



universität  
wien

# MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

## Optimizing the Operation Range of E-Bikes in Routing Systems

verfasst von / submitted by

Simon Tobias Haumann, B.A.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of  
Master of Science (MSc)

Zürich, 2017 / Zurich 2017

Studienkennzahl lt. Studienblatt /  
degree programme code as it appears on  
the student record sheet:

A 066 856

Studienrichtung lt. Studienblatt /  
degree programme as it appears on  
the student record sheet:

Masterstudium Kartographie und Geoinformation

Betreut von / Supervisor:

Prof. Dr. Martin Raubal (ETH Zürich)

Mitbetreut von / Daily Advisor:

M.Sc. Dominik Bucher (ETH Zürich)



## List of contents

List of Figures.....	vi
List of Tables .....	x
List of Abbreviations .....	xii
List of Symbols.....	xiv
Abstract.....	xvi
Kurzfassung.....	xviii
Acknowledgment.....	xx
<b>1. Introduction.....</b>	<b>2</b>
1.1. Objectives .....	3
1.2. Outline of the Thesis .....	4
<b>2. Related Works.....</b>	<b>6</b>
2.1. Literature Overview.....	6
2.2. Similar projects .....	10
<b>3. Methods.....</b>	<b>12</b>
3.1. Data Acquisition .....	12
3.2. Selection of Software .....	13
3.3. Specific Hardware.....	15
3.4. Study Area .....	16
<b>4. Development of the Energy Consumption Model.....</b>	<b>18</b>
4.1. Energy Model.....	18
4.2. Data Pipeline .....	22
4.2.1. Basic Preparation.....	22
4.2.2. Preparation in ArcGIS .....	23
4.2.3. Data Conditioning.....	29
4.2.4. Altitude Value Extraction .....	31
4.2.5. Impeding Forces.....	33
4.2.6. Determining Motor efficiency.....	35
4.2.7. Electrical Motor Power and Energy Consumption.....	36

<b>5. Routing Applications .....</b>	<b>42</b>
<b>5.1. Dijkstra Application.....</b>	<b>42</b>
5.1.1. Back-End.....	42
5.1.2. Front-End.....	46
<b>5.2. Bellman-Ford Application .....</b>	<b>48</b>
5.2.1. Back-End.....	48
5.2.2. Front-End.....	49
<b>6. Evaluation.....</b>	<b>54</b>
<b>6.1. Parameters.....</b>	<b>54</b>
<b>6.2. Test Set .....</b>	<b>64</b>
6.2.1. Test Session A.....	66
6.2.2. Test Session B.....	70
<b>7. Results .....</b>	<b>74</b>
<b>8. Discussion.....</b>	<b>80</b>
<b>8.1. Findings.....</b>	<b>80</b>
<b>8.2. Conclusion, Application Areas and Outlook.....</b>	<b>80</b>
<b>9. References .....</b>	<b>88</b>
<b>Appendix .....</b>	<b>I</b>
<b>E-Mail Bosch (Martin Wille) .....</b>	<b>I</b>
<b>E-Mail EMPA (Marcel Gauch) .....</b>	<b>II</b>
<b>Educational Use Data swisstopo .....</b>	<b>III</b>
<b>Programming Code ArcGIS Model Builder.....</b>	<b>IV</b>
energyconsumptionmodel_egomovement_whiteknight.py .....	IV
energyconsumptionsubmodel_whiteknight_1.py.....	VII
energyconsumptionsubmodel_whiteknight_2.py.....	XIV
energyconsumptionmodel_stromer_st2.py.....	XX
energyconsumptionsubmodel_st2_1.py .....	XXIX
energyconsumptionsubmodel_st2_2.py .....	XXXVI
energyconsumptionsubmodel_st2_3.py .....	XXXIX
<b>Programming Code Dijkstra Application .....</b>	<b>XLV</b>
index_v20_norec_Wh.html.....	XLV
<b>Programming Code Bellman-Ford Application.....</b>	<b>XLVIII</b>



Graph.rs.....	XLVIII
Spatialpoints.rs .....	LV
Main.rs.....	LVI
Endpoint.rs .....	LVII
Index.html.....	LXI
<b>Test Session A.....</b>	<b>LXVII</b>
03.01.2017 .....	LXVII
04.01.2017 .....	LXX
11.01.2017 .....	LXXIII
12.01.2017 .....	LXXV
13.01.2017 .....	LXXVI
16.01.2017 .....	LXXVIII
17.01.2017 .....	LXXIX
23.01.2017 .....	LXXXII
<b>Test Session B.....</b>	<b>LXXXIV</b>
15.02.2017 .....	LXXXIV
16.02.2017 .....	LXXXVI
17.02.2017 .....	LXXXVII
19.02.2017 .....	LXXXVIII
22.02.2017 .....	LXXXIX
23.02.2017 .....	XC
24.02.2017 .....	XCI
25.02.2017 .....	XCIII
27.02.2017 .....	XCIV
14.03.2017 .....	XCVII
16.03.2017 .....	XCVIII
17.03.2017 .....	XCIX
21.03.2017 .....	C
<b>Affidavit.....</b>	<b>CI</b>



## List of Figures

Figure 1: Development of the stock of electric bikes in Europe from 2008 to 2012 (Paul & Bogenberger 2014)..... 3

Figure 2: The growing number of countries with bike-sharing programs (Paul & Bogenberger 2014). ..... 3

Figure 3: An example on how the summation of an additional edge cost value in a graph in order to eliminate negative edge costs can ultimately result in a different path. .... 8

Figure 4: The study area. The map shows the street network illustrating the location and extent..... 16

Figure 5: All physical forces considered in this work (Abagnale et al. 2015a, edited). .... 19

Figure 6: Input Parameter of the Tool “Create Enterprise Geodatabase”. ..... 24

Figure 7: Succeeded processing of the Tool “Create Enterprise Geodatabase” ..... 24

Figure 8: The energy consumption model for the bike without recuperation..... 26

Figure 9: "Environment Settings" of each model. .... 27

Figure 10: The beginning of the energy consumption calculation in the first submodel. .. 29

Figure 11: Input Parameters of the tool "Download OSM Data (XAPI)" ..... 30

Figure 12: Schematic illustration of the calculation of slope. .... 31

Figure 13: The altitude value extraction procedure. .... 32

Figure 14: The end of the first submodel..... 34

Figure 15: The beginning of the second submodel..... 35

Figure 16: An extract of the computation of the motor efficiency of the bike used in Test Session B. .... 36

Figure 17: The end of the energy consumption model for bikes with recuperation ..... 37

Figure 18: The end of the energy consumption model for bikes without recuperation ..... 38

Figure 19: The energy consumption [Wh] of each street segment in the study area..... 40

Figure 20: The hypothetical energy consumption per kilometer [Wh/km] for each street segment.....	40
Figure 21: Comparison between an energy-based path (left side) and the shortest path (right side).....	42
Figure 22: The screenshot shows the required arguments to store the function.....	44
Figure 23: A new workspace with the name pgRouting is created.....	45
Figure 24: The newly created store is named after the edge costs v20_norec_Wh applied for this trial. ....	45
Figure 25: The SQL View which is used to access data through the created wrapper.....	46
Figure 26: Comparison between an energy-based path (left side) and the shortest path (right side).....	48
Figure 27: Invocation of the Bellman-Ford Application in order to start it.....	50
Figure 28: Bellman-Ford application with a reachability request.....	52
Figure 29: Bellman-Ford application with a routing request from ETH Höggerberg to ETH Center.....	52
Figure 30: The average friction and drag for the entire study area. ....	56
Figure 31: The average friction and drag for the entire study area. ....	56
Figure 32: The amount of power [W] to overcome by the electric bicycle .....	57
Figure 33: The amount of power [W] to overcome by the electric bicycle. ....	58
Figure 34: GPX-Records of the the test tracks. ....	65
Figure 35: Energy consumption per street segment on the test tracks. ....	65
Figure 36: The Electric Bicycle employed in Test Session A. ....	66
Figure 37: The pgRoutingLayer QGIS-Plugin.....	67
Figure 38: A screenshot from “Geo Tracker”, one of the smartphone applications available (first picture).....	68

Figure 39: The Electric Bicycle employed in Test Session B..... 70

Figure 40: The display of the second e-bike model shows data logging of the electric bicycle such as the ongoing aggregation of total energy expenditure. Note the difference between total consumption at the beginning (4701 Wh) and the end (4751 Wh) of each trip – resulting in an energy expenditure of 50 Wh for this trip..... 71

Figure 41: The diagram compares measured and modelled energy consumption [Wh] from Test Session A (without recuperation) (cf. Table 12)..... 78

Figure 42: The diagram compares measured and modelled energy consumption [Wh] from Test Session B (with recuperation) (cf. Table 12)..... 78

Figure 43: The histogram shows the distribution of the values for the percentage slope gradient in the study area. .... 83

Figure 44: The histogram shows the distribution of the values for slope angle in degree in the study area. .... 83

Figure 45: The scatterplot compares the slope angle in degree and the length of every street segment in the study area..... 84



## List of Tables

Table 1: The specifications of the employed data. ....	12
Table 2: The software specifications. ....	14
Table 3: The hardware specifications.....	15
Table 4: The specifications of the electric bicycles used in the test drives.....	15
Table 5: The model's parameters, classified by symbol and unit.....	55
Table 6: The motor efficiency in Test Session B for different torques and velocity levels. 60	
Table 7: The average motor power [W] and human power [W] for the study area. ....	61
Table 8: Available electric charge of the battery used in Test Session A at different temperatures $T_{amb}$ . ....	62
Table 9: Model Parameters for the model White Knight from EGO Movement. ....	69
Table 10: Model Parameters for the model ST2 from Stromer. ....	72
Table 11: Control Measurements Table. ....	75
Table 12: The entire test set containing measured values from Test Session A and B....	76





## List of Abbreviations

<b>AGILE</b>	Association of Geographic Information Laboratories in Europe
<b>(X)API</b>	(Extended) Application Programming Interface
<b>CH</b>	Contraction Hierarchies
<b>DEM</b>	Digital Elevation Model
<b>EMPA</b>	Swiss Federal Laboratories for Materials Science and Technology
<b>EPSG</b>	European Petroleum Survey Group
<b>ESRI</b>	Environmental Systems Research Institute
<b>ETH</b>	Swiss Federal Institute of Technology Zurich
<b>EV</b>	Electric Vehicle
<b>FSO (BFS)</b>	Federal Statistical Office (Bundesamt für Statistik)
<b>GeoTIFF</b>	Georeferenced Tagged Image File Format
<b>GIS</b>	Geographic Information System
<b>GNSS</b>	Global Navigation Satellite System
<b>GUI</b>	Graphical User Interface
<b>GPL</b>	General Public Licence
<b>GPS</b>	Global Positioning System
<b>GPX</b>	GPS Exchange Format
<b>HSV</b>	Hue Saturation Value
<b>HTML</b>	Hypertext Markup Language
<b>IDE</b>	Integrated Development Environment
<b>IMS</b>	Internet Map Server
<b>ITS</b>	Intelligent Transport System
<b>LC</b>	Left Click
<b>OGD</b>	Open Government Data
<b>OSM</b>	Open Street Map
<b>PVEC</b>	Pedelec Velocity Control
<b>ORDBMS</b>	Object-Relational Database Management System

<b>RC</b>	Right Click
<b>REST</b>	Representational State Transfer
<b>RGB</b>	Red Green Blue
<b>RPM</b>	Revolutions per Minute
<b>SFOE (BFE)</b>	Swiss Federal Office of Energy (Bundesamt für Energie)
<b>SOC</b>	State of Charge
<b>SQL</b>	Structured Query Language
<b>SRC</b>	Source
<b>SRID</b>	Spatial Reference System Identifier
<b>SRTM</b>	Shuttle Radar Topography Mission
<b>WFS</b>	Web Feature Service
<b>WGS</b>	World Geodetic System
<b>WMS</b>	Web Map Server
<b>XML</b>	Extensible Markup Language

## List of Symbols

$\alpha$	Slope Angle
$\beta$	Temperature-Dependent Factor
$\gamma$	Constant Factor Rider Power Input
$\eta_{EM}$	Motor Efficiency
$\eta_G$	Gearbox Efficiency
$A$	Reference Area
$C$	Wheel Perimeter
$c_{rr}$	Rolling Coefficient
$c_w$	Drag Coefficient
$E_{EM}$	Energy Consumption of the Electric Motor
$F_d$	Drag / Air Resistance
$F_f$	Friction / Rolling Resistance
$F_g$	Gravitation / Climbing Resistance
$F_h$	Human / Rider's Force
$F_T$	Tractive Force
$g$	Gravitational Constant
$h$	Average Height between the two End Nodes of a Street Segment
$l$	Length of a Street Segment
$m$	Total Mass (i.e., bicycle and rider)
$P_{amb}$	Ambient Air Pressure
$P_h$	Human Power
$P_{EM}$	Power of the Electric Motor
$P_M$	Power of the Electric Motor (Temperature Adjusted)
$P_i$	Power required by all Auxiliary Components
$R_a$	Universal Gas Constant
$r_w$	Wheel Radius
$T_{amb}$	Ambient Temperature
$T_{EM}$	Motor Torque
$T_h$	Human Torque

$T_v$	Wheel Torque
$v_{DD}$	Velocity in Driving Direction
$v_{wx}$	Wind Speed
$w_v$	Angular Velocity of the Wheel
$z$	Height Difference between the two End Nodes of a Street Segment

## Abstract

Limited driving range and the subsequent range anxiety is still one of the greatest obstacles for the use of electric vehicles (EV). Slow-moving progress in increasing battery capacities necessitates for new and novel solutions to the problem. This thesis presents a routing system capable of computing the ideal route in terms of energy consumption for electric bicycles. An underlying static model calculates energy requirements for arbitrary street segments based on contextual information. For example, this information could include the structure of the road network, a digital elevation model or vehicular parameters such as weight, velocity, or the presence of a recuperation mechanism. A routing application was developed that was capable of accurately displaying either an energy optimized route or an estimation of the remaining cruising range. The existence of the application allows for the estimation (which acts as a simulation of energy consumption) to be validated and optimized through test drives. One use-case of this technology is within a navigation system, where it can enable automatic switching from a shortest-path route in progress into energy-saving mode when a target destination becomes unreachable due to reasons such as insufficient charge. The methods outlined in this work can optimize the route, taking into account the required energy for the remaining distance and the given low state of charge (SOC) to enable riders to more reliably reach their destinations. It extends the potential of e-bike routing.

**Keywords:** *routing, e-mobility, electric bicycle, electric vehicle, range prediction, reachability, energy consumption model, Bellman-Ford*



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



universität  
wien



## Kurzfassung

Die begrenzte Reichweite und daraus entstehende Reichweitenangst kann als eine der grössten Hindernisse bei der Nutzung von elektrischen Fahrzeugen (EV) angesehen werden. Der gleichzeitig nur sehr schleppende Fortschritt in der Entwicklung von grösseren Batteriekapazitäten verlangt nach innovativen Lösungen für dieses Problem. Diese Thesis stellt ein Routing System vor, welches für elektrische Fahrräder jene Routen mit dem geringsten Energieverbrauch berechnen kann. Das dem System zugrundeliegende statische Modell erzeugt den Energiebedarf für beliebige Strassensegmente auf Basis kontextbezogener Informationen. Diese bestehen u.a. aus dem Strassennetz, einem digitalen Höhenmodell oder fahrzeugseitigen Parametern wie Gewicht, Geschwindigkeit oder einem Mechanismus zur Rückgewinnung der Energie. Darauf aufbauend wurde eine Applikation entwickelt, welche einem entweder eine energieoptimierte Routenwahl oder die Schätzung der verbleibenden Reichweite anzeigt. Mittels Testfahrten konnten die simulierten Daten validiert und optimiert werden. Ein mögliches Anwendungsfeld dieser Technologie sind Navigationssysteme. Während man sich entlang der kürzesten Route bewegt, könnte ein automatischer Wechsel in einen Energiesparmodus vollzogen werden, sobald das Ziel durch einen zu geringen Akkustand ausser Reichweite gerät. Das in dieser Thesis vorgestellte Verfahren kann die Route unter Berücksichtigung des noch absehbaren Energieverbrauchs für die verbleibende Distanz bei entsprechendem Akkustand optimieren und Fahrern das Erreichen ihres ausser Reichweite geglaubten Ziels ermöglichen. Dies eröffnet neue Möglichkeiten im E-Bike Routing.

**Schlüsselwörter:** *Routing, E-Mobilität, elektrische Fahrräder, elektrische Fahrzeuge, Reichweitenprognose, Erreichbarkeit, Energieverbrauchsmodell, Bellman-Ford*



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



universität  
wien





## Acknowledgment

I would like to thank my supervisor Prof. Dr. Martin Raubal (ETH Zurich) for his openness to the idea of supervising an external student, his quick resolve and interest in the project. Equally, I am much obliged to Ass.-Prof. Dr. Andreas Riedl (University of Vienna) and the entire Department of Geography and Regional Research (IfGR) for giving me this opportunity and the valuable and instructive master's program.

I thank my advisor Dominik Bucher (ETH Zurich) for his magnificent support throughout this thesis. Not only did I appreciate his steady advice every single time I needed it, but also his contributions that sometimes went beyond the tasks of a supervisor and helped me keeping myself on track. At the same time, I am thankful for giving me space to develop and realize my own ideas. Thank you for believing in this project and in me solving it!

Many thanks to Daniel Kastl and other participants of the FOSS4G 2016 in Bonn for the interesting discussions, enabling me to make a jump start into the world of pgRouting (and thanks to Patrick Dilger (ETH Zurich) who enabled the visit with a last-minute funding). I thank Marcel Gauch from the Swiss Federal Laboratories for Materials Science and Technology (EMPA) for his help finding a solution for the energy consumption measurement issue. Furthermore, I would like to give thanks to David Jonietz (ETH Zurich), Christian Sailer (ETH Zurich), Fabian Göbel (ETH Zurich), René Buffat (ETH Zurich), Ruth Kläy-Bührer (ETH Zurich), Monika Niederhuber (ETH Zurich), Christoph Schöneberger (ESRI Switzerland), and Bernhard Schneider (NewRide).

I felt very happy to win cooperation partners to improve my model. I would like to thank Marie So and Daniel Meyer from *EGO Movement* for their help and sincere interest in the technical part of the thesis. Tiffany Kraus and the whole *smide* crew for your quick and straightforward support. Special thanks go to Christophe Wiedmer from *Stromer*, who had a solution for any technical concern and special requests regarding their electric bicycles.

Cheers to Annika, Cynthia, Anja, Fabian, Dominik, Daniel, Jonathan and the whole Bülachhof community for their power supply, inspiring conversations, and shelter on any occasion. Special thanks to Fabian and Aurelia for performing additional test rides. Heike, Alex, Vicky and Angie – thank you for your thorough proofreading. I would like to thank all of my friends for their continuous support and patience all along.

My Family. Carmen, Jürgen, Kai, Alexander, Fabian, Heike and Annia – I am deeply grateful for your support throughout the years. Without you, it would literally not have been possible. You built a new home out of my old one – back in Switzerland.

For Micha. You were a great friend to me. Farewell.



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



universität  
wien



## 1. Introduction

As cities become more densely populated due to rising urbanization, and societies try to decrease their dependency on fossil energy sources, it becomes increasingly important to substitute many of the trips previously covered by car with smaller and more energy-efficient modes of transportation. Electric bicycles fit nicely into this gap, as they have several benefits. They take little space on the roads. Perhaps most significantly, they are lower in costs under many interpretations – energy demand, cost of purchase or rental, insurance, licenses, registration, road taxes, or parking (MUETZE & TAN 2007; MCLOUGHLIN ET AL. 2012). At the same time, they may be fueled by renewable energy. Also, required physical strain decreases as compared to a regular bicycle (thus increasing the personally reachable range). Those advantages manifest themselves in growing sales numbers in both European countries (PAUL & BOGENBERGER 2014; Figure 1) and other regions in the world, for example China (FAIRLEY 2005), Japan, Taiwan and the United States (MUETZE & TAN 2007). In Switzerland two percent of all households own an electric bicycle (in 2010; likely to have increased ever since; FEDERAL STATISTICAL OFFICE (FSO) 2017). Contributing uses cases seem to stem not only from individual purchases for personal use but also business uses. Some examples of this include delivery or courier purposes, or companies providing their employees with shared e-bikes (e.g., as part of a health program). Also, there is an emerging amount of stations-based and even free-floating e-bike sharing systems within cities (Figure 2). Current examples for Switzerland are the station-based system “Publi-Bike” (PUBLI-BIKE 2017), the free-floating system “smide” (SMIDE 2017) in Zurich or the eCargo-Bike System “carvelo2go” (CARVELO2GO 2017). These trends make it increasingly important to be able to effectively assess driving ranges of bicycles taking into account not only parameters of the bike itself, but also of the person using it, the roads available for travel and any other contextual data that may influence the journey. Concomitant, the relentless digitization of every aspect of our daily lives within the scope of intelligent transport systems (ITS)<sup>1</sup> increase the suitability of applications as the envisaged one in this work.

<sup>1</sup> ITS address processing of traffic and transportation data through telecommunication and information technologies.

**Figure 1:** Development of the stock of electric bikes in Europe from 2008 to 2012 (PAUL & BOGENBERGER 2014).



**Figure 2:** The growing number of countries with bike-sharing programs (PAUL & BOGENBERGER 2014).



### 1.1. Objectives

Based on a variety of other electrical and physical consumption models (cf. HOCH 2015; ABAGNALE ET AL. 2015a; MUETZE & TAN 2007; OLIVA ET AL. 2013; WANG ET AL. 2015), I will build a static model of an electric bicycle, which is embedded in a framework for route computations. The resulting framework features a processing pipeline tailored to fast querying for electrical bicycle routes. Prominent steps in the pipeline are the evaluation of multiple parameters in parallel, and building up static cost graphs based on a road network and a digital elevation model (DEM). In order to evaluate the model performing test drives, two

prototypical applications are built with the aim of route planning and cruising range estimation.

Within the scope of this work, three overarching research questions are discussed:

- i) What are the peculiarities of electrical bicycles in terms of energy consumption, and related route and range computations? What specific parameters are necessary to model the energy consumption of an e-bike? (Chapter 4 and Section 6.1)
- ii) What is an adequate implementation of an energy consumption model for routing systems and range estimation for electrical bicycles? (Chapter 5)
- iii) Is there an extension of the capabilities of common routing models? (Chapter 6, 7 and 8)

## 1.2. Outline of the Thesis

The thesis consists of the following structure: In order to analyze the previously defined research questions, I am first going to review the existing literature on this topic (Chapter 2) and – due to the thesis' partly application-based character – present similar projects accordingly. Chapter 3 describes the study area and the methodology employed in the thesis and hence, its framework including specific hardware, the chosen data construct, development tools and other applied software. Chapter 4 deals with the creation of a theoretic approach of an energy model suitable for e-bikes, followed by the implementation into a programmatic model. The calculations are followed by query preparation and updating the initial files within an object-relational database management system (ORDBMS), which is then used within a routing system. Therefore, the development of two prototypical applications in Chapter 5 is used to evaluate, validate, and optimize the model performing test drives (Chapter 6). After presenting and visualizing the results in Chapter 7, the final Chapter summarizes the elaborated work and concludes with an outlook on possible further research.





## 2. Related Works

I provide an overview of existing work in this field, which encompasses identifying peculiarities of electric bicycles, summarizing similar energy models of all kinds of electric vehicles (EV) to be able to adopt them, and an investigation on appropriate algorithms in Section 2.1. Since one of the thesis results is a prototypical application, I present similar projects in the private sector in the subsequent Section 2.2.

### 2.1. Literature Overview

The profound potential for energy-based routing has previously been stated by ERICSSON ET AL. 2006, initially in reference to fossil fuel use and later with specific regard to EVs (SACHENBACHER ET AL. 2011). For EVs in particular, this routing approach will gain even more importance in the future, with limited battery capacity resp. limited operation range and long recharge times still representing major user concerns (cf. NEAIMEH ET AL. 2013; ARTMEIER & HASELMAYR 2010). This might lead to mental obstacles such as range anxiety (STEINHILBER ET AL. 2013). In contrast to the research mentioned, I have chosen an approach where I treat the use of energy consumption directly as weights for the graph rather than using it as a factor of the length (ERICSSON ET AL. 2006). This method enables me to display genuine consumption upon each request in terms of cost transparency.

To be able to compute an energy-based routing, it's essential to know the energy demand on a certain route. Therefore, I need to establish an energy consumption model for electric bicycles. A large body of work covers similar energy models for electric cars (cf. HOCH 2015; OLIVA ET AL. 2013), both from an engineering point of view (e.g., to assess the influence of different parts on the overall energy consumption), as well as for routing and cruising range estimation using Geographic Information Systems (GIS) (cf. NEAIMEH ET AL. 2012; NEAIMEH ET AL. 2013; KARRAIS 2014). Due to the progress in the field, scientists have started to take altered approaches of determining factors into account. NEAIMEH ET AL. 2013, for example, focused on a maximized cruising radius integrating the limitations with regard to battery capacity of an EV when adjusting a certain route, rather than finding the shortest or quickest path. While the above-mentioned models are to a large degree applicable to the topic of electric bicycles as well, there are several differences: the smaller complexity of the drivetrain (e.g., partly the lack of an energetic recovery system resp. recuperation system, or the mostly constant energy consumption and smaller amount of switched on auxiliary components – e.g., compared to adjustable air conditioning or radio in cars), the additional power supplied by the rider, or the lack of battery temperature management causing higher or lower energy consumption (KARIMI & LI 2013; YUKSEL & MICHALEK 2015; LI ET AL. 2016).

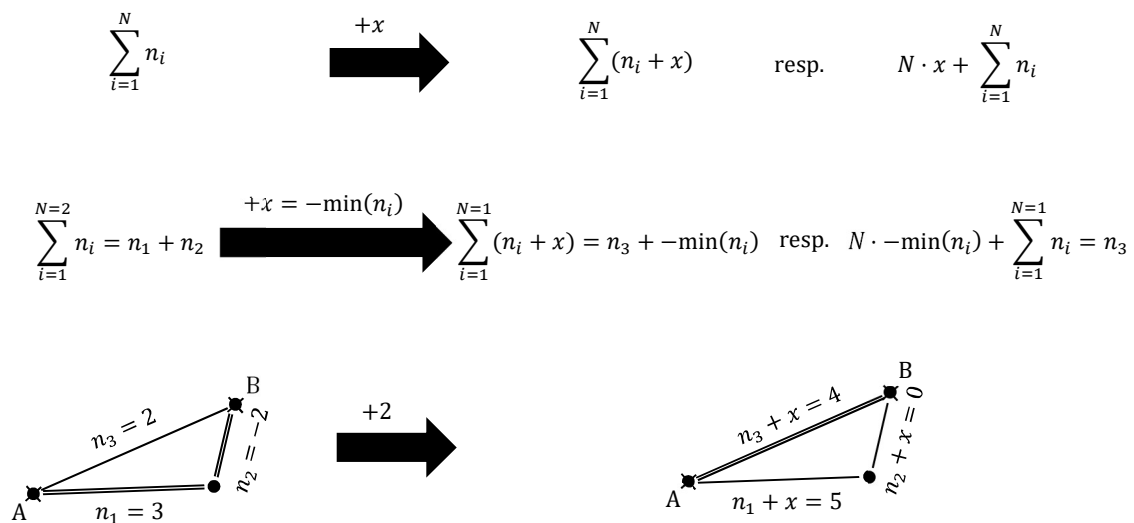
For example, the recent research on electric bicycle energy models by ABAGNALE ET AL. 2015b; ABAGNALE ET AL. 2015a; CARDONE ET AL. 2016 only consists of a motor and gearbox in sequence. BA HUNG ET AL. 2017 evaluated the effect and interaction of a variety of input parameters on the required power to propel an electric bicycle. They aimed at providing suggestions on how to optimize the energy consumption. MUETZE & TAN 2007 also examines the performance of electric bicycles. While STORANDT 2012 has already identified the need for energy saving routes for vehicles in the context of the examined ordinary bicycles, her work focusses on avoiding steep climbs as influencing parameter only. However, the study aims at optimizing a specific routing algorithm.

Routing algorithms can solve the shortest path problem in a network. Such a network is typically described as a graph, consisting of nodes (i.e., street intersections), which are connected by edges (i.e., street segments, bike lanes, etc.). A cost value is assigned to each edge weight which could be the length of the street segment or – in this thesis – the energy consumption of an electric bicycle on the street segment. To solve the shortest path problem the algorithm computes the path with the least edge costs between two selected nodes in the network. In an undirected graph the edges have no orientation, while edges in a directed graph have a specific orientation. Street networks require directed graphs as street segments can be traversed sometimes in both directions and sometimes only in one direction. Furthermore, the energy consumed by a electric bicycle is higher when going uphill than when going downhill (BA HUNG ET AL. 2017), especially when a recuperation mechanism recovers energy while braking. This fact can be considered only when distinguishing between forward and backward traversal.

The mentioned peculiarities require the use of a routing algorithm suitable for my cause. The recuperation mechanism prevents the use of a routing algorithm such as Dijkstra (DIJKSTRA 1959) or its generalizing, extending, or speed-up variants such as the A\*- algorithm (HART ET AL. 1968), Contraction Hierarchies (CH) (GEISBERGER ET AL. 2008) or several other route planning algorithms (DELLING ET AL. 2009), as the battery may be charged during a downhill segment (SACHENBACHER ET AL. 2011). Therefore, if recovered energy is represented by a negative value, the corresponding edge weight will be negative. Simply adding a large positive number to all edge weights and applying the Dijkstra algorithm (and subtracting the appropriate amount after compiling the route) can ultimately result in a different (and thus not optimal) route, as Figure 3 illustrates. The Bellman-Ford algorithm (BELLMAN 1958) is a popular alternative able to process negative edge costs. Nevertheless, it does not allow negative edge cycles (i.e., a cycle containing only negative edge weights, or even

a cycle causing a negative value when adding up the containing edge weights). Considering the topography while passing downhill segments in a street network with the aim to go back to the origin, a negative edge cycle in a three-dimensional environment is physically impossible. In other words, as the energy required to traverse a street segment (i.e., assigned to the street segment as edge weight) is only negative if the slope gradient is, it will not cause a cycle of negative edge values.

