrical power is transformed into mechanical power of wheels. The capacity of the battery pack and dissipation of energy in the powertrain determines maximum length of the driving distance.

## 1.2 Problem Statement

The energy consumption of an electrical vehicle differs according to drive strategies, therefore, it is possible to introduce a drive strategy to save as much energy as possible. This can be achieved by optimization algorithm, which calculates the optimal speed profile, representing the drive strategy along a given track based on the vehicle model. The project includes an application for track design and vehicle specification, the optimization algorithm, and an environment for the evaluation of the optimization results.

## 1.3 Goals of the Master Thesis

1. Create an application in Matlab for track design, using Google maps, and GUI to specify parameters of the electrical vehicle. The application also suggests speed limit in respect to the character of the roads and curves.
2. Design and implement an optimization algorithm to minimize the energy consumption of the electrical vehicle along the given track in respect to the speed limit.
3. Create real-time model for drive evaluation and visualization in Matlab/Simulink environment, with possible expansion for bench simulation in VTP Roztoky.

## 1.4 Structure of the Master Thesis

The thesis is organized into 9 chapters as follows:

1. *Introduction*: Describes the motivation together with the goals of the thesis.

## 1. INTRODUCTION

---

2. *Track*: Characterizes the track with possible sources of data and calculations of speed limit.
3. *Vehicle dynamics*: Forces analysis applied on a vehicle in motion and the forward vehicle model and powertrain simulation deriving.
4. *Energy model*: Introduces simplifications of the dynamic model to fit the purpose of the optimization algorithm.
5. *Optimization*: Forms the optimization problem and introduces heuristic optimization algorithm and reference drive based on controller with prediction.
6. *Programming*: Brief reference to problems of Matlab application development with suggested solutions and examples.
7. *Application*: Introduces the developed application with some features and parts.
8. *Results*: Applies the algorithm on few drives of different character and presents the achieved results.
9. *Conclusions*: Summarizes the results of the research, suggests possible topics for further research, and concludes the thesis.

---

# Track

## 2.1 Introduction

Track can be understood as a smooth continuous function describing the trajectory of vehicle motion, or the motion of the mass point in a 3 dimensional space as a variable of:

- time  $t$ , then we can define the position in ortogonal space with axes  $x,y,z$  as:

$$P(x, y, z) = f(t) \quad (2.1)$$

- traveled distance  $d$ , then we can define the position in ortogonal space with axes  $x,y,z$  as:

$$P(x, y, z) = f(d) \quad (2.2)$$

For the purpose of optimization, we use the description of track where the trajectory is considered only in a 2-dimensional space with the axes  $z$  which refers to altitude  $h$  and  $x$  as a function of the traveled distance  $d$ . The position of the mass point is then:

$$P(x, h) = f(d) \quad (2.3)$$

We will adopt this simplification due to the vehicle dynamics description in chapter 3 which ignores the lateral force applied on the vehicle in motion as these forces are a minor source of energy dissipation compared to the complicated calculation which would make the optimization algorithm almost infinite. However, the curvature of the track is not ignored at all inasmuch as we use the  $y$  axes to introduce the limits of maximum cornering speed explained in following section. The track is further divided into small sections of constant length  $d$  and constant angle of inclination  $\alpha$ . This fact causes no big inaccuracy while using appropriate small section length, apparent in figure 2.1.

## 2.2 Source of data

We have chosen this type of description along the previously mentioned reasons bearing in mind possible sources of data. There is no way the track can be described by mathematical function, and the description of real track is, therefore, always given as a series of sampled points with discrete values.

The most common description is GPS coordinates, which consist of altitude, longitude and latitude. Almost every device designed for measuring GPS coordinates provides information

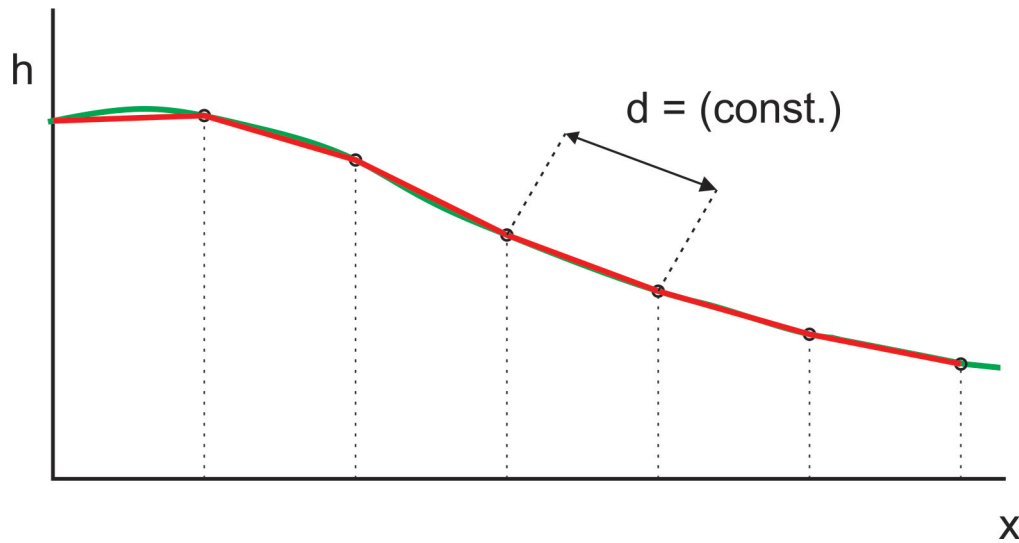


Figure 2.1: Track discretization

about the traveled distance, which gives us all the data necessary for the track description mentioned in the section 2.1.

For the purpose of obtaining data before taking a drive, it is possible to use one of many web services 'APIs', which provides GPS coordinates of the designed track in user interface. Furthermore, the advantage of using web services APIs is that they provide a lot of useful information about traffic situation, speed limit, and the character of the road. The most significant APIs are Google maps and OpenStreetMaps. When compared, the Google maps provide more accurate information with easy scripting as there are loads of examples on the official support pages, however, for information about speed limit and traffic situation, Google charges large amount of money. In the case of OpenStreetMap, everything is absolutely for free, nevertheless, the consistency of data differs between city road and countryside and also depends on the country of usage. Another trouble is missing documentation and the minimal number of examples which makes using OpenStreetMaps very uncomfortable and difficult.

## 2.3 Speed limits

As mentioned at the end of previous chapter, there is no satisfactory source of speed limit. However, maximum speed is not given only by traffic signs, but also by the curvature of the chosen road as well. Motion of vehicle in curves is described in physics as the movement of mass point along the circumference of a circle, called circular motion. During a circular motion, there is a force applied on the vehicle and passengers called centrifugal force  $\vec{F}_{cen}$ .

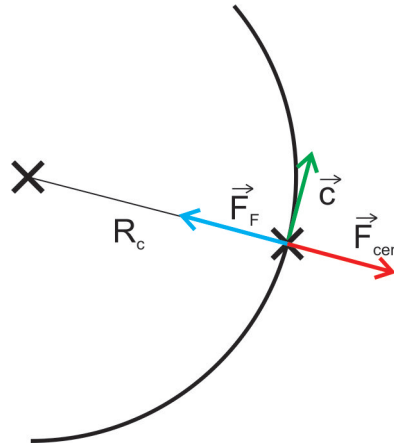


Figure 2.2: Centrifugal force applied on mass point in curve.

$$\vec{F}_{cen} = m \cdot \frac{|\vec{c}|^2 \cdot \vec{n}_c}{R_c} \quad (2.4)$$

As we can see in equation 2.4, the magnitude of the force is quadratically proportional to the magnitude of the velocity  $|\vec{c}|$  of the vehicle and inversely proportional to the radius of the curvature  $R_c$ . The centrifugal force vector has the normal direction of velocity vector  $\vec{n}_c$ , which is apparent in figure 2.2. The radius of the curvature is calculated from the GPS coordinates, but first it is necessary to transform the altitude, latitude and longitude coordinates to the cartesian coordinate system. Relation between these systems is apparent in figure 2.3. GPS coordinates are understood as spherical coordinates described by latitude  $\phi$  and longitude  $\lambda$  corresponding to the azimuthal and polar angle and altitude, which by adding Earth radius ( $R = 6378km$ ) corresponds to the radial distance. Transformation equations are then as follows:

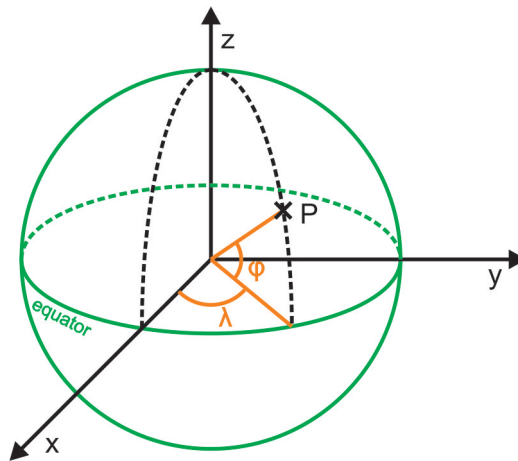


Figure 2.3: Spherical and cartesian coordinates.

$$r = R + \text{altitude}; \quad (2.5)$$

$$x = r \cdot \cos(\phi) \cdot \cos(\lambda) \quad (2.6)$$

$$y = r \cdot \cos(\phi) \cdot \sin(\lambda) \quad (2.7)$$

$$z = r \cdot \sin(\phi) \quad (2.8)$$

However, the radius of curvature is calculated in  $x$  and  $y$  axes, and recalculation of these axes in respect to the axes  $z$  has to be done.

### 2.3.1 Geometrical circle approximation

As a circle is defined by minimum of three points, the calculation is made for each triplet of consecutive points labeled  $P_1(x_1, y_1)$ ,  $P_2(x_2, y_2)$ ,  $P_3(x_3, y_3)$ . Geometrically, we know that the center of a circle  $M(x_m, y_m)$  lies on lines that pass through the midpoints of chords  $\overline{P_1P_2}$  and  $\overline{P_2P_3}$  and are perpendicular to each chord. Equations of the cord lines are:

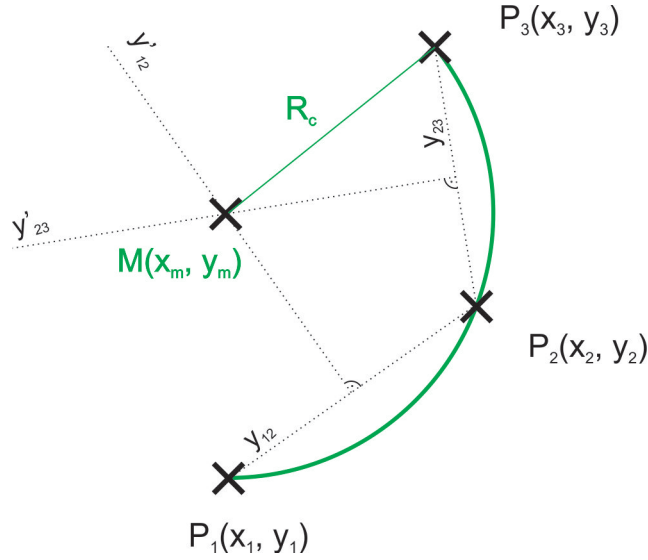


Figure 2.4: Geometrical construction of the circle.

$$m_{12} = \frac{y_2 - y_1}{x_2 - x_1} \quad (2.9)$$

$$m_{23} = \frac{y_3 - y_2}{x_3 - x_2} \quad (2.10)$$

$$y_{12} = m_{12} \cdot (x - x_1) + y_1 \quad (2.11)$$

$$y_{23} = m_{23} \cdot (x - x_2) + y_2 \quad (2.12)$$

Equation of the perpendicular lines are:

$$y'_{12} = -\frac{1}{m_{12}} \cdot \left(x - \frac{x_1 + x_2}{2}\right) + \frac{y_1 + y_2}{2} \quad (2.13)$$

$$y'_{23} = -\frac{1}{m_{23}} \cdot \left(x - \frac{x_2 + x_3}{2}\right) + \frac{y_2 + y_3}{2} \quad (2.14)$$

Both of these lines pass through the midpoint of the circle which is identical to their point of intersection. The geometrical construction is clear from figure 2.4. Coordinate  $x_m$  is found by equation:

$$x_m = \frac{m_{12} \cdot m_{23} \cdot (y_3 - y_1) + m_{12} \cdot (x_2 + x_3) - m_{23} \cdot (x_1 + x_2)}{2 \cdot (m_{12} - m_{23})} \quad (2.15)$$

Coordinate  $y$  is easily calculated by substituting  $x$  into the equation of one of lines  $y'_{12}$  or  $y'_{23}$ . Finally the radius of the curvature is the distance between midpoint and any point of taken triplet.

$$R_c = \sqrt{(x_1 - x_m)^2 + (y_1 - y_m)^2} \quad (2.16)$$

Geometrical approximation of every triplet of consecutive points is possible only in the case of extremely accurate set of data. However, GPS coordinates of Google maps API are precise within meters, which is comparable to the desired sampling and, thus, the result of geometrical approximation is nonsense. For this purpose, it is by far better to calculate the curvature of cornering by algebraic approximation using multiple consecutive points.

### 2.3.2 Taubin-Raphson approximation

In general, algebraic methods use larger sets of data than the previous method and minimize the sum of squares of algebraic distances between each point of given set and fitted circle and, thus, this method is one of Least-Squares fitting methods. This fact can be written as follows.

$$F_1(a, b, R) = \sum_{k=1}^n [(x_k - a)^2 + (y_k - b)^2 - R^2]^2 \quad (2.17)$$

Where  $F_1$  is the object of minimization, variables  $a$  and  $b$  are coordinates of center, and  $R$  is radius of fitted circle. Generally speaking, the problem of fitting is described as polynomial function  $P(x_i, y_i)$  and function  $F_1$  can be rewritten as.

$$F_1 = \sum_{k=1}^n [P(x_i, y_i)]^2 \quad (2.18)$$

Improved methods called the gradient weighted algebraic fit are based on minimizing 2.19

$$F_2 = \sum_{k=1}^n \frac{[P(x_i, y_i)]^2}{\|\nabla P(x_i, y_i)\|^2} \quad (2.19)$$

where  $\nabla P(x_i, y_i)$  is the gradient of the function  $P(x_i, y_i)$  in 2.18. The function  $F_2$  works better than  $F_1$ , because using the Taylor expansion we have

$$\frac{[P(x_i, y_i)]}{\|\nabla P(x_i, y_i)\|} = d_i + O(d_i^2) \quad (2.20)$$

where  $d_i$  is the geometric distance of the point  $(x_i, y_i)$  to the curve  $P(x_i, y_i)$ . The function  $F_2$  is therefore a linear approximation to the function  $F_1$ . For a circle we can write

$$P(x_i, y_i) = (x_i^2 + y_i^2) - 2 \cdot a \cdot x_i - 2 \cdot b \cdot y_i + a^2 + b^2 - R^2 \quad (2.21)$$

$$\nabla P(x_i, y_i) = (2x_i - 2a, 2y_i - 2b) \quad (2.22)$$

$$\|\nabla P(x_i, y_i)\| = 4(x_i^2 + y_i^2) - 8ax_i - 8by_i + 4a^2 + 4b^2 \quad (2.23)$$

The minimization objective function is then

$$F_2 = \sum_{k=1}^n \frac{(x_i^2 + y_i^2) - 2ax_i - 2by_i + a^2 + b^2 - R^2}{4(x_i^2 + y_i^2) - 8ax_i - 8by_i + 4a^2 + 4b^2} \quad (2.24)$$

This is a nonlinear problem that can be solved iteratively, hence, we use approximation which allows us to use simpler solution. Taubin's approximation averages the variables in the denominator. When substituting  $(x_i^2 + y_i^2) = z_i$ , we get to the function

$$F_2 = \sum_{k=1}^n \frac{[z_i - 2ax_i - 2by_i + a^2 + b^2 - R^2]^2}{4z_i - 8ax_i - 8by_i + 4a^2 + 4b^2} \quad (2.25)$$

Now we can switch to the algebraic circle parameters  $A, B, C, D$  by following substitutions:

$$a = -\frac{B}{2A}, \quad b = -\frac{C}{2A}, \quad R^2 = -\frac{B^2 + C^2 - 4AD}{4A^2} \quad (2.26)$$

and we obtain function  $F_3$  as

$$F_3(A, B, C, D) = \sum_{k=1}^n \frac{[Az_i + Bx_i + Cy_i + D]^2}{[4A^2z_i + ABx_i + ACy_i + B^2 + C^2]} \quad (2.27)$$

The minimization of function 2.27 is equivalent to the minimization of function  $F_4$  in 2.28

$$F_4(A, B, C, D) = \sum_{k=1}^n [Az_i + Bx_i + Cy_i + D]^2 \quad (2.28)$$

with the constraint 2.29

$$4A^2z_i + ABx_i + ACy_i + B^2 + C^2 = 1 \quad (2.29)$$

The method is deeply explained in [2]

### 2.3.3 Taubin-Raphson implementation

First, we eliminate the parameter  $D$  by centering the data set considering  $\bar{x}_i = \bar{y}_i = 0$  and using  $D = -Az_i$  we get the function:

$$F_5(A, B, C) = \sum_{k=1}^n [A(z_i - \bar{z}) + Bx_i + Cy_i]^2 \quad (2.30)$$



and constraint

$$4A^2\bar{z} + B^2 + C^2 = 1 \quad (2.31)$$

It is convenient to introduce parameter  $A_0$  as

$$A_0 = 2\sqrt{\bar{z}}A \quad (2.32)$$

Now we minimize function  $F_6$  with the constraint as follows:

$$F_6(A_0, B, C) = \sum_{k=1}^n [A_0 \frac{z_k - \bar{z}}{2\sqrt{\bar{z}}} + Bx_k + Cy] \quad (2.33)$$

$$A_0^2 + B^2 + C^2 = 1 \quad (2.34)$$

Secondly, we make reduction to eigenvalue problem by rewriting the function 2.33 in a matrix form:

$$F_5 = \|\mathbf{X}_0 \mathbf{A}_0\|^2 = \mathbf{A}_0^T (\mathbf{X}_0^T \mathbf{X}_0) \mathbf{A}_0 \quad (2.35)$$

where  $\mathbf{X}_0$  and  $\mathbf{A}_0$  can be written as

$$\mathbf{A}_0 = (A_0, B, C)^T \quad (2.36)$$

$$\mathbf{X}_0 = \begin{bmatrix} \frac{z_1 - \bar{z}}{2\sqrt{\bar{z}}} & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ \frac{z_n - \bar{z}}{2\sqrt{\bar{z}}} & x_n & y_n \end{bmatrix} \quad (2.37)$$