**Figure 3:** An example on how the summation of an additional edge cost value in a graph in order to eliminate negative edge costs can ultimately result in a different path. *The first line:* Adding a number  $x$  to every edge appears in the cost as an additional term equal to  $N \cdot x$ . This additional cost value  $x$  is dependent on  $N$ , which is the number of all edges of the path taken. In general the number of edges has no influence on the cost of a path, only the sum of individual cost of all edges. Therefore this method can not be applied. The notation in *the second line* is adapted for the example in *the third line*: To eliminate negative costs, the additional cost must be at least the minimum of the edge weights in the graph. While in the first graph the shortest path from A to B is along  $n_1 = 3$  and  $n_2 = -2$  (total cost of the path = 1), the shortest path in the altered second graph is now along  $n_3 = 4$  (total cost of the path = 4).



The Bellman-Ford algorithm might also require more calculations in terms of time complexity<sup>2</sup>. According to BAST ET AL. 2015 the application of algorithms for route planning inevitably implies a trade-off between query time on the one hand and preprocessing time and storage at the other hand. GALLO & PALLOTTINO 1982 resp. PALLOTTINO 1984 introduced a variant of Bellman-Ford with a worst time complexity, but a better performance though (ZHAN & NOON 1998). Also, the mentioned work of STORANDT 2012 focuses mainly on the application and improvement of the underlying CH algorithm. However, the use of speed-up techniques is not of major concern in this work. As routes for electric bicycles are comparatively short

<sup>2</sup> Time complexity in computer science describes the growth of the running time of a function (i.e. the algorithm) proportional to the size of the input (i.e. the length of its string) (SIPSER 2006). While  $O$  in  $O(n^x)$  denotes a tight upper bound on the time complexity,  $n$  reflects the size of the input.

(as range is limited by the utilized battery) and make thereby use of relatively small parts of the street network. I updated the applicability of an appropriate routing algorithm. Concomitant with graphs being preprocessed, query time resp. the use of speed-up techniques is not the focus of this work. As a consequence, routing for a bicycle without recuperation mechanism is computed simply with a Dijkstra implementation, whereas the subsequent developed Bellman-Ford enables a wider use for both bikes with and without recuperation systems.

As a model for electrical bicycles is largely dependent on personal parameters (e.g., weight, rider power supply), various capturing of the parameters in different steps of the cost calculation requires the preprocessing of the cost value and prevents me from updating edge weights dynamically. While recent research has shown methods for dynamic graph updates (e.g., SCHULTES & SANDERS 2007), it usually focuses on updating only a few edges. For that reason, I opted for building a processing pipeline, which computes multiple parameterized graphs in parallel.

According to the works of ABAGNALE ET AL. 2015a; ABAGNALE ET AL. 2015b; CARDONE ET AL. 2016, literature distinguishes between two kinds of actuators for electric bicycles: Pure electric bicycles which are triggered by a handlebar throttle on the one hand (cf. FAIRLEY 2005; SOMCHAIWONG & PONGLANGKA 2006; YANG ET AL. 2009), and so-called pedelecs which react on pedaling of the rider on the other hand (cf. DU ET AL. 2009). Although the test set from Section 6.2 is carried out using pedelecs<sup>3</sup> and the energy model from Section 4.1 designed accordingly (as this type is more common in Europe and therefore the study area introduced in Section 3.4 according to MUETZE & TAN 2007), my approach could be applied on both versions of electric bicycles by just omitting the human force for the former. Also, by adjusting the parameters and the overall setting, the model can be used for other EVs.

---

<sup>3</sup> I use the terms *electric bicycle* resp. *e-bike* synonymous to *pedelec* in this work.

## 2.2. Similar projects

Besides academic research, major developments can be found in the private sector. While “Smart Directions” by “Mapbox” enables to compute the most energy efficient route for electric cars or scooter (so far for San Francisco solely), it is not exclusively made for electric bicycles (MAPBOX n.d.). “Nyon” by “Bosch” is capable of showing the remaining range for electric bicycles with “Topo-Reichweite” considering the topography on the shortest route (launched in April 2016; BOSCH 2016). Surprisingly however, neither do these offer an energy based routing, which predicts an approximation of the required energy, nor a recommendation on which route to take, so that one can still reach the desired target location at the remaining state of charge (SOC) of the battery, although developments in this direction might be tending (WILLE 2016). In this context, the so-called “eBike Reichweiten-Assistent” is noteworthy as well. Based on a variety of input parameters, it estimates the general reachability displaying a value for the range only without any spatial context (Bosch products solely, BOSCH n.d.). “Morbih’en vélo” follows a similar approach as this thesis, but they use pgRouting to find the safest route riding an ordinary bicycle (MORBIH’EN VÉLO n.d.). Just as much as those products differ from my approach, it might not be expedient to define a state of the art for this kind of application.



### 3. Methods

For the selection of the right framework, I focused mainly on the interoperability of the chosen tools in order to meet the requirements of the desired universal applicability. Certainly, criteria such as adaptability during the process of the work and previous knowledge of the underlying coding language mattered as well. The use of open source data and tools promise greater combability which inherently allows an easier use of the calculated data in external programs and arbitrary applicability. Moreover, I followed a holistic contemplation in the stage of data (Section 3.1) and software (Section 3.2) selection to ensure proper functioning of every chosen element. The chapter provides both an overview of required hardware (Section 3.3) and justification for the selection of the study area (Section 3.4).

#### 3.1. Data Acquisition

To establish a routing environment, I had to acquire data both for a street network and a DEM which should meet the following requirements: Latest data, high accuracy/resolution, optimal spatial coverage, available complementary, automated processing and storage in a ORDBMS possible. Two options fulfil those conditions: Open Street Map (OSM) and Open Government Data (OGD). Although for the latter a Web Feature Service (WFS) for bikeways is available for the chosen study area introduced Section 3.4 (CITY OF ZURICH 2017) and could be incorporated as a Tool in ArcMap, the provider limits any request to 1000 features. This precludes any implemented reconciliation with OSM Data to improve the selection of accurate street segments simultaneously. However, OSM Data does not have the mentioned restrictions. Through manually defined predicates before download, I can select negotiable roads solely, which reduces query time as well. Raw OSM data from the source mentioned in Table 1 is provided in the World Geodetic System (WGS) WGS 84 (EPSG SRID: 4326).

**Table 1:** The specifications of the employed data. The table contains the type, provider, geodetic datum and source of the very same.

Data	Provider	Geodetic Datum	Source
road network	OSM	WGS 84	<a href="http://www.overpass-api.de/api/xapi_meta?">http://www.overpass-api.de/api/xapi_meta?</a>
DEM (swissALTI3D)	swisstopo	CH1903 LV03	Educational Use Data swisstopo in Appendix

Defined goal for future work is to fully automate the model which also concerns the DEM (cf. Section 8.2). However, for the extraction of altitude values I used the swissALTI3D of the canton of Zurich provided by the Federal Office of Topography in Switzerland swisstopo



with a resolution of 2 meters to ensure best possible evaluation thereafter instead of e.g., free and open-source data from the Shuttle Radar Topography Mission (SRTM). The geodetic datum is CH1903 LV03 (EPSG SRID: 21781).

### 3.2. Selection of Software

Considering the use of open data, a whole series of developing tools came into question such as OSRM, OpenTripPlanner, Graphhopper, only to name a few (OPENSTREETMAP WIKI 2016). I decided to use PostgreSQL to store my data. On the one hand, by its own admission it is the world's most advanced open source database and is used very often. Besides, with PostGIS, it has a very powerful spatial extension – indispensable for this work. On the other hand, the additional geospatial routing system pgRouting provides a large number of functionalities for developers. It can process OSM Data, for which I chose osm2pgrouting, and edge costs can be calculated and updated dynamically. ArcSDE, which enables the inclusion of spatial data into the ORDBMS, establishes the connection to ArcGIS Model Builder, where the calculation of the energy consumption takes place. I can incorporate OSM Data using the additional OpenStreetMap Toolbox. ArcGIS Model Builder combines both a graphical user interface (GUI) for a better review and fast adaption possibilities in the development process (instead of using a pure python script). The routing extension ArcGIS Network Analyst is based on the Dijkstra algorithm (ESRI 2016b) however an implementation of the Bellman-Ford algorithm (cf. Section 2.1) was required. Unfortunately, due to the proprietary nature of the tool, it was infeasible to implement it within ArcGIS. Consequently, I abstained from performing the routing within ArcGIS and used the programming language Rust for a stand-alone application. In contrast to the capabilities of ArcGIS, the reason for the additional usage of QGIS is more functionalities in the connection between the ORDBMS and the GIS to perform and visualize database queries such as DB Manager and pgRoutingLayer. However, ArcGIS enables me, for example, to implement Python scripts using ArcPy which is why I had to choose several integrated development environments (IDE). While PyCharm operates as a script tool editor and debugger with direct linkage to ArcGIS, I managed and administrated the ORDBMS by using DataGrid along with pgAdmin and DB Manager. For the development of applications, I used WebStorm and Notepad++, as well as Visual Studio Code mainly for the development of the Bellman-Ford Application. The linkage between Tableau and the ORDBMS facilitates analysis and visualization of the calculated values when altering parameters upon each run of the model in the evaluation phase. To avoid measurement inaccuracies as a consequence



of the use of third party applications for tracking, I used two free and open-source smartphone applications redundantly. Table 2 summarizes the selection of the software.

**Table 2:** The software specifications. The table contains the name of each program, its version, proprietor and license and how it is employed for my purposes.

Name	Version	Proprietor (license)	Function
PostgreSQL	9.4	open source (PostgreSQL)	ORDBMS
	2.3	open source (GPL)	spatial database extender
pgRouting	2.3	open source (GPL)	routing system
osm2pgrouting (mapconfig_for_bicycles.xml)	2.2.0	open source (GPL)	tool - imports OSM Data into pgRouting / defines topology
pgAdmin	1.16.1	open source (GPL)	GUI for ORDBMS
ArcGIS for Desktop	10.4	proprietary (ESRI)	GIS (GUI for cost calculation)
ArcGIS for Server	10.4		
ArcSDE	10.2		spatial database engine
ArcGIS Editor for OSM	10.4		toolbox
QGIS	2.16.1	open source (GPL)	GIS (GUI for routing)
DB Manager	0.1.20		GUI for ORDBMS
pgRoutingLayer	2.1.0		Plugin to display pgRouting layers directly
Geoserver	2.10.1	open source (GPL)	IMS
PyCharm	2016.2.3	proprietary (JetBrains)	IDE
DataGrid	2016.2.6	proprietary (JetBrains)	IDE
WebStorm	2016.2.4	proprietary (JetBrains)	IDE
Visual Studio Code	1.11.0	open source /proprietary (Microsoft)	source code editor
Notepad++	7.2.2	open source (GPL)	source code editor
Tableau	10.1	proprietary (Tableau Software)	data visualization (ORDBMS)
Open GPS Tracker		proprietary (René de Groot)	application for tracking test drives / generating GPX-Files
Geo Tracker – GPS tracker		proprietary (Ilya Bogdanovich)	application for tracking test drives / generating GPX-Files

### 3.3. Specific Hardware

**Table 3:** The hardware specifications. The table contains the hardware employed, its manufacturer and usage resp. specifications.

Hardware	Manufacturer / Model	Usage / Specs
Notebook	Lenovo Yoga	Intel(R) Core(TM) i7-6500U CPU @ 2,50GHz 2,59 GHz and 16.0 GB RAM with Win- dows 10 64-bit operating sys- tem
Energy cost meter	Arendo	Measurement of the Energy Consumption of the Battery
Smartphone	OnePlus Two	Tracking
E-Bikes	EGO Movement White Knight Stromer ST2	Table 4

Throughout this thesis, I required specific hardware, especially e-bikes in order to perform test drives (Table 4). I used a GNSS (Global Navigation Satellite System) enabled device to track the route and an energy cost meter to ascertain energy consumption (Table 3).

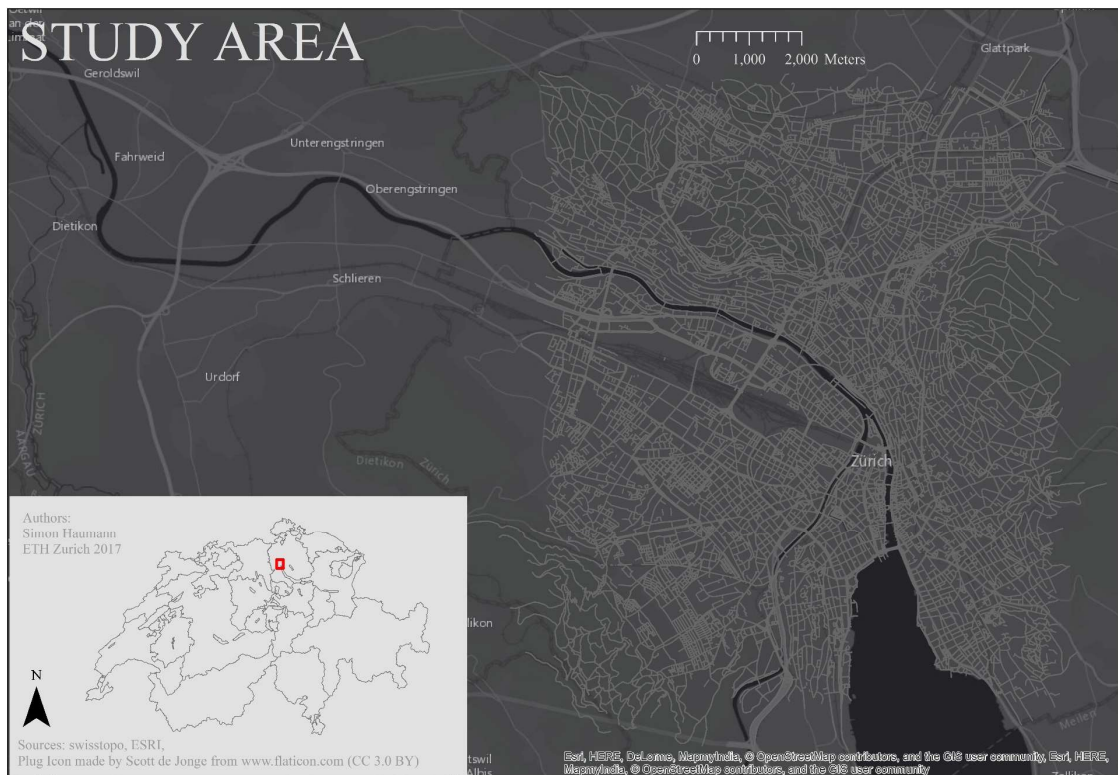
**Table 4:** The specifications of the electric bicycles used in the test drives.

Specifications	White Knight (EGO Movement)	ST2 (Stromer)
Weight [kg]	23	27
Wheel Diameter [inches]	28	26
Max. Gradeability [degree]	15	-
Motor		
Nominal Voltage [V]	36	46.8
Nominal Current [A]	-	24 (max.)
Nominal Power [W]	350	500
Auxiliary Components [W]	1.08 (display)	0.15 (backlight) 1 (controller) 1.02 (display) 2 (daytime running lights) 4.2 (headlight)
Battery		
Nominal Voltage [V]	-	48
Rated Capacity [Ah]	10.319	15.9
Capacity [Wh]	360	814
Maximal Speed [km/h]	25	45 (curbed to 35 for the tests)

### 3.4. Study Area

Zurich City was chosen as survey area, because it contains a dense road network and steep as well as flat regions. Therefore, it is suitable to evaluate the impact of different influence quantities such as *gravitation*  $F_g$  (cf. Section 4.1 (1)) carrying out test drives. The extent in WGS 84 (EPSG: 4326) is defined with 47.42 (Top), 8.57 (Right), 47.35 (Bottom), 8.485 (Left). It contains 22859 street segments (Figure 4).

**Figure 4:** The study area. The map shows the street network illustrating the location and extent.





## 4. Development of the Energy Consumption Model

In this chapter, I am going to explain the development process of the energy consumption model, which calculates edge costs, to be processed in the applications using appropriate routing algorithms thereafter. I am going to describe the theoretic setup of an energy model for electric bicycles (Section 4.1), which is then – after conducting inevitable preparation steps – transformed into a programmatic form in Section 4.2.

### 4.1. Energy Model

An electric bicycle powertrain has to supply enough energy to overcome *gravitation*, *drag*, and *friction* (i.e., road resistance), as shown in Figure 5. Steep slopes cause a large gravitational force, which can either lead to energy consumption (when going uphill), or energy production (when going downhill):

$$(1) \quad F_g = m \cdot g \cdot \sin(\alpha)$$

In this formula,  $m$  is the total mass (i.e., bicycle and rider),  $g$  is the gravitational constant, and  $\alpha$  is the slope angle. The friction is commonly defined as:

$$(2) \quad F_f = c_{rr} \cdot m \cdot g \cdot \cos(\alpha)$$

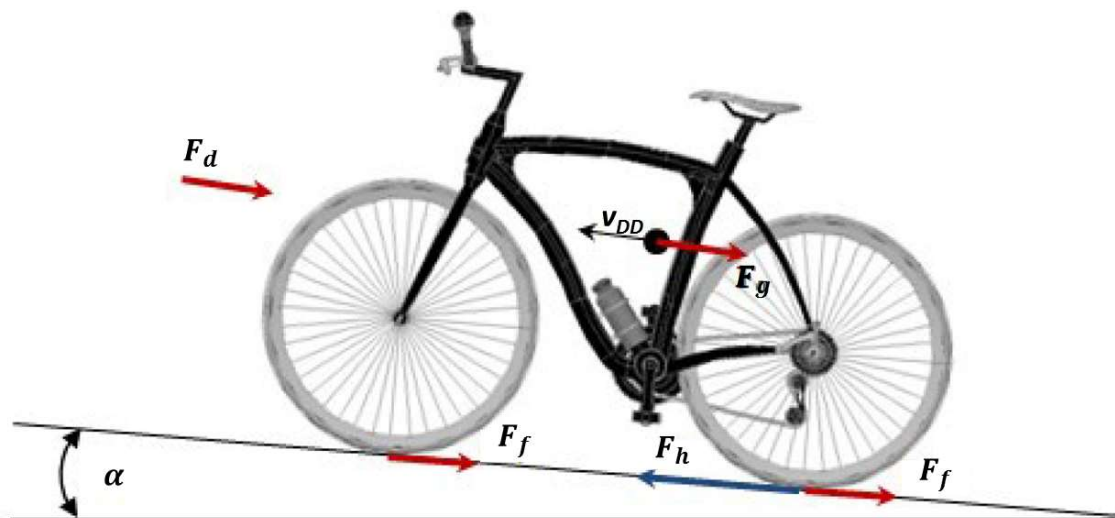
Being proportional to the gravitational force, it is enriched with a rolling coefficient  $c_{rr}$ . Finally, the definition of drag or air resistance is:

$$(3) \quad F_d = \frac{P_{amb}}{2R_a T_{amb}} c_w A (v_{DD} - v_{wx})^2$$

Here,  $P_{amb}$  denotes the ambient air pressure,  $R_a$  the universal gas constant,  $T_{amb}$  the ambient temperature,  $c_w$  the drag coefficient,  $A$  the reference area,  $v_{DD}$  the driving speed in driving direction and  $v_{wx}$  an optional wind speed. For simplicity, I assume the wind speed to always be zero in the elaborated implementation.

Power generated via the electric powertrain must overcome all these forces ( $F_T = F_g + F_d + F_f$ ), in order to propel the bicycle forward. Note that I will omit the *acceleration resistance*, which is used to model how a vehicle accelerates and decelerates. For my static model, acceleration is neglected. Instead, a measured average velocity is assumed to be the prospective target velocity. Later integration is discussed in Section 8.2.

**Figure 5:** All physical forces considered in this work (ABAGNALE ET AL. 2015a, edited). The arrows denote vectors which illustrate the direction of each force interfering.



There are different systems for controlling the ratio between rider and motor power input. For example, ABAGNALE ET AL. 2015a propose the pedelec velocity control (PVEC), in which the motor provides full support up to a selectable target velocity. Since this method only works in a dynamic model, I instead model the rider power input  $F_h$  as a constant factor  $\gamma$  of the tractive force  $F_T$ . This way, a specific value emerges for each graph, which can be seen as average rider power input and modified at will in an optimization process (cf. Section 6.1). Ultimately, a user-dependent value also limits the maximal rider power supply.

$$(4) \quad F_h = F_T \cdot \gamma$$

Accordingly, the overall wheel torque  $T_v$  consists of  $T_{EM} + T_h$ , where  $T_{EM}$  is the motor torque (cf. formula (5)).

The above forces induce a certain torque on the wheels, denoted as  $T_v = F_T r_w$ , where  $r_w$  is the wheel radius. Using the angular velocity of the wheel ( $w_v = v_{DD}/r_w$ ), the equation for motor power generation / consumption results in:

$$(5) \quad P_{EM} = \begin{cases} \frac{T_{EM} w_v}{\eta_{EM}(T_{EM}, w_v) \cdot \eta_G} + \sum_{i=1}^n P_i & T_{EM} \geq 0 \\ T_{EM} w_v \cdot \eta_{EM}(T_{EM}, w_v) \cdot \eta_G + \sum_{i=1}^n P_i & T_{EM} < 0 \end{cases}$$

Where  $\eta_{EM}$  is the motor efficiency (depending on torque and velocity) and  $\eta_G$  the gearbox efficiency, and the sum of the power  $P_i$  consumed by auxiliary components (cf. HOCH 2015; ABAGNALE ET AL. 2015a; OLIVA ET AL. 2013). For torques greater than zero, the motor is in consumption mode, and for torques smaller than zero, it is in production mode.

For bicycles which lack a recuperation system, the motor simply never operates in production mode, i.e.,  $P_{EM}$  will never be negative, and the battery will never be recharged. In my model, this restraint is a parameter of the bicycle, which limits  $P_{EM}$  accordingly.

As most people charge their battery indoors, i.e., in a warmer environment, the power consumption appears to be larger due to the battery cooling down. YUKSEL & MICHALEK 2015 investigated the effects of temperature differences (i.e., comparison between different regions in that case) on EVs stressing the poorer performance of EVs at lower temperatures since electrochemical reactions are temperature dependent. LI ET AL. 2016 examined the effect of temperature among and compared to other factors influencing the energy consumption of EVs. They are both stating that heating and cooling influences range not only due to less available energy from the battery, but also causes an increasing energy demand. Note that I only considered the cooling effect in this work since the tests were carried out in winter, and for simplification it is modelled with a linear decrease. The effect of the battery cooling down is approximated by a temperature-dependent factor  $\beta$ :

$$(6) \quad P_M = P_{EM} \cdot (1 + (25 - \Delta T_{amb}) \cdot \beta)$$

The electric motor power ultimately results in the energy consumption required for each road segment of length  $l$ :

$$(7) \quad E_{EM} = P_M \cdot \frac{l}{v_{DD}}$$



## 4.2. Data Pipeline

To find routes with low energy consumption, the model from Section 4.1 must be applied to a street network. Such a model application either computes all required values dynamically during the routing process, or builds a static graph of edge costs. My model follows the second approach, as routing on a static graph is much less computationally intensive, and thus applicable for large graphs, such as a street network. The recurrence of numerous personalized input parameters (cf. Section 6.1) in different stages of the cost calculation process makes it almost impossible to fully process the model dynamically upon request resp. would lead to unacceptable query times.

I transformed the theoretical approach for an energy model from Section 4.1 into a programmatic model using ArcGIS Model Builder in a PostgreSQL environment (cf. Section 3.2). It can preprocess all cost value required for the application in one step. Despite choosing Zurich City as study area, a later adaption and optimization could easily be performed by changing input parameters for any desired location.

The graph building application consists of the following stages, which are built as an automated data pipeline beginning in Section 4.2.3: road network preparation, extraction of altitude values from a DEM, calculation of intermediate model values (impeding forces, wheel torque and angular velocity), and computation of electrical motor power and energy consumption. Naturally, all edge costs must be computed both for a forward and backward traversal, for different bike parameters (such as recuperation mode switched on or off), and different target velocities. To perform model iterations only for specific parameters at the end of the workflow, the model is divided into submodels, which the program runs consecutively.

### 4.2.1. Basic Preparation

The first step is to set up a framework. The installed versions of PostgreSQL (with PostGIS and pgRouting) and ArcGIS must match in order to function properly, which is why the versions must meet specific requirements (PostgreSQL 9.3 resp. 9.4 for ArcGIS <9.5 by the time I started to write this thesis; ESRI 2016a). I created a new database named *ebike* which will store the data (SQL expression):

```
CREATE DATABASE ebike;
```

This database is defined as maintenance database (in pgAdmin3: disconnect server > RC Server > Properties > maintenance DB: ebike). All other (default) databases need to be disconnected to avoid any mis-allocation of the required schemas and extensions:

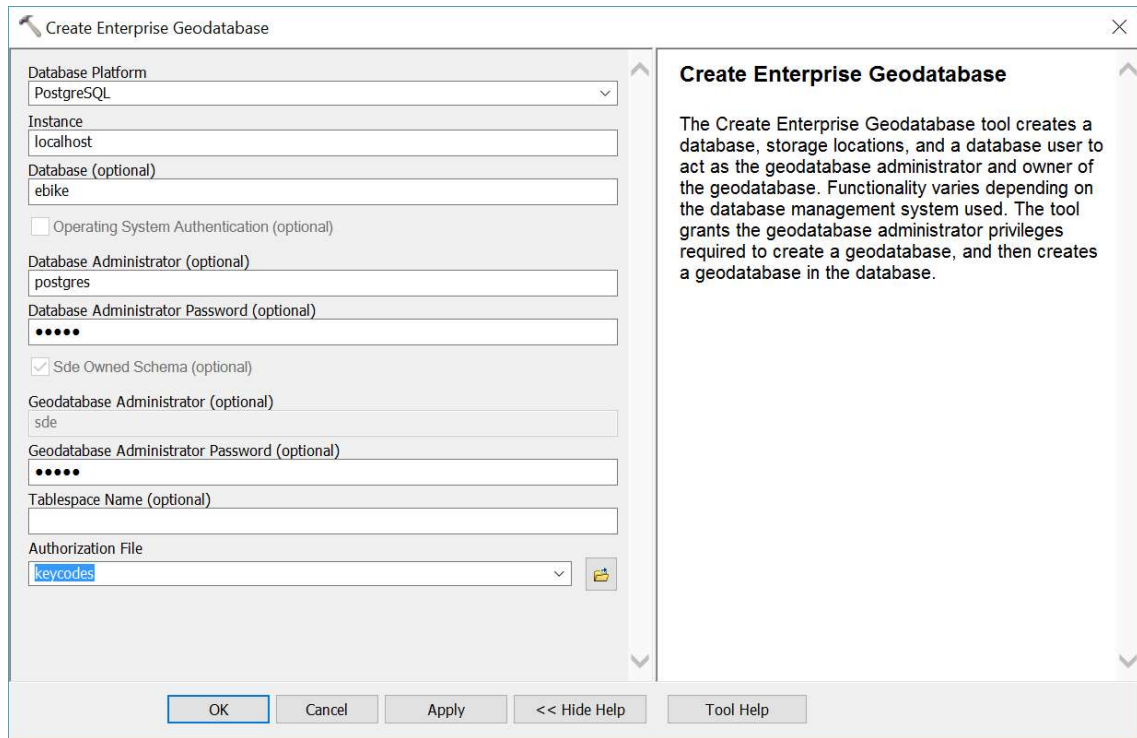
```
CREATE SCHEMA postgres;
CREATE EXTENSION postgis WITH SCHEMA public;
CREATE EXTENSION pgrouting WITH SCHEMA public;
```

As shown above, functions must be stored in schema *public*, so that the ORDBMS and ArcGIS can communicate over ArcSDE. However, due to restrictions by ArcGIS, data generated in Model Builder can neither be stored in the default schema *public* nor in (the later created) *sde*, but in the new created schema *postgres*. However, the above-mentioned extensions and related functions can only be stored in the default schema *public*. Otherwise, ArcGIS can't access them. I copy *st\_geometry.dll* (usually stored) in C:\Program Files (x86)\ArcGIS\Desktop10.4\DatabaseSupport\PostgreSQL\9.4\Windows64 into C:\Program Files\PostgreSQL\9.4\lib (default paths). In the next step, I create a folder *Workspace* (which will contain *Current* and *Scratch Workspace* as explained below) and a subfolder *Scratch*.

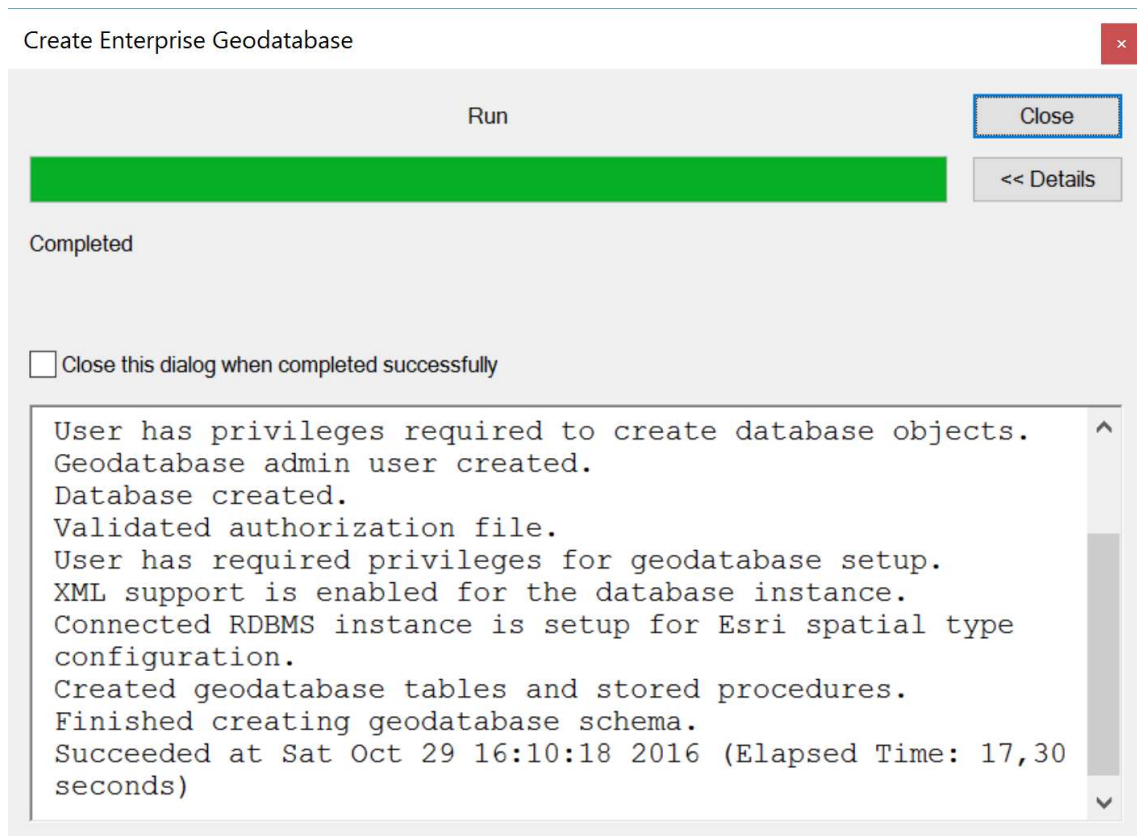
#### 4.2.2. Preparation in ArcGIS

ArcGIS needs connection to the ORDBMS in order to access the inherent data. Thus, I execute "Create Enterprise Geodatabase". This tool creates i.a. a new schema *sde* which ensures compatibility between PostgreSQL and ArcGIS. Therefore, a new database, storage locations and a database user to act as the geodatabase administrator and owner of the geodatabase are established automatically. The tool grants the geodatabase administrator privileges required to create a geodatabase in the database. I used input parameter and stated in Figure 6. Figure 7 shows the succeeded execution.

**Figure 6:** Input Parameter of the Tool “Create Enterprise Geodatabase”. The database platform is PostgreSQL, the instance a localhost, the database named ebike, the database administrator postgres with the password ebike. A new geodatabase administrator sde with the password ebike is created. The field authorization file must contain the file of a active the ArcGIS Server license (cf. Section 3.2).



**Figure 7:** Succeeded processing of the Tool “Create Enterprise Geodatabase”.



The next step is to create a database connection file in the *Workspace* folder using “Create Database Connection”, which ultimately acts as *Current Workspace*, specified as default geodatabase (RC on *databaseconnectionfile.sde* > Make Default Geodatabase). In this database, I compile a new toolbox *ebike.postgres.energy\_consumption\_model* (RC on *databaseconnectionfile.sde* > New > Toolbox). This newly created toolbox will contain all models and submodels used for the calculation of the energy consumption (RC on *ebike.postgres.energy\_consumption\_model* > New > Model). For this purpose, I use the following pattern:

*energyconsumptionmodel\_[manufacturer]\_[e-bike-model]*

resp. *energyconsumptionsubmodel\_[e-bike-model]\_[number of submodel]*

Example:

*energyconsumptionmodel\_egomovement\_whiteknight.py*

resp. *energyconsumptionsubmodel\_whiteknight\_1.py*

*energyconsumptionsubmodel\_whiteknight\_2.py*

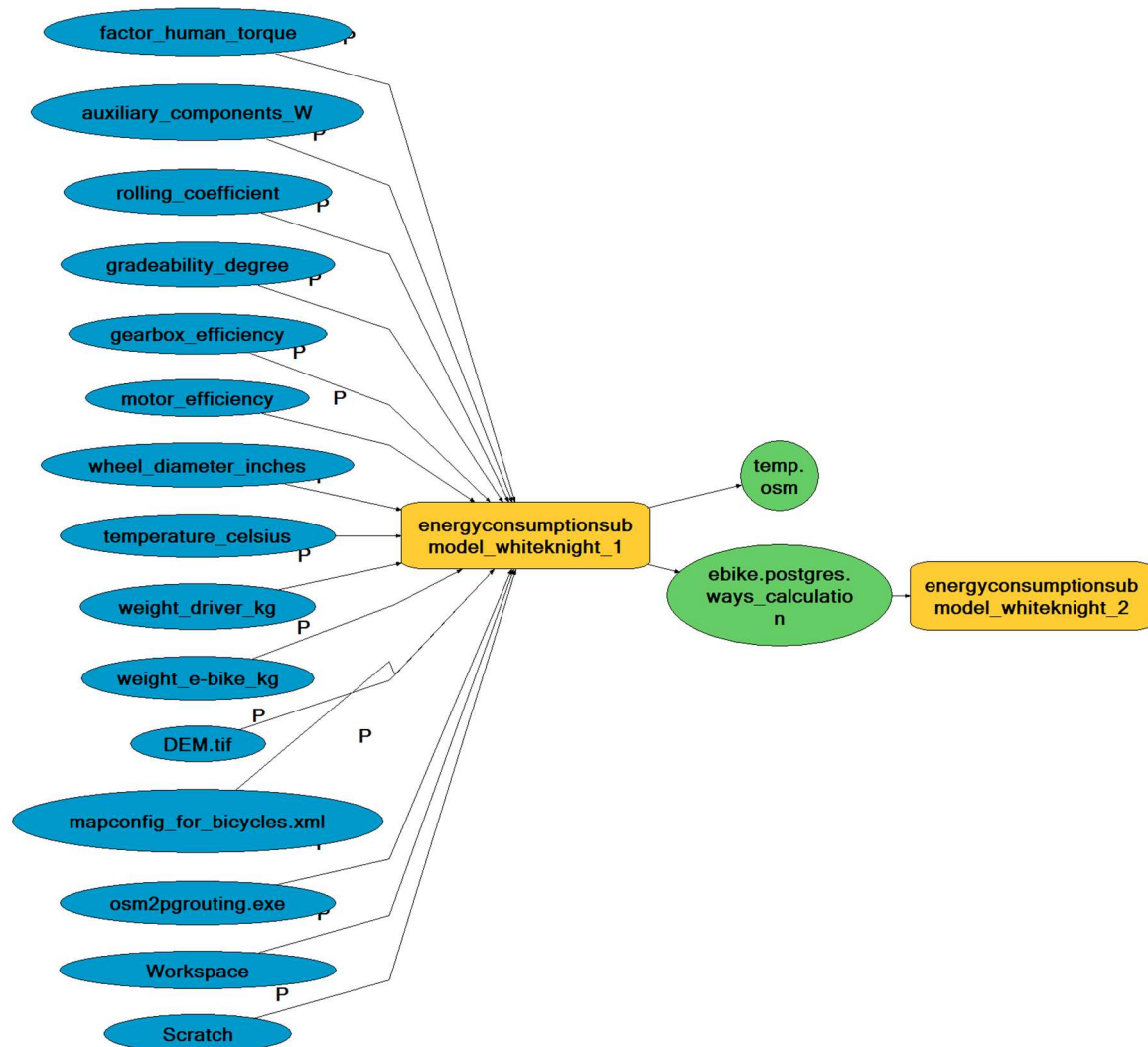
*energyconsumptionmodel\_stromer\_st2.py*

resp. *energyconsumptionsubmodel\_st2\_1.py*

*energyconsumptionsubmodel\_st2\_2.py*

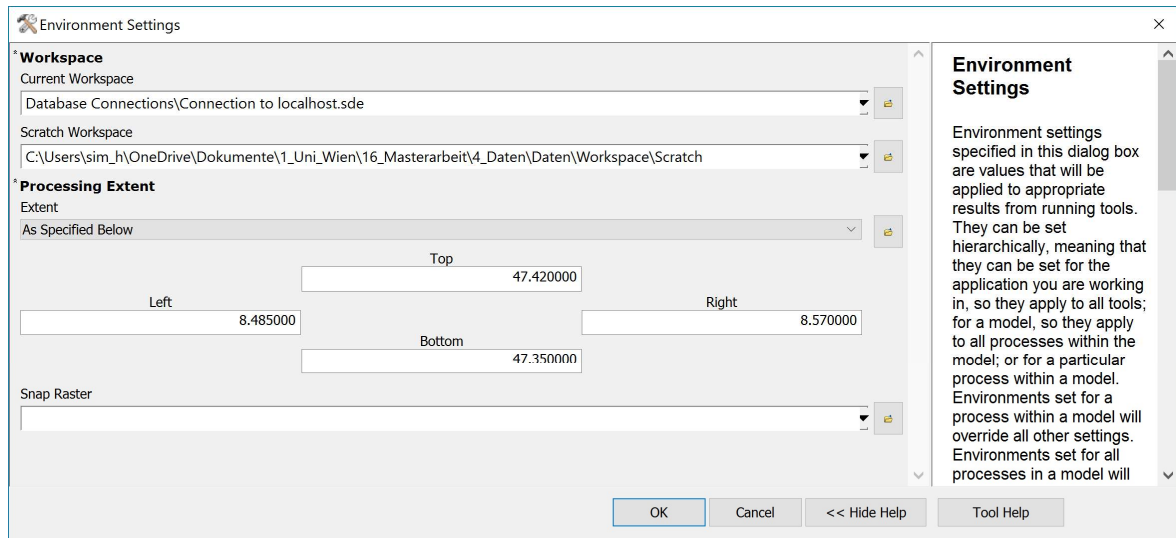
*energyconsumptionsubmodel\_st2\_3.py*

**Figure 8:** The energy consumption model for the bike without recuperation (White Knight from EGO Movement), divided into two submodels which are computed subsequently in an automated data pipeline. The parameters on the left side correspond the established ones in Section 4.1 and are filled with values from Table 9 in Section 6.2.1.



The label [e-bike model] denotes the electric bicycles from Section 3.3 tested in Chapter 6, which have marginal differences in calculation (cf. Programming Code ArcGIS Model Builder in Appendix). In each superior model, I created a *Scratch Workspace* (%Scratch%) and a *Current Workspace* (%Workspace%) and several other variables, as the example in Figure 8 as well as Figure 10 (Section 4.2.3) illustrates. Those variables were established in Section 4.1 and summarized in Table 9 and Table 10 in Section 6.2. On the one hand, *Scratch workspace* is used to access raw data such as the DEM and stores Python scripts or with the database incompatible data such as a temporary OSM file (removed by using the tool “Delete” unless determination through RC on file > Intermediate possible). On the other hand, *Current Workspace* covers the connection to the database (specified in LC on Model > Model Properties > Environments) where the calculated data is stored. All data is stored using relative path names. Figure 9 shows the specified “Environment Settings”.

**Figure 9:** "Environment Settings" of each model. The path of both the current and the scratch workspace as well as the processing extent is defined here.



The variables contain the input model parameters, established in Section 4.1 and filled with values from Section 6.1. Defining variables makes it possible to access them in arbitrary parts of the model and adjust values at will. To simplify understanding due to the amount of input and output variables, all labels have the same pattern:

[variable]\_[presence of recuperation]\_[direction of traversal]\_[unit]

Example:  $P_{amb}$       pressure\_r\_hPa  
 $E_{EM}$       v%velocity\_kmh%\_rec\_r\_Wh



The following annotations might facilitate the comprehension:

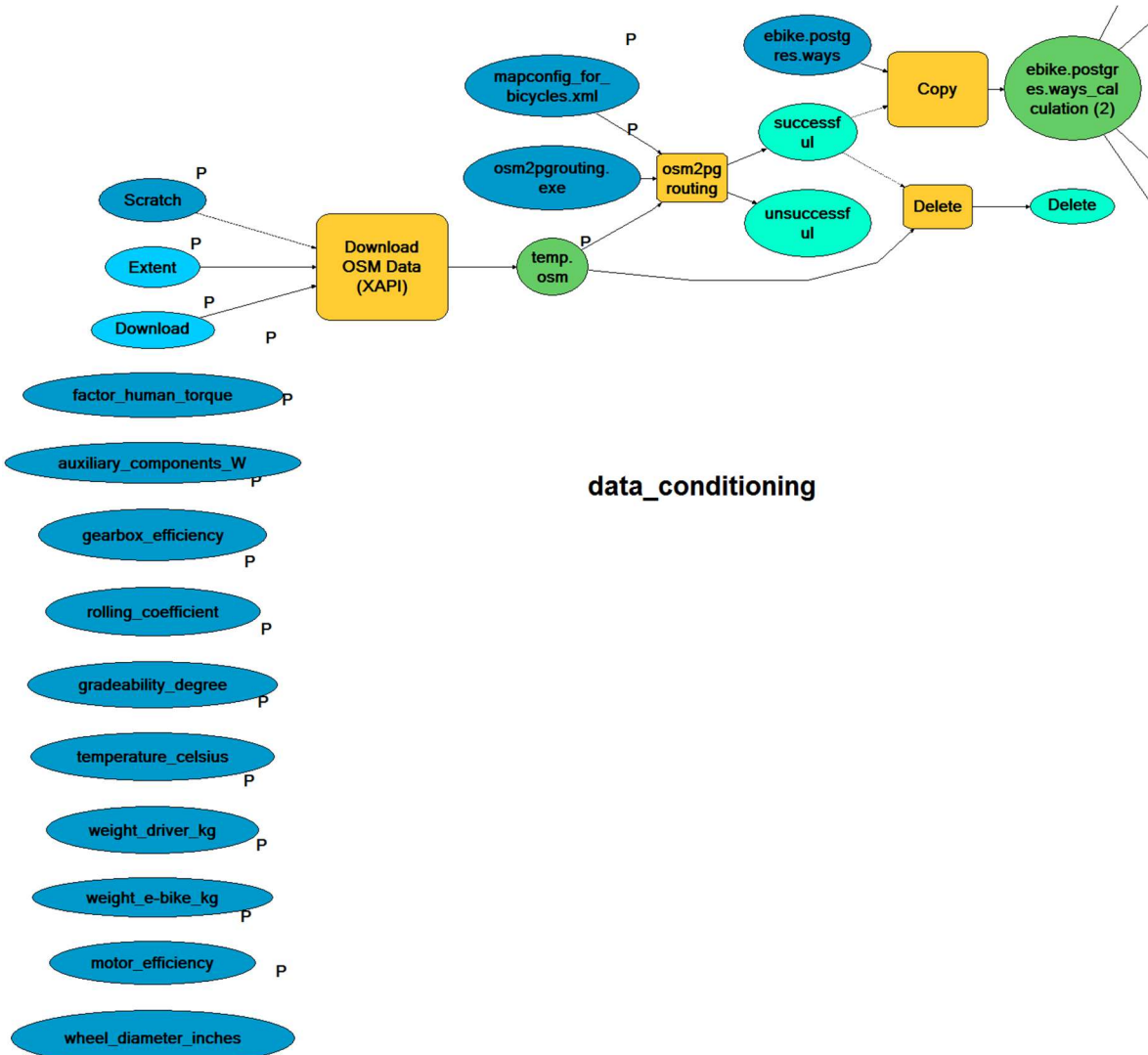
The unit is stated at the end of each expression throughout the entire model (e.g., `weight_e-bike_kg`). As the unit is only stated if there is one (e.g., `pressure_hPa` vs. `motor_efficiency` (dimensionless)), `[direction of traversal]` in front of it appears only when going backwards `[r]` (e.g., `pressure_hPa` for forward traversal vs. `pressure_r_hPa` for backward traversal). Computing reverse costs is mandatory for the use in directed graphs such as a street network (cf. Section 2.1).

The end of the model constitutes an exception (Section 4.2.7): To reduce the length of the expression, you can note the variable only by its unit (e.g.,  $P_{EM}$  by `[W]` or  $E_{EM}$  by `[Wh]`), whereas the variable iteration becomes its label (e.g., `v%velocity_kmh%` resp. `v20` for a velocity of 20km/h) – e.g. `v20_norec_Wh`. The optional presence of recuperation (`rec/norec`) is established at the end of the model, which is why it does not emerge in any previous stage.

With this kind of notation, an arbitrary extension of the model (e.g., additional iterations for weight levels or temperature) becomes viable.

4.2.3. Data Conditioning

**Figure 10:** The beginning of the energy consumption calculation in the first submodel. It contains all model parameters followed by several data preparation stages (which include automated data retrieval, and pre-processing of the street graph).



The data conditioning stage starts by downloading OSM data for a defined extend (see Section 3.4) through an Extended Application Programming Interface (XAPI) directly from one of the OSM Servers ([http://www.overpass-api.de/api/xapi\\_meta?](http://www.overpass-api.de/api/xapi_meta?)), as shown in Figure 11. A request predicate is set to include negotiable roads only and reduce processing time subsequently:

highway=primary|primary\_link|secondary|tertiary|residential|living\_street|track|pedestrian|path|cycleway|footway|byway|unclassified|secondary\_link|tertiary\_link|lane|track|opposite\_lane|opposite|grade1|grade2|grade3|grade4|grade5|roundabout



Figure 11: Input Parameters of the tool "Download OSM Data (XAPI)"

I import the originated OSM file into the pgRouting database using an implemented script, which defines a suitable routing topology at the same time. This python script shown below executes the `osm2pgrouting` command using the subprocess module. The implementation pursues the approach of fully automated processing and enables easy customizability through the shift of input parameters (e.g., database access data could be incorporated into the model as well). After implementing it into the model, the script could use any configuration file, e.g., in this case one optimized for bicycles (see Section 3.2), whose path has to be located in the *bin* folder of PostgreSQL/9.4. Among other arguments, I defined a Boolean expression which continues the process if succeeded.

```
import subprocess
import sys
import arcpy

try:
    osm2pgrouting = arcpy.GetParameterAsText(0)
    osm_file      = arcpy.GetParameterAsText(1)
    conf_file     = arcpy.GetParameterAsText(2)
    print osm2pgrouting
    print osm_file
    print conf_file

    response = subprocess.check_output([osm2pgrouting,
                                       '--file', osm_file,
                                       '--conf', conf_file,
                                       '-p', '5432', '--dbname', 'ebike',
                                       '--user', 'postgres', '--password',
                                       'ebike', '--clean']).decode("utf-8")

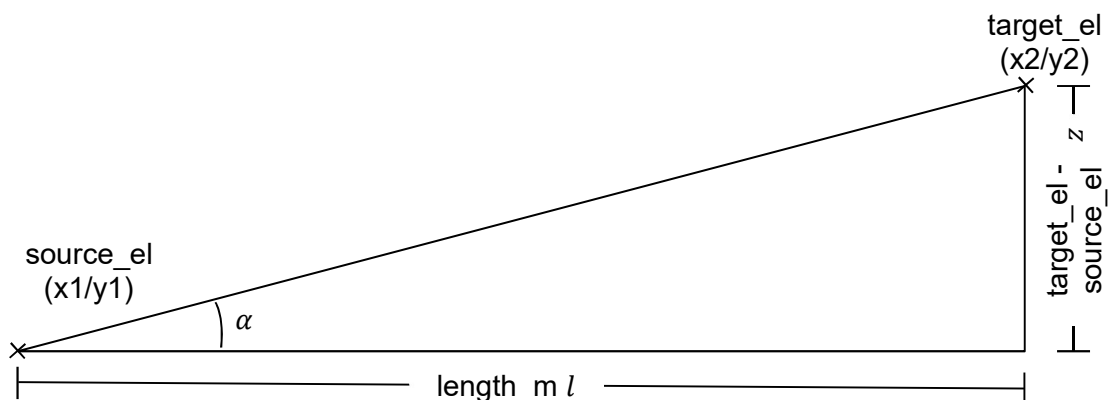
    print response
except subprocess.CalledProcessError as e:
    print "Have an error in processing:", sys.exc_info()[0]
    print "Ping stdout output:\n", e.output
    arcpy.SetParameterAsText(4, "false")
else:
    arcpy.SetParameterAsText(3, "true")
```

Using Model Builders' own relation "precondition", the model deletes the downloaded temporary OSM file and initiates the replication of the generated file ways (edge table as the key to conduct routing) likewise. Although the program claims, the input and output data element were identical, further proceeding with the original file causes errors. By updating the original file at the completion of the calculation with relevant columns only, I hereby reduce the amount of data necessary for the subsequent routing. These steps ultimately complete the data preparation stage.

#### 4.2.4. Altitude Value Extraction

To obtain the slope angle for each street segment, I need to extract altitude values from a DEM and project them onto the street network. Figure 12 illustrates the trigonometric approach to calculate the required value. I conducted the following steps (cf. Figure 13).

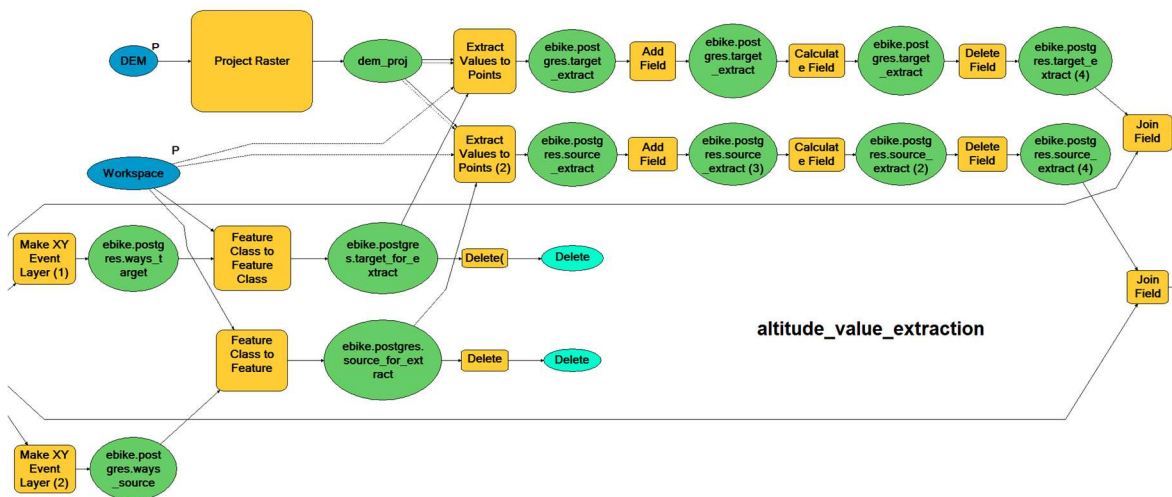
**Figure 12:** Schematic illustration of the calculation of slope. The elevation point at the source location *source\_el* of a specific street segment with the length *length\_m* has the coordinates  $x1/y1$ , the one at the target location *target\_el* has  $x2/y2$ . While *length\_m* symbolizes the adjacent side  $l$ , the difference of the values for *target\_el* and *source\_el* results into opposite side  $z$ .



First of all, height values need to be extracted from the raster (DEM) and added to the street network nodes (OSM). For this purpose, points are required to store the altitude values at the beginning and end of each street segment. Because the coordinates of the start and end node of each segment are provided through  $x1/y1$  for the source and  $x2/y2$  for the target location already, I can use "Make XY Event Layer" to convert this information into point features (instead of using the tailored "Feature Vertices To Points"). The integration of the tool "Feature Class to Feature Class" is mandatory to proceed, since the raw (temporary) point feature layer cannot serve as input feature for the following "Extract Values to

Points”. Once the DEM has been converted into the same coordinate system (CH1903 to WGS 1984), it is stored as the single compatible ESRI GRID (instead of the original GeoTIFF format)<sup>4</sup>. Its elevation values are interpolated at the point location through “Extract Values to Points” (check optional “Interpolate values at the point locations” to minimize potential miscalculations) which computes a default field called *rastervalu*. Since I need to execute this function twice both for the source and target and for reasons of comprehension, I renamed the generated default label into *source\_el* resp. *target\_el*. I used “Add Field”, “Calculate Field” and “Delete Field” (to delete the generated *rastervalu*) in the stated order (the much simpler “Alter Field” caused severe errors in further calculations).

Figure 13: The altitude value extraction procedure.



The transfer of elevation values causes scattered inaccurate height allocations due to interpolation errors through nearby bridges, walls, or scarps. Hence, slope angle contains some extreme values (cf. Figure 43Figure 44Figure 45 in Section 8.2). Those can be considered as outliers. Since I did not yet alter values above a certain threshold value manually (e.g., according to constructional regulations for maximal slopes), no data values are assigned to values above a certain gradeability of the motor of the the electric bicycle (cf. Section 4.2.7). This approach excludes outliers from the routing procedure at the same time. As a result, few street segments which are actually less steep and thus negotiable are not taken into account. Due to small quantity of the affected values (under 1 % of the values above 15 and below -15 degrees of street segment for the study area), this effect can be considered negligible for the time being, whereas other solutions are discussed in Section 8.2.

<sup>4</sup> Clipping or an alignment on the extent might become important when, due to further automation, downloading the raster directly (cf. Section 8.2). It is negligible for the experimental setup.

After joining the intermediate data (using the field *gid*), the slope (in percentage as *slope\_percentage* and angle as *slope\_degree*) is calculated. For every single step from this point forward, all intermediate data is calculated for both forward and backward traversal (cf. Figure 14 in Section 4.2.5). In the sequel, the Python code contained in “Calculate Field” is shown consecutively numbered per equation in Section 4.1 and additional intermediate calculations in the term (x.1), (x.2), and so on. Initially, the model computes (0.1) slope percentage as a simple ratio between *target\_el* – *source\_el*, which is inserted into an arc tangent function yielding (0.2) the angle of slope accordingly (MUETZE & TAN 2007; BUCKLEY 2008):

$$(0.1) \quad slope_{percentage} = \frac{z}{l} \cdot 100$$

```
(0.1) (!target_el!-!source_el!)/!length_m!*100 #slopeangle_percentage
```

$$(0.2) \quad slope_{degree} = \tan^{-1}\left(\frac{z}{l}\right)$$

```
(0.2) math.degrees(math.atan((!target_el!-!source_el!)/!length_m!)) #slopeangle_degree
```

#### 4.2.5. Impeding Forces

Subsequently, the model computes (1) climbing resistance  $F_g$ , and (2) rolling resistance  $F_f$ :

```
(1) (%weight_driver_kg%+%weight_e-bike_kg%)*9.806*
math.sin(math.radians(!slopeangle_degree!)) #climbing_resistance
```

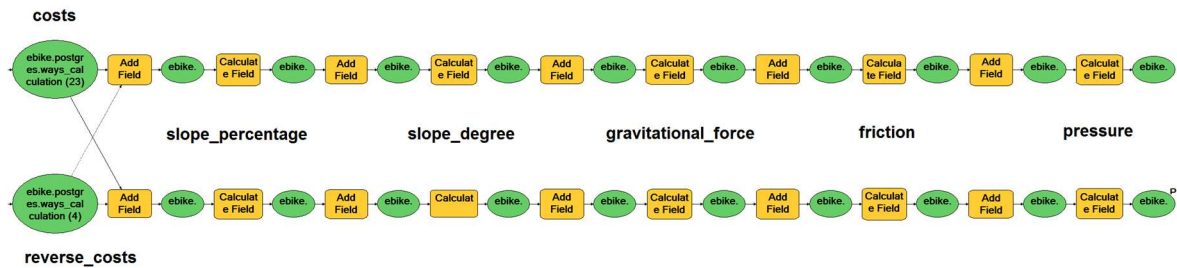
```
(2) %rolling_coefficient%*(%weight_driver_kg%+%weight_e-bike_kg%)
*9.806*math.cos(math.radians(!slopeangle_degree!)) #rolling_resistance
```

The code in (3.1) denotes the computation of the ambient air pressure  $P_{amb}$  through the international height formula, a premise for the calculation of drag  $F_d$ . It contains the standard atmosphere 1013.25 hPa, the temperature gradient of 0.0065 K / m, the average height between the two end nodes of a street segment  $h$ , and the ambient temperature  $T_{amb}$ :

$$(3.1) \quad P_{amb} = 1013.25 \cdot \left(1 - \frac{0.0065 \cdot h}{T_{amb}}\right)^{5.255}$$

```
(3.1) 1013.25*math.pow(1-(0.0065*((!target_el!+!source_el!)/2))
/(!temperature_celsius%+273.15),5.255) #ambient_air_pressure
```

**Figure 14:** The end of the first submodel with the calculation of slope in degree required to obtain the impeding forces gravitation and friction and the pressure required to obtain drag for both forward and backward traversal respectively.



Please note: Because of the performed iteration the model is split into parts as Figure 8 shows exemplary. The computation of (3) drag  $F_d$  indicates the beginning of the second part which iterates dependent variables for defined velocity levels  $v_{DD}$  `%velocity_kmh%` using the iterator “For” (Figure 15).

```
(3) !pressure_hPa!*100/(2*287.058*(%temperature_celsius%+273.15))
    *1.15*0.55*math.pow((%velocity_kmh%/3.6),2) #drag_resistance
```

Finally, I add up all previously computed impeding forces to the tractive force  $F_T$ :

```
(4.1) !climbres_N!+!rollres_N!+!dragres_v%velocity_kmh%_N! #tractive_force
```

To consider (4) the rider power input  $F_h$ , I assume the human torque to be a constant factor  $\gamma$  of the tractive force  $F_T$ , ultimately resulting into (4.2) motor torque  $T_{EM}$ :

```
(4.2) def factorm(tracforcevvelocity_kmh_N, factor_human_torque, wheel_diameter_inches):
    if tracforcevvelocity_kmh_N > 0:
        return (tracforcevvelocity_kmh_N-(tracforcevvelocity_kmh_N*factor_hu-
        man_torque))*((wheel_diameter_inches*0.0254)/2)
    else:
        return tracforcevvelocity_kmh_N*((wheel_diameter_inches*0.0254)/2)
    #motor_torque
```

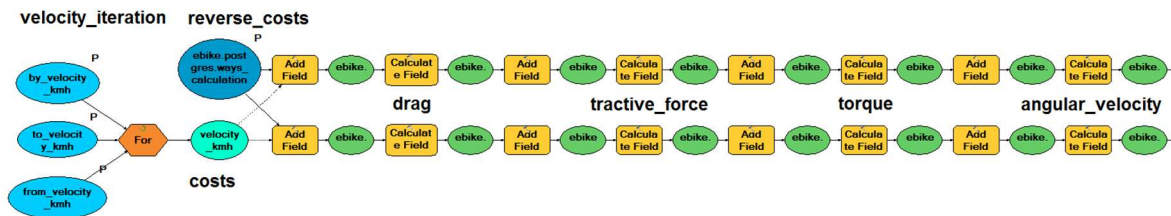
Since the rider needs to pedal only when the tractive force is positive, it does not need to be subtracted when it is negative resp. the motor is in generator mode. The pre-logic script code is executed by:

```
factorm(!tracforcev%velocity_kmh%_N!, %factor_human_torque%, %wheel_diameter_inches%)
```

```
(4.4) (%velocity_kmh%/3.6)/((%wheel_diameter_inches%*0.0254)/2) #angular_velocity
```

Together with the (4.4) angular velocity of the wheel  $w_v$ , we are now able to calculate the required electric motor power.

**Figure 15:** The beginning of the second submodel with the iteration of different velocity levels with the computation of the remaining impeding forces, summed up to the tractive force. Hence, wheel torque and angular velocity are calculated.