The minimum of  $F_5$  is attained on the unit eigenvector of the matrix  $\mathbf{X}_0^T \mathbf{X}_0$  corresponding to its smallest eigenvalue. This matrix is symmetrical and positive-semidefinite, thus, all its eigenvalues are real and non-negative. Furthermore, this matrix is non-singular, positive-definitive, unless the data points lie of a circle or a line. For a faster solution, the eigenvalue  $\lambda$  of the matrix  $\mathbf{X}_0^T \mathbf{X}_0$  is found by solving its characteristic equation

$$\det(\mathbf{X}_0^T \mathbf{X}_0 - \lambda \mathbf{I}) = 0 \quad (2.38)$$

Equation 2.38 is rewritten into polynomial form of the 3rd degree in  $\lambda$ :

$$P(\lambda) = c_3 \lambda^3 + c_2 \lambda^2 + c_1 \lambda + c_0 = 0 \quad (2.39)$$

Coefficients  $c_0, c_1, c_2, c_3$  are as follows

$$\begin{aligned} c_3 &= 4\bar{z} \\ c_2 &= -\bar{z}\bar{z} - 3\bar{z}^2 \\ c_1 &= \bar{z}(\bar{z}\bar{z} - \bar{z}^2) + 4\bar{z}(\bar{x}x\bar{y}y - \bar{x}y^2) - \bar{x}\bar{z}^2 - \bar{y}\bar{z}^2 \\ c_0 &= \bar{x}\bar{z}^2\bar{y}y + \bar{y}\bar{z}^2\bar{x}x - 2\bar{x}\bar{y}\bar{z}\bar{x}y - (\bar{x}\bar{y}y - \bar{x}y^2)(\bar{z}\bar{z} - \bar{z}^2) \end{aligned} \quad (2.40)$$

As the eigenvalues of  $\mathbf{X}_0^T \mathbf{X}_0$  are real and non-negative, the characteristic polynomial equation 2.39 always has three non-negative real roots and  $P(0) \leq 0$ . In the non-singular case, the

$P(0) < 0$ , and then  $P''(\lambda) < 0$  in the interval between 0 and the smallest root. Newton-Raphson method with initial guess  $\lambda = 0$ , hence, always converges to the smallest desired root. Parameters of the fitted circle are calculated as:

$$a = \frac{2(\overline{xzy} - \lambda\overline{xz} - \overline{yzyx})}{\lambda^2 - \lambda\overline{z} + \overline{xy} - \overline{xyxy}} + x_0 \quad (2.41)$$

$$b = \frac{2(\overline{xzx} - \lambda\overline{xz} - \overline{yzyx})}{\lambda^2 - \lambda\overline{z} + \overline{xy} - \overline{xyxy}} + y_0 \quad (2.42)$$

$$R = \sqrt{a^2 + b^2} \quad (2.43)$$

The method is deeply explained in [3].

### 2.3.4 Velocity calculation

We can calculate the maximum velocity in a curve by limiting centrifugal acceleration applied on the vehicle and passengers inside. For the vehicle, the centrifugal force  $\vec{F}_{cen}$  causes skid of the vehicle. Magnitude of the centrifugal force is directly proportional to the mass of the vehicle  $m$ , square of the vehicle speed  $\vec{c}$ , and inversely proportional to the radius of curvature  $R_c$ .

$$\vec{F}_{cen} = m \cdot \frac{|\vec{c}|^2 \cdot \vec{n}_c}{R_c} \quad (2.44)$$

According to the principle of action and reaction, there is a force which is applied on the vehicle in the opposite direction to the centrifugal force. This force is caused by the friction between tires and the road, and its maximum magnitude is calculated as:

$$\vec{F}_{FM} = \mu \cdot m \cdot |\vec{g}| \cdot -\vec{n}_c \quad (2.45)$$

Where  $m$  is the mass of the vehicle,  $\vec{g}$  is gravitational acceleration and  $\mu$  is the coefficient of friction between the vehicle tires and surface of the road. The coefficient of friction depends on the surface of the road and the typical values are in the table 2.1. For each of these coefficients, maximal magnitude of centrifugal acceleration  $|\vec{a}_{cskid}|$  is calculated according to the equation 2.46 and its value is also in the table:

$$|\vec{a}_{cskid}| = |\vec{g}| \cdot \mu \quad (2.46)$$

Road type	$\mu$	$ \vec{a}_{cskid} $
Dry asphalt	0.9	8.83
Wet asphalt	0.6	5.89
Snow	0.2	1.96
Ice	0.05	0.49

Table 2.1: Friction coefficients and maximum lateral acceleration.

Once the centrifugal force exceeds friction force maximum, the vehicle gets into skid, which is undesirable and can be a cause of traffic accident.

Another factor limiting centrifugal acceleration is the comfort of the passengers. For railway, the limit of the centrifugal acceleration is  $|\vec{a}_{comfort}| = 2m \cdot s^{-2}$ . Taking to consideration dry or wet asphalt and snow, we are limited by the the comfortable lateral acceleration  $|\vec{a}_{cmax}| = 2m \cdot s^{-2}$ . Now it is possible to calculate the speed limit of each point of the track as follows:

$$v_{max} = \sqrt{R_c \cdot |\vec{a}_{cmax}|} \quad (2.47)$$



---

# Vehicle dynamics

## 3.1 Introduction

The motion of a vehicle on a road is a complex system described by several differential equations. The model of ideal rigid vehicle is difficult enough for purpose of optimization. There are loads of forces which differ in their magnitude, direction and point of application. Therefore, some simplifications are introduced in order to create an optimization algorithm fitting present computing capability and keep satisfactory precision. For the above reasons, straight motion of an ideal rigid vehicle on an inclined road is considered including acceleration force, gravitation force, air friction and rolling resistance. Resultant force of each considered force is applied in the center of gravity. The sum of these forces is the total resultant force applied on a vehicle in motion. This force has multiple variables, and the detailed description is the subject of the following sections, where each of these forces is examined separately.

## 3.2 Applied forces

### 3.2.1 Gravitation force

Any two objects with a mass attract each other with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them, apparent in figure 3.1.

$$\vec{F}_g = G \cdot \frac{m_1 \cdot m_2}{r^2} \quad (3.1)$$

The mass of a vehicle compared with the mass of Earth is insignificant and therefore, force which the vehicle attracts Earth is omitted. For purpose of optimization the satisfactory results gives a model of a gravitational field with gravitational constant.

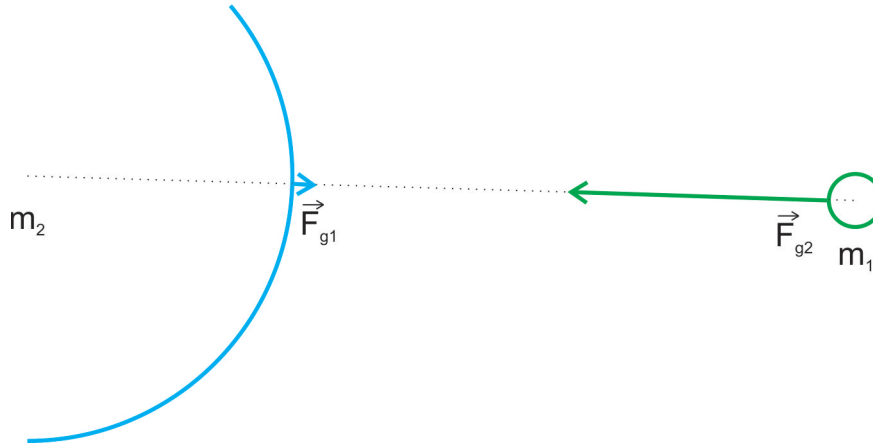


Figure 3.1: Gravitational force attraction between two objects

$$\vec{g} = G \cdot \frac{m_1}{\vec{R}^2} \quad (3.2)$$

Gravitation force is then:

$$\vec{F}_g = m \cdot \vec{g} \quad (3.3)$$

As the vehicle exerts a force of its mass on the road, the road simultaneously exerts a force of the same magnitude and in the opposite direction on the vehicle. For the model of a vehicle the component of the force opposite the direction of the motion is considered. [6]

$$\vec{F}'_g = \vec{F}_g \cdot \sin(\alpha) \quad (3.4)$$

### 3.2.2 Drag force

Drag force is the resultant force of forces acting opposite to the relative motion of any object moving in respect to the surrounding fluid (between two fluid layers or a fluid and a solid surface). The drag force is directly proportional to velocity for laminar flow according to the Stokes law, and squared for turbulent flow according to the Newton's law. The flow depends on the shape of the moving object and on the Reynolds number.

$$Re = \frac{c \cdot D}{\nu} \quad (3.5)$$

$$\nu = \frac{\mu}{\rho} \quad (3.6)$$

Where  $\nu$  is kinematic viscosity of the fluid (viscosity  $\mu$  divided by the density  $\rho$  of the fluid) and  $D$  is characteristic diameter. Calculation of the force applied on the vehicle in motion is computed in iterative steps to determine the type of flow and associated drag coefficient. Moreover, the flow type changes from the vehicle hood to the back of the vehicle. Precise examination of the flow along the vehicle is inappropriate for the purpose of real-time simulation and optimization. Hence, the Newton's resistance law based on conservation of energy is used. An object moving in a fluid pushes the originally still fluid. The force, which the vehicle exerts on the fluid, according to the principle of action and reaction, is equal to

the drag force in the opposite direction. The work of this force is equal to the kinetic energy of the moving fluid, hence: [8]

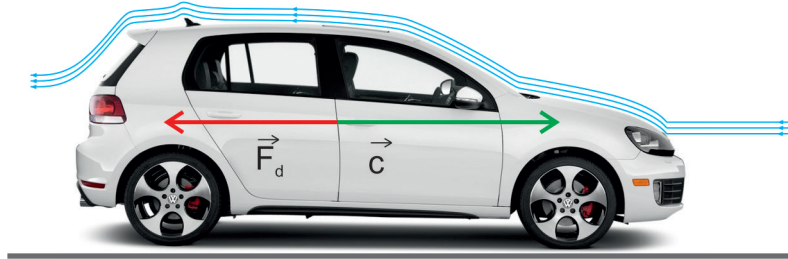


Figure 3.2: Drag between the fluid and the vehicle in motion

$$\vec{F}_d = \frac{1}{2} \cdot \rho \cdot A \cdot \vec{c}^2 \quad (3.7)$$

However, this force does not respect the shape of the moving object and properties of the fluid. Therefore, the drag coefficient  $C_d$  is introduced and the Newton's resistance law equation is as follows:

$$\vec{F}_d = \frac{1}{2} \cdot \rho \cdot C_D \cdot A \cdot \vec{c}^2 \quad (3.8)$$

Where  $A$  is the area of the cross section. Drag force is the most significant of motion resistant forces. [6]

### 3.2.3 Rolling resistance

Rolling resistance  $F_r$  is generated by a turning tire on the road. Direction of the force is opposite to the direction of the motion and the magnitude is directly proportional to the normal component of the gravitation force.

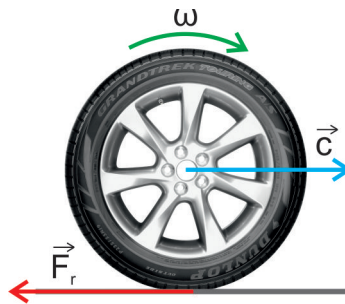


Figure 3.3: Resistance force applied on a rolling wheel

$$\vec{F}_r = \mu_r \cdot \vec{F}_{gn} \quad (3.9)$$

The value of the friction coefficient  $\mu_r$  depends on the tire velocity, inflation pressure, sideslip and camber angles and mechanical properties, wear, temperature, load, driving and braking force, and road conditions. It would put extreme demands on the simulation time to consider all the operating condition dependencies. In the case of velocity, the rolling friction coefficient increases with the square of the velocity. However, the part of the equation 3.10 with coefficient  $\mu_1$  is insignificant compared with the first part of the equation for values of velocity under  $100m/s$  which highly exceed speed limits of any road. [5]

$$\mu_r = (\mu_0 + \mu_1 |\vec{c}|^2) \quad (3.10)$$

Hence, we can omit that part and for the simulation model we can consider the rolling friction as follows:

$$\vec{F}_r = \mu_0 \cdot \left| \vec{F}_g \right| \cdot \cos(\alpha) \quad (3.11)$$

### 3.2.4 Acceleration force

Acceleration force is generated by the imbalance between load torque  $\vec{M}_L$  and torque of the powertrain on wheels  $\vec{M}_W$  according to the motion equation 3.12, where  $\vec{M}_D$  is the dynamic torque. The acceleration torque is on the side of load because of the principle of action and reaction.

$$\vec{M}_W = \vec{M}_D + \vec{M}_L + \vec{M}_a \quad (3.12)$$

In a steady state when the vehicle is not accelerating ( $\frac{dc}{dt} = 0$ ), the dynamic torque is equal to zero. In the case of torque surplus ( $\vec{M}_W - \vec{M}_L > 0$ ) the vehicle is accelerating, in the case of torque lack ( $\vec{M}_W - \vec{M}_L < 0$ ) the vehicle is decelerating. The dynamic moment represents the moment increasing the angular velocity of rotating mass which is specified by the rotational inertia  $J$ .

$$\vec{M}_D = J \cdot \frac{d\vec{\omega}}{dt} \quad (3.13)$$

Acceleration torque can be separated from the equation as follows:

$$\vec{M}_a = \vec{M}_W - \vec{M}_L - \vec{M}_D \quad (3.14)$$

The acceleration force is then simply calculated as torque divided by the perimeter of wheels  $r_w$ , and  $\vec{a}$  is the acceleration of the vehicle in the direction of the motion.

$$\vec{F}_a = \frac{\vec{M}_a}{r_w} = m \cdot \vec{a} \quad (3.15)$$

## 3.3 Forward vehicle model

In mathematical model, we substitute vehicle by mass point moving along the track with applied forces explained in the previous chapters.



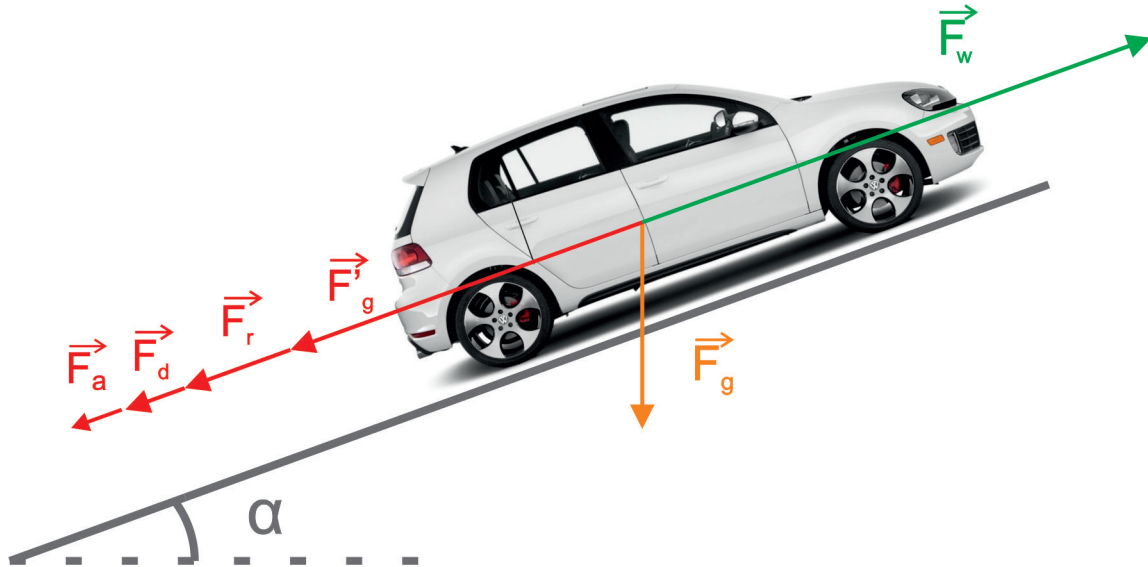


Figure 3.4: Forward vehicle model

For the motion of a vehicle, we can use equation of force balance.

$$\vec{F}_W + \vec{F}_a + \vec{F}_g' + \vec{F}_d + \vec{F}_r = 0 \quad (3.16)$$

As we can see in the picture 3.4, we only consider forces in the same direction and opposite the direction of a motion which allows us to transfer vectors of forces to scalars with appropriate signs. Where minus sign indicates force direction opposite the motion and plus sign indicates force in the direction of motion. Then we can rewrite the equation 3.16 to the scalar form.

$$F_M - F_a - F_g' - F_d - F_r = 0 \quad (3.17)$$

We can also multiply the equation 3.17 by the perimeter of wheels  $r_w$  and add dynamic moment and get equation of motion in scalar form. Using equations of each force leads to the full form of equation 3.19 describing the motion of a vehicle.

$$M_W - M_a - M_g' - M_d - M_r - M_D = 0 \quad (3.18)$$

$$M_W - m \cdot a - m \cdot g \cdot \sin(\alpha) - \frac{1}{2} \cdot \rho \cdot C_D \cdot A \cdot c^2 - \mu_0 \cdot m \cdot g \cdot \cos(\alpha) - J \cdot \frac{d\omega}{dt} = 0 \quad (3.19)$$

Using equation 3.20 and separating the acceleration  $a = \frac{dc}{dt}$  from the equation we gets simulation equation 3.21 of the vehicle motion.

$$\omega = \frac{c}{2 \cdot \pi \cdot r} \quad (3.20)$$

$$\frac{dc}{dt} = \left( \frac{J}{2 \cdot \pi \cdot r} + m \right) \cdot (M_W - m \cdot g \cdot \sin(\alpha) - \frac{1}{2} \cdot \rho \cdot C_D \cdot A \cdot c^2 - \mu_0 \cdot m \cdot g \cdot \cos(\alpha)) \quad (3.21)$$

In this chapter, we see the vehicle motion is described by differential equation 3.21. This means that for the validation of suggested result from optimization algorithm, it is necessary to run whole simulation of a drive. This is very time consuming and for purpose of optimization it is, therefore, essential to come up with some simplification, which provides easier calculation and possible analysis.