#### 4.2.6. Determining Motor efficiency

The motor efficiency for the bike employed in Test Session A is determined by one value, which is altered throughout the tests to increase accuracy. In Test Session B, the manufacturer provided specifications for the motor efficiency for specific torques and speeds for both consumption and production mode in revolutions per minute (RPM), which I transferred into model-own entities and then implemented into the model (Figure 16). The values for each corresponding velocity level expressed in Table 6 in Section 6.1 are transformed into a Python Code implemented into “Calculate Field”, as the example below shows:

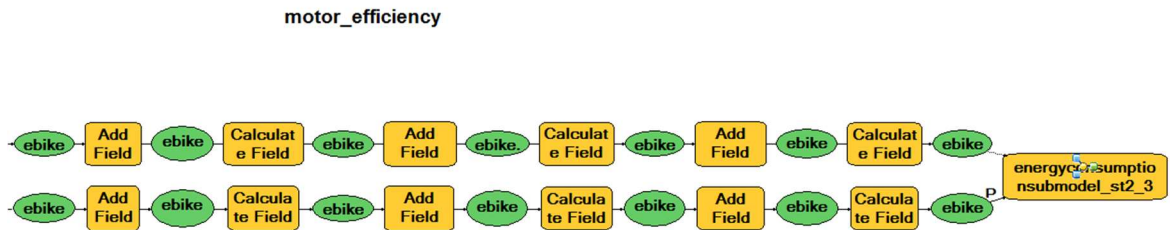
```
(4.3) def meff5(torquev5_Nm):
    if -12.5 >= torquev5_Nm:
        return 0.1686509334044
    elif -7.5 >= torquev5_Nm > -12.5:
        return 0.400272727088039
    elif 0 >= torquev5_Nm > -7.5:
        return 0.53365978248219
    elif 0 < torquev5_Nm < 7.5:
        return 0.623834964702797
    elif 7.5 <= torquev5_Nm < 12.5:
        return 0.579450836444724
    elif 12.5 <= torquev5_Nm < 17.5:
        return 0.514837509676393
    elif 17.5 <= torquev5_Nm < 22.5:
        return 0.472091740648567
    elif 22.5 <= torquev5_Nm < 27.5:
        return 0.409172192178427
    elif 27.5 <= torquev5_Nm < 32.5:
        return 0.358790286998961
    elif 32.5 <= torquev5_Nm < 37.5:
        return 0.32394748442385
    elif 37.5 <= torquev5_Nm:
        return 0.192834086191931
```

The pre-logic script code shown above (4.3) is then executed by:

```
meff5(!torquev5_Nm!)
```



Figure 16: An extract of the computation of the motor efficiency of the bike used in Test Session B.



#### 4.2.7. Electrical Motor Power and Energy Consumption

The last part of the model consists of the calculation of the required electric motor power  $P_{EM}$  and, following the reassessment of the effect of an increased energy consumption released by lower temperatures resulting into  $P_M$ , the computation of the overall energy consumption  $E_{EM}$  for each feature resp. street segment (Figure 17).

- ```
(4) def pemrecW(slopeangle_degree, gradeability_degree, torquevvelocity_kmh_Nm, angularvvelocity_kmh_s_inverse, motorefficiencyvvelocity_kmh, gearbox_efficiency, auxiliary_components_W):
    if slopeangle_degree >= gradeability_degree:
        return 999999
    elif slopeangle_degree <= -gradeability_degree:
        return 999999
    elif torquevvelocity_kmh_Nm < 0:
        return torquevvelocity_kmh_Nm * angularvvelocity_kmh_s_inverse * motorefficiencyvvelocity_kmh * gearbox_efficiency + auxiliary_components_W
    else:
        return torquevvelocity_kmh_Nm * angularvvelocity_kmh_s_inverse / (motorefficiencyvvelocity_kmh * gearbox_efficiency) + auxiliary_components_W
    #power electric motor with recuperation
```

The pre-logic script code shown above calculates (4) the electric motor power for engines with a recuperation mechanism (employed in Section 6.2.2) and is executed by:

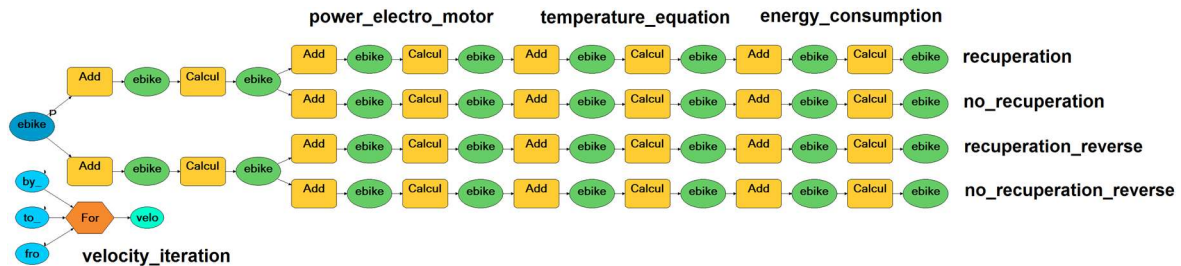
```
pemrecW(!slopeangle_degree!, %gradeability_degree%, !torquev%velocity_kmh%_Nm!, !angularv%velocity_kmh%_s_inverse!, !motorefficiencyv%velocity_kmh%!, %gearbox_efficiency%, %auxiliary_components_W%)
```

It is important to mention that all values beyond the above-mentioned maximal gradeability, specified by the e-bikes' manufacturer, are set to a no-data value.<sup>5</sup> This method ensures

<sup>5</sup> Since several routing systems (i.e., pgRouting) are not able to process pure no data values defined as such, I assigned 999999 instead.

that the routing exclude those segments and, simultaneously, eliminates scattered interpolation errors arisen from the extraction procedure. For that reason, negative slope (i.e., driving downhill) is considered as well. To spare query time, this step is implemented in the same script.

**Figure 17:** The end of the energy consumption model for bikes with recuperation (ensuing Figure 16 in Section 4.2.6), subsequently computing the power required by the electro motor (adjusted for lower temperatures), the energy consumption. The model considers that one could switch the recuperation on or off by calculating both cases.



```
(4) def pemnorecW(slopeangle_degree, gradeability_degree, torquevvelocity_kmh_Nm, angularvelocity_kmh_s_inverse, motorefficiencyvvelocity_kmh, gearbox_efficiency, auxiliary_components_W):
    if slopeangle_degree >= gradeability_degree:
        return 999999
    elif slopeangle_degree <= -gradeability_degree:
        return 999999
    elif torquevvelocity_kmh_Nm < 0:
        return auxiliary_components_W
    else:
        return torquevvelocity_kmh_Nm * angularvelocity_kmh_s_inverse / (motorefficiencyvvelocity_kmh * gearbox_efficiency) + auxiliary_components_W

#power electric motor without recuperation
```

```
pemnorecW(!slopeangle_degree!, %gradeability_degree%, !torquev%velocity_kmh% Nm!, !angularv%velocity_kmh%_s_inverse!, !motorefficiencyv%velocity_kmh%!, %gearbox_efficiency%, %auxiliary_components_W%)
```

To also model the power provided by electric bicycles with a lack of recuperation capability (employed in Section 6.2.1), required electric motor power for torques smaller than zero consists in this case only of the energy consumed by auxiliary components  $P_i$  such as display or light, as shown above. Figure 18 represents this case.

```
(6) def pemctemprecW(vvelocity_kmh_rec_W, temperature_celsius):
    if vvelocity_kmh_rec_W == 999999:
        return 999999
    if temperature_celsius >= 25:
        return vvelocity_kmh_rec_W
    else:
        return vvelocity_kmh_rec_W * (1+((25-temperature_celsius)*0.0047))

#temperature equation with recuperation
```



```
pemctemprecW(!v%velocity_kmh%_rec_W!, %temperature_celsius%)
```

After considering (6) the effect of increasing energy consumption due to lower temperatures, the multiplication of the power with the ratio between the specific length  $l$  and velocity  $v_{DD}$  ultimately results in (7) energy consumption for each feature resp. street segment.

```
(7) def pemrecWh(vvelocity_kmh_ctemp_rec_W, length_m, velocity_kmh):
    if vvelocity_kmh_ctemp_rec_W == 999999:
        return 999999
    else:
        return vvelocity_kmh_ctemp_rec_W*((length_m/1000)/velocity_kmh)

  #energy consumption with recuperation

pemrecWh(!v%velocity_kmh%_ctemp_rec_W!, !length_m!, %velocity_kmh%)
```

**Figure 18:** The end of the energy consumption model for bikes without recuperation, subsequently computing the power required by the electro motor (adjusted for lower temperatures), the energy consumption resp. energy consumption per kilometer for visualization purposes (Figure 19 and Figure 20).

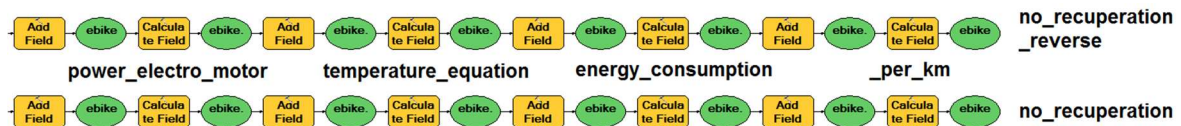


Figure 19 and Figure 20 illustrate the calculated edge weights. The execution time of the entire model is approximately 60 minutes for Test Session A and 100 minutes for Test Session B (see Table 3 in Section 3.3 for system specifications). To update the native file from the database consecutively, running the following SQL query is a crucial step:

```
-- new columns are added in the native edge cost file ways and filled with the calculated
data for energy consumption (for forward and backward traversal and every velocity iteration
respectively)

ALTER TABLE postgres.ways ADD v5_norec_Wh float8;
UPDATE postgres.ways SET v5_norec_Wh = (SELECT v5_norec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v5_rec_Wh float8;
UPDATE postgres.ways SET v5_rec_Wh = (SELECT v5_rec_Wh FROM postgres.ways_calculation WHERE
postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v5_norec_r_Wh float8;
UPDATE postgres.ways SET v5_norec_r_Wh = (SELECT v5_norec_r_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v5_rec_r_Wh float8;
UPDATE postgres.ways SET v5_rec_r_Wh = (SELECT v5_rec_r_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);

ALTER TABLE postgres.ways ADD v10_norec_Wh float8;
UPDATE postgres.ways SET v10_norec_Wh = (SELECT v10_norec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v10_rec_Wh float8;
UPDATE postgres.ways SET v10_rec_Wh = (SELECT v10_rec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
```

```

ALTER TABLE postgres.ways ADD v10_norec_r_Wh float8;
UPDATE postgres.ways SET v10_norec_r_Wh = (SELECT v10_norec_r_Wh FROM postgres.ways_calcu-
lation WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v10_rec_r_Wh float8;
UPDATE postgres.ways SET v10_rec_r_Wh = (SELECT v10_rec_r_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);

ALTER TABLE postgres.ways ADD v15_norec_Wh float8;
UPDATE postgres.ways SET v15_norec_Wh = (SELECT v15_norec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v15_rec_Wh float8;
UPDATE postgres.ways SET v15_rec_Wh = (SELECT v15_rec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v15_norec_r_Wh float8;
UPDATE postgres.ways SET v15_norec_r_Wh = (SELECT v15_norec_r_Wh FROM postgres.ways_calcu-
lation WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v15_rec_r_Wh float8;
UPDATE postgres.ways SET v15_rec_r_Wh = (SELECT v15_rec_r_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);

ALTER TABLE postgres.ways ADD v20_norec_Wh float8;
UPDATE postgres.ways SET v20_norec_Wh = (SELECT v20_norec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v20_rec_Wh float8;
UPDATE postgres.ways SET v20_rec_Wh = (SELECT v20_rec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v20_norec_r_Wh float8;
UPDATE postgres.ways SET v20_norec_r_Wh = (SELECT v20_norec_r_Wh FROM postgres.ways_calcu-
lation WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v20_rec_r_Wh float8;
UPDATE postgres.ways SET v20_rec_r_Wh = (SELECT v20_rec_r_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);

ALTER TABLE postgres.ways ADD v25_norec_Wh float8;
UPDATE postgres.ways SET v25_norec_Wh = (SELECT v25_norec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v25_rec_Wh float8;
UPDATE postgres.ways SET v25_rec_Wh = (SELECT v25_rec_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v25_norec_r_Wh float8;
UPDATE postgres.ways SET v25_norec_r_Wh = (SELECT v25_norec_r_Wh FROM postgres.ways_calcu-
lation WHERE postgres.ways_calculation.gid = postgres.ways.gid);
ALTER TABLE postgres.ways ADD v25_rec_r_Wh float8;
UPDATE postgres.ways SET v25_rec_r_Wh = (SELECT v25_rec_r_Wh FROM postgres.ways_calculation
WHERE postgres.ways_calculation.gid = postgres.ways.gid);

-- as some operations are only possible in the schema public due to database restrictions
(cf. Section 4.2.1) files are copied accordingly

CREATE TABLE public.ways AS
SELECT * FROM postgres.ways;

CREATE TABLE public.ways_vertices_pgr AS
SELECT * FROM postgres.ways_vertices_pgr;

CREATE TABLE public.relations_ways AS
SELECT * FROM postgres.relations_ways;

CREATE TABLE public.osm_relations AS
SELECT * FROM postgres.osm_relations;

CREATE TABLE public.osm_nodes AS
SELECT * FROM postgres.osm_nodes;

CREATE TABLE public.osm_way_classes AS
SELECT * FROM postgres.osm_way_classes;

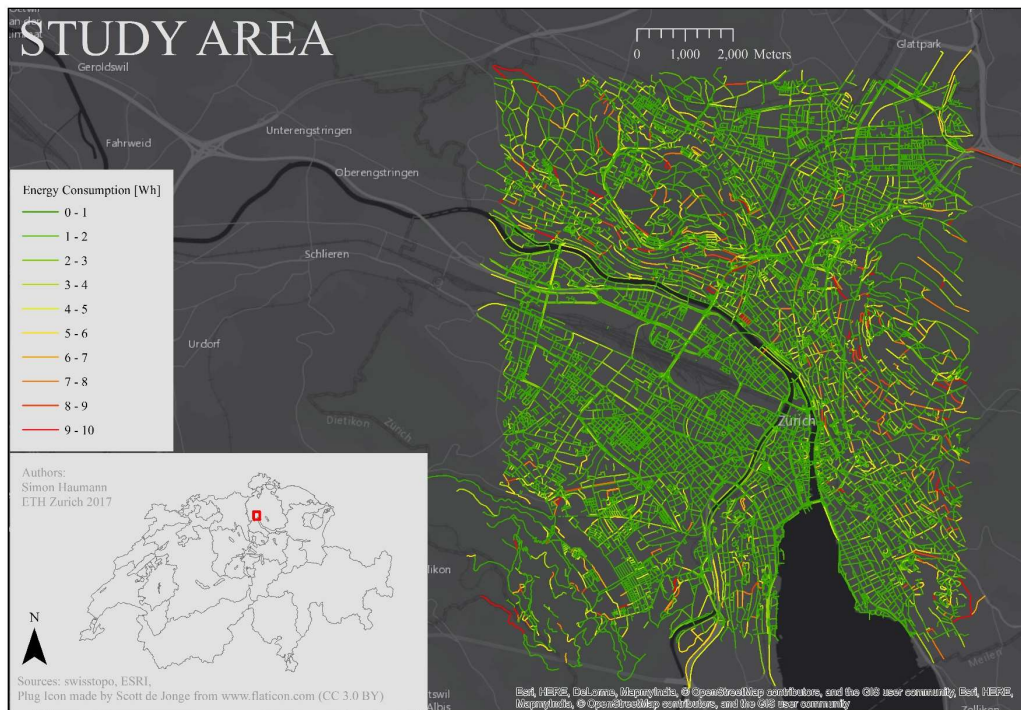
CREATE TABLE public.osm_way_tags AS
SELECT * FROM postgres.osm_way_tags;

CREATE TABLE public.osm_way_types AS
SELECT * FROM postgres.osm_way_types;

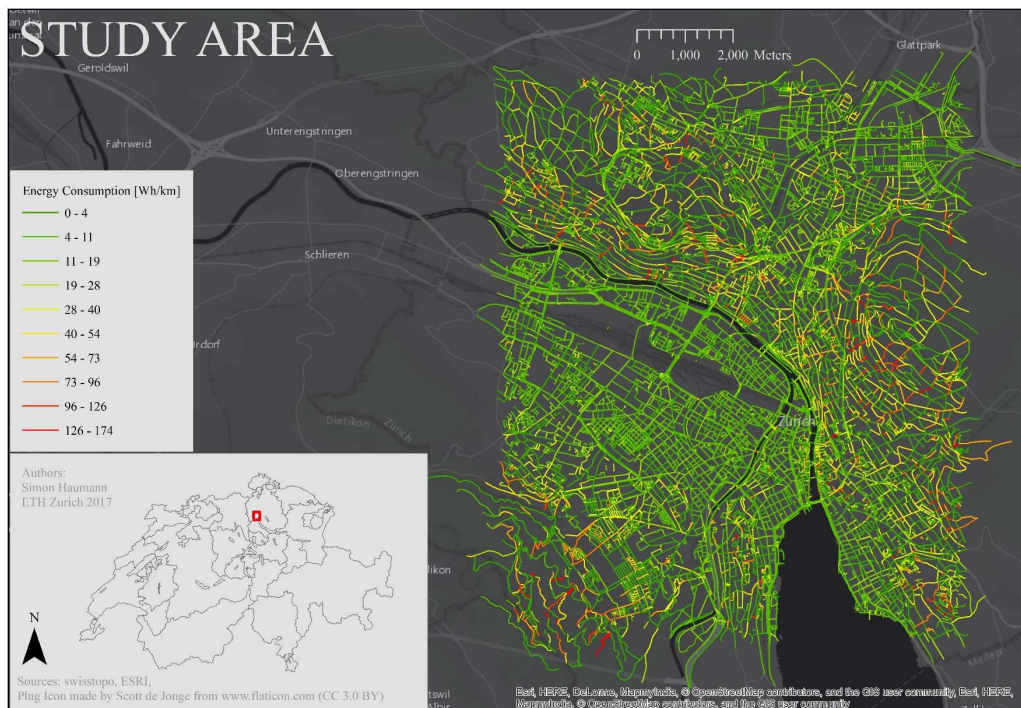
```



**Figure 19:** The energy consumption [Wh] of each street segment in the study area. Naturally, longer street segments inherit higher values of energy consumption. Figure 20 addresses this concern.



**Figure 20:** The hypothetical energy consumption per kilometer [Wh/km] for each street segment. This illustration facilitates a comparison between the expected energy demand of particular street segments despite their length.





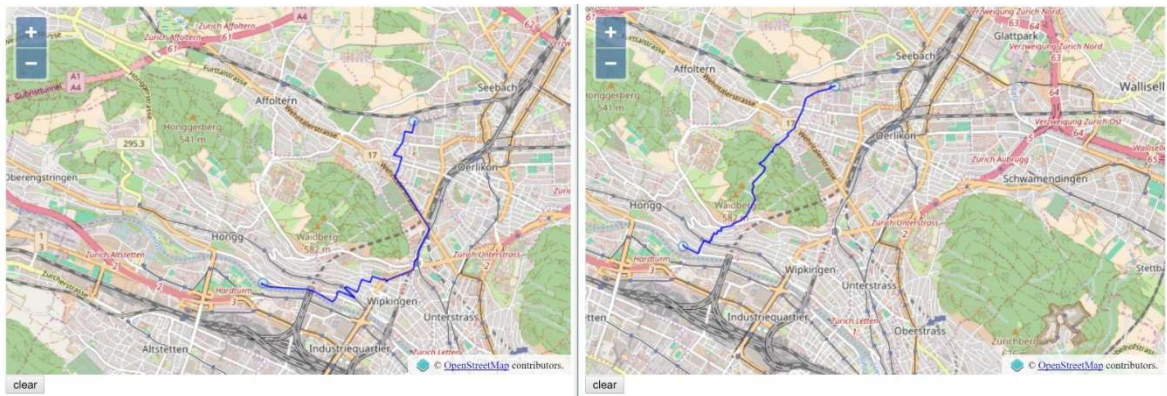


## 5. Routing Applications

In order to request a least energy consuming route, I need to build an application using the algorithms addressing the shortest path problem introduced in Section 2.1. In a first step, for bikes without recuperation and thereby generating only positive edge costs, I compute routes with pgRouting using the Dijkstra algorithm (Section 5.1). Second, my own development of a routing application using an implementation of the Bellman-Ford algorithm written in Rust enables me to involve negative edge costs caused by an energetic recovery system henceforth (Section 5.2).

### 5.1. Dijkstra Application

**Figure 21:** Comparison between an energy-based path (left side) and the shortest path (right side). Parameters taken from Test Session A ( $T_{amb} = 0$ ,  $m=123$ ,  $v_{DD} = 20$ ).



The first approach is to develop an application using the Dijkstra algorithm, applicable for bikes without recuperation (Figure 21). With time complexity of  $O(n^2)$ , according to ARTMEIER & HASELMAYR 2010 it is the best known algorithm for the processing of non-negative edge costs.

Disregarding the availability of several other route planning algorithms with speed-up technique to reduce query time, preprocessing time and space consumption (DELLING ET AL. 2009) as the A\*- algorithm (HART ET AL. 1968) in the pgRouting library, I used the existing implementation of Dijkstra in the pgRouting framework.

#### 5.1.1. Back-End

The following description of the setup of the pgRouting Dijkstra Application is inspired by the workshop of the FOSS4G 2016 in Bonn and adjusted for my own purposes (KASTL & VERGARA 2016). First, I create a function `pgr_v20_norec_Wh.sql`, which processes input parameters both from an edge dataset (in this case `ways`, specified later on) such as edge

costs (e.g., *v20\_norec\_Wh*), and from an attached node dataset, specified as *ways\_vertices\_pgr* (e.g., *longitude* and *latitude*). Taking those input parameters into account, the function runs the shortest path Dijkstra query after finding the nearest nodes to start and end-point coordinates. It flips the geometry if necessary, that target node of the previous road segment is the source of the following. Also, it calculates the azimuth from start to end node of each road segment. For this purpose, the function employs functions from the PostGIS library. *pgr\_v20\_norec\_Wh* ultimately returns a set of records: A sequence, gid, edge costs as energy costs for each segment, street names, a geometry and the heading in degree.

```
CREATE OR REPLACE FUNCTION pgr_v20_norec_Wh(
    IN edges_subset varchar,
    IN x1 double precision,
    IN y1 double precision,
    IN x2 double precision,
    IN y2 double precision,
    OUT seq INTEGER,
    OUT cost FLOAT,
    OUT name TEXT,
    OUT geom geometry,
    OUT heading FLOAT
)
RETURNS SETOF record AS
$BODY$

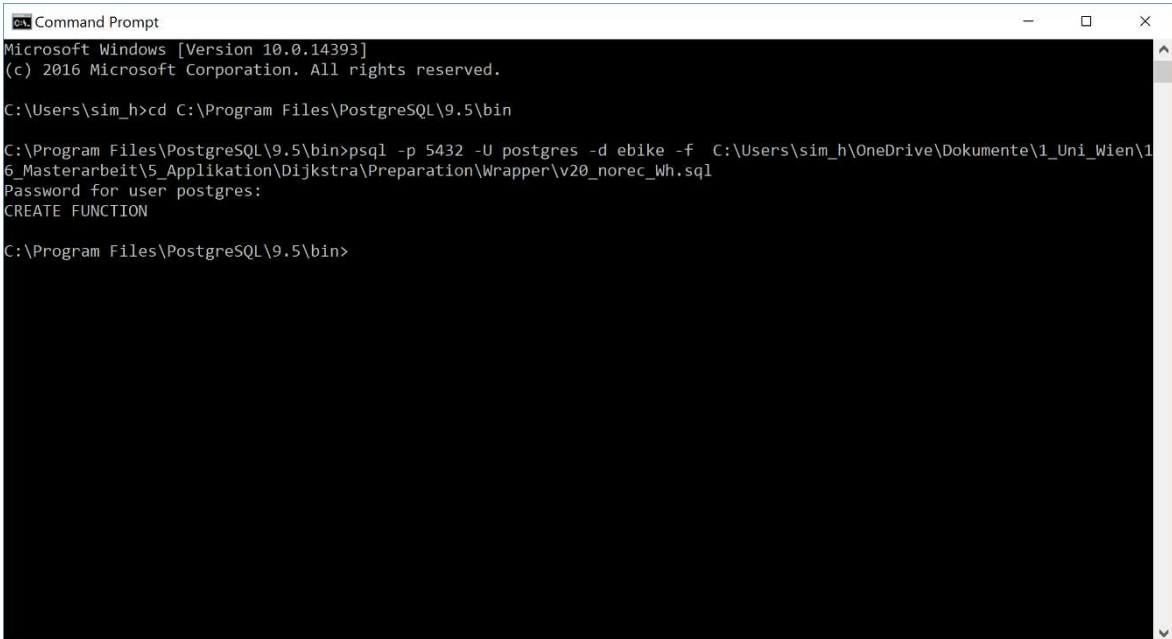
WITH
dijkstra AS (
    -- the following runs the shortest path Dijkstra query, which contains input parameters
    -- for the algorithm such as source and target
    SELECT * FROM pgr_dijkstra(
        'SELECT gid as id, source, target, v20_norec_Wh AS cost, v20_norec_r_Wh AS re-
        verse_cost FROM ' || $1,
        -- allocation for finding the nearest nodes to start and endpoint coordinates
        -- source
        (SELECT id FROM ways_vertices_pgr
         ORDER BY the_geom <-> ST_SetSRID(ST_Point(x1,y1),4326) LIMIT 1),
        -- target
        (SELECT id FROM ways_vertices_pgr
         ORDER BY the_geom <-> ST_SetSRID(ST_Point(x2,y2),4326) LIMIT 1),
        -- directed, reverse costs
        true)
    ),
with_geom AS (
    SELECT dijkstra.seq, dijkstra.cost, ways.name,
        -- flips the geometry if necessary, that target node of the previous road segment
        -- is the source of the following
        CASE
            WHEN dijkstra.node = ways.source THEN the_geom
            ELSE ST_Reverse(the_geom)
        END AS route_geom
    FROM dijkstra JOIN ways
    ON (edge = gid) ORDER BY seq
    )
    -- calculates the azimuth from start to end node of each road segment
    SELECT *,
    ST_azimuth(ST_StartPoint(route_geom), ST_EndPoint(route_geom))
    FROM with_geom;
$BODY$
LANGUAGE 'sql';
```



To store the above described wrapper into the ORDBMS, the user needs to execute the following statement in the command line as shown in Figure 22 (change the direction into the target folder of psql; usually `cd C:\Program Files\PostgreSQL\9.5\bin\`):

```
psql -p 5432 -U postgres -d ebike -W ebike -f C:\...\v20_norec_Wh.sql
```

**Figure 22:** The screenshot shows the required arguments to store the function, executed in the command line.



```

Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\sim_h>cd C:\Program Files\PostgreSQL\9.5\bin

C:\Program Files\PostgreSQL\9.5\bin>psql -p 5432 -U postgres -d ebike -f C:\Users\sim_h\OneDrive\Dokumente\1_Uni_Wien\16_Masterarbeit\5_Applikation\Dijkstra\Preparation\Wrapper\v20_norec_Wh.sql
Password for user postgres:
CREATE FUNCTION

C:\Program Files\PostgreSQL\9.5\bin>

```

The next step is to publish the wrapper as a Web Map Service (WMS), for which I use the Internet Map Server (IMS) Geoserver. Therefore, I connect to the administration page (default: user: admin; password: geoserver), create a new workspace (Figure 23), a new store (Figure 24) and a new layer (Figure 25). I configure the following SQL View in this layer:

```

SELECT ST_MakeLine(route.geom) FROM (
    SELECT geom FROM pgr_v20_norec_Wh ('ways', %x1%, %y1%, %x2%, %y2%
) ORDER BY seq) AS route

```

Afterwards, I have to ensure transformation of the coordinate system, since the employed OpenLayers basemap (EPSG SRID: 3857) has a different SRID EPSG than the database (EPSG SRID: 4326) (cf. Section 3.1). Therefore, I change “Declared SRS” accordingly, select “Reproject native to declared” in “SRS handling”, click “Compute from data” and “compute from native bounds” and save the form.

**Figure 23:** A new workspace with the name pgRouting is created.

## Edit Workspace

Edit existing workspace

Name

Namespace URI

The namespace uri associated with this workspace

**Figure 24:** The newly created store is named after the edge costs *v20\_norec\_Wh* applied for this trial. Additionally, connection parameters are defined such as the database type, the host, the port, the database, the schema, the user and the password.

PostGIS  
 PostGIS Database

**Basic Store Info**

Workspace \*

Data Source Name \*

Description

Enabled

**Connection Parameters**

dbtype \*

host \*

port \*

database

schema

user \*

passwd

Namespace \*



**Figure 25:** The SQL View which is used to access data through the created wrapper (cf. Figure 22) in the layer.

**Edit SQL view**  
 Update the definition of the SQL view and its metadata

View Name

SQL statement

```
SELECT ST_MakeLine(route.geom) FROM (
  SELECT geom FROM pgr_v20_norec_Wh ('ways', %x1%,
    %y1%, %x2%, %y2%
  ) ORDER BY seq) AS route
```

SQL view parameters  
 Guess parameters from SQL Add new parameter Remove selected

| Name                        | Default value                  | Validation regular expression            |
|-----------------------------|--------------------------------|------------------------------------------|
| <input type="checkbox"/> y1 | <input type="text" value="0"/> | <input type="text" value="^-?[0-9]+\$"/> |
| <input type="checkbox"/> x1 | <input type="text" value="0"/> | <input type="text" value="^-?[0-9]+\$"/> |
| <input type="checkbox"/> y2 | <input type="text" value="0"/> | <input type="text" value="^-?[0-9]+\$"/> |
| <input type="checkbox"/> x2 | <input type="text" value="0"/> | <input type="text" value="^-?[0-9]+\$"/> |

Escape special SQL characters

Attributes  
 Refresh  Guess geometry type and srid

| Name        | Type       | SRID                              | Identifier               |
|-------------|------------|-----------------------------------|--------------------------|
| st_makeline | LineString | <input type="text" value="4326"/> | <input type="checkbox"/> |

### 5.1.2. Front-End

A front-end document (cf. *index\_v20\_norec\_Wh.html* in Appendix) ensures access for a potential user. A routing request can be performed by simply clicking on two arbitrary points of the map. This basemap derives from the OpenLayers library for which the study area is set as a default view. To be able to perform the routing request, several variables must be predefined. On the one hand, a vector layer is created which is used to display the also newly created start and destination features. On the other hand, a variable *params* calls the required layers from Geoserver. The generation of an additional transform function ensures the mapping in the correct coordinate system.