### 3.4 Powertrain modeling

Powertrain of electric vehicle consists of battery, inverter, electric motor and mechanical transmission, apparent in figure 3.5. These parts are very complicated and their behavior is hardly modeled even for purpose of design. To investigate all power dissipation in the powertrain, we would need to use the most detailed models of each part, so for purpose of modeling power dissipated in powertrain we will use a table of efficiency. The power flow in electric motor is characterized by torque  $M$  and revolution  $N$  in mechanical domain and by voltage  $U$  and current  $I$  in electrical domain. This table is commonly measured on test benches, well known as the so called efficiency map. Knowing mechanical power on wheels, we use only mechanical domain of the map and the efficiency of powertrain can be expressed as:

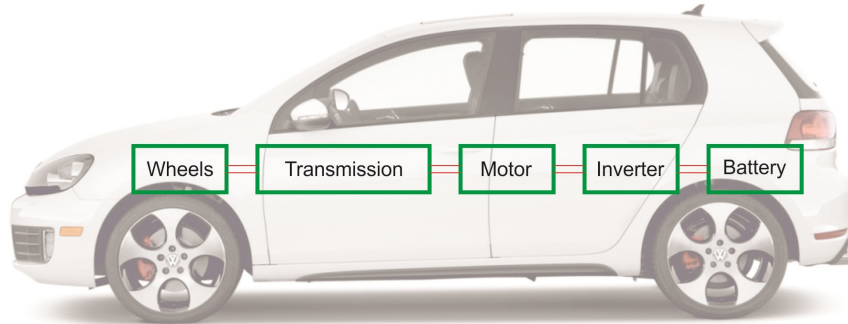


Figure 3.5: Electric vehicle powertrain composition

$$\eta = f(M, N) \quad (3.22)$$

However, the efficiency map represents only energy dissipation, but the vehicle is also limited on motion by maximum torque, revolution and power of the engine. Maximum torque is limited by maximum temperature of insulation as the torque is proportional to the electrical current flowing in the winding, which transforms into heat in the winding resistance.

Maximum revolutions are limited by the mechanical construction of the motor as the centrifugal force is proportional to the square of the velocity. This cannot be omitted, but it is simply simulated by limiting these values in the model. If not included in the efficiency map, both of these values must be recalculated to the side of the motor, as the powertrain include transmission. The transmission ratio is mostly set to transform the revolution to higher values on the side of the motor and higher torque on the side of wheels. As the energy consumed from the battery is investigated, we must determine the direction of power flow. If the powertrain drives wheels, the power flowing from battery  $P_{bat}$  is calculated as power on wheels  $P_w$  increased by power dissipated in powertrain. If the powertrain recuperates, the power flowing to battery is calculated as power on wheels decreased by power dissipated in powertrain. This can be introduced as follows:

$$P_{bat} = P_w \cdot \eta^{-sign(P_w)} \quad (3.23)$$

This equation simply describes the fact, that the power dissipation in powertrain increases the power drained from battery, and on the other hand decreases the power of recuperation.



## Energy model

### 4.1 Introduction

Energy model is the essential simplification mentioned in the previous chapter 3.3. It is based on the division of the route into sections from the chapter 2.1. Generally speaking, we decompose the drive into many drives by sections, which we can handle separately. To make this possible, we must first deal with aspects preventing us from doing so.

### 4.2 Velocity of the vehicle

The dynamic model in the previous chapter 3 considers velocity as a continuous smooth function of traveled distance  $s$ :

$$c = f(s) \quad (4.1)$$

However, similar to the track characteristic values, we will also sample the function of velocity. Every section is characterized by start point and end point, where in each of these points velocity  $c(k)$ , altitude  $h(k)$  and distance between these points  $d$ , which is designed as a constant, are defined. As in the case of altitude, the velocity is changing linearly from the start point to the end point of a section and the end point of a section is the start point of following section. This fact transforms the smooth velocity function to function 4.2 of discrete variable  $k$ , which refers to the index of the section.

$$c(s) = c(k) + \frac{c(k+1) - c(k)}{d} \cdot s \quad (4.2)$$

The set of all velocity values describing all sections gives velocity profile, which may be subject of optimization, apparent in figure 4.1.

### 4.3 Energy of section

With all the so far made steps, there are no more difficulties to calculate the energy of every single section, which is generally given by altitude and velocity in every section. The energy of every section is calculated by integration of force over its distance. As well as forces, we can also examine kinetic energy  $E_k(k)$ , gravitational energy  $E_g(k)$ , drag energy dissipation  $E_d(k)$  and roll energy dissipation  $E_r(k)$  separately. The total energy of the section  $E_T(k)$  is then the sum of these energy components 4.3.

$$E_T(k) = E_k(k) + E_g(k) + E_d(k) + E_r(k) \quad (4.3)$$

In the following subsections, we will derive formulas of each component of total energy.

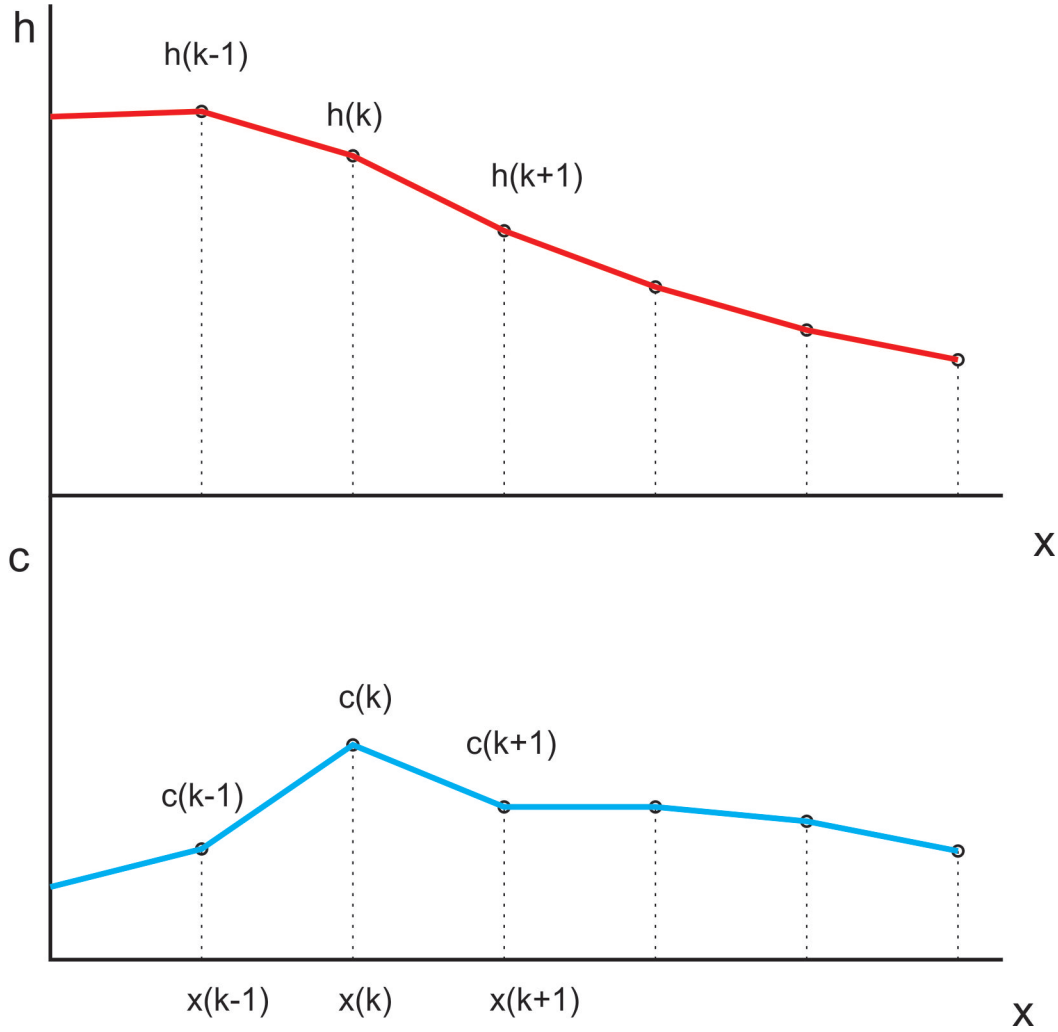


Figure 4.1: Speed profile introduction.

### 4.3.1 Kinetic energy

Kinetic energy of a vehicle is the energy that it possesses due to its motion. We can define it as the work needed to accelerate the vehicle of mass  $m$  from velocity of the start point  $c(k-1)$  to the velocity of the end point  $c(k)$  in equation 4.4.

$$E_k(k) = \frac{1}{2} \cdot m \cdot (c^2(k) - c^2(k-1)) \quad (4.4)$$

### 4.3.2 Gravitational energy

Gravitational force applied on the vehicle in the motion was examined in the previous chapter 3.2.1. When we multiply the force by element of track  $dh$  in its direction, we get an element of gravitational energy.

$$dE_p = m \cdot g \cdot dh \quad (4.5)$$

By integration of the energy element 4.5 over the section we get the gravitational energy 4.6.

$$E_p(k) = \int_{h(k-1)}^{h(k)} m \cdot g \, dh. \quad (4.6)$$

As we can see, the energy depends simply on the altitude of start point and altitude of end point.

$$E_p(k) = m \cdot g \cdot (h(k) - h(k-1)) \quad (4.7)$$

### 4.3.3 Drag resistance energy

Element of drag resistance dissipation energy  $dE_d$  4.8 is calculated from drag resistance force 3.2.2 and element of track in the direction of motion  $ds$ .

$$dE_d = B \cdot c^2(s) \cdot ds \quad (4.8)$$

Total energy is given as integration over the track from start point to end point.

$$E_d(k) = \int_{d(k-1)}^{d(k)} B \cdot c^2(s) \, ds. \quad (4.9)$$

We can substitute for the velocity  $c(s)$  from equation 4.2

$$E_d(k) = B \cdot \int_{d(k-1)}^{d(k)} \left( c(k+1) + \frac{c(k) - c(k-1)}{d} \cdot s \right)^2 \, ds. \quad (4.10)$$

After integration we get the total energy as follows:

$$E_d(k) = B \cdot (c^2(k-1) \cdot d + c(k-1) \cdot (c(k) - c(k-1)) \cdot d + \frac{1}{3} \cdot (c(k) - c(k-1))^2 \cdot d) \quad (4.11)$$

### 4.3.4 Roll resistance energy

Roll resistance dissipation energy is caused by roll resistance force 3.2.3. According to the similar procedure as in the case of drag resistance energy, we have equation for the element of the energy 4.12

$$dE_r = \mu_0 \cdot m \cdot g \cdot \cos(\alpha) \cdot ds + \mu_1 \cdot m \cdot g \cdot c^2(s) \cdot \cos(\alpha) \cdot ds \quad (4.12)$$

Then integration over the track section of length  $d$  is made

$$E_r(k) = \int_{d(k-1)}^{d(k)} \mu_0 \cdot m \cdot g \cdot \cos(\alpha) \, ds + \int_{d(k-1)}^{d(k)} \mu_1 \cdot m \cdot g \cdot c^2(s) \cdot \cos(\alpha) \, ds \quad (4.13)$$

The slope of the road is given as follows

$$\cos(\alpha) = \frac{d}{x(k-1) - x(k)} \quad (4.14)$$

Thus, after substitution from equation 4.2 we get

$$E_r(k) = \frac{\mu_0 \cdot m \cdot g \cdot x}{d} \cdot \int_{d(k-1)}^{d(k)} ds + \frac{\mu_1 \cdot m \cdot g \cdot x}{d} \cdot \int_{d(k-1)}^{d(k)} \left( c(k-1) + \frac{c(k) - c(k-1)}{d} \cdot s \right)^2 \, ds \quad (4.15)$$

By integration and substitution of limits of integration we get the equation of total roll resistance energy of section

$$E_r(k) = \mu_0 \cdot m \cdot g \cdot x + \mu_1 \cdot m \cdot g \cdot x \cdot \frac{1}{3} \cdot (c^2(k) + c(k) \cdot c(k-1) + c^2(k-1)) \quad (4.16)$$

### 4.3.5 Total energy

Total energy consumed during drive on a given track is simply calculated as the sum of energy components over all section, where  $k$  is the index of section and  $n$  is the total count of section.

$$E_{total} = \sum_{k=1}^n E(k) \quad (4.17)$$

## 4.4 Powertrain simulation

To simulate the powertrain energy dissipation we use the same efficiency map representation as in the case of dynamic model 3.4. However, as the drive is investigated section by section, we do not have continuous values of torque neither power. Calculation of these values means integration over the section leading back to the dynamic model. When the section length is small enough, we can consider values of torque, power and revolution as constants. The power is, therefore, simply calculated as energy of the section divided by value of time.

$$P(k) = \frac{E(k)}{t(k)} \quad (4.18)$$

Revolution of the motor can be simply calculated by averaging the velocity of the section. The velocity must be recalculated from the side of wheels to the side of the motor by the transmission ratio  $c_{tr}$ :

$$N(k) = \frac{c(k-1) + c(k)}{2} \cdot \frac{2\pi}{60} \cdot c_{tr} \quad (4.19)$$

From previously calculated power and revolution of the section, we can calculate torque:

$$M(k) = \frac{P(k)}{N(k)} \quad (4.20)$$

Each pair of these values specifies the operation point of the motor, and we can, therefore, find the value of efficiency in the efficiency map. As mentioned in the previous chapter, the operation of the motor is limited by the power, torque and revolution and thus, we must oversee whether none of these values exceeds the limits of the motor. Despite the fact we calculate average values of the section and the actual values can differ, the motor can be temporarily overloaded in each of these values. As the section length is small enough, the overload is always considered as temporary and, moreover, the value of overload is not significant.

As the power flows through the powertrain in both directions, we must distinguish whether the powertrain drives the wheels, and the energy on wheels is increased by the dissipated energy, or the powertrain recuperates, and the energy on wheels is decreased by the dissipated energy. Considering the  $E_{total}$  as energy on wheels,  $E(k)$  as energy consumed from battery, and positive value of power  $P(k)$  as flow from battery to wheels, the consumed energy of the  $k$ -th section is computed as:

$$E(k) = E_{total}(k) \cdot \eta(M(k), N(k))^{-sign(P(k))} \quad (4.21)$$



## 4.5 Time calculation

As we have chosen distance  $d$  as variable of function describing the vehicle motion, which brought about many simplifications, on the other hand the time calculation of the vehicle motion is quite complicated. However, time calculation is necessary to determine the arrival time, which may be one of the conditions characterizing the drive. Total time, similar to the energy calculation, is given as the sum of times to pass each section. In the calculation, we can start with the definition of velocity

$$c(s) = \frac{ds}{dt} \quad (4.22)$$

We can separate element of time  $dt$  on one side of equation

$$dt = \frac{1}{c(s)} \cdot ds \quad (4.23)$$

Time to pass a section can be obtained by integrating both sides of the equation

$$t(k) = \int_0^d \left( \frac{1}{c(s)} \right) ds \quad (4.24)$$

The last step, we substitute for velocity from equation 4.2

$$t(k) = \int_0^d \left( \frac{1}{c(k-1) + \frac{c(k)-c(k-1)}{d} \cdot s} \right) ds \quad (4.25)$$

After integration we get the final expression of time to pass a section as follows:

$$t(k) = \frac{d}{c(k) - c(k-1)} \cdot \ln\left(\frac{c(k)}{c(k-1)}\right) \quad (4.26)$$

Complication that are brought about by choosing distance domain may not be clear right now, but if we want to make inverse calculation to obtain velocity on the end of the section  $c(k)$  as a function of time  $t(k)$ , there is no way to get the function  $c(k) = f(t(k))$  by using elemental function as the variable  $c(k)$  appears in the logarithm and also in the fraction in front of the logarithm. Hence, we must use the so called Lambert function  $W(x)$ , which is an inverse relation of the function:

$$f(x) = x \cdot e^x \quad (4.27)$$

$$x = f^{-1}(x \cdot e^x) = W(x \cdot e^x) \quad (4.28)$$

We can now find the inverse function to the original problem by using the Lambert function as:

$$c(k) = - \frac{d \cdot W\left(-\frac{c(k-1) \cdot t(k) \cdot e^{\frac{c(k-1) \cdot t(k)}{d}}}{d}\right)}{t(k)} \quad (4.29)$$

The Lambert function is commonly implemented in almost all mathematical oriented programming languages or can be easily included by mathematical package if necessary. Nevertheless, the previously mentioned problem is not the only one. The energy model introduced in

this chapter can partly substitute the dynamic model, however, sections cannot be examined completely separately. It is clear that change in velocity of a section affects all parameters of following or previous section, depending whether we change  $c(k)$  or  $c(k - 1)$  of the  $k$ -th section. Problem of time calculation is, therefore, far more complicated. Changing the original velocity  $c(k)$  to new value  $c'(k)$  causes change of time of the  $k - th$  and  $(k + 1)th$  section and, hence, the total time changes by value  $\Delta T$ .

$$\Delta T = \frac{d}{c'(k) - c(k - 1)} \cdot \ln\left(\frac{c(k + 1)}{c'(k)}\right) + \frac{d}{c(k + 1) - c'(k)} \cdot \ln\left(\frac{c(k + 1)}{c'(k)}\right) - t(k) - t(k + 1) \quad (4.30)$$

Where  $t(k)$  and  $t(k + 1)$  are the original time associated to the original velocity  $c(k)$ . There is no possible way to get inverse function as  $c(k) = f(\Delta T(k, k + 1))$ . For this purpose, we must use numerical methods to obtain the value of inverse function.

## 4.6 Numerical methods

Numerical analysis provides equation solving without an analytical solution or equation where analytical solution requires high computing performance. These methods are largely based on linear approximation combined with iterative calculation to achieve minimal deviation. Taking into consideration the type of problematical equations and the simplicity of their derivations, we can choose Newton-Raphson method. The advantage of this method is that it provides very precise solution in a few iterative steps, which leads to a higher performance of optimization algorithm.

### 4.6.1 Newton-Raphson method

As was mentioned in the previous section, most of the numerical methods use linear approximation. In the case of Newton-Raphson method, the approximation is a tangent of the observed function. The principle of this method is clear from the graph in the figure 4.2, where we start with the initial guess  $x_0$  approximating the function by tangent  $f'(x_0)$  to get the result of the first iteration  $x_1$ , which serves for further calculation as an initial guess.

The function in figure 4.2 has only one variable  $x$ , however, the method naturally works with a set of functions of multiple variables. Implementation of this method for a general function of  $n$  variables is hard to represent graphically, thus, we rather use mathematical representation. We form the matrix of equation as:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{y} \quad (4.31)$$

First of all, we must come up with some reasonable guess  $\mathbf{X}_0 = [x_{10}, x_{20}, \dots, x_{n0}]$  as close to the solution as possible.

In the second step, we calculate the value of the function for the guess established in the previous step.

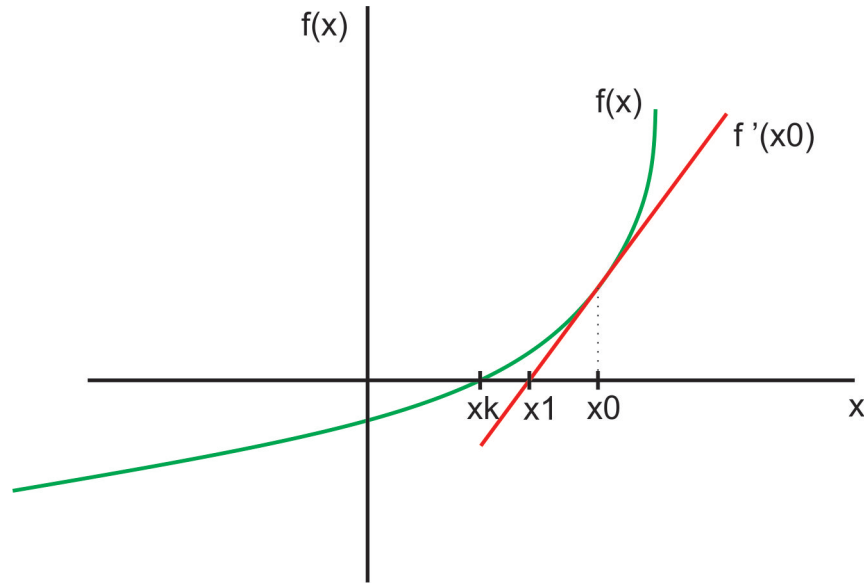


Figure 4.2: Newton-Raphson method principle

$$\mathbf{f}(\mathbf{x}_0) = \mathbf{y}_0 \quad (4.32)$$

Now we can calculate the deviation from the desired value as

$$\mathbf{D} = \mathbf{y} - \mathbf{y}_0 \quad (4.33)$$

In the third step, we calculate the Jacobian matrix which refers to the derivatives of all the functions and variables.

$$\mathbf{J} = \frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{df_1(x_1, x_2, \dots, x_n)}{dx_1} & \dots & \frac{df_1(x_1, x_2, \dots, x_n)}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{df_n(x_1, x_2, \dots, x_n)}{dx_1} & \dots & \frac{df_n(x_1, x_2, \dots, x_n)}{dx_n} \end{bmatrix} \quad (4.34)$$

In the last step, we calculate solution  $\mathbf{x} = [x_1, x_2, \dots, x_n]$

$$\mathbf{x} = \mathbf{x}_0 - \mathbf{J} \cdot \mathbf{D} \quad (4.35)$$

If the precision is not sufficient, we can set  $\mathbf{x}_0 = \mathbf{x}$  and continue in the next iteration from the first step until the deviation  $\mathbf{D}$  is in the desired band. Numerical calculations in this project are always solving one equation of one variable and thus we can use all the previously mentioned equations substituting the matrices by single variables [4].



---

# Optimization

## 5.1 Introduction

The reason to find optimal speed profile along a given track was already mentioned. Relation between the given track, speed profile and resultant energy was also defined earlier in the thesis. First we have to think about important facts, which can help us to determine the exact optimization problem. As experience says, slower the vehicle goes the less energy is consumed. Although it may not be clear from established models, this fact is not always true. As the motor is represented by efficiency map, we can see that operating in small values of revolutions or torque is connected to low efficiency. Once the saved energy from decreasing velocity falls under the increase of consumption caused by changing operational point and efficiency, the slower motion is no longer efficient. However, the optimal efficiency map is the subject of motor design and we will therefore omit these marginal cases. Still, from what was said, we must add a very important constraint, which will assign each speed profile to a group of equivalent solutions. This constraint is arrival time and we would like to find optimal speed profile from set of profiles with the same arrival time and the lowest value of consumed energy.

## 5.2 Optimization problem

The objective function of optimization of the energy consumption of the electric vehicle on the given track is, as expected, the function of the energy calculation subject to the constraints given by powertrain limits. The powertrain limits were introduced earlier in this thesis as power  $P_{max}$ , torque  $M_{max}$  and revolutions. Moreover, the vehicle speed is limited by speed limit  $v_{max}(k)$ . These limits are determined by traffic signs and comfortable centrifugal acceleration mentioned in track chapter 2. The motor revolutions are related to the velocity of the vehicle, thus, if the motor is designed well, the power and torque limits are exceeded earlier than revolutions limit. In addition, the acceleration applied on the passengers is not limited only in lateral, but also in forward direction. The value of maximum forward acceleration is, as in the case of lateral acceleration, completely subjective and is based on demands on the comfort of drive. For energy model, the optimization problem can be formed as:

$$\begin{aligned}
& \underset{v(k)}{\text{minimize}} && E(v) \\
& \text{subject to} && v(k) \leq v_{max}(k) \\
& && M(k) \leq M_{max} \\
& && P(k) \leq P_{max} \\
& && a(k) \leq a_{max} \\
& && T \leq T_{max}
\end{aligned}$$

The variable  $k$  refers to the index of track section. Optimization methods routinely solves quadratic problems, which would be exactly the objective function of energy on wheels [1]. However, as is introduced in powertrain section 3.4, the energy drained from the battery is adjusted by the power dissipation in the powertrain obtained from the efficiency map. This efficiency map represents huge non-linearity, apparent in figure 5.1.

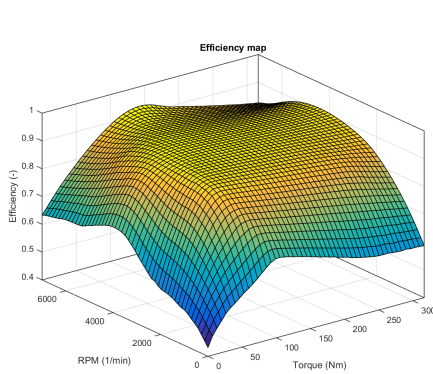


Figure 5.1: Efficiency map

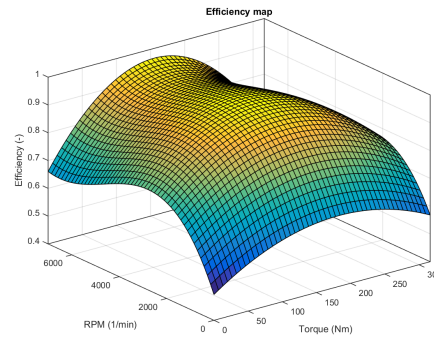


Figure 5.2: Polynomial surface of the efficiency map

There are two possible options to deal with this problem. First, to fit the efficiency map by a polynomial surface 5.2. The higher the degree of the polynomial is, the more precise results are obtained. However, to achieve the objective function fitting requirements of optimization techniques, the polynomial surface function must be as simple as possible. These two requirements diverge and we have to choose between precision of calculations and simplicity of the objective function, moreover, the efficiency map is already such a big simplification that more decrease in precision would make the model almost invalid. Second possibility is to leave the optimization and assemble heuristic algorithm. The disadvantage of this option is that the algorithm might easily find local minimum where it could get stuck and there is no possibility to determine whether the minimum is local or global. Despite of this, we will rather choose this option as no extra precision compromises must be done.

## 5.3 Heuristic algorithm

### 5.3.1 Introduction

Heuristic algorithms are based on various ways of searching for the best result of possible solutions. The algorithm starts with initial solution trying to improve the result in iterative steps. The new solution can be chosen completely random or it can be chosen by logic based

on previous steps. The logic choosing of the next step determines the heuristic method. Unlike the optimization methods, the heuristic methods do not have strict procedure and they can be easily modified to improve performance of the algorithm. Therefore, we will introduce custom assembled algorithm inspired in existing heuristic methods, which fits the character of objective function subjected to given constraints.

### 5.3.2 General scheme

The idea of the algorithm is to evaluate each point of the track in the domain of the objective function, which is the energy. The evaluation is done by calculating the demands of the track points as a result of velocity changes in these points. The velocity changes are not constant, and they are calculated to bring about constant change in arrival time  $\Delta T$ , for each track point.

#### 1. Determine time step

The introduced time step  $\Delta T$  is designed as constant in order to make all points demands equivalent. Consequently the time step represents change in the velocity on the section. Hence, speed in each point is recalculated for  $\Delta T$  and  $-\Delta T$  in respect to the fact that changing of  $k$ -th velocity  $c(k)$  changes section time  $t(k)$  and  $t(k-1)$ . These facts are clear from figure 5.3.

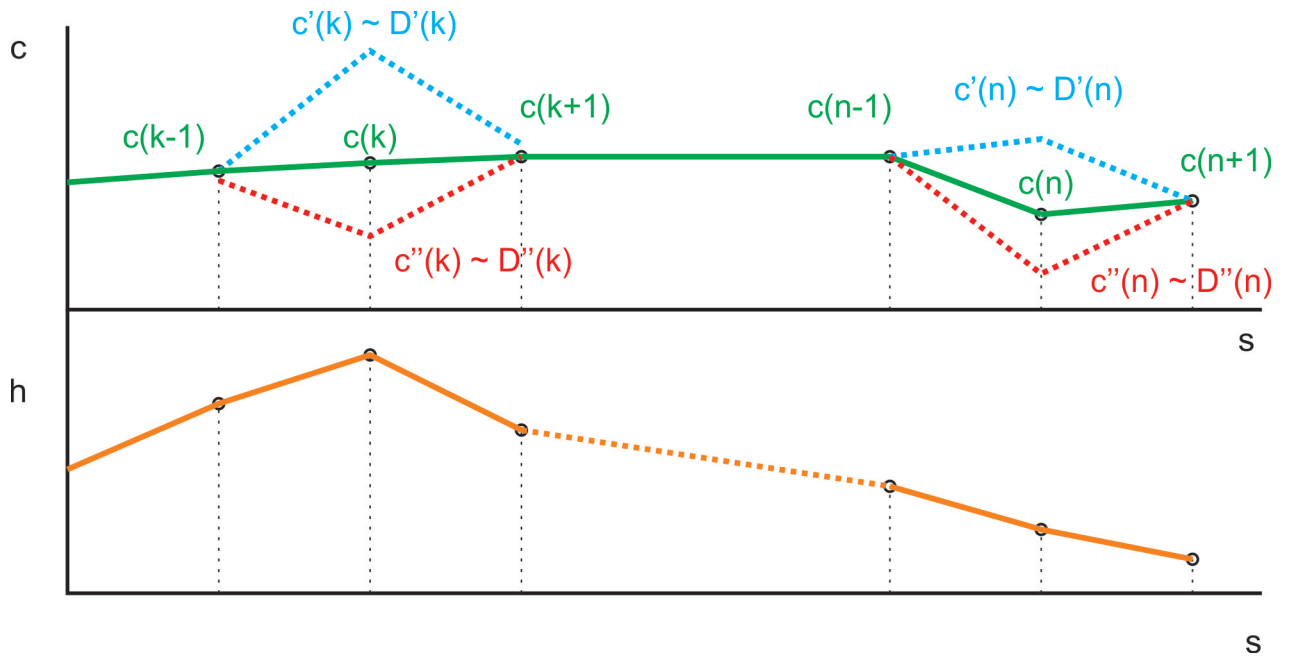


Figure 5.3: Speed profile recalculated in respect to the time step.

As mentioned, the value of velocity  $c'(k)$  and  $c''(k)$  is calculated in respect to validity of following equations:

$$t(k-1) + t(k) - \Delta T = t'(k-1) + t'(k) \quad (5.1)$$

$$t(k-1) + t(k) + \Delta T = t''(k-1) + t''(k) \quad (5.2)$$

Where  $t(k)$  refers to the velocity  $c(k)$ ,  $t'(k)$  refers to the velocity  $c'(k)$  and  $t''(k)$  refers to the velocity  $c''(k)$ . Velocities  $c'(n)$ ,  $c''(n)$  are calculated in the same way, and all points in general.

## 2. Evaluate points of the track by energy demands

Energy demand  $D'(k)$  of  $k$ -th point is calculated as the difference between energy in consideration to velocity  $c(k)$  and energy in consideration to velocity  $c'(k)$ :

$$D'(k) = (E(k) + E(k-1)) - (E'(k) + E'(k-1)) \quad (5.3)$$

The energy demand  $D'(k)$  would be enough to evaluate the  $k$ -th point when we consider the slope of the tangent:

$$\frac{D'(k)}{\Delta T} = \frac{D''(k)}{-\Delta T} \quad (5.4)$$

However it is useful to calculate also energy demand  $D''(k)$  in consideration to velocity  $c''(k)$ . The reason is that we get the exact value of the energy change, which can be used while recalculating the sections energy when the speed is changed. These energy demands are calculated in each point of the speed profile.

## 3. Variation of the most demanding and least demanding point

Once the energy demands are calculated, the most demanding and the least demanding points are chosen. If we consider the  $k$ -th point as the most demanding and the  $n$ -th point as the least demanding, we can determine whether changing velocity in these points brings about any improvement in energy consumptions:

$$D''(k) + D'(n) > 0 \quad (5.5)$$

If the inequality is valid, changing the velocity in the  $k$ -th point to  $c''(k)$  and the velocity in the  $n$ -th point to  $c'(n)$ , represents better solution of the optimization problem. Moreover, if the original speed profile observe the constraint  $T \leq T_{max}$ , then due to the constant value of  $\Delta T$ , the newly set speed profile meets the constraint automatically.

The scheme of the algorithm explains the simple idea of the heuristic algorithm. Nevertheless, the implementation of this idea is by far more sophisticated as all of the constraint must be validated and we also have to introduce a method to avoid getting stuck in the nearest local minimum.



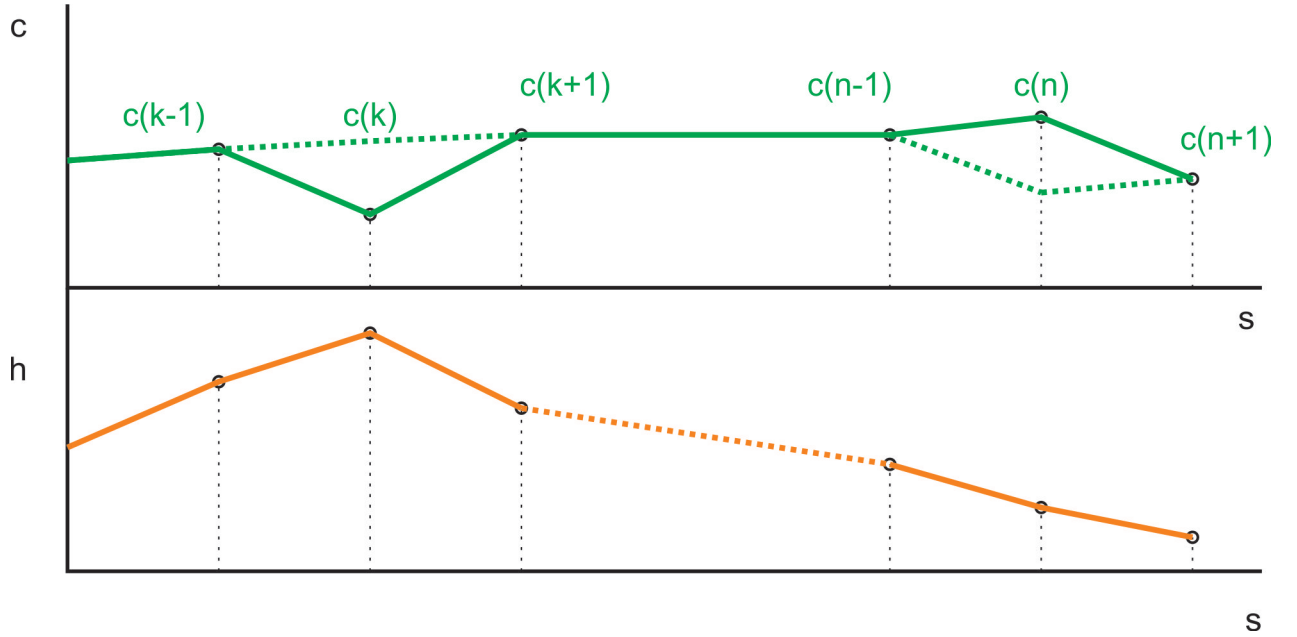


Figure 5.4: Resultant speed profile of one iterative step of the algorithm

### 5.3.3 Constraints validation

The track section demand is simply calculated as the difference between the energy of the original speed profile and the energy of the profile in consideration to the suggested velocity  $c'(k)$ . While determining the operation point of the powertrain on  $k - th$  and  $(k - 1) - th$  section, the torque  $M'(k - 1)$ ,  $M'(k)$  and power  $P'(k - 1)$ ,  $P'(k)$  are calculated, according to 4.20 and 4.18. These values can be directly compared to the constraint, same as the suggested speed  $c'(k)$ . The only value which needs more calculation is the acceleration  $a'(k - 1)$  and  $a'(k)$  as:

$$a'(k - 1) = \frac{c'(k) - c(k - 1)}{t'(k - 1)} \quad (5.6)$$

$$a'(k) = \frac{c(k + 1) - c'(k)}{t'(k)} \quad (5.7)$$

If one of the constraint is violated by the calculated values, the change of velocity  $c'(k)$  is forbidden and the section is removed from the process of velocity variation.

### 5.3.4 Local minimum prevention

A disadvantage of heuristic algorithms is getting stuck in the local minimum. As a matter of fact, the optimization problem 5.2 introduced in this thesis is almost insolvable by the assembled algorithm, because of the local minimums. The track data and speed profile are sampled with a very dense grid, and as all constraints are overseen in every iterative step of the algorithm, the change in the speed profile is extremely marginal, therefore, the algorithm tends to get stuck in the local minimum. To deal with this problem, we could admit temporary violation of the constraints, which would be treated after more iterative steps, or we could reduce the speed profile sampling grid to cause major changes without violating any

constraints, but we would still make all calculations, primarily verification of constraints, in consideration to the dense grid. The grid can be progressively softened to the original density. This technique is apparent in figures 5.5 and 5.6

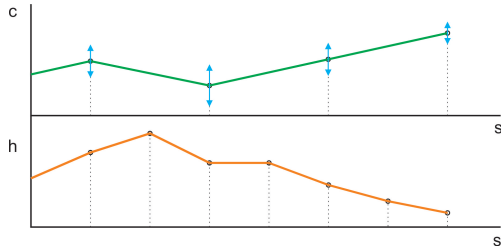


Figure 5.5: Reduced grid

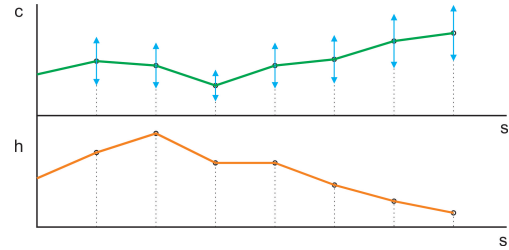


Figure 5.6: Soften grid

The method of grid reduction is rather chosen as each iteration provides valid solution, and major changes improve the approach to the global minimum. Unfortunately, this method does not allow us to start with any previously achieved solution and the speed profile must be build up from very beginning. The speed profile is preferably built up gradually by setting intermediate time constraint. The algorithm can be then described by the following scheme, starting with few points of speed profile grid and considering energy demand evaluation associated to the time step  $\Delta T$  in each of these points:

#### 1. Set intermediate time

Strategy of the intermediate time setting determines how fast the algorithm will reach the original time constraint, or how dense the grid is when the original time constraint is reached. In each iteration of the scheme, the intermediate time is reduced.