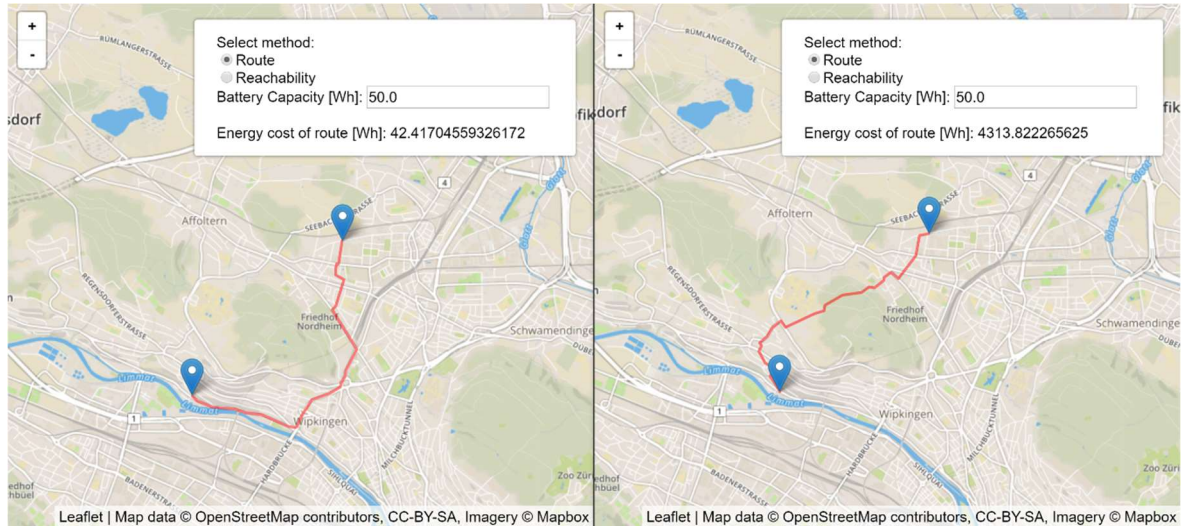
Registering a click event listener determines the starting point through the first click and the destination point through the second click. The transformation of the coordinates of the retrieved WMS image from native to server projection is ensured by the application of the mentioned transform function. Subsequently, data as a WMS image from Geoserver is retrieved, matched and displayed. Finally, an additional function capable to removing all elements from the map in order to start a new request is defined.

The final step would be to display the aggregated costs of each route which is why I would have to transform the WMS into a WFS. Instead, this functionality is integrated in the Bellman-Ford Application in Section 5.2 (responsible for the evaluation in Test Session B in Section 6.2.2). However, the initial evaluation of the model in Test Session A in Section 6.2.1 is carried out using the pgRouting Plugin in QGIS, which provides detailed cost listing of the energy consumption per street segment. Hereby, the Dijkstra algorithm proves the applicability of the calculated edge costs for energy-based routing initially.

Explained in the following Section 5.2, the advantage of the Bellman-Ford Application is more flexibility as the user can quickly change edge costs (compared to the preliminary application presented in this chapter), which were computed for different velocity levels in the static model. In detail, because of the implementation of the Bellman-Ford algorithm, the application can now process negative edge costs and therefore consider the recuperation ability of certain electric bicycles.

## 5.2. Bellman-Ford Application

**Figure 26:** Comparison between an energy-based path (left side) and the shortest path (right side). Parameters taken from Test Session B ( $T_{amb} = 0$ ,  $m = 127$ ,  $v_{DD} = 20$ ) similar to the setting of Figure 21. Obviously, the costs of the shortest route denote the total length [m] of the journey instead of energy cost of the route [Wh].



In contrast to, for example, SACHENBACHER ET AL. 2011, who used a variant of the Bellman-Ford algorithm with the so-called Pallottino strategy (PALLOTTINO 1984), we<sup>6</sup> conducted a straight-forward implementation of the Bellman-Ford algorithm into another prototypical application (Figure 26, Figure 28 and Figure 29). Able to compute arbitrary (i.e. negative) edge weights, its worst time complexity  $O(n^3)$  (compared to Dijkstra) is intended to be compensated by using pre-processed data only in our application, supported by the fast-responding character through its setting. Latter means fast query times that is achieved by the use of the programming language Rust, which claims to have minimum runtimes (RUST n.d.). As the Bellman-Ford algorithm is not part of the library of pgRouting yet, we developed a stand-alone application explained below.

### 5.2.1. Back-End

The applications' source (*src*) consists of *main.rs*, *endpoints.rs*, *spatialpoints.rs* and *graph.rs* (cf. Programming Code Bellman-Ford Application in Appendix). The latter covers the implementation of the Bellman-Ford algorithm and performs the routing. It reads declared nodes and edges from the provided database, creates a new graph also adding and returning an OSM ID as starting point for the subsequent Bellman-Ford query. Following this, *graph.rs* (cf. *Graph.rs* in Appendix) initiates *spatialpoints.rs* (cf. *Spatialpoints.rs* in Appendix) to build an R tree. A so-called R tree is a spatial index which defines squares, to

<sup>6</sup> The second application was developed in collaboration with my supervisor Dominik Bucher.

quickly find and access the closest node, starting from every requested location given longitude and latitude. *Graph.rs* then gets the node IDs from a longitude and latitude, the internal ID from an OSM ID, and the location from an internal ID. The application returns a vector containing longitude and latitude, finally performing a routing request from source to target with the detection of the aggregated costs from each computed graph as *total\_cost*. The implementation of the Bellman-Ford algorithm returns a tuple, containing a vector of predecessors and a vector of distances to the source node.

Additionally, the application computes the reachability of all nodes in the graph, and returns those which are reachable. It returns a vector of vectors, where the arguments are as follows: Longitude, latitude, remaining energy.

Eventually, *main.rs* (cf. *Main.rs* in Appendix) – the main function and entry point to the program – assigns all arguments from the command line and starts the program. The procedure consists of loading the data from *graph.rs*, setting up the router for the web server. In this step, it accesses *endpoint.rs* which is responsible for the communication with the front-end. Here, *endpoint.rs* (cf. *Endpoint.rs* in Appendix) transforms the result of the route and reachability calculation into a GeoJSON string which can be retrieved in the Front-End. Therefore, it computes a route given a start and end latitude and longitude resp. a start and end OSM ID. Part of this is a request that returns all reachable nodes in a vicinity.

### 5.2.2. Front-End

The html-part of the overall web application (cf. *Index.html* in Appendix) defines the two functionalities route and reachability as boxes to select the desired method. To request the latter, the user can claim a certain battery capacity or SOC in Wh. The output plot of the energy costs per route request is placed below and contains a placeholder (“Please compute a route by clicking on the map!”) for instance.

In the following JavaScript code, we set the default view to Zurich City with an appropriate zoom level, added a function to add or remove elements on the map and defined a helper function to convert HSV color ramps to RGB. This time, we choose a mapbox layer and the Leaflet library rather than OpenLayers for the underlying basemap.

Subsequently, we defined what happens, when a user clicks on the map: Either, the user is in routing mode, where the application simply computes and displays routes, or he is in the reachability mode, where a contour plot of reachable nodes is drawn. For the latter, we

determined maximal and minimal capacity, which is required for coloring. Then we created 10 equally spaced breaks to visualize the declining SOC, the farther someone diverges from the origin (from green to red).

Since the application relies on a different field data type of the *longitude* and *latitude* values, contained in the edge file *ways\_vertices\_pgr*. Instead of using the numeric format, the user needs to alter on double precision, so we can start the application.

```
ALTER TABLE ways_vertices_pgr ALTER COLUMN lat TYPE DOUBLE PRECISION;
ALTER TABLE ways_vertices_pgr ALTER COLUMN lon TYPE DOUBLE PRECISION;
```

To run the wrapper (the so-called cargo package, a build automation system of Rust) in the command line, the user need to change the direction into the target folder (`cd C:\...`). The user has to make sure to be in the right directory, as the implementation uses the current directory to look for *index.html*, i.e., under *src/static* (Figure 27).

```
bellman_osm.exe database_user database_password database_name
ways_vert_table_name ways_table_name forward_cost_column back-
ward_cost_column
```

```
Example: bellman_osm_0.1.1.exe postgres ebike ebike ways_vertices_pgr ways
v20_rec_Wh v20_rec_r_Wh
```

**Figure 27:** Invocation of the Bellman-Ford Application in order to start it.

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\sim_h>cd C:\Users\sim_h\OneDrive\Dokumente\1_Uni_Wien\16_Masterarbeit\5_Applikation\Bellman-Ford\
C:\Users\sim_h\OneDrive\Dokumente\1_Uni_Wien\16_Masterarbeit\5_Applikation\Bellman-Ford>bellman_osm_0.1.1.exe postgres e
bike ebike ways_vertices_pgr ways v30_rec_Wh v30_rec_r_Wh
Loading the data
  duration: 0s

Starting Bellman-Ford ...
└─ Starting from 3051, having 17521 nodes.
└─ Bellman iterations: 76
└─ Backtracking from 9976, having 45898 edges. Total cost: 33.64584.
└─ duration: 0s
```

Then, the user needs to open a browser and points it at <http://127.0.0.1:9000>, click on the map, or use the endpoints <http://127.0.0.1:9000/api/route>, <http://127.0.0.1:9000/api/route-using-ids> and <http://127.0.0.1:9000/api/reachability>. These three endpoints accept parameters as follows:

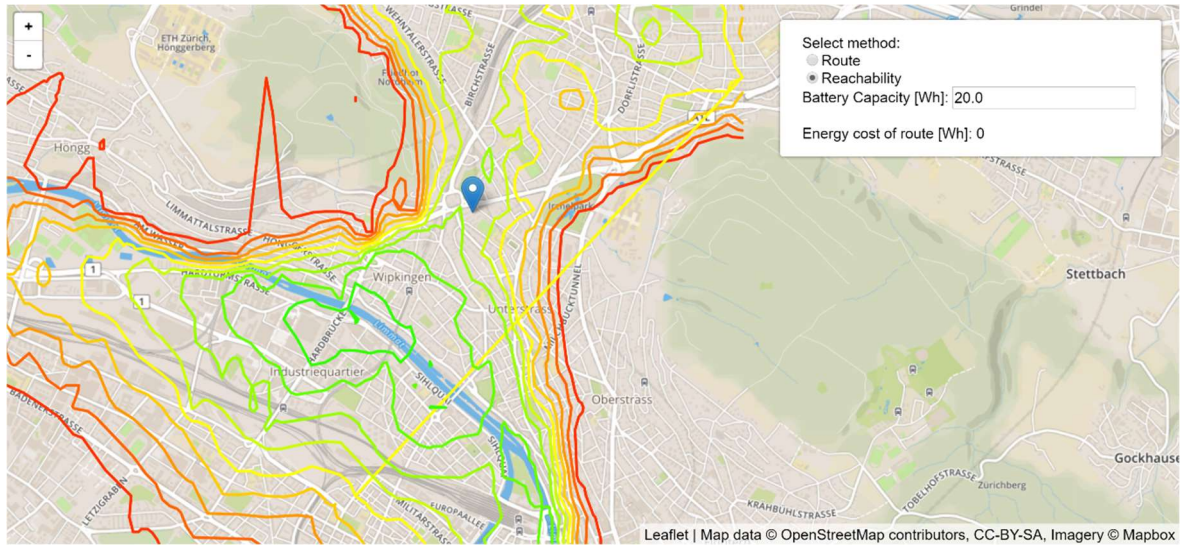
```
http://127.0.0.1:9000/api/route?source-lon=8.54564666748047&source-  
lat=47.407295617526366&target-lon=8.531398773193361&target-  
lat=47.366617842193385
```