2. **Speed up until the intermediate time reached** Velocity of the least demanding sections is repeatedly increased in a cycle until the intermediate time is reached. Each iteration of this inner cycle decreases the arrival time by the time step  $\Delta T$ , and all constraints are treated as in the case of section variation.

#### 3. Find intermediate solution

After the intermediate time is reached, the heuristic algorithm finds, for the current grid, the least demanding solution according to the previously mentioned scheme.

#### 4. Soften the grid

The grid is softened by splitting each section in half. This step requires recalculation of the demands of all the newly established sections. This operation decreases the ability of making major changes, however, the closer to the final solution we are, the less this ability is required.

#### 5. Find intermediate solution

After the grid is softened, the previously established solution is no longer considered as the least demanding, therefore, the least demanding solution must be found by the same algorithm as in the third point of this scheme.

One would deem point 3 or 5 as redundant, but keeping all intermediate solutions as close to the optimal solution as possible turns out to be crucial. The scheme runs in cycle, and once the intermediate time is equal to the time constraint, the point 2, the speed up, is omitted and the algorithm just applies the variation of the section velocity by their demands and softens the grid until the original grid density is reached. We can see, the first scheme is implemented in the second scheme and in either case, once most of all sections are forbidden by constraints, the time step  $\Delta T$  may be arbitrarily changed.

## 5.4 Reference drive

To evaluate the result of the optimization of algorithm, the reference drive must be established, in order to simulate the behavior of the drive. We would like to assemble an algorithm, which has at least the same results in consumption as a real driver, or better. This is achieved by a control algorithm based on the set of rules established from experience.

### 5.4.1 Control algorithm

The control algorithm operates with energy model, thus, the velocity consists of sections. The vehicle can accelerate, keep the velocity, glide or brake. This forms a set of commands, and on every section one of these commands is applied. Each of the commands represents calculation of the velocity of the endpoint of a section.

1. To **accelerate A**, the velocity is increased by the highest possible rate. The rate of acceleration is limited by the maximum torque and the power of the motor and also by the maximum acceleration given by demands of the comfort of the drive. To determine the velocity of the endpoint of a section as result of acceleration, each of the limiting factors must be considered and the velocity of the endpoint of a section is calculated separately as a result of maximum torque, power and acceleration applied on the section. From the set of these three velocity values, the lowest one is selected as this value represents acceleration to the limit of the vehicle or comfort.
2. To **keep the velocity K**, the velocity of the startpoint is set to the endpoint of a section. However, this value must be validated by limits of the motor. If one of the limits is exceeded, the value of the velocity of the endpoint is recalculated to the limit.
3. **Glide G** represent drive with no power of the powertrain. Hence, the value of velocity of the endpoint is calculated in respect to the zero value of the energy of the section.
4. **Braking B** is an inverse command to accelerating. Braking is used to avoid speed limit violation. To accomplish this, the algorithm is using prediction in control. The prediction calculates a few points of the track in advance using the vehicle model with brake command. The results of this simulation are used to determine the current action. As the vehicle has velocity on the  $k - th$  section, the velocity for the following section  $v(k+1)$  must be determined in respect to the speed limit of the following sections of the track, which must not be violated during braking. Hence, the desired velocity  $v(k+1)$  must be validated by simulation of braking from this value of velocity on the following section, until the vehicle is stopped. If the vehicle stops on the  $n - th$  section, it is obvious that changing velocity on  $(k+1) - th$  section to the value  $v(k+1)$  affects only sections of band  $\langle k+1, n \rangle$ .

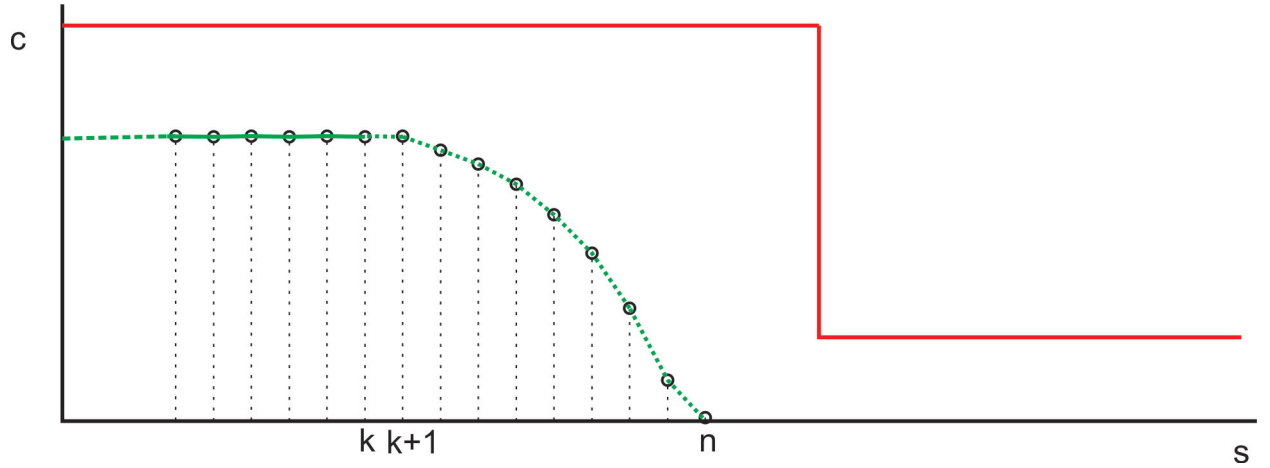


Figure 5.7: Brake test to validate velocity of the point (k+1)

However, violation of the speed limit of any section in the band brings about restriction of the value of velocity  $v(k + 1)$ . If the violation of the speed limit occurs on the  $l - th$  section, the velocity of the vehicle is set to the value of the limit on this section and the braking is reversely simulated from this point back to point  $k + 1$ . By this process, the value of the velocity  $v(k + 1)$  is determined and it is guaranteed that the speed limit on the  $l - th$  section is not violated, but the vehicle will have the velocity  $v(l)$  equal to the speed limit at this point. This is apparent from figure 5.8.

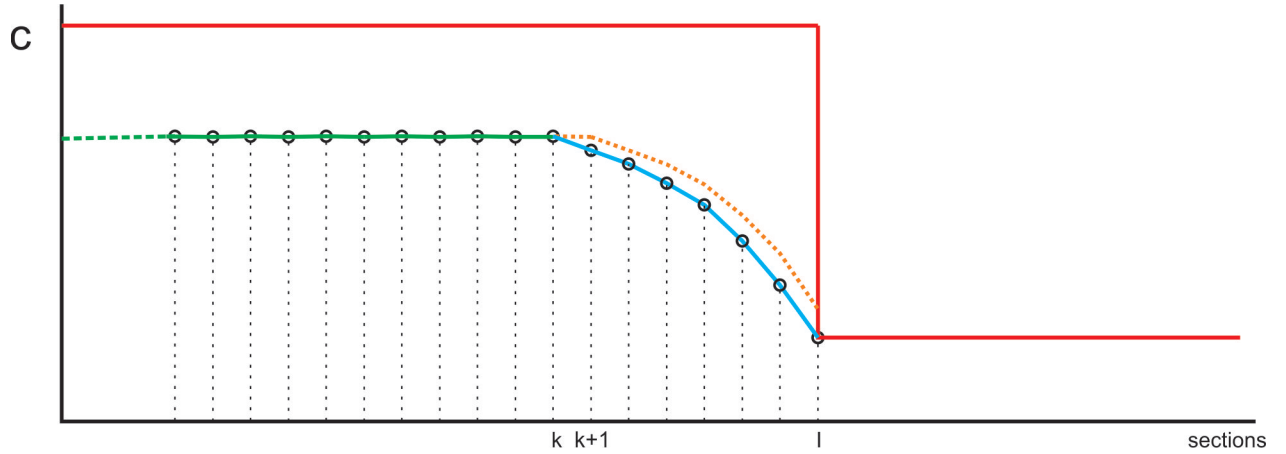


Figure 5.8: Inverse speed profile construction as speed limit violation occurred

Implementation of the control algorithm is based on a simple idea. As mentioned, the slower the vehicle moves the less energy is consumed. If we start with this supposition, the most efficient would be to drive with a constant value of velocity all over the track. Hence, the average velocity  $\bar{c}$  is determined by the desired arrival time  $t_{des}$  and total length of the track  $s$  as:

$$\bar{c} = \frac{s}{t_{des}} \tag{5.8}$$

Traveling slower than the average velocity requires increasing the velocity on the other parts of the track to attain the same arrival time. However, as the energy consumption is non-linearly proportional to the traveling velocity, one can determine the saved energy on the part of the track where the velocity is lower than average as lower than spent energy on the part of the track where the velocity must be higher than average. The most efficient is considered to travel at the average velocity. Then, the control algorithm is formed by simple logic. When the vehicle moves slower than the established average velocity, the command is accelerate. Once the average velocity is reached, the vehicle keeps this speed. When moving downhill or the velocity is higher than average, the vehicle glides, however, when the suggested velocity is lower than average, the vehicle keeps the average velocity. Every suggested velocity  $c(k+1)$  must be validated by the brake test introduced earlier. The brake test is the superior command in order not to violate the speed limit.

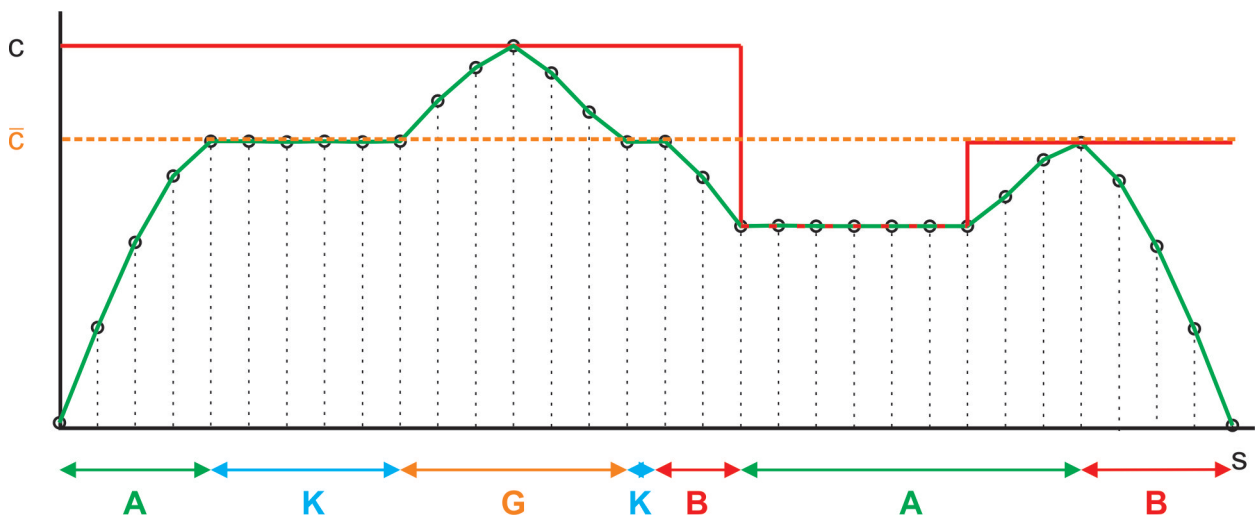


Figure 5.9: Example of reference control algorithm behavior.

As the resultant velocity is not equal to the established average velocity on every section, due to acceleration and braking, the arrival time related to the resultant speed profile differs from the desired time. The average velocity value must be properly recalculated and the speed profile is, therefore, built up in more iterative steps of the algorithm. The algorithm provides speed profile with exact arrival time without violating any constraints formed earlier. Hence, we can consider the result, clearly not the optimal one, as the reference drive simulating driver behavior.



---

# Programming

## 6.1 Introduction

The project is developed in the Matlab environment. As it is clear from the established tasks, the project is a large-scale, and it is crucial to use OOP (object-oriented programming) to keep the project well-arranged and easily expandable. The project is divided into the computer part, graphical user interface and data gathering part. However, it is not possible to obtain GPS data directly from the Matlab, so the data gathering part must be developed in HTML and javascript languages, then implemented in the Matlab. In the following chapters, each of these parts of the project is described separately. All codes are included on enclosed CD, therefore, this chapter describes difficulties and samples of code characterizing principles.

## 6.2 Computer parts

The computer part provides all calculation done on a given track and vehicle. The heuristic algorithm is implemented as well as reference drive control algorithm. The computer part of the project consists of the following classes:

- @Drive - superclass defines the project and associates the other classes.
- @Vehicle - defines the character of the vehicle and environment by constants.
- @Track - gathers all data of the given track
- @Speed - specifying the speed profile and implements driver simulation controller
- @Energy - provides all calculation of the energy and time
- @Optim - implements the heuristic algorithm introduced in the thesis

As mentioned, the @Drive is the superclass of all computer classes, which represent the properties of this superclass. The vehicle, speed and track classes are defined as handle class, in order to have direct access to the vehicle, speed and track information in every class and keep the only instance of these objects. The handle class is the superclass for all classes that follow handle semantics. [7] Handle represents a reference to an object and it is simply passed to every single class as a property to provide direct access and keep the object as unique.

```
classdef energy < handle
    % The energy class provides calculations of total energy and its
    % components on given track and with given speed profile

    % Energy class properties
    properties (SetObservable, AbortSet)
        drive;           % drive object handle
        track;           % track object handle
        vehicle;         % vehicle object handle
        speed;           % speed object handle

        potential;       % potential energy component
        kinetic;          % kinetic energy component
        drag;             % drag energy component
        roll;             % roll energy component
        total;           % total energy consumed
    end

    % Energy class methods
    methods
        % Energy class constructor
        energy(drive)

        potentialCalc(energy) % potential energy component calculation
        kineticCalc(energy)  % kinetic energy component calculation
        dragCalc(energy)     % drag energy component calculation
        rollCalc(energy)     % roll energy component calculation
        calc(energy)         % total energy calculation
        timeCalc(energy)     % time calculation
    end
end
```

Listing 6.1: Energy class definition.

Close view on each of the classes and their methods is useless as these methods provide pure calculations described earlier in the thesis. In listing 6.1 there is an example of the Energy class with properties and methods. Some of the properties represent the handles of the objects of drive, track, vehicle and speed for simple access to data necessary for calculations in the class methods. The Energy class itself is also the handle class.

### 6.3 Graphical user interface

The issues of GUI (graphical user interface) in Matlab is described in many publications, however, almost all of them use a built-in graphic editor GUIDE, which is not optimal for larger-scale projects. Hence, advanced GUI technique is introduced to create a well-arranged and easily expandable GUI. These two demands lead to the use of OOP, as in the case of computer classes. In the GUI, each of the classes represents one window with its graphical objects. To explain the technique, the most simple window of the object is selected.

The window in figure 6.1 welcomes the user when the application is started up. There are only four graphical objects in the window, three buttons and a title. This window represents class @startup of the project in interface package folder +interface. The class definition with its typical properties and methods are clear from listing 6.2.



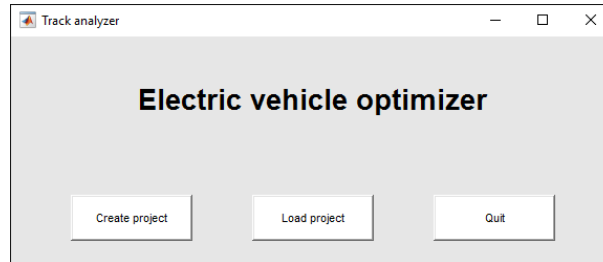


Figure 6.1: Start up application window

### 6.3.1 Properties

Each of the classes of GUI includes a set of typical properties to simplify the handling of the window:

- The **EVO** property in listing 6.2 specifies the project where the window belong. This superclass connects the interface to the computer classes.
- **position** determines where the windows appears when opened. Value of the position is either normalized or in pixels.
- **handles** is a structure with handles of every graphical object of the window and their children. The handle structure allows the developer comfortable changes of any object or property.
- **isOpen** represents open flag to determine whether the window is opened or not.

```

classdef startup < handle
    % Startup window class
    % Provides user to make selection between opening existing project or
    % create new one

    properties
        EVO                % superclass of the project
        position            % position of the windows
    end

    properties (SetAccess = protected)
        handles            % handles of all graphical objects
        isOpened           % windows opened flag
    end

    methods
        % Constructor of the class
        function obj = startup(objEVO)
            obj.EVO = objEVO;
            obj.position = [.35 .4 .3 .2];
        end

        open(startup)                % create the figure with graphical objects
        closeRequestFcn(startup)      % close window request callback
        createProject(startup)        % open project callback
    end
end

```

```
        loadProject(startup)                % load project callback
    end
end
```

Listing 6.2: Definition of the startup class

## 6.3.2 Methods

As in the case of properties, the classes of GUI have also characteristic methods. The input of the class method is always the handle of the object to allow the method to manipulate and access properties and to call other methods. The most important method is the constructor of the class, where the object is assigned to the project. The class also includes callbacks of the graphical objects.

### 6.3.2.1 Open method

Open is the most important of all the methods. This method opens the figure and create all graphical objects. As we can see in listing 6.3, figure is opened first and graphical objects are one-by-one assigned to the figure with proper properties. Handle of each object is inserted in the structure. Once all the graphical objects are created, callbacks are assigned easily. Finally, the handle structure is referred to the class handle property and open flag property is set to 1.

```
function open(startup)
    if (startup.isOpened)
        figure(startup.handles.figure)
    else
        % Create figure
        handles.figure = figure(...
            'Name','Track analyzer',...
            'Units','normalized',...
            'Position',startup.position,...
            'NumberTitle','off',...
            'MenuBar','none',...
            'Tag','startup', ...
            'Color', startup.EVO.gui.figureBackgroundColor, ...
            'HandleVisibility','off');

        % Create button object
        handles.createButton = uicontrol(...
            'Parent', handles.figure,...
            'Style','pushbutton',...
            'String','Create project',...
            'units','normalized',...
            'BackgroundColor',startup.EVO.gui.buttonBackgroundColor,...
            'ForegroundColor',startup.EVO.gui.buttonFontColor, ...
            'Position',[.1 .1 .2 .2]);

        %Callbacks
        handles.createButton.Callback = ...
            @(a, b)startup.createProject;

        % Assign handles and set the open flag
        startup.handles = handles;
```

```

        startup.isOpened = true;
    end
end

```

Listing 6.3: Example of open method of the startup application window.

The code published in listing 6.3 is not a complete open method of the startup window. However, the principles of the open method are represented enough. The handle structure of complicated windows requires to sort handles to sub-structures of buttons, axes, lines, etc. for better organization.

## 6.4 Data gathering

Nowadays, the track data is easily measured by most of smartphones and exported to various file formats like gpx, kml, csv, etc. Matlab has built-in functions to read these files, thus, gathering data by this method brings about no problems, and importing these files is obviously a part of the application. However, the idea of gathering data in advance is not a convenient solution, hence, the application provides possibility to gather the data from web servers. As mentioned earlier in the thesis, Google maps service is chosen for this purpose. Direct implementation in Matlab is not possible and there are no toolboxes allowing this, hence, the map must be implemented by a working around it.

### 6.4.1 Google maps

The Matlab has a built-in web browser with a special feature allowing partial interaction between web pages and Matlab. Hence, we can start with a web application for the purpose of the track design. The web application includes HTML part, where the layout of the page is designed, and JS part, where all functions of the page are performed.

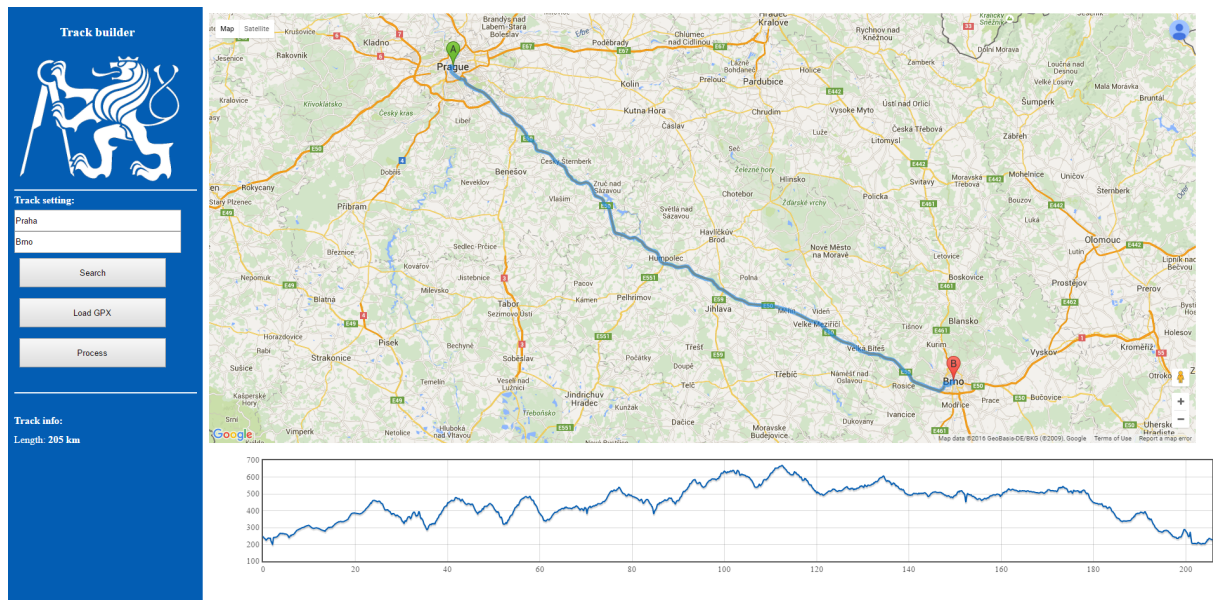


Figure 6.2: Web application for track design

As we can see in figure 6.2, the web page includes a few objects, and the HTML script is simple as well, however, the JS code is far more complicated. Google maps service distin-

guishes several interfaces to obtain different geographical data. To gather data sufficient for the calculation, the application uses Google Maps Directions API to design the track and collect GPS coordinates, and Google Maps Elevation API to collect elevation level of the track. Both of these interfaces use JavaScript Object Notation (JSON) for input and output, and the use of the interface is quite comfortable. Moreover, complete documentation is provided, including examples of use. The use of Google Maps API is, for free users, limited in number and length of the request. Therefore, the JS must split the data obtained from Direction API to more requests for Elevation API and compose the result back.

### 6.4.2 Transferring data

The web application is opened in Matlab browser. Unlike conventional browsers Matlab browser have special features to manipulate the Matlab console from JS of the web page. This can be conveniently used to transfer data from JS to Matlab workspace. In listing 6.4 a sample of JS code to display values of elevation to Matlab console is seen.

```
function pullData(){
    var strElev = '';
    strElev = elevations.join(' ');

    document.location = "matlab:set (disp(" + strElev + "));"
}
```

Listing 6.4: Javascript method manipulating with the console of Matlab

This function is only available in Matlab browser, and it is similarly used to assign these values to a variable. However, to pull up the data, the JS function must be called asynchronously from Matlab. This part is not documented in any official documentation, but it is possible. When creating the web browser in the application, it is crucial to keep the handle. The web browser has hidden method, which is called, similarly to the previously mentioned function, to execute a command in the console of the browser.

```
wizard.handles.webBrowser.executeScript('javascript:pullData()');
```

Listing 6.5: Matlab method manipulating with the console of web browser.

Thanks to the method presented in listing 6.5, there is no more difficulties to implement data transfer in either directions. Both methods use a command in the form of string, however, if the string is designed well, the data are accepted directly in numerical type, and there is no need of data type conversion.

# Application

## 7.1 Introduction

The application is designed regularly to fit the character of the given task and the used solution. It provides creation of a project with a design of a track and a setting of parameters, optimization of a speed profile along the track, and simulation of a drive with direct visulisation. Export of results and import of data is naturally included.

## 7.2 Project wizard

The project is created in 4 steps by classical wizard.

1. The name of the project and location is selected. In this step, the type of the project is also selected to determine the regular speed limit (city, countryside, highway)
2. The track is designed in the implemented web page including Google maps. This step also provides importing GPS coordinates of a track from gpx file in a standard form.

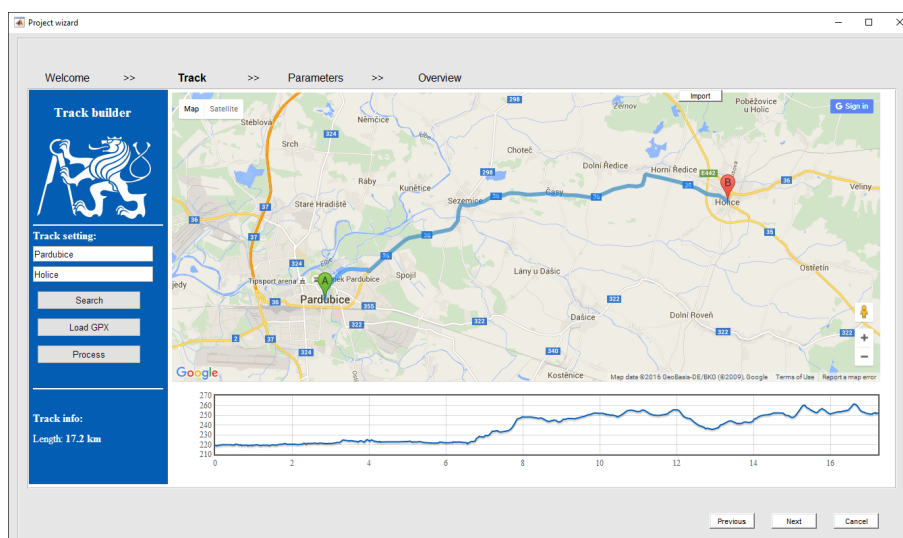


Figure 7.1: Implemented web application for track design.

- The complete parameters of vehicle and enviromental conditions are specified in this step, including the import of the efficiency map from Excel document. Example of the document structure is enclosed on CD.

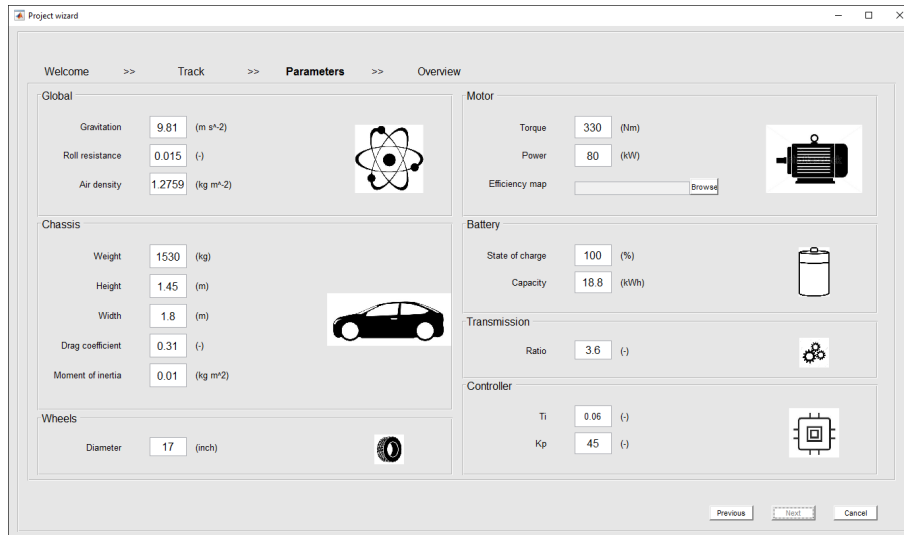


Figure 7.2: Specification of the vehicle parameters

- Finally all inserted data are displayed to check with the calculated speed limits. The limits may be imported or added from the Excel file, or it can be adjusted to the given speed profile. Once the wizard finishes, the project file is created with the designed track and given parameters, and the project is opened in the project viewer.

### 7.3 Project viewer

The project viewer is the main part of the application. The window 7.3 allows us to view the given track, run the desktop real-time simulation of the drive and observe the current track and motor values. The simulation runs in the background in simulink model based on introduced dynamic model calculation. Different speed profiles are chosen in the control panel on the left side of the viewer, in order to evaluate the results of different drive strategies obtained from the optimization algorithm, driver simulation or imported from file. The menu in the left upper corner provides common project operations as save, load and create new project. Moreover, the results of the simulation may be exported to csv. The option to change track, or parameters of the vehicle and environment is available only during the creation of the project for very simple reason. Once any of the project parameters is changed, the results given by optimization of the algorithm are invalid. This is apparent in the case of motor torque change. If the calculation of the optimal speed profile consider a value of torque, which would be subsequently reduced, the vehicle may no longer manage to pass the track with the prescribed speed profile. Therefore, changing any parameter makes the project worthless and it is recommended to create a new project, instead.



Figure 7.3: Project viewer with running simulation.

## 7.4 Optimization tool

This part of the application represents the core of the thesis, the optimization. It provides different suggestions of speed profiles calculated in respect to the energy model introduced earlier. There is also the possibility to import measured speed profiles from csv file. The control panel is situated on the left side with the possibility to change, add or delete a speed profile, set desired time and run one of the methods of speed profile suggestions. The approximate resultant time and consumed energy is calculated using the energy model. The axes situated on the right side display the speed profile and the speed limits along the track. When any of the calculations is running, the axes display the intermediate steps of the algorithm, which make the optimization algorithm very illustrative. Every created speed profile in this tool is transferred to the project viewer to be run in the dynamic model.

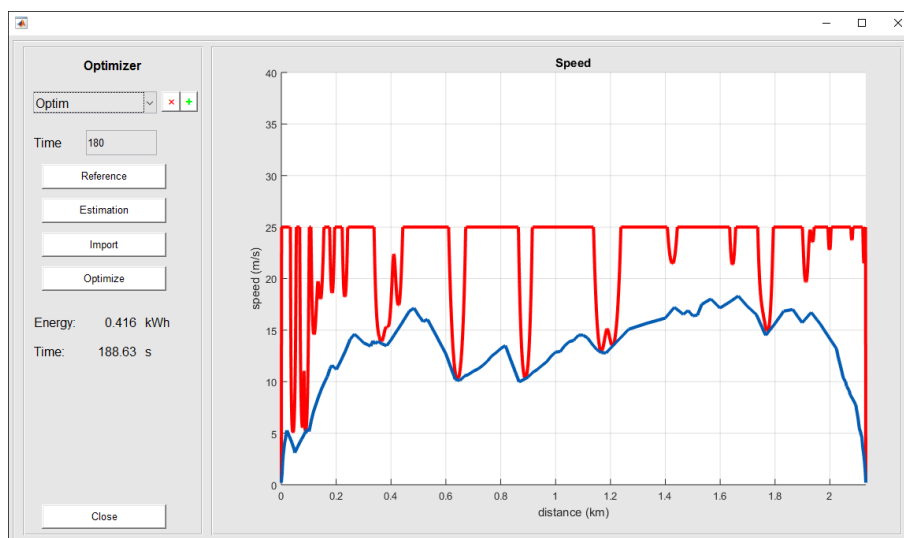


Figure 7.4: Optimizer tool of the application.

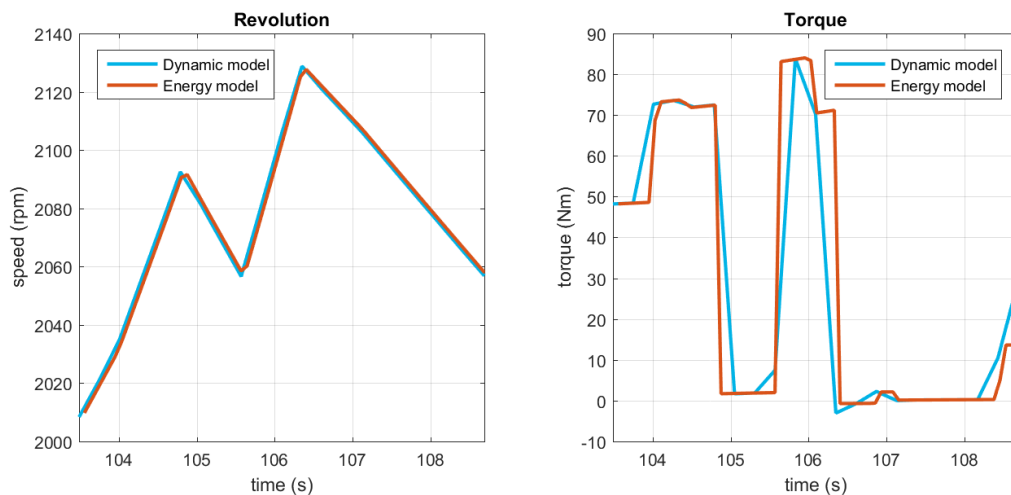




# Results

## 8.1 Model comparison

The principle of simplification introduced in energy models brings about the difference between results produced by dynamical and energy model. Divergence of operating values is apparent in sample of drive in figure 8.1. The speed profile suggested by the using of energy model represents prescribed speed for the dynamic model, thus, the divergence is given by the quality of controller tuning. Power and torque are considered as constants by sections, which makes the difference, as their dynamic behavior is omitted. Therefore, the operational point of the motor is not accurately determined continuously during the drive, but as a constant by sections, although the value in dynamic drive continuously changes. These facts cause slightly different results between total energy of dynamic model and energy model. Nevertheless, the character of the vehicle of dynamical model is satisfactorily reflected by the energy model, and the result of the optimization algorithm can be considered as valid.



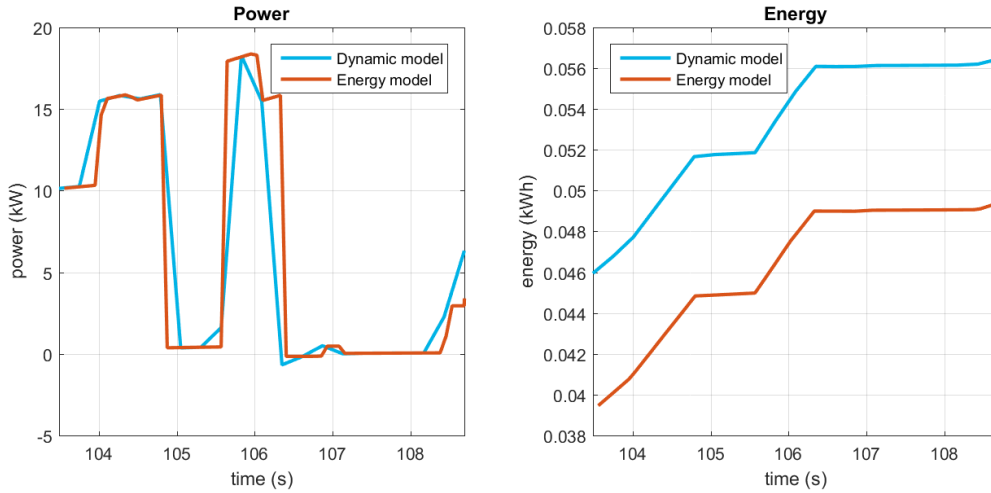


Figure 8.1: Comparison of the energy and dynamical model of the vehicle

## 8.2 Test drives

The final part of this thesis is devoted to the results of the application. The goal of the optimization algorithm is to produce energy saving, which is represented as the difference in consumed energy between the reference drive and optimal drive. The behavior of the average driver is introduced earlier as predictive control algorithm. Therefore, the result of this algorithm is considered as the reference drive, although an experienced driver would take a drive with less energy consumed. The amount of saved energy clearly depends on the chosen track, and to put the algorithm under test, tracks of different elevation profile and length are chosen.

### 8.2.1 Vehicle

The chosen vehicle and environmental condition are set as a constant for all testing drives. However, the application may also serve to evaluate the vehicle design in respect to energy consumption on a test track. The test vehicle is e-Golf with parameters obtained from the catalogue and efficiency map measured on powertrain of similar parameters. These parameters are listed in the table 8.1


e-Golf	Component	Parameters	Values
	<b>Chassis</b>	Weight	1530 (kg)
		Height	1.45 (m)
		Width	1.8 (m)
	<b>Wheels</b>	Drag coefficient	0.31 (-)
		Moment of inertia	0.01 (kg m <sup>2</sup> )
	<b>Motor</b>	Diameter	17 (inches)
		Torque	330 (Nm)
	<b>Transmission</b>	Power	80 (kW)
		Ration	3.6 (-)

Table 8.1: Parameters of the vehicle and environment.