```
http://127.0.0.1:9000/api/route-using-ids?source-id=1&target-id=5
```

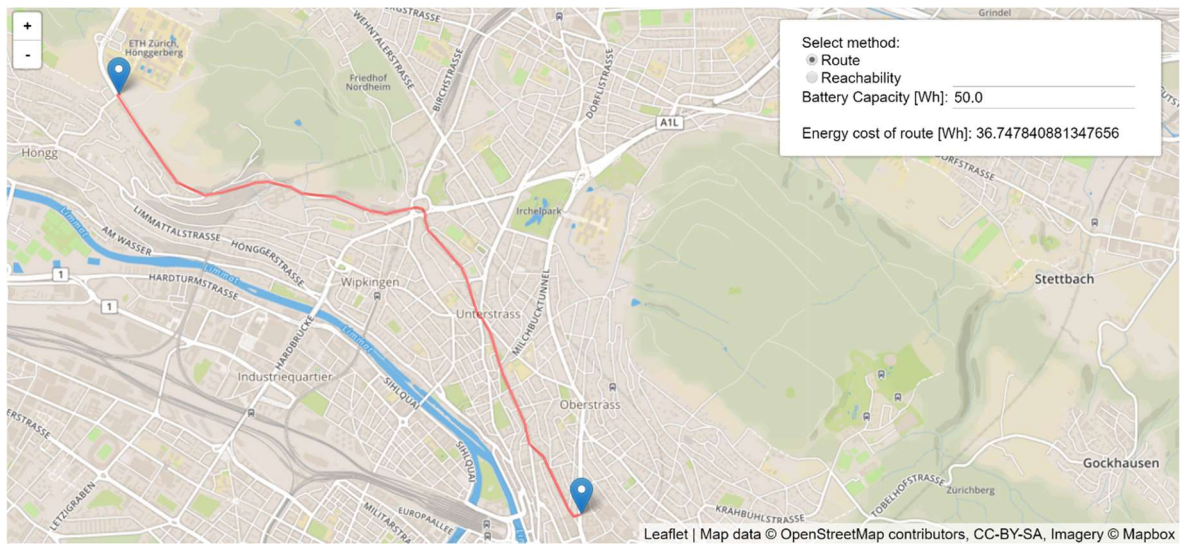
```
http://127.0.0.1:9000/api/reachability?source-lon=8.50170135498047&source-  
lat=47.37429091011091&capacity=50.0
```



**Figure 28:** Bellman-Ford application with a reachability request. The example shows the maximum remaining cruising range with a SOC of 20 Wh from an arbitrary location in Zurich.



**Figure 29:** Bellman-Ford application with a routing request from ETH Hönggerberg to ETH Center. Parameters taken from Test Session A ( $T_{amb} = 0$ ,  $m=123$ ,  $v_{DD} = 20$ ). Note the difference to comparable modelled values from Test Session A in Table 12 in Chapter 7, caused by recuperation in downhill segments.







## 6. Evaluation

The following chapter is structured in two parts. In the first Section 6.1, I provide a full description of the values of the parameters and an explanation for why they were chosen and how they influence the resistance forces. Then, I describe the procedure of the Test Drive Sessions A and B, in Sections 6.2.1 and 6.2.2 respectively.

### 6.1. Parameters

Many of the model parameters presented in this thesis were taken from either literature or the model specifications of the bicycles used. In order to determine the remaining parameter values, I conducted field tests with electric bicycles. The experimental nature of the tests allowed me to tune parameters such as motor efficiency, rolling resistance, and the influence of the rider. An overview of all the parameters considered is presented in Table 5. This is followed by a summary of the parameter values that were applied in different test situations in Table 9 (Section 6.2.1) and Table 10 (Section 6.2.2).

I will begin by explaining the interaction and the amount of influence each of the resistance forces contributed. To determine which specific parameters these forces consist of, a derivation of the quantity of each parameter is provided afterwards.

WILSON 2004 describes the influence of each force on the overall resistance in different use cases. This is summarized by MUETZE & TAN 2007:

- i) At speeds greater than approximately 10 km/h (considered as threshold value) on flat ground, air resistance  $F_d$  commonly has a greater effect than rolling resistance  $F_f$ :

$$F_g = 0; F_d > F_f$$

- ii) At speeds smaller than approximately 10 km/h on flat ground, more power is needed to overcome rolling resistance  $F_f$  than air resistance  $F_d$ :

$$F_g = 0; F_f > F_d$$

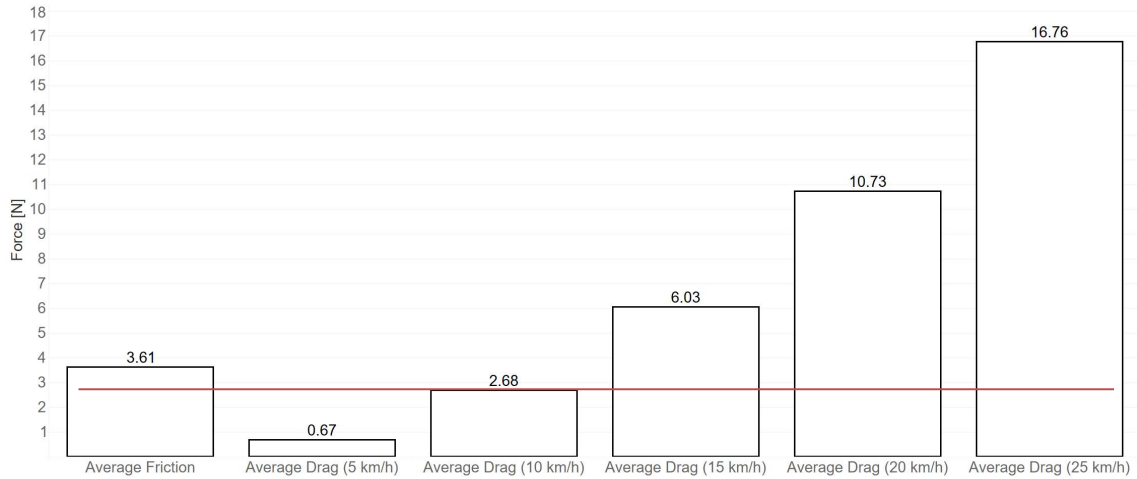
- iii) On hilly terrain where climbing resistance is high, both air resistance  $F_d$  and rolling resistance  $F_f$  are insignificant. If the velocity becomes high, as with other vehicles (e.g. cars), drag could override the climbing resistance  $F_g$ . However this is usually not the case with e-bikes, which are used at lower speeds (cf. Figure 32 and Figure 33):

$$F_g > F_f; F_g > F_d$$

**Table 5:** The model's parameters, classified by symbol and unit. To ensure user-friendly units are considered in the early development process for an application, sometimes conventional units are listed (e.g., [inches] for the diameter of a bicycle wheel, are used for the input values and are subsequently converted during the geo-processing phase in Section 4.2).

| Model Parameter                     | Symbol      | Unit                     |
|-------------------------------------|-------------|--------------------------|
| weight a) driver                    | $m$         | [kg]                     |
| b) e-bike                           |             |                          |
| standard gravity                    | $g$         | [m/s <sup>2</sup> ]      |
| slope angle                         | $\alpha$    | [degree]                 |
| rolling coefficient                 | $c_{rr}$    |                          |
| temperature                         | $T_{amb}$   | [celsius] resp. [kelvin] |
| ambient air pressure                | $P_{amb}$   | [hPa] resp. [Pa]         |
| universal gas constant              | $R_a$       |                          |
| drag coefficient                    | $c_w$       |                          |
| reference area                      | $A$         | [m <sup>2</sup> ]        |
| velocity levels                     | $v_{DD}$    | [km/h] resp. [m/s]       |
| factor human torque                 | $\gamma$    |                          |
| wheel radius (resp. wheel diameter) | $r_w$       | [inches] resp. [m]       |
| wheel perimeter                     | $C$         | [m]                      |
| motor efficiency                    | $\eta_{EM}$ |                          |
| gearbox efficiency                  | $\eta_G$    |                          |
| gradeability                        |             | [degree]                 |
| temperature-dependent factor        | $\beta$     |                          |
| auxiliary components                | $P_i$       | [W]                      |

**Figure 30:** The average friction and drag for the entire study area, using parameters from Test Session A ( $T_{amb} = 0, m = 123$ ). For speeds slightly above 10 km/h, average drag becomes more significant than average friction. The red line illustrates the transition. Note that the average value for friction in this diagram is calculated from all street segments (including flat and steep segments). Therefore, the value is usually slightly higher than 10 km/h.



I seek to reach the same 10km/h threshold value for the entire study area by adjusting the rolling coefficient according to the literature discussed previously. Figure 30 and Figure 31 show that the transition point, where the influence of drag overpowers the influence of friction, lies slightly above the velocity of 10 km/h (average drag  $F_d$  in the study area = 2.68, average rollres  $F_f = 3.61$  for Test Session A; average drag  $F_d$  in the study area = 2.71, average rollres  $F_f = 3.73$  for Test Session B).

**Figure 31:** The average friction and drag for the entire study area, using parameters from Test Session B ( $T_{amb} = 0, m = 127$ ). For speeds slightly above 10 km/h, average drag becomes more significant than average friction. The red line illustrates the transition. Note that the average value for friction in this diagram is calculated from all street segments (including flat and steep segments). Therefore, the value is usually slightly higher than 10 km/h.

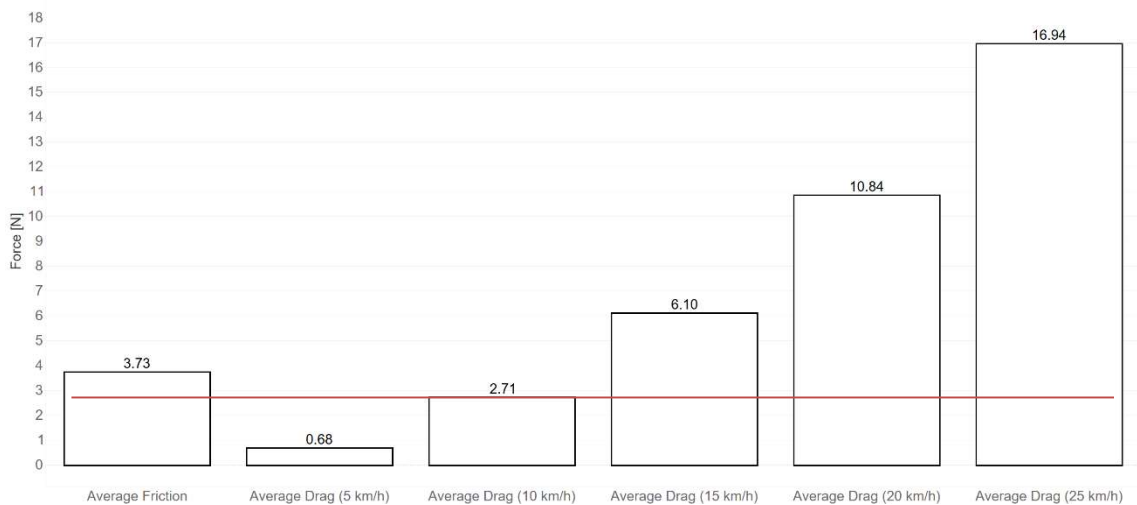
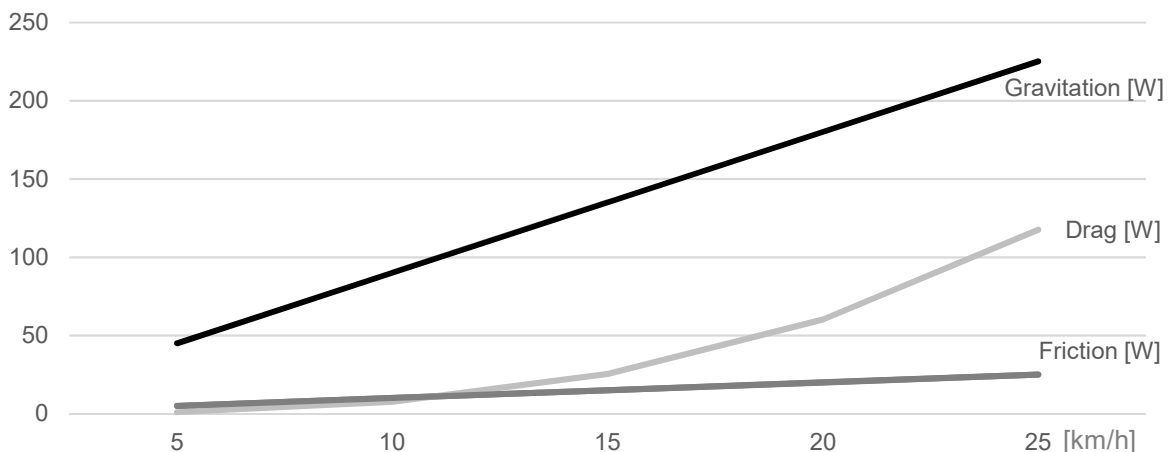
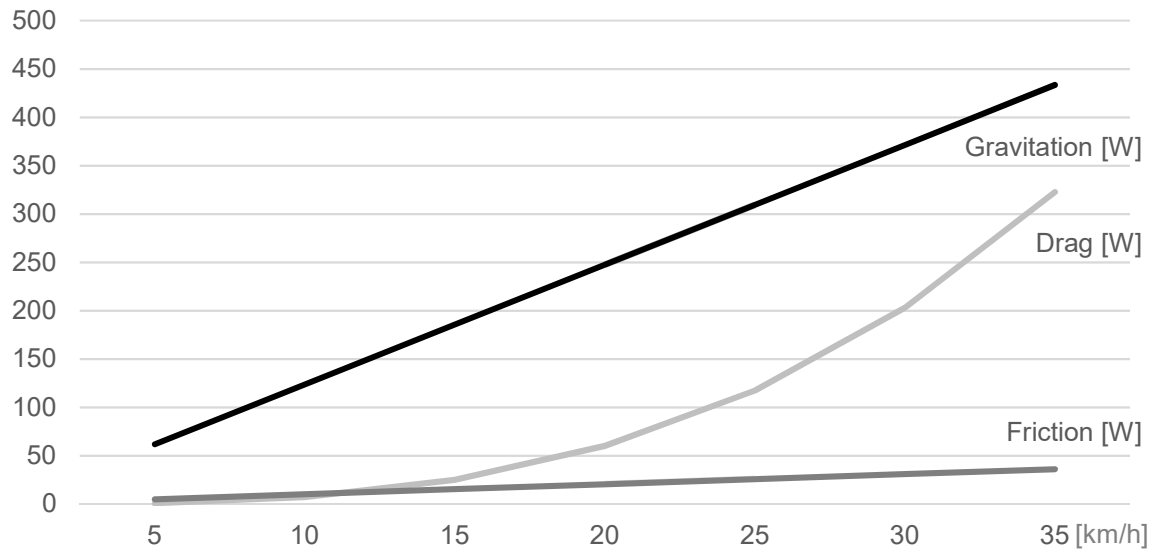


Figure 32 and Figure 33 thus provides a comparison between *all* incorporated powers at certain speeds, using the mean of the calculated values in the study area. While friction and gravitation result in a linear increase, drag has an exponential curve and exceeds gravitation at a certain speed. Considering that the maximum speed of electric bicycles is 25 km/h (Test Session A) resp. 35 or 45 km/h (Test Session B; cf. Table 4 in Section 3.3), gravity is the largest influencing factor. However, the smaller the corresponding values for the mass  $m$  and the slope angle  $\alpha$  become, the lower the speed at which drag exceeds gravitation becomes. BA HUNG ET AL. 2017 findings also show that the mass  $m$  and the slope angle  $\alpha$  are one of the most important parameters affecting the tractive force  $F_T$ . Following this examination of the acting forces, the following paragraphs give a description of the applied values given to the underlying parameters that contribute to the extent of each force.

**Figure 32:** The amount of power [W] to overcome by the electric bicycle, split up by the influence each force compared at different velocities [km/h] as average values for the entire study area. Parameters taken from Test Session A ( $T_{amb} = 0$ ,  $m = 123$ ). Since gravitation is either positive or negative (depending if forward or backward traversal is performed) the employed values constitute averages over absolute values (the mean is composed of the mean of negative and the mean of positive values).



**Figure 33:** The amount of power [W] to overcome by the electric bicycle, split up by the influence each force compared at different velocities [km/h] as average values for the entire study area. Parameters taken from Test Session B ( $T_{amb} = 0$ ,  $m = 127$ ). Since gravitation is either positive or negative (depending if forward or backward traversal is performed) the employed values constitute averages over absolute values (the mean is composed of the mean of negative and the mean of positive values).



The parameters from Table 5 can be divided into four groups. The first group consists of constant values for the *whole test et* either taken from literature or derived from physical constants. The values taken from literature are the rolling coefficient, the drag coefficient, the reference area, the human torque factor and the gearbox efficiency. This is followed by the values derived from physical constants which are the standard gravity constant and the universal gas constant. The second group of parameters are the values from producers' specifications that vary between the *test sessions*. These are the weight of the bike, the wheel diameter, the wheel perimeter, the motor efficiency, the maximal gradeability, the temperature-dependent factor and the energy expenditure of auxiliary components. The third group are the parameters that change *upon each test drive*, namely the weight of the driver, the temperature and the average velocity. Aside from these statically determined parameters, *the model also requires dynamically computed parameters* such as the angle of slope and the ambient air pressure.

*Friction  $F_f$ :* According to WILSON 2004 the rolling coefficient for bicycles lies in between 0.002 and 0.01 on a smooth but hard surface. The exact value can change and is dependent on environment variables such as inflation pressure, wheel diameter, tire construction, surface material or the presence of an intermediate layer between the rolling object and the surface. The presence of heavy snowfall throughout parts of Test Sesion A is an example of such an intermediate layer. Under similar conditions, the value of the rolling coefficient

for electric bicycles has been found to range from 0.003 (ABAGNALE ET AL. 2015a) to 0.004 (LOMONOVA ET AL. 2002) and even as high as 0.014 (MORCHIN 1994). Having neither producer specifications nor an opportunity to quantify the exact value through appropriate experimentation, the rolling coefficient was approximated. As a result, it exists as one of the parameter that can be adjusted and tuned in the model. For this approximation, I took into account the previous findings regarding the influence of friction on the rolling coefficient (as the findings in relation to Figure 30 and Figure 31 discuss).

To calculate *drag*  $F_d$  in the model, I require values for the frontal area  $A$  as well as for the drag coefficient  $c_w$ . The drag coefficient  $c_w$  is a dimensionless parameter that describes the degree of resistance from an object's surface in a fluid. The frontal area  $A$  is defined by the forward-facing surface area of the bicycle and rider that is directly exposed and opposed to the incoming air resistance. Depending on the rider's position (recumbent to an entirely upright position), values found for the frontal area  $A$  in literature vary significantly: 0.33 / 0.34 / 0.36 (WILSON 2004), 0.4 (MORCHIN 1994), 0.5 (LOMONOVA ET AL. 2002), 0.55 (WILSON 2004). Typical values for the drag coefficient  $c_w$  range from 0.5 (LOMONOVA ET AL. 2002) to 0.77 (MUETZE & TAN 2007a) and from 1 (MORCHIN 1994) to 1.15 / 1.2 (WILSON 2004).

However, as the drag coefficient and frontal area dynamically influence the model, they are unable to be quantified in a static setting. Although this is theoretically possible to derive from wind tunnel measurements that involve all e-bike models and human subjects, this is not practically feasible. As a substitute, approximations of  $A = 0.55$  and  $c_w = 1.15$  were taken from a single test setting (upright commuting bike) in WILSON 2004. These values represent the most compatible and standard values. These were then assumed to be constant throughout all test drives (cf. Table 9 and Table 10).

The *motor efficiency* of the electric bicycle employed in Test Session A is about 0.8 according to manufacturer's specifications. However, this value was only treated as an initialization value for the model. It was then treated as a tuning parameter, to adjust the model and the subsequent output values throughout test drives. In contrast, the values of the parameter for Test Session B are predefined by more detailed specifications provided by the manufacturer (Table 6). In terms of the gearbox efficiency  $\eta_G$ , the value from ABAGNALE ET AL. 2015a is assumed to be generally valid as a benchmark. It was not possible to quantify this value for this work.

**Table 6:** The motor efficiency in Test Session B for different torques and velocity levels. The allocation of values to each velocity level (torquev\_Nm) results from speed in revolutions per minute (RPM) for a specific wheel perimeter ( $C = 2.074$ ). For torques greater than zero, the motor is in consumption mode, and for torques smaller than zero, it is in generator mode.

| Speed                |      |         |         |         |         |         |         |         |   |
|----------------------|------|---------|---------|---------|---------|---------|---------|---------|---|
| [rpm]                | 50   | 100     | 150     | 200     | 250     | 300     | 350     | 400     |   |
| [km/h]               | 0    | - 9.33  | - 15.55 | - 21.77 | - 27.99 | - 34.22 | - 40.44 | - 46.66 | - |
| when $C$             | 9.33 | 15.55   | 21.77   | 27.99   | 34.22   | 40.44   | 46.66   | 49.77   |   |
| [km/h] of torquev_Nm | 5    | 10   15 | 20      | 25      | 30      | 35   40 | 45      |         |   |
| Torque [Nm]          |      |         |         |         |         |         |         |         |   |
| below                |      |         |         |         |         |         |         |         |   |
| <-37.5               |      |         | 0.24    | 0.41    | 0.50    | 0.57    | 0.62    | 0.65    |   |
| <-32.5 >-37.5        |      |         | 0.33    | 0.48    | 0.57    | 0.63    | 0.67    | 0.70    |   |
| <-27.5 >-32.5        |      | 0.14    | 0.41    | 0.55    | 0.62    | 0.68    | 0.72    | 0.74    |   |
| <-22.5 >-27.5        |      | 0.31    | 0.53    | 0.64    | 0.69    | 0.74    | 0.77    | 0.79    |   |
| <-17.5 >-22.5        |      | 0.46    | 0.62    | 0.71    | 0.75    | 0.78    | 0.80    | 0.81    |   |
| <-12.5 >-17.5        | 0.17 | 0.56    | 0.68    | 0.75    | 0.79    | 0.81    | 0.83    | 0.84    |   |
| <-7.5 >-12.5         | 0.40 | 0.66    | 0.75    | 0.79    | 0.81    | 0.82    | 0.83    | 0.83    |   |
| <0>-7.5              | 0.53 | 0.69    | 0.72    | 0.77    | 0.77    | 0.76    | 0.75    | 0.74    |   |
| >0<7.5               | 0.62 | 0.72    | 0.75    | 0.78    | 0.77    | 0.77    | 0.78    | 0.75    |   |
| >7.5 <12.5           | 0.58 | 0.70    | 0.76    | 0.78    | 0.80    | 0.82    | 0.82    |         |   |
| >12.5 <17.5          | 0.51 | 0.65    | 0.73    | 0.76    | 0.79    | 0.81    | 0.82    |         |   |
| >17.5 <22.5          | 0.47 | 0.62    | 0.70    | 0.75    | 0.78    | 0.81    | 0.82    |         |   |
| >22.5 <27.5          | 0.41 | 0.57    | 0.65    | 0.71    | 0.74    | 0.77    |         |         |   |
| >27.5 <32.5          | 0.36 | 0.51    | 0.61    | 0.67    | 0.71    | 0.76    |         |         |   |
| >32.5 <37.5          | 0.32 | 0.48    | 0.57    | 0.64    |         |         |         |         |   |
| >37.5                | 0.19 | 0.31    | 0.41    |         |         |         |         |         |   |

Furthermore, I define a constant factor for the human torque  $\gamma$  resp. the corresponding human torque  $T_h$  which represents the overall average per ride resp. street segment. WILSON 2004 estimated tractive power on different velocity levels and for two types of riders (regarding their physical strength and condition). However, I cannot use data related to ordinary bicycles, as the required human power input without motor assistance is disproportionately higher and hence makes it fundamentally incomparable. For the same reason I cannot apply quantities from literature (300 W for an athlete and 75 W for a nonathlete rider) from the work about electric bicycles by MORCHIN 1994, as he derives his values from an article covering non-EVs (cf. GROSS ET AL. 1983). However, ABAGNALE ET AL. 2015a;

ABAGNALE ET AL. 2015b proposes a constant modelled torque at a desired target velocity (2 Nm), which is significantly lower than during the acceleration phase – with a torque value of 10 Nm resp. 15 Nm for different rider models. The lack of information about a related torque value provided by the electric motor prevents me from transforming those values into my own model. SCHNEIDER 2009 measured the amount of pedaling power compared to motor power for a variety of currently available electric bicycles on behalf of the Swiss Federal Office of Energy (SFOE). He undertook an investigation about the comparative power on a certain route by measuring the subject's power curve while riding a racing bicycle on the same route as a reference. The range of human power in this test series is 50 W at approximately 16 km/h, 100 W at approximately 20 km/h, 150 W at approximately 25 km/h, and 200 W at approximately 30 km/h. These values could serve as indication for the magnitude of potential human torque resp. human power values.

Based on the researched values, I aim to derive an original approach for determining the human power as a constant factor  $\gamma$  according to the resistance forces as shown in Section 4.1 (4). Table 7 shows the calculated average values with adapted parameters, which follow the assumption of a potential human force. Without further and more accurate evaluation, it remains an approximate value.

**Table 7:** The average motor power [W] and human power [W] for the study area at varying speeds for Test Session A and B. I adjusted the human torque factor  $\gamma$  on 0.1 for the entire Test Series according to the measurements. Therefore, the values may differ from those found in literature.

| Test Session | Velocity [km/h]<br>( $v_{DD}$ ) | Motor Power [W]<br>( $P_{EM}$ at a certain $v_{DD}$ ) | Human Power [W]<br>( $P_h$ at a certain $v_{DD}$ ) |
|--------------|---------------------------------|-------------------------------------------------------|----------------------------------------------------|
| A            | 15                              | 220                                                   | 24                                                 |
|              | 20                              | 330                                                   | 37                                                 |
|              | 25                              | 477                                                   | 53                                                 |
| B            | 15                              | 223                                                   | 25                                                 |
|              | 20                              | 271                                                   | 30                                                 |
|              | 25                              | 317                                                   | 35                                                 |
|              | 30                              | 418                                                   | 46                                                 |
|              | 35                              | 547                                                   | 61                                                 |

Finally, I need to define the temperature-dependent factor  $\beta$ . The data is provided by the manufacturer of the battery used in Test Session A and the same specification was assumed in Test Session B as the manufacturer did not provide the relevant details. As shown in Table 8, electric charge decreases with decreasing temperatures. As it is possible to assume that energy demand is inversely proportional to the available electric charge (cf. Section 2.1), energy demand is, by extension, also inversely proportional to temperature.



**Table 8:** Available electric charge of the battery used in Test Session A at different temperatures  $T_{amb}$ . I assumed that less available electric charge results in a higher energy demand (inversely proportional). The linear decrease is 0.0047 per degree celsius. The data was provided by the manufacturer.

| Temperature $T_{amb}$ [celsius] | Electric Charge [Ah] | Available Electric Charge [%] |
|---------------------------------|----------------------|-------------------------------|
| 25                              | 10.319               | 100                           |
| 24                              | 10.2706              | 99.530962                     |
| 23                              | 10.2222              | 99.061925                     |
| 22                              | 10.1738              | 98.592887                     |
| 21                              | 10.1254              | 98.123849                     |
| 20                              | 10.077               | 97.654812                     |
| 19                              | 10.0286              | 97.185774                     |
| 18                              | 9.9802               | 96.716736                     |
| 17                              | 9.9318               | 96.247698                     |
| 16                              | 9.8834               | 95.778661                     |
| 15                              | 9.835                | 95.309623                     |
| 14                              | 9.7866               | 94.840585                     |
| 13                              | 9.7382               | 94.371548                     |
| 12                              | 9.6898               | 93.90251                      |
| 11                              | 9.6414               | 93.433472                     |
| 10                              | 9.593                | 92.964435                     |
| 9                               | 9.5446               | 92.495397                     |
| 8                               | 9.4962               | 92.026359                     |
| 7                               | 9.4478               | 91.557321                     |
| 6                               | 9.3994               | 91.088284                     |
| 5                               | 9.351                | 90.619246                     |
| 4                               | 9.3026               | 90.150208                     |
| 3                               | 9.2542               | 89.681171                     |
| 2                               | 9.2058               | 89.212133                     |
| 1                               | 9.1574               | 88.743095                     |
| 0                               | 9.109                | 88.274058                     |
| -1                              | 9.0606               | 87.80502                      |
| -2                              | 9.0122               | 87.335982                     |
| -3                              | 8.9638               | 86.866944                     |
| -4                              | 8.9154               | 86.397907                     |
| -5                              | 8.867                | 85.928869                     |
| -6                              | 8.8186               | 85.459831                     |
| -7                              | 8.7702               | 84.990794                     |
| -8                              | 8.7218               | 84.521756                     |
| -9                              | 8.6734               | 84.052718                     |
| -10                             | 8.625                | 83.583681                     |
| ...                             | ...                  | ...                           |

For simplification purposes, I am neglecting a few influence quantities such as the headwind  $v_{wx}$  which has an impact on energy consumption (LI ET AL. 2016) but can only be incorporated in a real-time application (cf. Section 8.2).

## 6.2. Test Set

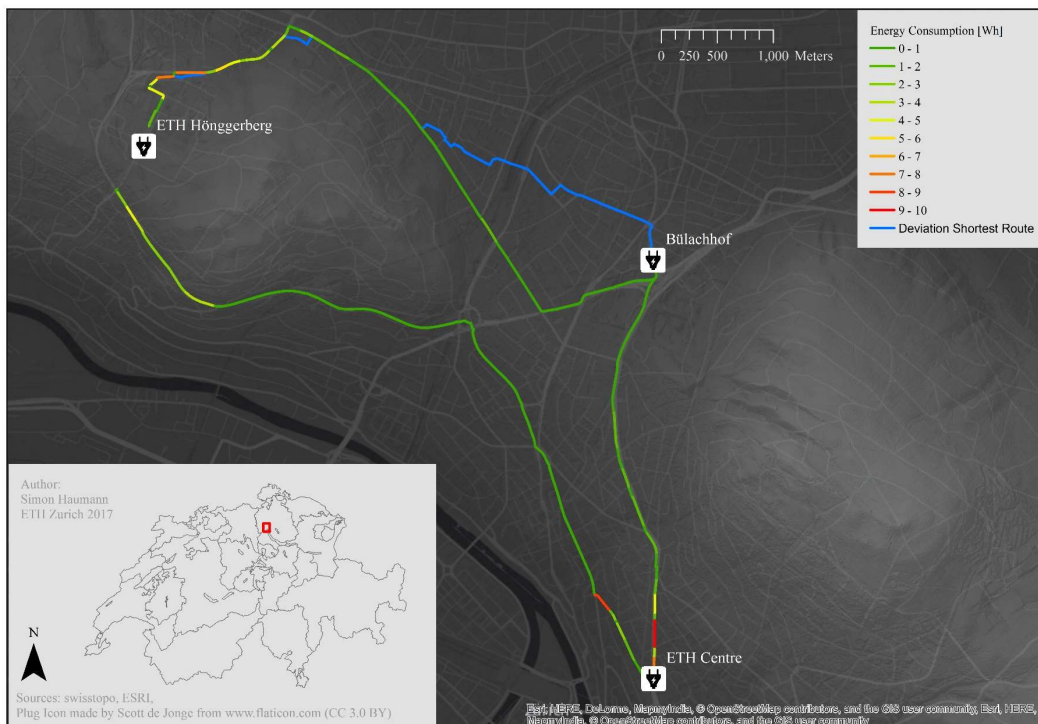
To verify the calculated values and tune the model, I ran repeated test drives on three routes. I obtained those routes by performing a route request between selected origin and target locations using the applications developed in the previous chapter. Figure 35 shows the energy demand of each street segment and how this route deviates from a common shortest route request which uses distance as edge costs. Those routes represent ordinary commuter distances, which are usually rather short compared to commutes via other vehicles such as car or train. The set of routes consist of either uphill stretches: Bülachhof (height *source\_el*: 473 meters a.s.l.) to ETH Hönggerberg (height *source\_el*: 525 meters a.s.l.) and ETH Center (height *source\_el*: 450 meters a.s.l.) to Bülachhof. In contrast, the route from ETH Hönggerberg to ETH Center consists mainly of downhill segments. I tested the energy demand of two different e-bike models on these routes. In Test Session A, an engine without recuperation ability was used (EGO Movement White Knight). The model employed in Test Session B had a recuperation mechanism (Stromer ST 2) (cf. Section 3.3).

The tests are carried out with the adapted input parameters, which take into consideration the temperature during a test ride and the combined weight of the subject and the electric bicycle. The measurement for the average velocity could be taken from either the smartphone applications or the internal torque sensor displayed for each trip shown in Figure 38 in Section 6.2.1. However, the first few test rides revealed the so-called “average speed while moving” by Geo Tracker provided the best results (cf. Figure 38 in Section 6.2.1). As a result of this finding, I match this value with the modelled target velocity of the compiled route rather than calculating the mean from the mentioned set of measurements (the values measured by the smartphone applications and by the internal torque sensor). Several other parameters that I do not measure were assumed from literature or producers’ specifications, as explained in Section 6.1. The tire pressure was held constantly at 2.6 bar. I use the highest power assistance level to exploit full potential of the electric motor in these tests. However, further investigation that integrates different assistance levels into the model remains possible (cf. Section 8.2). Additionally, a GNSS-Tracking performed by the smartphone applications store the driven trajectory (Figure 34). A comparison with the routing request ascertains the correct route was taken by the rider.

Figure 34: GPX-Records of the the test tracks.



Figure 35: Energy consumption per street segment on the test tracks. The Map shows all routes with the modelled energy consumption per street segment (where the parameters to create the route are taken from Test Session A). A comparison between least energy consuming and shortest route (length\_m as edge costs) reveals very few detours for the chosen test set.





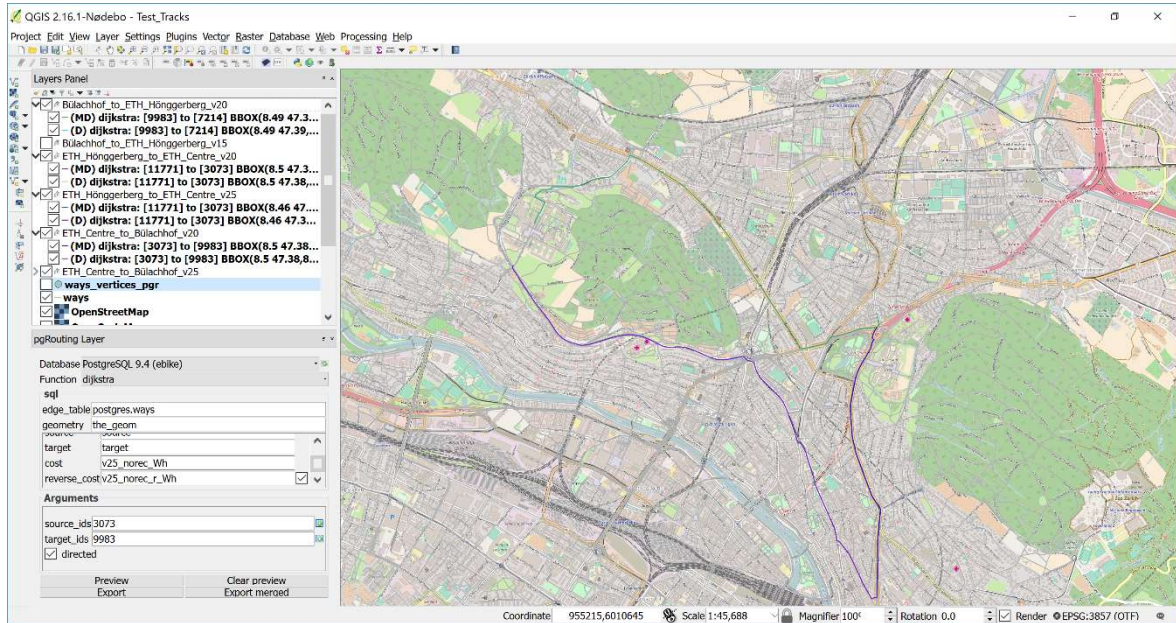
### 6.2.1. Test Session A

**Figure 36:** The Electric Bicycle employed in Test Session A.



To reflect the majority of electric bicycles currently available, I employed a bike without recuperation mechanism for the first series of tests (Figure 36). For each test drive, the route and average speed were tracked and matched with modelled values afterwards. Through a retrospective approach, I determined energy consumption during recharge mode using an energy cost meter at the respective target location. Naturally, I started always at a full SOC. This procedure is illustrated by plugs in Figure 35 in Section 6.2.

**Figure 37:** The pgRoutingLayer QGIS-Plugin. Each shortest path algorithm requires input parameters including an edge table, geometry, id, target, source, cost and reverse cost. Source and target ID's serve as arguments for each routing request. The application returns a graph with aggregated costs (i.e., for the three test tracks shown here). Upon request, the original coordinate system 4326 must be set and the full extent of the desired route must fit into the currently shown display (inside a specified bounding box).



The plugin pgRoutingLayer in QGIS enables me to retrieve routes quickly and adjust parameters accordingly. It allows even faster adaptations than the developed Bellman-Ford-Application. This explains why it was hence used for the initial Test Session A where particular parameters required tuning through a great number of iterative recalculations of the model. Using data recorded during the test drives, I tuned the motor efficiency and the human torque factor which both influence the edge costs declared in Section 4.2, to increase the model's fit. Through iterative processing of the model with altered inputs, I could minimize the deviation between measured and modelled values. Figure 37 shows an exemplary routing request.

**Figure 38:** A screenshot from “Geo Tracker”, one of the smartphone applications available (*first picture*). It displays the average velocity of the trip. *The second picture* shows the average velocity measured by the internal torque sensor on the display of the e-bike itself. *The third picture* shows separate energy cost meter. The display plots i.a. the measured amount of energy during recharge [kWh]. Since I started each Test Drive with a fully charged battery, the measured value in recharge mode at the respective target location corresponds to the energy consumption of the driven route.





**Table 9:** Model Parameters for the model White Knight from EGO Movement.

| Model Parameter [unit]                                 | Value                           | Reference                                 |
|--------------------------------------------------------|---------------------------------|-------------------------------------------|
| weight $m$                                             | driver [kg]                     | Table 12                                  |
|                                                        | e-bike [kg]                     | 23                                        |
|                                                        |                                 | producers' specs                          |
| standard gravity $g$ [m/s <sup>2</sup> ]               | 9.806                           |                                           |
| slope angle $\alpha$ [degree]                          | model-dependent                 |                                           |
| rolling coefficient $c_{rr}$                           | 0.003                           | WILSON 2004; adjusted empirically         |
| temperature $T_{amb}$ [celsius]                        | Table 12                        |                                           |
| ambient air pressure $P_{amb}$ [hPa]                   | model-dependent                 | International height formula              |
| universal gas constant $R_a$                           | 287.058                         |                                           |
| drag coefficient $c_w$                                 | 0.55                            | WILSON 2004                               |
| reference area $A$ [m <sup>2</sup> ]                   | 1.15                            | WILSON 2004                               |
| velocity levels $v_{DD}$ [km/h]                        | 5, 10, 15, 20, 25<br>(Table 12) | model iteration                           |
| factor human torque $\gamma$                           | 0.1                             | empirical approximation                   |
| wheel diameter (resp.<br>wheel radius $r_w$ ) [inches] | 28                              | producers' specs                          |
| motor efficiency $\eta_{EM}$                           | 0.45                            | producers' specs; adjusted empirically    |
| gearbox efficiency $\eta_G$                            | 0.98                            | ABAGNALE ET AL. 2015a                     |
| gradeability [degree]                                  | 15                              | producers' specs; adjusted empirically    |
| temperature-dependent<br>factor $\beta$                | 0.0047                          | Producers' specs; empirical approximation |
| auxiliary components $P_i$ [W]                         | 1.08                            | producers' specs (display only)           |



### 6.2.2. Test Session B

**Figure 39:** The Electric Bicycle employed in Test Session B.



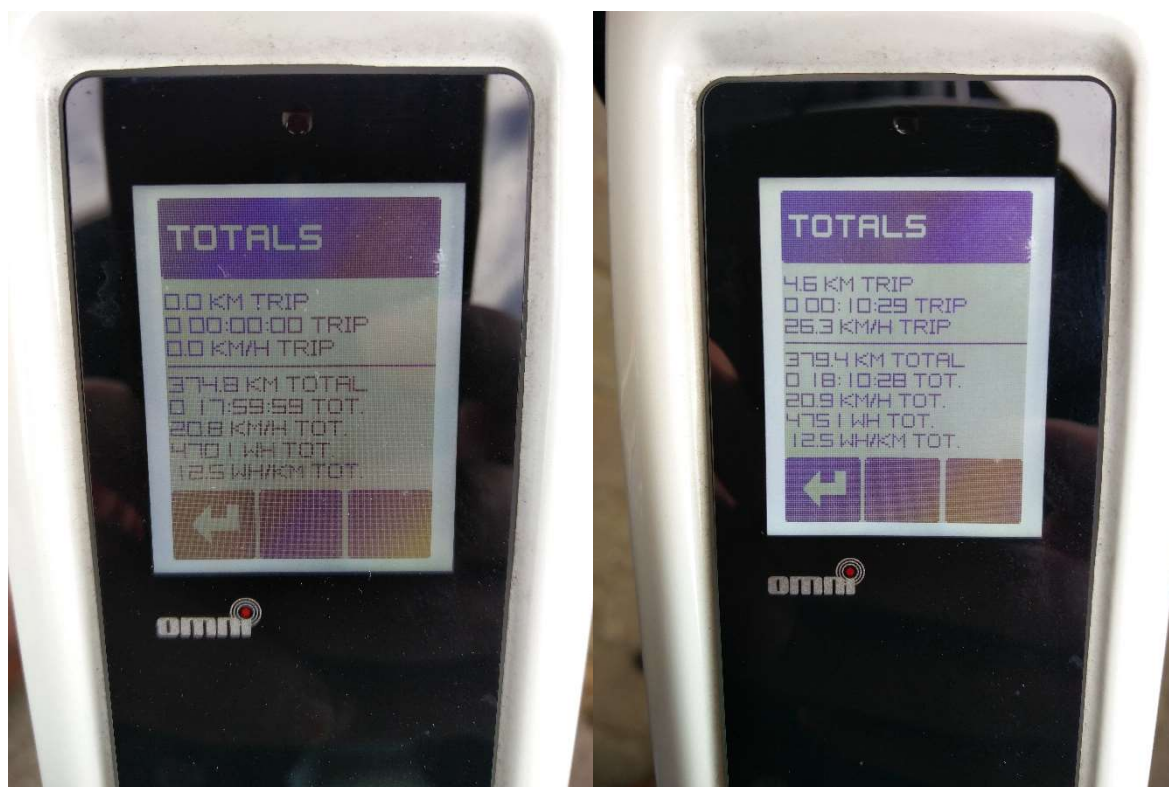
The bicycle employed in Test Session B (Figure 39) differs from the one in Test Session A with regards to three characteristics. Firstly, it has an integrated recuperation mechanism, so the motor is capable of generating power instead of just consuming it. Secondly, the bike can accelerate to velocities up to 45 km/h (which is curbed to 35 km/h for our Tests; cf. Table 4 in Section 3.3). Thirdly, the battery was not fully loaded at the beginning of the trip since recuperation is not possible at a full SOC.

The motor efficiency determined for this test session is shown in Section 4.2.6 and 6.1. Even though one could easily improve the model by tuning factors such as human torque, I refrained from doing so in order to compare this untuned test session to the tuned Test Session A in Chapter 7. The deviations between the two tests allow me to illustrate the necessity of tuning the model.

Initially I aimed to employ a consistent method for the entire test set, however this was not possible. After the first trial test drives, I recognized irregularities in the output values from the outlined procedure (retrospective determination of energy consumption in the recharge

mode with an energy meter). This was most probably caused by an incompatible charging transformer and battery. However, even if the transformer and battery were compatible, a consistent procedure would not be possible due to fundamental differences in the e-bike models used in each test session. As mentioned previously, an engine without recuperation cannot recover energy when starting at a full SOC (which was the starting state for the batteries used in the test drives). In consequence, the entire remaining test session uses an alternative method to plot energy consumption. This method allows the e-bike model's continuous measurements of the aggregated total energy expenditure to be easily read (Figure 40). Despite having such a simple measurement procedure, there are restrictions for this bike model as well. The displayed value only changes every time the SOC diminishes by one percent, i.e. at the interval of approximately 8.14 Wh (7 Wh throughout measurements) out of 814 Wh SOC when fully charged. As a result, inaccuracies cannot be ruled out – especially on short routes.

**Figure 40:** The display of the second e-bike model shows data logging of the electric bicycle such as the ongoing aggregation of total energy expenditure. Note the difference between total consumption at the beginning (4701 Wh) and the end (4751 Wh) of each trip – resulting in an energy expenditure of 50 Wh for this trip.



As highlighted in the subsequent chapter 7, I aim to compare measured values with modelled predictions. As the aim of the test drives is to improve the accuracy of the energy consumption prediction, the same predefined route had to be used for each test drive in

order to be comparable across test sessions. However different cost values (for parameters such as recuperation and velocity) may be calculated by the application. At times, this is caused by altered parameters but this is not necessarily the case either. The effect of these altered cost values is that in some cases, the compiled route outputted by the application after the initial routing request and may result in a route that deviates from the predefined route. I calculate the energy consumption for the test routes by splitting up the request into multiple parts and adding them together.

**Table 10:** Model Parameters for the model ST2 from Stromer.

| Model Parameter [unit]                              | Value                                | Reference                                                                                                   |
|-----------------------------------------------------|--------------------------------------|-------------------------------------------------------------------------------------------------------------|
| weight $m$                                          | driver [kg]                          | Table 12                                                                                                    |
|                                                     | e-bike [kg]                          | 27                                                                                                          |
|                                                     |                                      | producers' specs                                                                                            |
| standard gravity $g$ [m/s <sup>2</sup> ]            | 9.806                                |                                                                                                             |
| slope angle $\alpha$ [degree]                       | model-dependent                      |                                                                                                             |
| rolling coefficient $c_{rr}$                        | 0.003                                | WILSON 2004; adjusted empirically                                                                           |
| temperature $T_{amb}$ [celsius]                     | Table 12                             |                                                                                                             |
| ambient air pressure $P_{amb}$ [hPa]                | model-dependent                      | International height formula                                                                                |
| universal gas constant $R_a$                        | 287.058                              |                                                                                                             |
| drag coefficient $c_w$                              | 0.55                                 | WILSON 2004                                                                                                 |
| reference area $A$ [m <sup>2</sup> ]                | 1.15                                 | WILSON 2004                                                                                                 |
| velocity levels $v_{DD}$ [km/h]                     | 5, 10, 15, 20, 25, 30, 35 (Table 12) | model iteration                                                                                             |
| factor human torque $\gamma$                        | 0.1                                  | empirical approximation                                                                                     |
| wheel diameter (resp. wheel radius $r_w$ ) [inches] | 26                                   | producers' specs                                                                                            |
| wheel perimeter $C$ [m]                             | 2.074                                | producers' specs                                                                                            |
| motor efficiency $\eta_{EM}$                        | model-dependent                      | producers' specs                                                                                            |
| gearbox efficiency $\eta_G$                         | 0.98                                 | ABAGNALE ET AL. 2015a                                                                                       |
| gradeability [degree]                               | 15                                   | producers' specs; adjusted empirically                                                                      |
| temperature-dependent factor $\beta$                | 0.0047                               | producers' specs; empirical approximation                                                                   |
| auxiliary components $P_i$ [W]                      | 8.37                                 | producers' specs (headlight: 4.2; backlight: 0.15; daytime running lights: 2, display: 1.02; controller: 1) |





## 7. Results

This chapter presents the results of the conducted test set. I will begin by examining the precision of the measured values from the repeated test drives. Following this discussion, I will compare the measured values to the predicted values outputted by the model to ascertain the accuracy of the results. Table 12 presents this comparison between the data predicted and observed, for both Test Sessions. The round trips presented in Table 11 serve as control measurements only.