## 8.2.2 Flat drive

The track of this drive is situated on lowland, with almost a flat elevation profile. The speed limit of this track is limited only in centrifugal acceleration omitting the traffic signs.

### 8.2.2.1 Track

Start (altitude)	End (altitude)	Length	Desired time
Pardubice (220 m)	Sezemice (223 m)	5 km	300 s

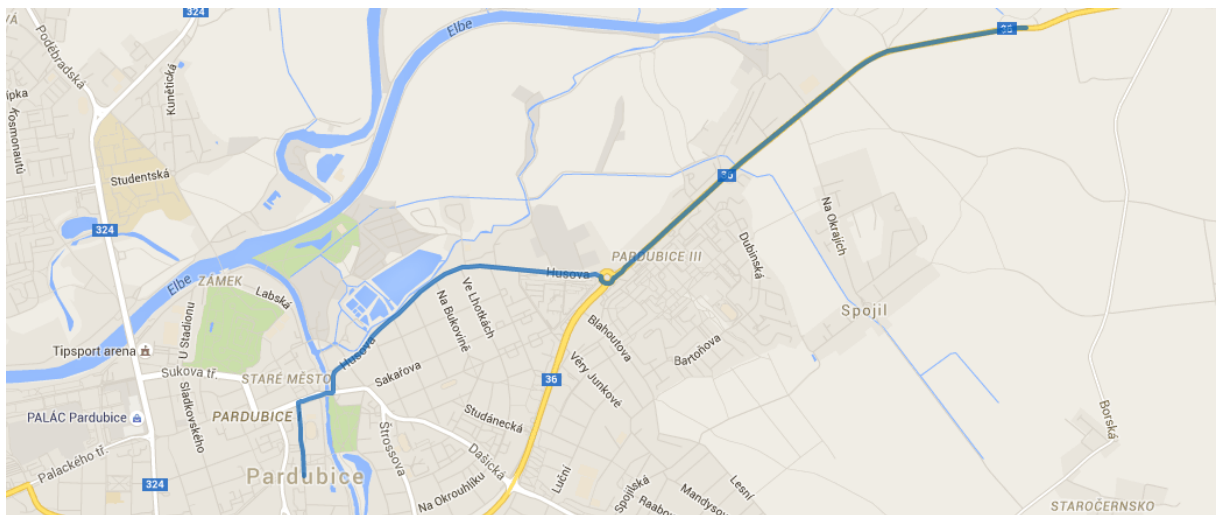


Figure 8.2: Map of flat drive track.

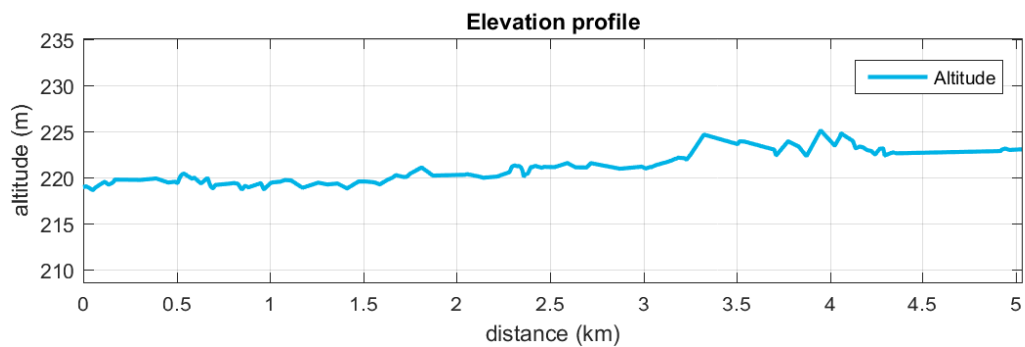


Figure 8.3: Elevation profile of the flat track.

## 8.2.2.2 Speed profile

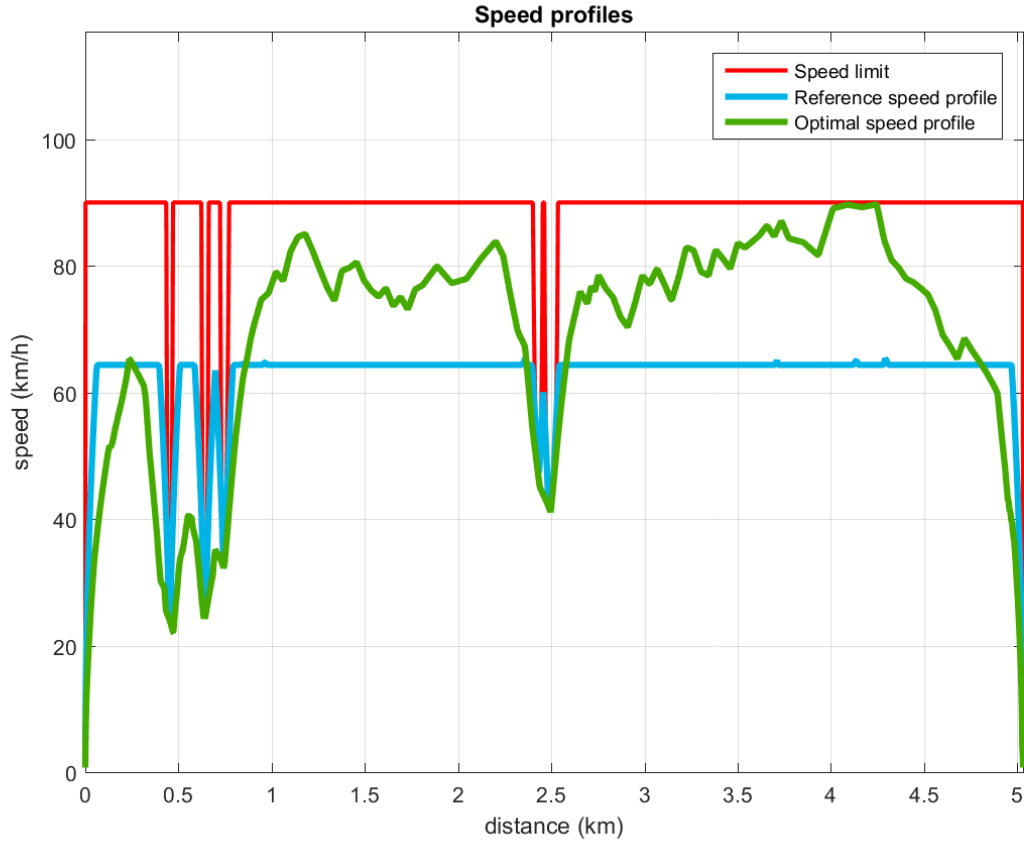


Figure 8.4: Suggested reference and optimal speed profile of the flat track drives.

## 8.2.2.3 Result

Results				
	Reference drive	Optimal drive	Saved energy	
Arrival time	299.4 s	298.9 s		
Energy model consumption	0.882 kWh	0.828 kWh	0.054 kWh	6.5 %
Dynamic model consumption	1.015 kWh	0.852 kWh	0.163 kWh	<b>19.1 %</b>

Table 8.2: Overview of results of the flat track drive strategies.

Speed profiles differ mainly in the distribution of speed within the speed limit drops. The braking is not so hard, and gaps between two neighbouring speed limit drops are neglected.

### 8.2.3 Hill drive

The track of this drive is situated on highland, with high level of road inclination. The speed limit of this track is limited in centrifugal acceleration and also in respect to the traffic signs.

#### 8.2.3.1 Track

Start (altitude)	End (altitude)	Length	Desired time
Snezne (700 m)	Kratka (705 m)	2.6 km	250 s

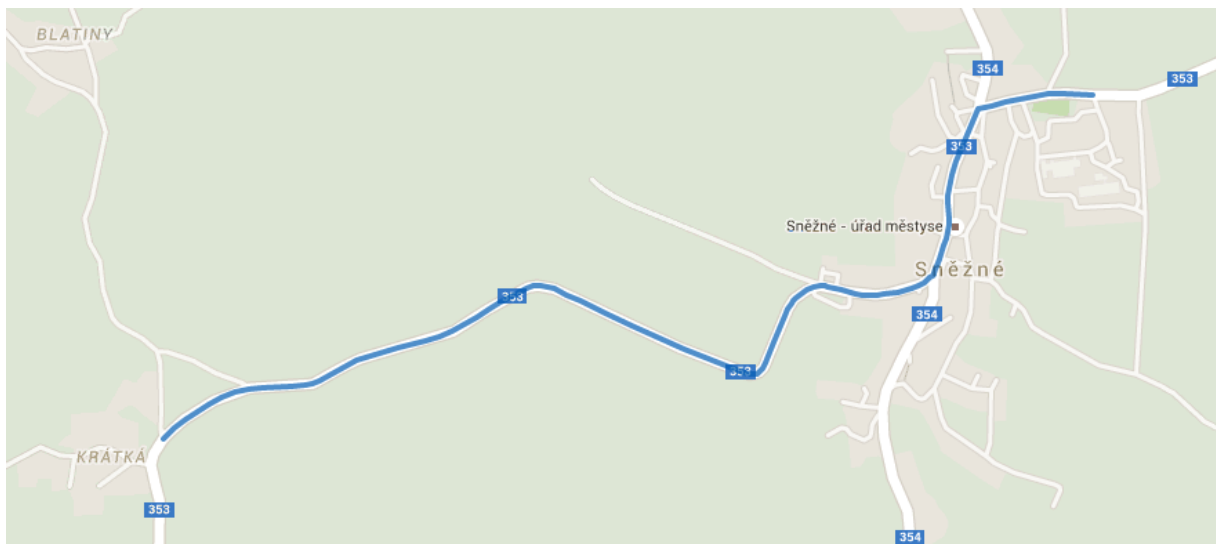


Figure 8.5: Map of hill drive track.

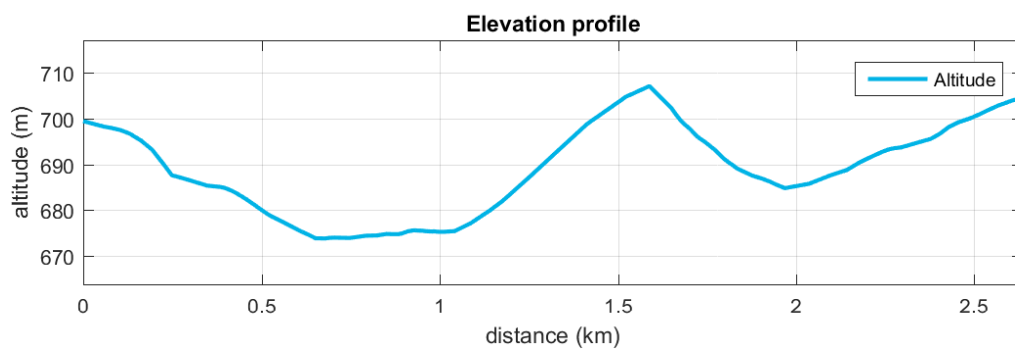


Figure 8.6: Elevation profile of the hill track.

## 8.2.3.2 Speed profile

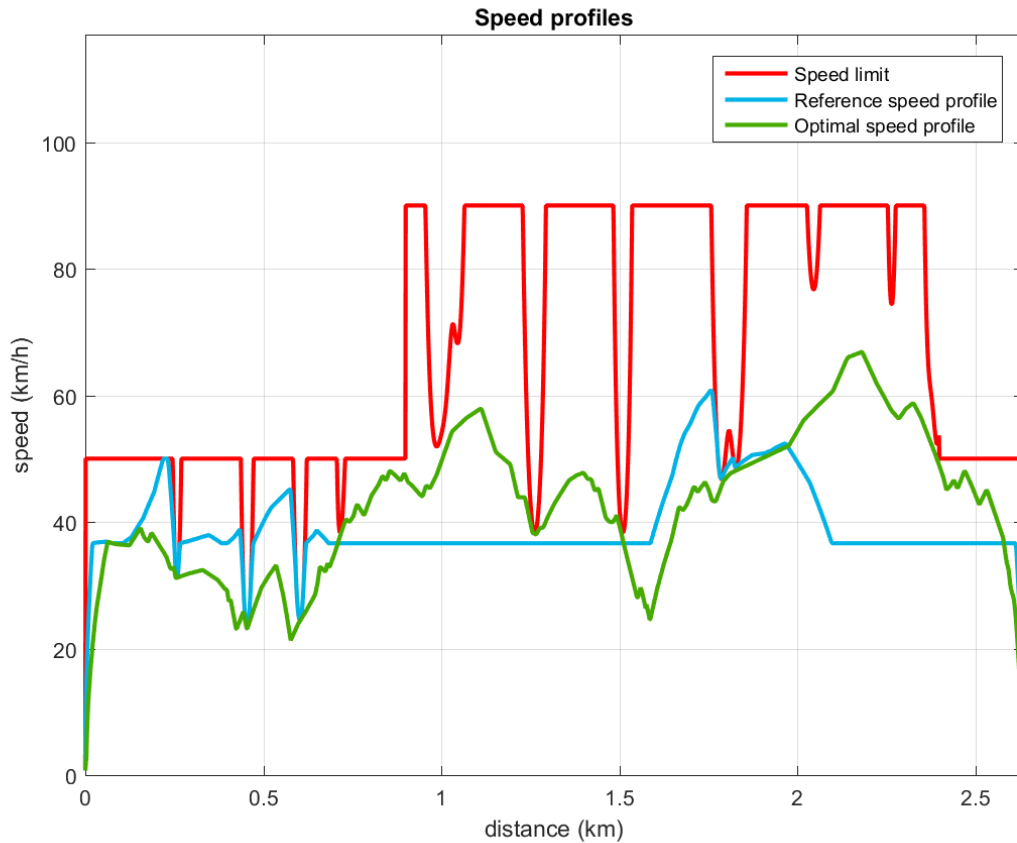


Figure 8.7: Suggested reference and optimal speed profile of the hill track drives.

## 8.2.3.3 Result

Result				
	Reference drive	Optimal drive	Saved energy	
Arrival time	249.3 s	249.9 s		
Energy model consumption	0.378 kWh	0.355 kWh	0.023 kWh	6.5 %
Dynamic model consumption	0.436 kWh	0.381 kWh	0.056 kWh	<b>14.7 %</b>

Table 8.3: Overview of results of the flat hill drive strategies.

The speed profiles differ according to behavior of a driver about the top of the hill, where the load changes from negative to positive. The optimal drive decreases the speed on the top to avoid braking when driving downhill to comply with the speed limits.

### 8.2.4 Reverse drive

The track of this drive is the same as in the hill drive. The drive differs according to the direction of the drive, which is reversed.

#### 8.2.4.1 Track

Start (altitude)	End (altitude)	Length	Desired time
Kratka (705 m)	Snezne (700 m)	2.6 km	250 s

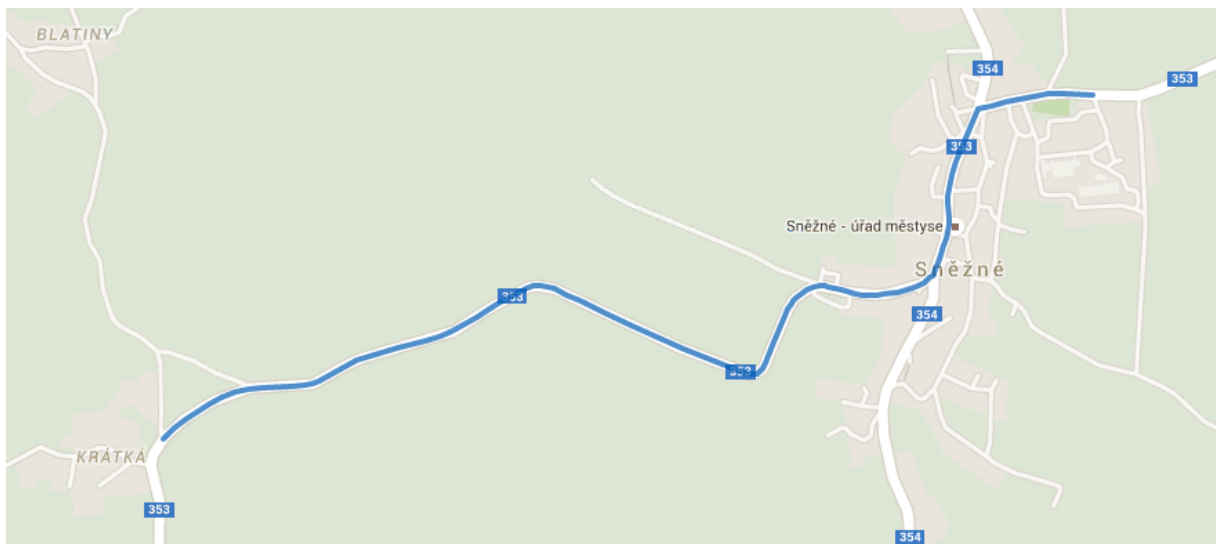


Figure 8.8: Map of reverse drive track.

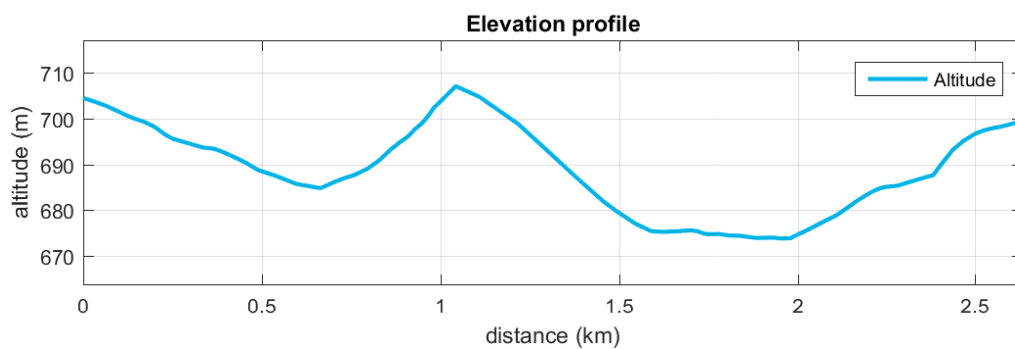


Figure 8.9: Elevation profile of the reverse track.

## 8.2.4.2 Speed profile

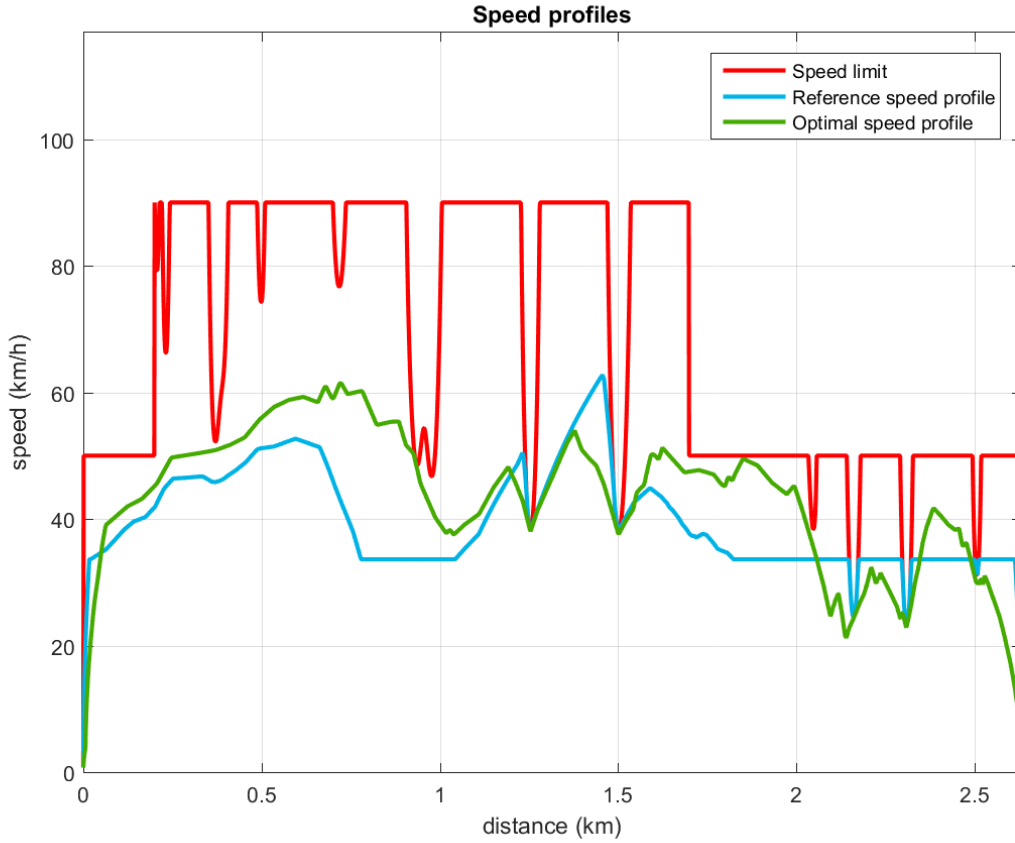


Figure 8.10: Suggested reference and optimal speed profile of the reverse track drives.

## 8.2.4.3 Result

Result				
	Reference drive	Optimal drive	Saved energy	
Arrival time	249.6 s	249.4 s		
Energy model consumption	0.305 kWh	0.299 kWh	0.006 kWh	2.0 %
Dynamic model consumption	0.377 kWh	0.321 kWh	0.055 kWh	<b>17.2 %</b>

Table 8.4: Overview of results of the flat reverse drive strategies.

Besides the speed limit, the elevation character of the track also has some influence on the built-up process of the optimal speed profile. Nevertheless, the behavior is almost the same as in the Hill drive.



## 8.2.5 Long drive

The track of this drive is also situated on highland, with high level of road inclination, omitting the traffic signs. However, the distance is much longer than in previous drives.

### 8.2.5.1 Track

Start (altitude)	End (altitude)	Length	Desired time
Snezne (750 m)	Zdar nad Sazavou (650 m)	20 km	1260 s

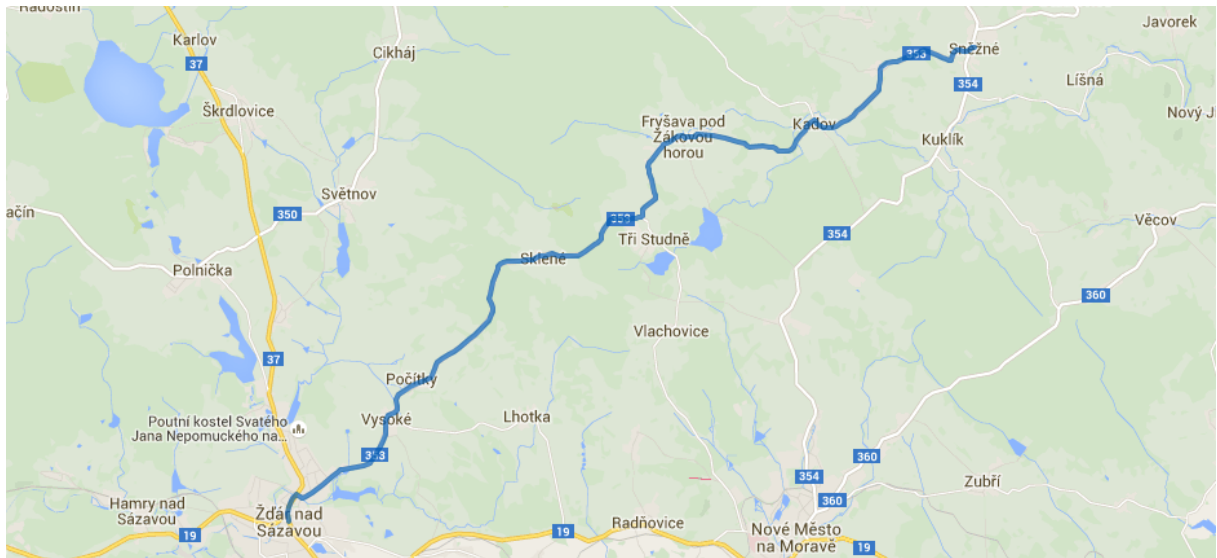


Figure 8.11: Map of long drive track.



Figure 8.12: Elevation profile of the long track.

## 8.2.5.2 Speed profile

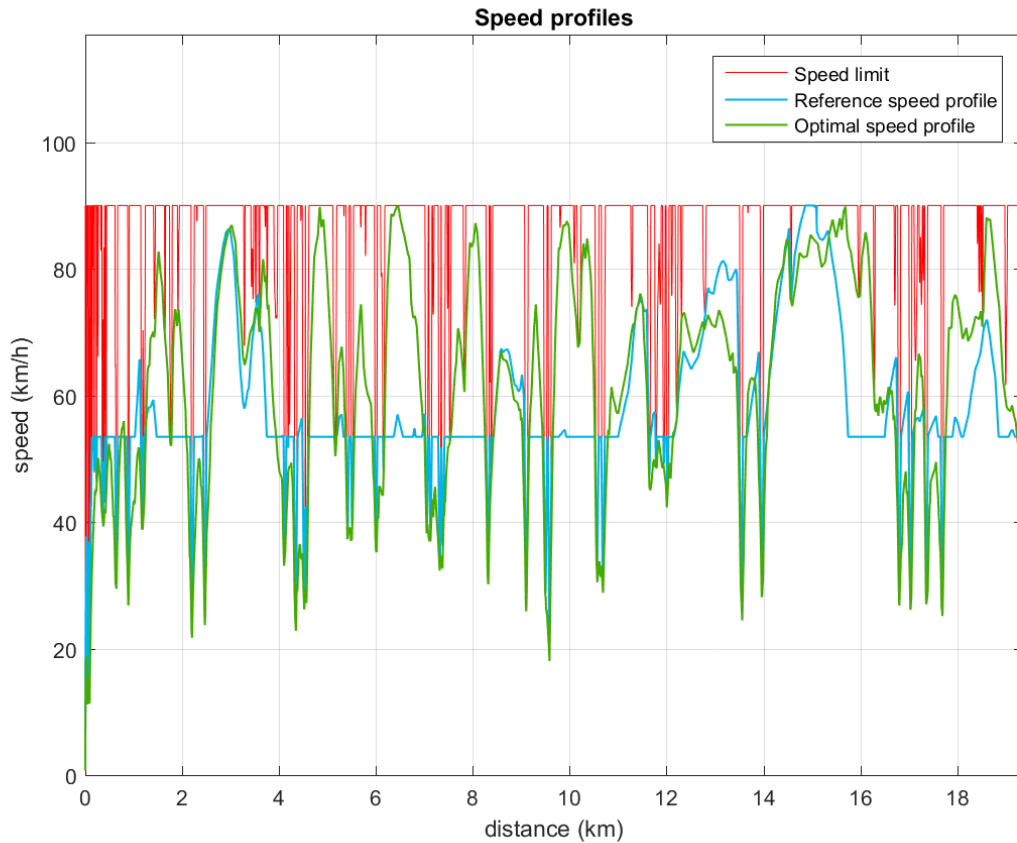


Figure 8.13: Suggested reference and optimal speed profile of the long track drives.

## 8.2.5.3 Result

Desired time = 1260 s				
	Reference drive	Optimal drive	Saved energy	
Arrival time	1260.9 s	1259.7 s		
Energy model consumption	2.734 kWh	2.684 kWh	0.050 kWh	1.9 %
Dynamic model consumption	3.776 kWh	2.919 kWh	0.857 kWh	<b>29.4 %</b>

Table 8.5: Overview of results of the long track drive strategies.

A higher value of saved energy is the result of many drops of speed limits in combination with randomly placed downhill.



### 8.2.6.2 Speed profile

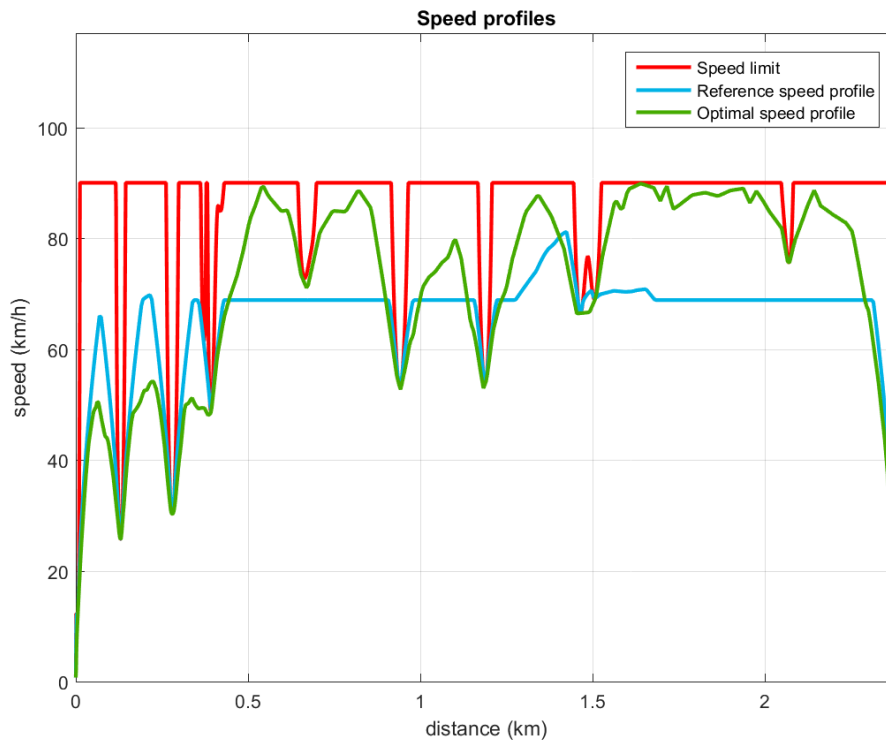


Figure 8.16: Example of reference and optimal speed profile for 140s arrival time.

### 8.2.6.3 Results

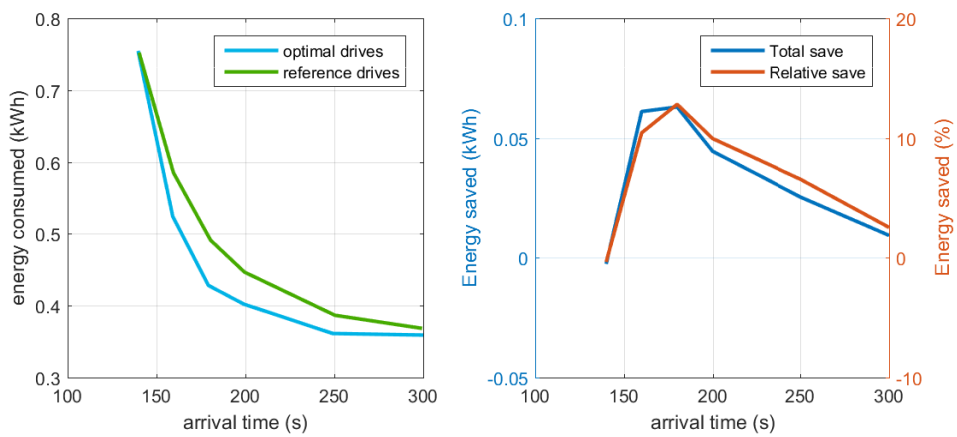


Figure 8.17: Influence of different arrival times on consumed energy.

The arrival time has strong influence on the amount of save energy. As apparent from figure 8.17, the amount of saved energy is zero for arrival times near the limits of the system and it increases to a maximum and suddenly decreases as the difference between reference drive and optimal is lower and lower. In the figure 8.16, we can see an example for arrival of time 140 seconds. The difference between the reference drive and optimal drive is huge, despite the fact that the values of consumed energy are almost equal.

### **8.3 Conclusion**

The value of saved energy is strongly dependent on a few factors. First of them is the influence of speed limits along the track. These limits make the biggest difference between reference drive and optimal drive. An advantage of the optimal drive is the decision to omit gaps between two neighbouring speed limit drops. The second main influence is arrival time. As expected, when the drive balances near the limits of the system, there is less options to optimize, while most of the sections are forbidden due to the limits. The third influence is the elevation character of the track, which causes the divergence in demands of the sections. All of these dependencies forms the optimal solution unique for whatever track or arrival time.



---

# Conclusion

## 9.1 Summary

The subject established in the introduction is divided into chapters in a logical sequence. In each chapter, a part of the subject is introduced and examined with a suggested solution.

Google maps represent a reliable source of track data with graphical user interface for track design, which is successfully implemented into Matlab application. These data includes GPS coordinates and traveled distance, moreover, the application includes an option to import data from GPX file. The analysis of the gathered data provides calculation of speed limit along the track determined by the edge of the vehicle skid and comfort of the drive.

The dynamic model of the vehicle is introduced in respect to the motion resistance forces applied on the vehicle. The model includes powertrain represented in the most efficient way to meet current computing capability to run in real-time. The dynamic model is further simplified for the purpose of optimization by introducing energy model. The key simplification of dividing track into section with constant slope of track and linear change of velocity on each section is established. This simplification brings about the transformation of differential equations of dynamic model into algebraic equations of energy model, which are by far more suitable for optimization.

To optimize the defined problem, the heuristic optimization algorithm is assembled due to the huge non-linearity of powertrain representation i.e. 'the efficiency map'. The optimization algorithm is based on the comparison of the energy demands of the track sections to techniques to avoid getting stuck in the nearest local minimum. The algorithm provides solution in respect to defined constraints.

The application is developed in Matlab/Simulink with features programmed in multiple environments to fulfill the tasks of the thesis. The application is designed into three parts for track design and vehicle parameters specification, suggestion of optimal and reference speed profile, and evaluation of the suggested speed profiles in real-time simulation.

The amount of energy saved is related to the difference between reference and optimal speed profiles and is dependent on the character of the track, especially, speed limit and elevation profile. The assumption that the reference drive is designed appropriately is proved by the decrease of amount in the energy saved when the drive is not limited in speed and the track is almost flat. The amount of saved energy converges to zero, as the desired arrival time approaches the limits of the system.

## 9.2 Contributions of the Master Thesis

Bearing in mind that the heuristic optimization algorithm provides solution, which may not be global but local, the suggested solution may be used as an initial guess or reference drive for further optimization techniques. However, the suggested drive strategy brings about significant energy saving in comparison to the reference drive, which may not be omitted. However, the optimization algorithm is very time consuming and any change of the track or vehicle parameters makes the solution invalid, thus, there is very limited option of use in traffic as any breach of the prescribed speed profile requires recalculation of the solution in respect to the newly established initial conditions. Nonetheless, this does not make the solution unusable, as there are loads of applications which provide perfect conditions for implementation. One of them is a robotic vehicle transporting materials in a factory. Moreover, almost all factories uses just-in-time method, which perfectly meets the possibility of given time constraint. Another application of the algorithm is rail vehicles, which have defined time schedule between stations, however, this application requires further adjustment of the models.

With the possibility to specify parameters of the vehicle, the application may serve to estimate the energy consumption of the vehicle while designing, and it provides sensitivity analysis. The option to import the data of the track, speed limit and speed profile makes the application able to compare the data measured in the vehicle to the dynamic model of the application or to compare imported speed profiles to suggested ones.

## 9.3 Future Work

For further research in this subject, it is recommended to explore and accomplish the following:

- Implement power analysis and measurement on test bench in VTP Roztoky.
- Design and create a device for measuring the data of track and motor directly in electrical vehicle.
- Evaluate the suggested speed profile in test bench and in the electrical vehicle.
- Modify the model for rail vehicles and use the algorithm to suggest optimal speed profile between two stations.



---

## Bibliography

- [1] Stephen Boyd. *Convex Optimization*. Cambridge University Press, New York, 2004.
- [2] N. Chernov and C. Lesort. Least squares fitting of circles. *Journal of Mathematical Imaging and Vision*, 2005.
- [3] Nikolai Chernov. *Circular and Linear Regression: Fitting Circles and Lines by Least Squares*. Taylor & Francis, 2010.
- [4] James F. Epperson. *An Introduction to Numerical Methods and Analysis*. Wiley, 2013.
- [5] Reza N. Jazar. *Vehicle dynamics : theory and application*. ANSI/IEEE STD 754-2008. New York : Springer, 2014.
- [6] Matthew Sands Richard P. Feynman, Robert B. Leighton. *The Feynman Lectures on Physics, Definitive Edition*. Addison Wesley; 2 edition, 2005.
- [7] Inc. The MathWorks. *MATLAB Object-Oriented Programming*, 2014.
- [8] Bruce R; Okiishi Theodore H Young, Donald F; Munson. *A brief introduction to fluid mechanics*. Wiley, 2011.



---

## Enclosed CD

### A.1 Content of the CD

The enclosed CD includes following folder structure:

- **Project**

This folder includes entire application with all m-files and web page application for track design.

- **Results**

Various project files are located here, along with projects presented in this thesis 8.

- **Import**

This folder includes samples of files of all kinds, which may be imported to the application.

- **Thesis**

Electronic version of the thesis is saved here. The folder also includes a record of the running optimization algorithm. The record is speeded up, thus, one second of the record refers to one minute in real-time.