The comparison between similar measurements from different test rides (i.e., regarding weight, velocity and temperature) reveals that the measurement results are intrinsically consistent with minor deviations (cf. Table 12). However, several results expose a bias, especially for Test Session B. For example, in the third test ride from ETH Center to Bülachhof in Test Session B, the journey had an average velocity that was higher than previous rides. This means that it should have had a higher energy consumption than previous rides too. Surprisingly, the observed energy consumption was in fact, less than half as high. Other examples are the last two test rides from Bülachhof to ETH Center in Test Session B, or the last two test rides from ETH Hönggerberg to ETH Center in Test Session A, where similar biased observations were made.

Inaccuracies might occur because of the chosen measurement methods (cf. Section 6.2). The actual power consumption during discharge and the read power consumption during charge in Test Session A might not be entirely congruent. Under some circumstances, this is caused by neglecting the interaction between amperage (i.e., constant current) and voltage (which depends on the respective SOC of the Battery). Human error is also likely to have affected the recorded results. With the bike model used in this test session, a recording could only be taken at the exact time point when recharging was complete. A delay could cause the result to be affected by trickle charging, which could result in an overestimated energy consumption reading. While the method chosen for Test Session B is a live measurement taken during discharge, inaccuracies may still arise from the fact that total energy consumption is only displayed in intervals (as explained in Section 6.2.2). If nothing else, the approximation of the velocity might lead to an over- or underestimation of the actual energy consumption (e.g., a measured average velocity of 17.51 km/h is assigned to 20 km/h i.e.  $v_{20\_rec\_Wh}$  for example, while the actual energy consumption in this case lies in between the value of  $v_{15\_rec\_Wh}$  and  $v_{20\_rec\_Wh}$ ).

Due to the minor quantity of measurements, I abstain from quantifying this phenomenon statistically. Moreover, the multivariate character of the test set (different input parameters for each test drive) would aggravate this attempt. Increased accuracy and precision would be an important requirement for further refinements of the model and could be achieved through a larger test set or even a different and/or more accurate measurement method.

**Table 11:** Control Measurements Table. The energy consumption is recorded as a total for the three test routes. The overall length might seem to be slightly longer but this is simply due to the gap between two routes at ETH Höggerberg where the bike was walked from the end point of one route to the start of the next. The gpx-records does not include this gap (cf. Figure 34 in Section 6.2). Note that the energy consumption is lower when there is a subject with a lower weight.

| Origin - Target | Test Session | Weight Driver [kg] | Temperature [Celsius] | Ø Velocity [km/h] | Energy Consumption [Wh] |          |
|-----------------|--------------|--------------------|-----------------------|-------------------|-------------------------|----------|
|                 |              |                    |                       |                   | Measured                | Modelled |
| Bülachhof       | A            | 100                | -1                    | 20                | 170                     | 176      |
| ETH Höggerberg  |              | 55                 | 0                     | 20                | 111                     | 135      |
| - ETH Center -  | B            | 100                | 13                    | 20                | 149                     | 96       |
| Bülachhof       |              | 55                 | 20                    | 20                | 56                      | 71       |

**Table 12:** The entire test set containing measured values from Test Session A and B (cf. Appendix for detailed listing of each test drive). Certain parameters are dynamic parameters that change with test drive performed. They include the modelled value, weight of the driver, temperature and velocity are dynamic parameters changing at each test drive performed. Others remain constant throughout a Test Session (cf. Table 9 and Table 10).

| Origin - Target                    | Test Session | Weight Driver [kg] | Temperature [Celsius] | Ø Velocity [km/h] | Energy Consumption [Wh] |          |
|------------------------------------|--------------|--------------------|-----------------------|-------------------|-------------------------|----------|
|                                    |              |                    |                       |                   | Measured                | Modelled |
| Bülachhof -<br>ETH Hönggerberg     | A            | 100                | 3                     | 15                | 71                      | 74       |
|                                    |              |                    | 3                     | 20                | 85                      | 86       |
|                                    |              |                    | 3                     | 20                | 85                      | 86       |
|                                    |              |                    | 0                     | 20                | 89                      | 87       |
|                                    |              |                    | -4                    | 20                | 83                      | 89       |
|                                    | B            | 5                  | 25                    | 85                | 65                      |          |
|                                    |              | 10                 | 20                    | 64                | 57                      |          |
|                                    |              | 13                 | 20                    | 50                | 56                      |          |
|                                    |              | 7                  | 25                    | 70                | 64                      |          |
|                                    |              | 7                  | 25                    | 63                | 64                      |          |
| ETH Hönggerberg<br>-<br>ETH Center | A            | 100                | 3                     | 20                | 49                      | 47       |
|                                    |              |                    | 3                     | 20                | 48                      | 47       |
|                                    |              |                    | 3                     | 20                | 54                      | 47       |
|                                    |              |                    | -4                    | 20                | 53                      | 48       |
|                                    |              |                    | -4                    | 25                | 42                      | 60       |
|                                    | B            | 10                 | 20                    | 42                | 17                      |          |
|                                    |              | 5                  | 30                    | 35                | 34                      |          |
|                                    |              | 10                 | 30                    | 35                | 32                      |          |
|                                    |              | 10                 | 25                    | 35                | 23                      |          |
|                                    |              | 10                 | 25                    | 35                | 23                      |          |
| ETH Center -<br>Bülachhof          | A            | 100                | 3                     | 20                | 40                      | 40       |
|                                    |              |                    | 3                     | 20                | 41                      | 40       |
|                                    |              |                    | 0                     | 20                | 42                      | 41       |
|                                    |              |                    | -4                    | 25                | 52                      | 50       |
|                                    |              |                    | -4                    | 20                | 38                      | 42       |
|                                    | B            | 10                 | 20                    | 35                | 25                      |          |
|                                    |              | 10                 | 20                    | 28                | 25                      |          |
|                                    |              | 10                 | 25                    | 14                | 28                      |          |
|                                    |              | 3                  | 30                    | 42                | 35                      |          |
|                                    |              | 10                 | 30                    | 28                | 34                      |          |

Out of five repeated test drives for each of the three routes, the average deviation between measured and modelled values is around 7 % for Test Session A and 25 % for Test Session B (i.e., higher or lower than measured). Hence, the overall deviation for the entire test set lies at around 16 %.

Besides the mentioned measurement inaccuracies, these deviations can also partly be explained by the omission of acceleration/deceleration resistance. A possible implementation is discussed in Section 8.2. Moreover, despite more accurate motor efficiency values used in Test Session B, the deviation between modelled and measured values is distinctly higher. As Test Session B was not tuned, this points out the importance of fine tuning the defined tuning parameters in the given model. In this context, the human torque factor assumed in the model might differ from the true value. As long as human torque is not measured directly in real-time (which is not feasible in this work), the actual influence of it remains an approximation. Finally, the implementation of a compensation factor that takes into account the capacity loss due to higher energy consumption in low temperatures could not be examined empirically in this work because of similar temperatures throughout the Test Sessions.

For initial evidence of the model's accuracy and precision, I adduce test results conducted by EMPA (with same bike from Test Session B) in the course of the program "commercial usage of EV in companies" (GAUCH 2017).

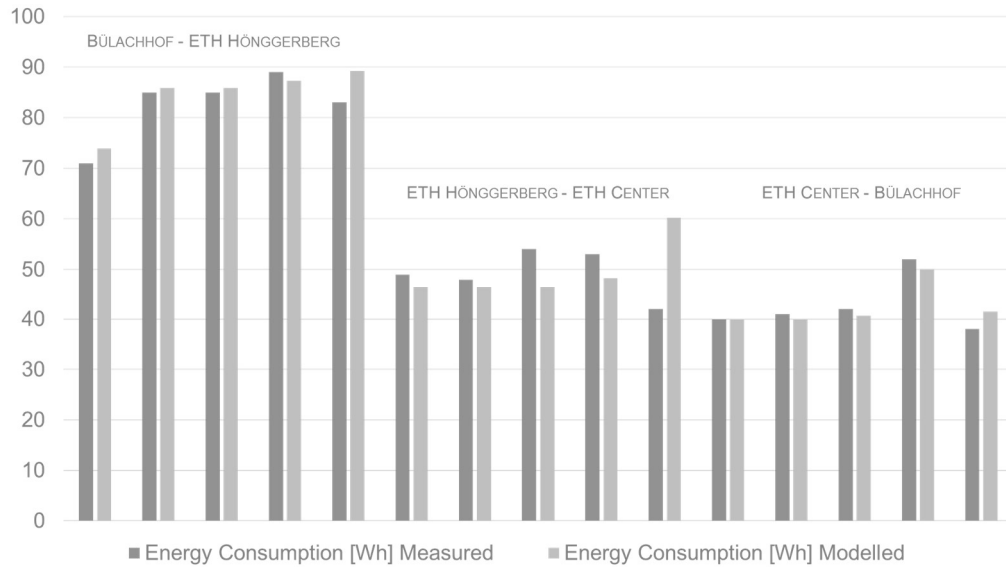
The measured energy consumption of 1120 Wh per 100 kilometers by EMPA covers the projected measured energy consumption of 1062 Wh per 100 kilometers resp. 983 Wh per 100 kilometers for the modelled values<sup>7</sup> in this thesis. This comparison provides only a first step towards validating the model since parameters such as the average velocity or the topography might differ between the test set carried out by EMPA and the one presented in this thesis.

<sup>7</sup> Average of the five test drives per route in Test Session B (calculation  $x[\text{Wh}] * 100[\text{km}] / y[\text{km}]$ , where  $x$  is the mean energy consumption for a route (cf. Table 12) and  $y$  is the length of the route):

|                               | Measured Wh/100km:      | Modelled Wh/100km:      |
|-------------------------------|-------------------------|-------------------------|
| Bülachhof – ETH Hönggerberg:  | $66 * 100 / 4.5 = 1467$ | $61 * 100 / 4.5 = 1356$ |
| ETH Hönggerberg – ETH Center: | $36 * 100 / 5 = 720$    | $26 * 100 / 5 = 520$    |
| ETH Center – Bülachhof:       | $27 * 100 / 2.7 = 1000$ | $29 * 100 / 2.7 = 1074$ |
| Total Average:                | 1062                    | 983                     |

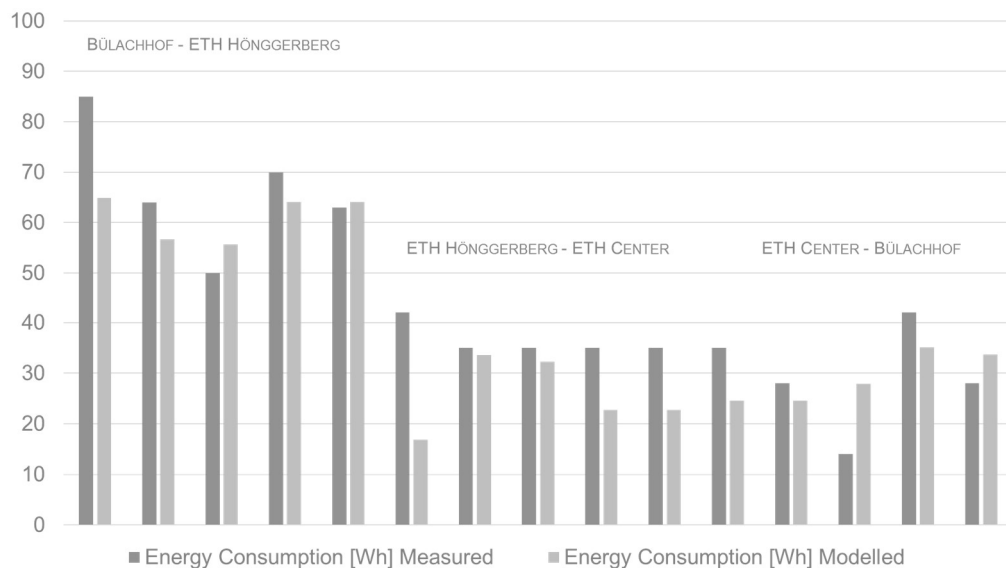


**Figure 41:** The diagram compares measured and modelled energy consumption [Wh] from Test Session A (without recuperation) (cf. Table 12).



It is also important to consider the effect of recuperation on energy consumption. As mentioned in Section 6.2, the track from ETH Hönggerberg to ETH Center has the most downhill segments, with a total descent of 75 meters in altitude. This makes it the most appropriate route to establish the difference between a ride with recuperation and without recuperation. In comparing Test Session A where recuperation is absent and Test Session B and where it is present (cf. Figure 41 and Figure 42 respectively), it becomes clear that both the modelled values and the actual energy consumption (measured values) are significantly lower with recuperation.

**Figure 42:** The diagram compares measured and modelled energy consumption [Wh] from Test Session B (with recuperation) (cf. Table 12).





## 8. Discussion

In Section 8.1 of this final chapter, I will review all proposed research questions and provide a conclusion to the conducted work in Section 8.2. This will be followed by an outlook on what further research could be accomplished. Also, this final Section considers optimizations and refinements of the model, outlines the feasibility of automation and discusses possible further development of end-user applications.

### 8.1. Findings

All initially stated research questions were fully investigated. A new energy model developed from a variety of existing ones was successfully implemented into a programmatic approach in Chapter 4. Additionally, I could also determine all parameters that this model required from either existing sources or from empirical data gathered, was able to successfully feed data into the framework and discuss the related peculiarities of electric bicycles, as shown in Section 6.1 (i). After initial exploratory analysis using an existing routing development tool for e-bikes without recuperation in Chapter 5 (ii), it was determined that the process was feasible and that the calculated edge weights could be used. This allowed me to further the investigation and construct an independent routing approach capable of processing negative edge weights. Finally, Chapter 6, 7 and 8 analyses the advantages of my approach in detail (iii). I could develop an extension to common routing models, that allows energy efficiency to be taken into account alongside existing routing algorithms that assess minimum distance and time. Furthermore, the consideration of energy cost allows for improved reachability.

### 8.2. Conclusion, Application Areas and Outlook

I have presented a static e-bike model which includes an automated processing pipeline for route graph building and a host application. After the application completes a necessary evaluation phase to validate and optimize the model for a given type of electric bicycle, it can be used for route planning, navigation systems, reachability analyses, or even urban planning. An example for urban planning would be to use the framework as criteria for the designation of new bicycle routes, as one might be willing to ride a slightly longer route to avoid steep hills, regardless of whether you are riding an e-bike or an ordinary bicycle (cf. STORANDT 2012; as shown in my thesis, slope has, at a certain point, the biggest influence on the calculation of edge weight in the presented model). Moreover, it is possible to determine locations for new e-bike stations in station-based bike sharing system or to ensure

optimal allocation of e-bikes in free-floating bike sharing systems. In particular, applications where fast querying on a personalized graph is necessary benefit from the approach.

For further research, it will be necessary to determine parameters more accurately and evaluate the model's overall applicability. The modeling of human power input and temperature influence in particular, will need further empirical validation. Objectives for additional research are automation, optimization, and refinement of the model. Refinement could be achieved through the inclusion of traffic (e.g., through OPENTRAFFIC 2016) and street type data, parameters which determine acceleration/deceleration phases (e.g., traffic lights or pedaling frequency and strength), a more detailed SOC model or a more detailed implementation of weather conditions. Additionally, it could be improved by incorporating any or all of these elements in a live navigation application. Possible implementations are discussed in this Section.

A significant refinement of the model would be to implement the omitted acceleration resistance through inclusion of narrow curves and intersections. Intersections have a particularly substantial impact as one might assume that an electric bicycle stops at 50 % of all intersections tagged as having traffic lights through OSM classes. This would almost certainly increase the overall accuracy. Simultaneously, it increases the accuracy of parameters that aim to compensate this effect so far and would result i.a. in a higher factor for the human power input.

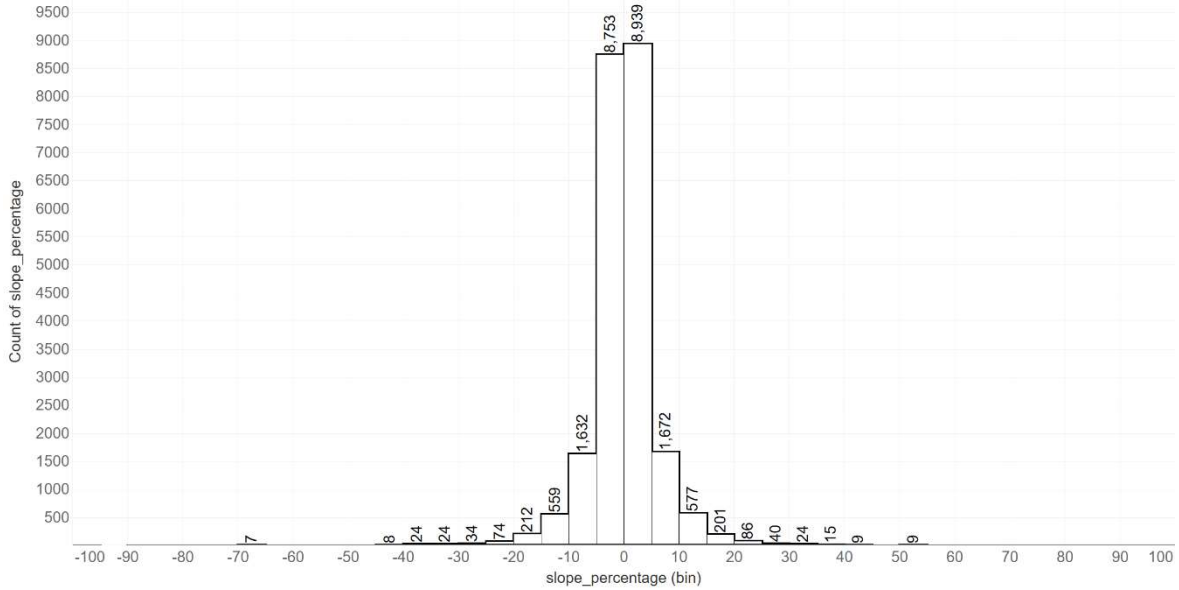
Another important refinement is improving the determination method for the human power input. This generally needs to be evaluated in greater detail using suitable methods. One such example of an improvement would be a test ride could be conducted with motor assistance switched off. Measuring the power provided by the driver, the value could be compared to the result when motor assistance is switched on (while all other parameters remain constant). Repeating this procedure, the accuracy of the human torque factor could be increased by adjusting it. Moreover, the effect of power assistance levels could be incorporated. For example, as the bike employed in Test Session A has five power assistance levels, the motor power would be reduced (20 % of the calculated value for the first power assistance level, 40 % for the second, etc.) and the factor for human power input increased accordingly. Undoubtedly, further evaluation of the extent of the influence of these parameters would further improve the model.

Due to the static nature of the model, the behavior of the rider (in particular in terms of their interaction with the bicycle) is not taken into consideration during processing. This can result

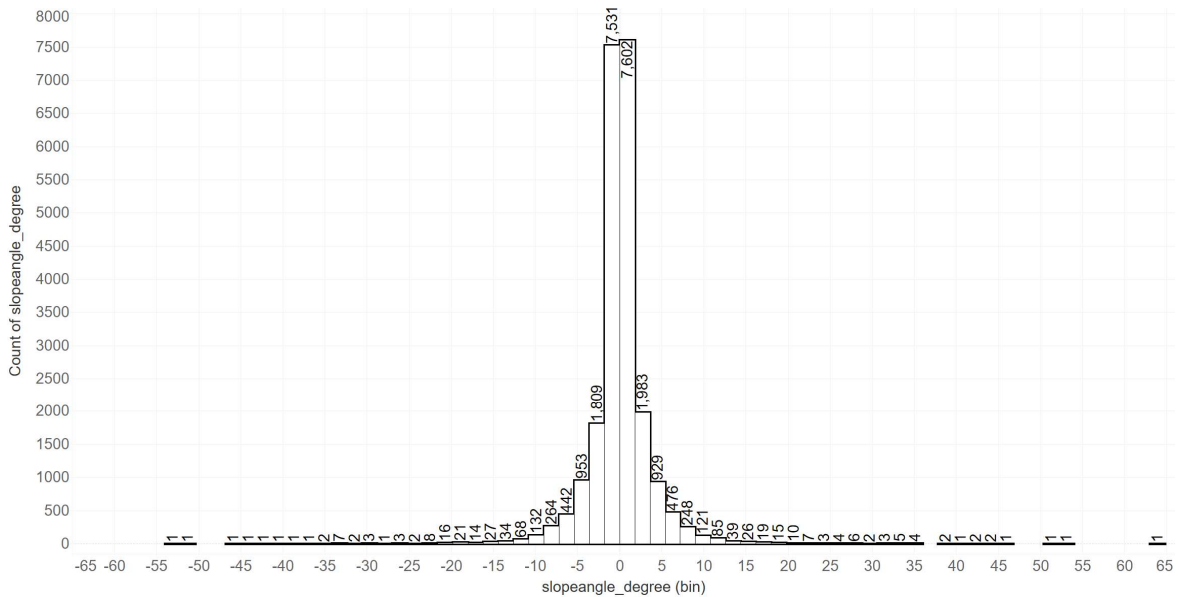
in inaccuracies in the model that are impossible to rectify. When driving downhill, one might not use the recuperation mechanism as extensively as modelled (a common example of this is when riders decide not to brake and thus exceed the modelled target velocity). On uphill segments on the contrary, the rider will not generally be able to maintain the modelled target velocity (this is further explained in the next paragraph). HOCH 2015, for example, proposes a theoretical black-box driver model that takes historic data into account and hence attempts to make the driver's behavior predictable. Together with a white-box EV model (comparable with Section 4.1), it results in a so-called "grey-box" EV consumption model that facilitates better prediction of the potential energy expenditure.

Maximum rated output of the electric bicycles (cf. Table 4 in Section 3.3) is not included as a parameter in the proposed model due to the static nature of the model. However, an implementation could be achieved by restricting the maximum rated output the engine is capable of delivering. Assuming there is a steep slope segment that requires 1000 W of power to propel the bicycle forward and that a rated power of 360 W can be provided by the electric motor, the remaining 640 W must therefore be supplied by the rider. However, this is simply not plausible. This discrepancy results from an inaccuracy in the underlying model. Primarily that by using a static velocity for the whole route graph disregards the fact that the speed of an electric bicycle mostly diminishes on higher slopes. An applicable approach would be to reduce the static velocity at a certain slope gradient. Switching automatically into a lower velocity level class (e.g., v20\_rec\_Wh into v15\_rec\_Wh) would ultimately result in a lower modelled value for the power required. To ensure the validity of this procedure, additional empirical validation would be needed.

**Figure 43:** The histogram shows the distribution of the values for the percentage slope gradient in the study area.



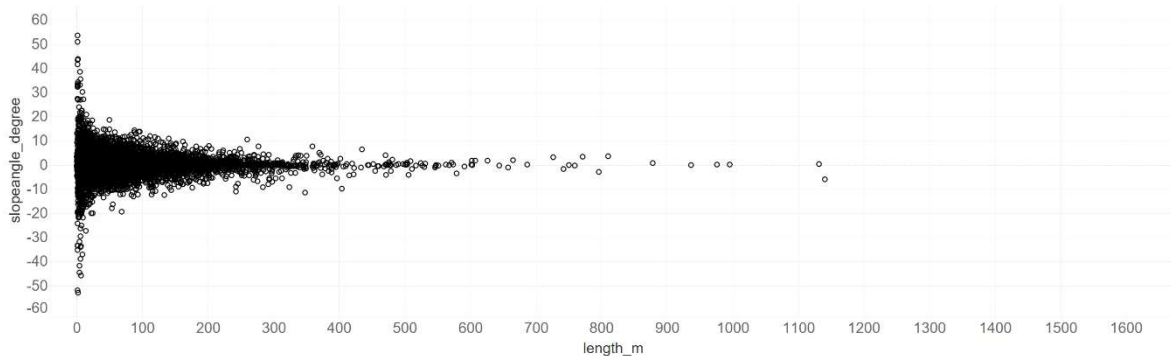
**Figure 44:** The histogram shows the distribution of the values for slope angle in degree in the study area.



The histograms in Figure 43 and Figure 44 show the distribution of the values for the slope angle in percent and degrees, respectively. In contrast, the scatterplot in Figure 45 compares the slope angle in degrees with the length of every street segment in the study area. Most of the segments are comparatively short with an overall mean in the dataset of 53 meters and a median 34 meters. The slope angle is within the 15 degrees of the given maximal gradeability of this test set (which is approximately 99% of all values). However, it reveals several outliers beyond the specified gradeability, particularly on short street segments. Causes might be at least partly interpolation inaccuracies during extraction process

of altitude values, as there exist no street segments with a slope angle of 60 degrees (cf. Section 4.2.4 and 6.1). An average of the height that was used for the calculated slope angle of long street segments is not an accurate representation of the true height (which may fluctuate). For further research, street segments could be divided into equidistant blocks, e.g., 10 meters (cf. NEAIMEH ET AL 2013). That way, I can avoid inaccurate predictions of energy consumptions which are caused by long segments with fluctuating elevation.

**Figure 45:** The scatterplot compares the slope angle in degree and the length of every street segment in the study area.



As far as gradeability is concerned, one could restrict slope over a certain predefined threshold value in order to not exclude street segments which are actually viable, as realized in the current approach in Section 4.2.7 (e.g., through legal construction restraints). Furthermore, instead of only integrating a maximum gradeability, maximum torque of the electric motor could be incorporated as well.

Also, the configuration file for bicycles has no usability in the model yet. For example, the existing classification of street classes is used to their practicability for bicycles (tagged through priority). The usage of the configuration file for bicycles would create no additional value at this stage of the work. However, one could classify street classes according to the friction of electric bicycles on their surface. The accurate rolling coefficient could be accessed at any given moment. Naturally, the definition of rolling coefficients would need further empirical inquiry beforehand. Furthermore, classifications contained in the configuration file such as *cycleway:right* or *cycleway:left* could be employed in one way regulations to obtain a more reasonable routing. This would aid the creation of a ready-to-use application for end-users.

Further suggestions for wider automation and refinement of the model could include the automatic retrieval of the DEM, e.g., through ArcREST (Representational State Transfer).



However, the accuracy of energy consumption prediction could diminish if a DEM with less resolution is chosen (cf. Section 3.1).

In this process of model refinement, the data pipeline from ArcGIS Model Builder could be transformed into a pure python script using the ArcPy modul (cf. Programming Code ArcGIS Model Builder in Appendix). This could improve processing time, but unfortunately complicates the development process. ArcGIS Model Builder simplified the development process in terms of allowing the developer to see an overview of the building and implementation process, as well as the tuning of particular parameters.

Finally, a variety of improvements for the application itself are conceivable in order to truly achieve a ready-to-use application. One such possible improvement is the coloring of the path in terms of energy consumption upon each routing request (cf. analog version in Figure 35 in Section 6.2). Another improvement would be to speed up the application through the techniques summarized in Section 2.1. For example, BAST ET AL. 2015 points out that the work of FUNKE & STORANDT 2013 is a significant improvement of CH. The related work of STORANDT 2012 on routing for bicycles could serve as a leverage point to improve applications for e-bikes without recuperation. Adding automated data pipelines to the static approach allows a library to be established which stores graphs on different parameter levels in parallel for specific regions. This could be incorporated into an automated service, such as a WFS. Therefore, the user is able to access an arbitrary region with his own input parameters. A requirement for the applicability for end-user applications is that the necessary input parameters are supplied automatically. Weight and temperature could be determined by a model iteration as well (using levels/intervals as shown for velocity in this work). The application could access these parameters automatically. Using a calculated velocity (either through the current GNSS-Location or through the internal torque sensor from a vehicle via data logging<sup>8</sup>), an automatic alignment of the modelled and measured average velocity (per trip) is feasible. With regards to weather conditions, the application could extract temperature (using a weather API and/or an integrated thermometer) and also current precipitation (e.g., heavy snowfalls during Test Session A), which may affect the rolling coefficient differently on different road surfaces. In a similar manner, headwind (cf. Section 4.1, Equation

---

<sup>8</sup> Data Logging means the record of data over time with additional measurement methods (internal or external) with a specific device, a data logger. A current example for a far-reaching data logging (as an interface together with the transmission of data using mobile communication) for an electric bicycle is Stromer OMNI, which has an own API. Smide provided the related bicycle, but it was not possible to apply the API for my purposes, e.g., to extend the prototypical application with additional functionality (Stromer desisted from sharing it with additional third-party users at the current stage).



(3)) could be taken into account through matching of respective azimuth (by use of gyroscope, GNSS) with the prevailing wind direction and speed (by use of sensors and/or a weather API). The automatic retrieval of weight via an on-board weighing device in the saddle or seat, value transmitted through a data logger would also be a conceivable improvement. Further on, data logging of the current SOC allows for continued alignment with the computed energy consumption.

Since time is of the utmost importance to most people when it comes to finding a route from A to B, it is likely one may choose the shortest or fastest route instead of the least energy consuming path (as such a path might require slightly more time). Take this into consideration, the eventual goal is to develop an application for real-time navigation systems which is able to switch between shortest route and energy-saving mode. Consequently, a previously assumed inaccessible target at a certain low SOC could still be reached. As a result, this research extends the potential of e-bike routing.

The thesis' results were reviewed and accepted as conference short paper "Energy-based Routing and Cruising Range Estimation for Electric Bicycles" at the AGILE 2017 (HAUMANN ET AL. 2017), which was elected best short paper. Moreover, the thesis was honored with the ESRI Young Scholar Award 2017, Switzerland.



## 9. References

- ABAGNALE, C. ET AL., 2015a. A dynamic model for the performance and environmental analysis of an innovative e-bike. *Energy Procedia*, 81, pp.618–627.
- ABAGNALE, C. ET AL., 2015b. Model-based control for an innovative power-assisted bicycle. *Energy Procedia*, 81, pp.606–617.
- ARTMEIER, A. ET AL., 2010. The optimal routing problem in the context of battery-powered electric vehicles. *Workshop: CROCS*, pp.1–13.
- BA HUNG, N., JAEWON, S. & LIM, O., 2017. A study of the effects of input parameters on the dynamics and required power of an electric bicycle. *Applied Energy*.
- BAST, H. ET AL., 2015. Route Planning in Transportation Networks. *Microsoft Research Technical Report*, pp.1–65.
- BELLMAN, R., 1958. On a Routing Problem. *Quarterly of Applied Mathematics*, 16, pp.87–90.
- BOSCH, <https://www.bosch-ebike.com/ch-de/service/reichweiten-assistent/>.  
Date accessed: 11.04.2017.
- BOSCH, 2016. <https://www.bosch-ebike.com/de/news-und-storys/news/more/details/was-ist-die-premiumfunktion-topo-reichweite-636/show/>.  
Date accessed: 11.04.2017.
- BUCKLEY, A., 2008. <https://blogs.esri.com/esri/arcgis/2008/06/12/expressing-slope/>. *ESRI*.  
Date accessed: 11.04.2017.
- CARDONE, M., STRANO, S. & TERZO, M., 2015. Optimal power-assistance system for a new pedelec model. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, pp.1–14.
- CARVELO2GO, 2017. <https://www.carvelo2go.ch/de/>. Date accessed: 11.04.2017.
- CITY OF ZURICH, 2017. <https://data.stadt-zuerich.ch/dataset/veloweg>.  
Date accessed: 11.04.2017.
- DELLING, D. ET AL., 2009. Engineering and Augmenting Route Planning Algorithms. *LNCS*, 2, p.161.
- DIJKSTRA, E.W., 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, pp.269–271.
- DU, W., ZHANG, D. & ZHAO, X., 2009. Research on battery to ride comfort of electric bicycle based on multi-body dynamics theory. *Proceedings of the 2009 IEEE International Conference on Automation and Logistics, ICAL 2009*, pp.1722–1726.
- ERICSSON, E., LARSSON, H. & BRUNDELL-FREIJ, K., 2006. Optimizing route choice for lowest fuel consumption - Potential effects of a new driver support tool. *Transportation Research Part C: Emerging Technologies*, 14, pp.369–383.
- ESRI, 2016a. <https://desktop.arcgis.com/de/system-requirements/latest/database-requirements-postgresql.htm>. Date accessed: 11.04.2017.

- ESRI, 2016b. <https://desktop.arcgis.com/en/arcmap/10.4/extensions/network-analyst/algorithms-used-by-network-analyst.htm>. Date accessed: 11.04.2017.
- FAIRLEY, P., 2005. China's cyclists take charge: electric bicycles are selling by the millions despite efforts to ban them. *IEEE Spectrum*, 42, pp.54–59.
- FEDERAL STATISTICAL OFFICE (FSO), 2017. <https://www.bfs.admin.ch/bfs/de/home/statistiken/mobilitaet-verkehr/verkehrsinfrastruktur-fahrzeuge/fahrzeuge/strassenfahrzeuge-bestand-motorisierungsgrad.html>. Date accessed: 11.04.2017.
- FUNKE, S. & STORANDT, S., 2013. Polynomial-time Construction of Contraction Hierarchies for Multi-criteria Objectives. *Alenex 2013*, pp.41–54. Available at: <http://dx.doi.org/10.1137/1.9781611972931.4>.
- GALLO, G. & PALLOTTINO, S., 1982. A new algorithm to find the shortest paths between all pairs of nodes. *Discrete Applied Mathematics*, pp.23–35.
- GAUCH, M., 2017. E-Mail Communication 20.02.2017. (cf. E-Mail EMPA (Marcel Gauch) in Appendix)
- GEISBERGER, R. ET AL., 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. *Proceedings of the 7th Workshop on Experimental Algorithms (WEA 2008)*, 2, pp.319–333.
- GROSS, A.C., KYLE, C.R. & MALEWICKI, D.J., 1983. The aerodynamics of human powered land vehicles. *Scientific American*, 249, pp.142–152.
- HART, P.E., NILSSON, N.J. & RAPHAEL, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), pp.100–107.
- HAUMANN, S.T., BUCHER, D. & JONIETZ, D., 2017. Energy-based Routing and Cruising Range Estimation for Electric Bicycles. In A. Bregt et al., eds. *Societal Geo-Innovation: short papers, posters and poster abstracts of the 20th AGILE Conference on Geographic Information Science*. Wageningen University & Research 9-12 May 2017, Wageningen, the Netherlands.
- HOCH, N., 2015. *Customer-Centric Travel Planning for Electric Vehicles*. ETH Zürich.
- KARIMI, G. & LI, X., 2013. Thermal management of lithium-ion batteries for electric vehicles. *International Journal of Energy Research*, 37(1), pp.13–24.
- KARRAIS, N., 2014. Modellierung der verbrauchsbasierten Erreichbarkeit mit GIS. In *Angewandte Geoinformatik 2014. Beiträge zum 26. AGIT-Symposium Salzburg*. Herbert Wichmann Verlag, VDE VERLAG GMBH, Berlin/Offenbach, pp. 56–67.
- KASTL, D. & VERGARA, V., 2016. <http://workshop.pgrouting.org/2.1.0-dev/en/index.html>. Date accessed: 11.04.2017.
- LI, W. ET AL., 2016. Determining the Main Factors Influencing the Energy Consumption of Electric Vehicles in the Usage Phase. *Procedia CIRP*, 48, pp.352–357.
- LOMONOVA, E.A. ET AL., 2002. Development of an improved electrically assisted bicycle. *IEEE Industry Applications Conference, 2002. 37th IAS Annual Meeting*, pp.384–389.

- MAPBOX, <https://www.mapbox.com/industries/transportation/>. Date accessed: 11.04.2017.
- McLOUGHLIN, I.V. ET AL., 2012. Campus Mobility for the Future: The Electric Bicycle. *Journal of Transportation Technologies*, 2(1), pp.1–12.
- MORBIH'EN VÉLO, <http://www.velo.morbihan.fr/pv/public/MorbihanVelo2>.  
Date accessed: 01.09.2016.
- MORCHIN, W.C., 1994. Battery-powered electric bicycles. In *Proceedings of NORTHCON '94*. pp. 269–274.
- MUETZE, A. & TAN, Y.C., 2007. Electric bicycles - A performance evaluation. *IEEE Industry Applications Magazine*, pp.12–21.
- NEAIMEH, M. ET AL., 2012. Investigating the effects of topography and traffic conditions on the driving efficiency of Electric Vehicles to better inform smart navigation. In *IET and ITS Conference on Road Transport Information and Control (RTIC 2012)*. pp. 1–6.
- NEAIMEH, M. ET AL., 2013. Routing systems to extend the driving range of electric vehicles. *Intelligent Transport Systems, IET*, pp.327–336.
- OLIVA, J.A., WEIHRAUCH, C. & BERTRAM, T., 2013. A Model-Based Approach for Predicting the Remaining Driving Range in Electric Vehicles. *Annual Conference of the Prognostics and Health Management Society*, pp.438–448.
- OPENSTREETMAP WIKI, 2016. <https://wiki.openstreetmap.org/wiki/Routing#Developers>.  
Date accessed: 11.04.2017.
- OPENTRAFFIC, 2016. <http://opentraffic.io/>. Date accessed: 11.04.2017.
- PALLOTTINO, S., 1984. Shortest path methods: Complexity, interrelations and new propositions. *Networks*, pp.257–267.
- PAUL, F. & BOGENBERGER, K., 2014. Evaluation-method for a Station Based Urban-pedelec Sharing System. *Transportation Research Procedia*, 4, pp.482–493.
- PUBLI-BIKE, 2017. <https://www.publibike.ch/>. Date accessed: 11.04.2017.
- RUST, <https://www.rust-lang.org/en-US/>. Date accessed: 11.04.2017.
- SACHENBACHER, M. ET AL., 2011a. Efficient Energy-Optimal Routing for Electric Vehicles. *Proc. Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp.1402–1407.
- SACHENBACHER, M. ET AL., 2011b. Efficient Energy-Optimal Routing for Electric Vehicles. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp.1402–1407.
- SCHNEIDER, B., 2009. *E-Bike Reichweitentest - Alltagstauglichkeit von Elektrobikes*, Bern.
- SCHULTES, D. & SANDERS, P., 2007. Dynamic highway-node routing. *Experimental Algorithms*, pp.66–79.
- SIPSER, M., 2006. *Introduction to the Theory of Computation*, Cambridge: Massachusetts Institute of Technology.
- SMIDE, 2017. <http://www.smide.ch/>. Date accessed: 11.04.2017.

- SOMCHAIWONG, N. & PONGLANGKA, W., 2006. Regenerative power control for electric bicycle. *2006 SICE-ICASE International Joint Conference*, pp.4362–4365.
- STEINHILBER, S., WELLS, P. & THANKAPPAN, S., 2013. Socio-technical inertia: Understanding the barriers to electric vehicles. *Energy Policy*, 60, pp.531–539.
- STORANDT, S., 2012. Route Planning for Bicycles — Exact Constrained Shortest Paths Made Practical Via Contraction Hierarchy. *Icaps*, pp.234–242.
- WANG, J., BESSELINK, I. & NIJMEIJER, H., 2015. Electric vehicle energy consumption modelling and prediction based on road information. *EVS28 International Electric Vehicle Symposium and Exhibition*, pp.1–12.
- WILLE, M., 2016. E-Mail Communication 19.12.2016.  
(cf. E-Mail Bosch (Martin Wille) in Appendix)
- WILSON, D.G., 2004. *Bicycling Science*, Cambridge: Massachusetts Institute of Technology.
- YANG, M.J. ET AL., 2009. A cost-effective method of electric brake with energy regeneration for electric vehicles. *IEEE Transactions on Industrial Electronics*, 56, pp.2203–2212.
- YUKSEL, T. & MICHALEK, J.J., 2015. Effects of regional temperature on electric vehicle efficiency, range, and emissions in the united states. *Environmental Science and Technology*, 49, pp.3974–3980.
- ZHAN, B.F. & NOON, C.E., 1998. Shortest path algorithms: an evaluation using real road networks. *Transportation Science*, 32, pp.65–73.





## Appendix

### E-Mail Bosch (Martin Wille)

12/04/2017

Gmail - Re: [Ticket#2016090693023914961] Ihre Anfrage vom Kontaktformular | Bosch Schweiz



Simon Haumann &lt;samsomon@googlemail.com&gt;

#### Re: [Ticket#2016090693023914961] Ihre Anfrage vom Kontaktformular | Bosch Schweiz

Bosch eBike Systems <contact@bosch-ebike.de>  
An: Simon Haumann <simon\_haumann@gmx.ch>

19. Dezember 2016 um 17:05

Sehr geehrter Herr Haumann,

die Premiumfunktion „Topo-Reichweite“ zeigt auf Nyon die mögliche Restreichweite unter Berücksichtigung der topografischen Gegebenheiten. Jedoch nicht, die energiegunstigste Strecke. Es gibt auch gibt auch keine Empfehlung welche Strecke zu fahren ist, um mit der vorhandenen Energie noch ans Ziel zu kommen.

Diese Funktion war eine Grundidee, die auf der Eurobike 2013 schon im Ansatz vorgestellt wurde. Siehe <https://www.youtube.com/watch?v=ZkOoSDxhkikA> bei 1 Minute 28 Sekunden, „Battery Range“. Das Nyon wurde im September 2014 in den Markt eingeführt. Die Topo-Reichweite war jedoch erst mit dem Software-Update Ende April 2016 verfügbar.

Mit welchen Ansätzen, der Reichweiten- und Streckenberechnung, wir uns auseinander setzten, darf ich Ihnen aus strategischen Gründen nicht mitteilen, selbst wenn Sie sich mit Ihrer Idee decken würde.

Für Ihre Masterthesis und Ihr Thema wünschen wir Ihnen viel Erfolg.

Mit freundlichen Grüßen  
Martin Wille  
Robert Bosch GmbH  
Bosch eBike Systems - Customer Service

Von: Simon Haumann <simon\_haumann@gmx.ch>  
An: Bosch eBike Systems <contact@bosch-ebike.de>  
Betreff: Re: [Ticket#2016090693023914961] Ihre Anfrage vom Kontaktformular | Bosch Schweiz

Sehr geehrte Damen und Herren,

Ich danke Ihnen vielmals für die Auskunft. Nachdem ich kürzlich die Funktion "Topo-Reichweite" von Nyon (<https://www.bosch-ebike.com/de/news-und-stories/news/more/details/was-ist-die-premiumfunktion-topo-reichweite-636/show/>) entdeckt habe, stellt sich mir nun jedoch die Frage, ob diese nicht genau wie beschrieben und von mir nun entwickelt funktioniert.

Ich wäre Ihnen für eine weitere Auskunft sehr dankbar und möchte mich dafür bedanken, dass Sie sich die Zeit nehmen.

Freundliche Grüsse  
Simon Haumann

<https://mail.google.com/mail/u/0/?ui=2&ik=534ce44f8d&view=pt&q=bosch&qf=true&search=query&msg=15917d4a839847f5&siml=15917d4a8398...> 1/1



universität  
wien



**ETH**

Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

## E-Mail EMPA (Marcel Gauch)

12/04/2017

Gmail - eBike 2



Simon Haumann <samsimon@googlemail.com>

eBike 2

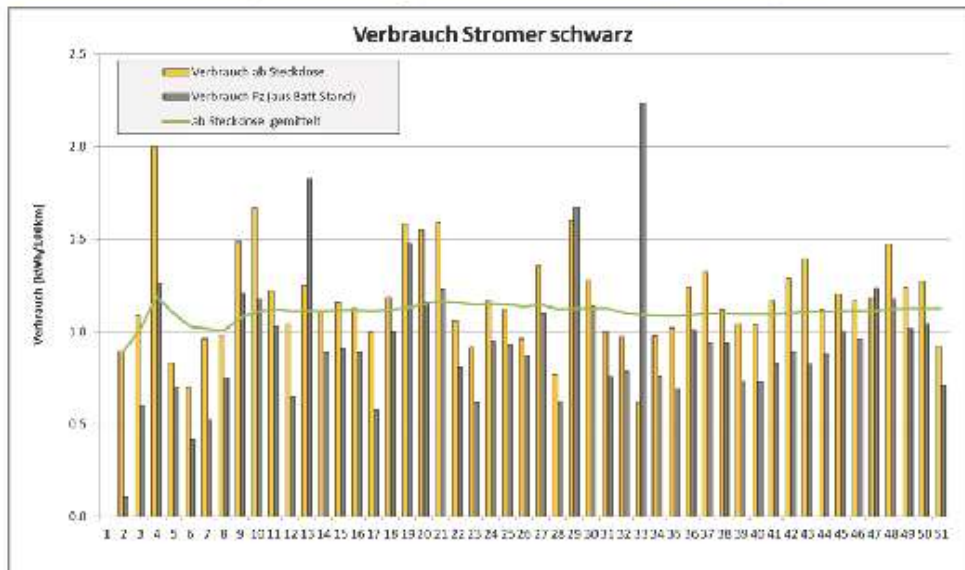
Gauch, Marcel <Marcel.Gauch@empa.ch>  
 An: Simon Haumann <simon\_haumann@gmx.ch>

20. Februar 2017 um 12:05

Hallo Herr Haumann

Sie können alles was ich angegeben habe verwenden. Beim Stromer daran denken, dass es die schnelle Variante war. Wir haben keine Angaben, was der Einfluss verschiedener Fahrer (je nach Stärke des Mitreitens) sein könnte. Diese Messung wäre sehr aufwendig. Ich denke aber, dass der mittlere Verbrauchswert über fast 2000km sowieso informativer ist.

Hier unsere Messresultate eines 45 km/h Stromers über 50 Messungen / fast 2000km: Verbrauchsmittelwert 1.12 kWh/100km ab Steckdose:



Herzliche Grüsse  
 Marcel Gauch

Von: Simon Haumann [mailto:simon\_haumann@gmx.ch]  
 Gesendet: Montag, 20. Februar 2017 11:56

[Dieser Text ausgeblendet]  
 [Dieser Text ausgeblendet]

<https://mail.google.com/mail/u/0/?ui=2&ik=534ce44f8d&view=pt&q=marcel%20gauch&qe=true&search=query&msg=15a5b3232436abb4&siml=15...> 1/1

## Educational Use Data swisstopo



Nutzungsvereinbarung für Swisstopo-Geodaten für ETH-Angehörige  
Usage of geodata for ETH employees and students

Name: HAUMANN Vorname: SIMON  
Last Name First Name  
Strasse: OBERWIESENSTRASSE 29 PLZ: 5332 Ort: REKINGEN  
Street Zip Code City  
Telefon: 076 549 22 55 E-Mail: simon.haumann@gmx.ch  
Phone

Departement/Institut: KARTOGRAPHIE UND GEONFORMATION  
Department/Institute

Bearbeitetes Gebiet:  
Field of research

Region ZÜRICH

Thema / Subject MOBILITY AND GIS

Benutzte Geodaten:  
Used Geodata

DHM 25  
SWISS ALTI 3D (2m)

Benutzungszweck (Titel einer geplanten Publikation, Diplomarbeit etc.):  
Purpose of use (title of a planned publication)

OPTIMIZING THE OPERATION RANGE OF E-BIKES IN ROUTING SYSTEMS

Die Daten dürfen nur Unterrichts- und Forschungszwecke im ETH-Bereich verwendet werden.  
The data shall be used for teaching and scientific purposes within ETH Domain only.

Die Originaldaten müssen nach Abschluss der Arbeiten gelöscht werden.  
After the project has been brought to a close the original data must be deleted.

Der Benutzer kennt den Inhalt des Merkblattes „Nutzungsbestimmungen für Swisstopo-Geodaten für ETH-Angehörige“.

The user has read the information sheet „Nutzungsbestimmungen für Swisstopo-Geodaten für ETH-Angehörige“.

Datum: 02.08.2016  
Date

Unterschrift: Simon Haumann  
Signature

Hinweis: Die Daten unterliegen dem Datenschutz und dienen allein der statistischen Auswertung und der Kontrolle über die benutzten Geodaten.

Note: The data is subject to data protection laws and used for statistical purposes and geodata usage control only.

Visum:



universität  
wien



**ETH**

Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

## Programming Code ArcGIS Model Builder

*energyconsumptionmodel\_egomovement\_whiteknight.py*

```

# -*- coding: utf-8 -*-
# -----
----
# energyconsumptionmodel_egomovement_whiteknight.py
# Created on: 2017-05-29 11:00:44.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionmodel_egomovement_whiteknight <factor_hu-
man_torque> <auxiliary_components_W> <rolling_coefficient> <gradeabil-
ity_degree> <gearbox_efficiency> <motor_efficiency> <wheel_diame-
ter_inches> <temperature_celsius> <weight_driver_kg> <weight_e_bike_kg>
<DEM_tif> <mapconfig_for_bicycles_xml> <osm2pgrouting_exe> <Workspace>
<Scratch>
# Description:
# -----
----

# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("C:/Users/sim_h/OneDrive/Dokumente/1_Uni_Wien/16_Mas-
terarbeit/4_Daten/Daten/Workspace/databaseconnectionfile.sde/ebike.post-
gres.energy_consumption_model")

# Script arguments
factor_human_torque = arcpy.GetParameterAsText(0)
if factor_human_torque == '#' or not factor_human_torque:
    factor_human_torque = "0.1" # provide a default value if unspecified

auxiliary_components_W = arcpy.GetParameterAsText(1)
if auxiliary_components_W == '#' or not auxiliary_components_W:
    auxiliary_components_W = "1.08" # provide a default value if unspeci-
fied

rolling_coefficient = arcpy.GetParameterAsText(2)
if rolling_coefficient == '#' or not rolling_coefficient:
    rolling_coefficient = "0.003" # provide a default value if unspeci-
fied

gradeability_degree = arcpy.GetParameterAsText(3)
if gradeability_degree == '#' or not gradeability_degree:
    gradeability_degree = "15" # provide a default value if unspecified

gearbox_efficiency = arcpy.GetParameterAsText(4)
if gearbox_efficiency == '#' or not gearbox_efficiency:
    gearbox_efficiency = "0.98" # provide a default value if unspecified

motor_efficiency = arcpy.GetParameterAsText(5)
if motor_efficiency == '#' or not motor_efficiency:
    motor_efficiency = "0.45" # provide a default value if unspecified

wheel_diameter_inches = arcpy.GetParameterAsText(6)
if wheel_diameter_inches == '#' or not wheel_diameter_inches:
    wheel_diameter_inches = "28" # provide a default value if unspecified

```



```

temperature_celsius = arcpy.GetParameterAsText(7)
if temperature_celsius == '#' or not temperature_celsius:
    temperature_celsius = "0" # provide a default value if unspecified

weight_driver_kg = arcpy.GetParameterAsText(8)
if weight_driver_kg == '#' or not weight_driver_kg:
    weight_driver_kg = "100" # provide a default value if unspecified

weight_e_bike_kg = arcpy.GetParameterAsText(9)
if weight_e_bike_kg == '#' or not weight_e_bike_kg:
    weight_e_bike_kg = "23" # provide a default value if unspecified

DEM_tif = arcpy.GetParameterAsText(10)
if DEM_tif == '#' or not DEM_tif:
    DEM_tif = "%Scratch%\DEM.tif" # provide a default value if unspecified

mapconfig_for_bicycles_xml = arcpy.GetParameterAsText(11)
if mapconfig_for_bicycles_xml == '#' or not mapconfig_for_bicycles_xml:
    mapconfig_for_bicycles_xml = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\mapconfig_for_bicycles.xml" # provide a default value
if unspecified

osm2pgrouting_exe = arcpy.GetParameterAsText(12)
if osm2pgrouting_exe == '#' or not osm2pgrouting_exe:
    osm2pgrouting_exe = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\osm2pgrouting.exe" # provide a default value if unspec-
ified

Workspace = arcpy.GetParameterAsText(13)
if Workspace == '#' or not Workspace:
    Workspace = "Database Connections\\Connection to localhost.sde" #
provide a default value if unspecified

Scratch = arcpy.GetParameterAsText(14)
if Scratch == '#' or not Scratch:
    Scratch = "C:\\Users\\sim_h\\OneDrive\\Dokumente\\1_Uni_Wien\\16_Mas-
terarbeit\\4_Daten\\Daten\\Workspace\\Scratch" # provide a default value
if unspecified

# Local variables:
temp_osm = "%Scratch%\\temp.osm"
ebike_postgres_ways_calculation = "%Workspace%\\ebike.postgres.ways_cal-
culation"

# Set Geoprocessing environments
arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Doku-
mente\\1_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.snapRaster = ""
arcpy.env.extent = "8.485 47.35 8.57 47.42"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"

# Process: energyconsumptionsubmodel_whiteknight_1
arcpy.energyconsumptionsubmodelwhiteknight1_energy_consump-
tion_model(Scratch, Workspace, "8.485 47.35 8.57 47.42", temp_osm,
osm2pgrouting_exe, mapconfig_for_bicycles_xml, DEM_tif, weight_e_bike_kg,
weight_driver_kg, temperature_celsius, wheel_diameter_inches, motor_effi-
ciency, gearbox_efficiency, gradeability_degree, rolling_coefficient,

```

```
ebike_postgres_ways_calculation, "http://www.overpass-  
api.de/api/xapi_meta?", auxiliary_components_W, factor_human_torque)  
  
# Process: energyconsumptionsubmodel_whiteknight_2  
arcpy.energyconsumptionsubmodelwhiteknight2_energy_consumption_model("5",  
"25", "5", ebike_postgres_ways_calculation)
```



*energyconsumptionsubmodel\_whiteknight\_1.py*

```

# -*- coding: utf-8 -*-
# -----
#
# energyconsumptionsubmodel_whiteknight_1.py
# Created on: 2017-05-29 11:01:05.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionsubmodel_whiteknight_1 <Scratch> <Workspace>
<Extent> <temp_osm> <osm2pgrouting_exe> <mapconfig_for_bicycles_xml>
<DEM> <weight_e_bike_kg> <weight_driver_kg> <temperature_celsius>
<wheel_diameter_inches> <motor_efficiency> <gearbox_efficiency> <gradeability_degree>
<rolling_coefficient> <ebike_postgres_ways_calculation_55> <Download_URL>
<auxiliary_components_W> <factor_human_torque>
# Description:
# -----
#

# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("C:/Users/sim_h/OneDrive/Dokumente/1_Uni_Wien/16_Mas-
terarbeit/4_Daten/Daten/Workspace/databaseconnectionfile.sde/ebike.post-
gres.energy_consumption_model")

# Script arguments
Scratch = arcpy.GetParameterAsText(0)
if Scratch == '#' or not Scratch:
    Scratch = "C:\\Users\\sim_h\\OneDrive\\Dokumente\\1_Uni_Wien\\16_Mas-
terarbeit\\4_Daten\\Daten\\Workspace\\Scratch" # provide a default value
if unspecified

Workspace = arcpy.GetParameterAsText(1)
if Workspace == '#' or not Workspace:
    Workspace = "Database Connections\\Connection to localhost.sde" #
provide a default value if unspecified

Extent = arcpy.GetParameterAsText(2)
if Extent == '#' or not Extent:
    Extent = "8.485 47.35 8.57 47.42" # provide a default value if un-
specified

temp_osm = arcpy.GetParameterAsText(3)
if temp_osm == '#' or not temp_osm:
    temp_osm = "%Scratch%\\temp.osm" # provide a default value if unspec-
ified

osm2pgrouting_exe = arcpy.GetParameterAsText(4)
if osm2pgrouting_exe == '#' or not osm2pgrouting_exe:
    osm2pgrouting_exe = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\osm2pgrouting.exe" # provide a default value if unspec-
ified

mapconfig_for_bicycles_xml = arcpy.GetParameterAsText(5)
if mapconfig_for_bicycles_xml == '#' or not mapconfig_for_bicycles_xml:
    mapconfig_for_bicycles_xml = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\mapconfig_for_bicycles.xml" # provide a default value
if unspecified

```



```

DEM = arcpy.GetParameterAsText(6)
if DEM == '#' or not DEM:
    DEM = "%Scratch%\\DEM.tif" # provide a default value if unspecified

weight_e_bike_kg = arcpy.GetParameterAsText(7)
if weight_e_bike_kg == '#' or not weight_e_bike_kg:
    weight_e_bike_kg = "23" # provide a default value if unspecified

weight_driver_kg = arcpy.GetParameterAsText(8)
if weight_driver_kg == '#' or not weight_driver_kg:
    weight_driver_kg = "100" # provide a default value if unspecified

temperature_celsius = arcpy.GetParameterAsText(9)
if temperature_celsius == '#' or not temperature_celsius:
    temperature_celsius = "20" # provide a default value if unspecified

wheel_diameter_inches = arcpy.GetParameterAsText(10)
if wheel_diameter_inches == '#' or not wheel_diameter_inches:
    wheel_diameter_inches = "28" # provide a default value if unspecified

motor_efficiency = arcpy.GetParameterAsText(11)
if motor_efficiency == '#' or not motor_efficiency:
    motor_efficiency = "0.45" # provide a default value if unspecified

gearbox_efficiency = arcpy.GetParameterAsText(12)
if gearbox_efficiency == '#' or not gearbox_efficiency:
    gearbox_efficiency = "0.98" # provide a default value if unspecified

gradeability_degree = arcpy.GetParameterAsText(13)
if gradeability_degree == '#' or not gradeability_degree:
    gradeability_degree = "15" # provide a default value if unspecified

rolling_coefficient = arcpy.GetParameterAsText(14)
if rolling_coefficient == '#' or not rolling_coefficient:
    rolling_coefficient = "0.003" # provide a default value if unspecified

ebike_postgres_ways_calculation_55_ = arcpy.GetParameterAsText(15)
if ebike_postgres_ways_calculation_55_ == '#' or not ebike_postgres_ways_calculation_55_:
    ebike_postgres_ways_calculation_55_ = "%Workspace%\\ebike.postgres.ways_calculation" # provide a default value if unspecified

Download_URL = arcpy.GetParameterAsText(16)
if Download_URL == '#' or not Download_URL:
    Download_URL = "http://www.overpass-api.de/api/xapi_meta?" # provide a default value if unspecified

auxiliary_components_W = arcpy.GetParameterAsText(17)
if auxiliary_components_W == '#' or not auxiliary_components_W:
    auxiliary_components_W = "1.08" # provide a default value if unspecified

factor_human_torque = arcpy.GetParameterAsText(18)
if factor_human_torque == '#' or not factor_human_torque:
    factor_human_torque = "0.1" # provide a default value if unspecified

# Local variables:

```

```

ebike_postgres_ways = "%Workspace%\ebike.postgres.ways"
successful = "true"
Delete_succeeded_3_ = successful
ebike_postgres_ways_calculation_2_ = "%Workspace%\ebike.postgres.ways_calculation"
ebike_postgres_ways_calculation_4_ = ebike_postgres_ways_calculation_2_
ebike_postgres_ways_calculation_23_ = ebike_postgres_ways_calculation_2_
ebike_postgres_ways_target = "%Workspace%\ebike.postgres.ways_target"
ebike_postgres_target_for_extract = "Database Connections\Connection to localhost.sde\ebike.postgres.target_for_extract"
dem_proj = "%Workspace%\ebike.postgres.dem_proj"
ebike_postgres_target_extract = "%Workspace%\ebike.postgres.target_extract"
ebike_postgres_target_extract_3_ = ebike_postgres_target_extract
ebike_postgres_target_extract_2_ = ebike_postgres_target_extract_3_
ebike_postgres_target_extract_4_ = ebike_postgres_target_extract_2_
ebike_postgres_ways_calculation_36_ = ebike_postgres_ways_calculation_23_
ebike_postgres_ways_calculation_47_ = ebike_postgres_ways_calculation_23_
ebike_postgres_ways_source = "%Workspace%\ebike.postgres.ways_source"
ebike_postgres_source_for_extract = "Database Connections\Connection to localhost.sde\ebike.postgres.source_for_extract"
ebike_postgres_source_extract = "%Workspace%\ebike.postgres.source_extract"
ebike_postgres_source_extract_3_ = ebike_postgres_source_extract
ebike_postgres_source_extract_2_ = ebike_postgres_source_extract_3_
ebike_postgres_source_extract_4_ = ebike_postgres_source_extract_2_
ebike_postgres_ways_calculation_10_ = ebike_postgres_ways_calculation_36_
ebike_postgres_ways_calculation_33_ = ebike_postgres_ways_calculation_10_
ebike_postgres_ways_calculation_46_ = ebike_postgres_ways_calculation_33_
ebike_postgres_ways_calculation_16_ = ebike_postgres_ways_calculation_46_
ebike_postgres_ways_calculation_9_ = ebike_postgres_ways_calculation_16_
ebike_postgres_ways_calculation_8_ = ebike_postgres_ways_calculation_9_
ebike_postgres_ways_calculation_12_ = ebike_postgres_ways_calculation_8_
ebike_postgres_ways_calculation_13_ = ebike_postgres_ways_calculation_12_
ebike_postgres_ways_calculation = ebike_postgres_ways_calculation_13_
unsuccessful = "false"
ebike_postgres_ways_calculation_15_ = ebike_postgres_ways_calculation_47_
ebike_postgres_ways_calculation_39_ = ebike_postgres_ways_calculation_15_
ebike_postgres_ways_calculation_25_ = ebike_postgres_ways_calculation_39_
ebike_postgres_ways_calculation_5_ = ebike_postgres_ways_calculation_25_
ebike_postgres_ways_calculation_11_ = ebike_postgres_ways_calculation_5_
ebike_postgres_ways_calculation_3_ = ebike_postgres_ways_calculation_11_
ebike_postgres_ways_calculation_7_ = ebike_postgres_ways_calculation_3_

```



```

ebike_postgres_ways_calculation__14_ = ebike_postgres_ways_calcula-
tion__7_
Delete_succeeded = "false"
Delete_succeeded__2_ = "false"

# Set Geoprocessing environments
arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Doku-
mente\\1_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"

# Process: Download OSM Data (XAPI)
tempEnvironment0 = arcpy.env.scratchWorkspace
arcpy.env.scratchWorkspace = Scratch
arcpy.XAPIDownload_osmtools(Download_URL, Extent, "*", "highway=pri-
mary|primary_link|secondary|tertiary|residential|living_street|track|pe-
destrian|path|cycleway|footway|byway|unclassified|secondary_link|ter-
tiary_link|lane|track|opposite_lane|oppo-
site|grade1|grade2|grade3|grade4|grade5|roundabout", temp_osm)
arcpy.env.scratchWorkspace = tempEnvironment0

# Process: osm2pgrouting
arcpy.osm2pgrouting_energy_consumption_model(osm2pgrouting_exe, temp_osm,
mapconfig_for_bicycles_xml)

# Process: Copy
arcpy.Copy_management(ebike_postgres_ways, ebike_postgres_ways_calcula-
tion__2_, "%Workspace\\ebike.postgres.ways")

# Process: Make XY Event Layer (1)
arcpy.MakeXYEventLayer_management(ebike_postgres_ways_calculation__2_,
"x2", "y2", ebike_postgres_ways_target, "GEOGCS['GCS_WGS_1984', DA-
TUM['D_WGS_1984', SPHE-
ROID['WGS_1984', 6378137.0, 298.257223563]], PRIMEM['Green-
wich', 0.0], UNIT['Degree', 0.0174532925199433]];-400 -400 1000000000;-
100000 10000;-100000 10000;8.98315284119522E-09;0.001;0.001;IsHighPreci-
sion", "")

# Process: Feature Class to Feature Class
arcpy.FeatureClassToFeatureClass_conversion(ebike_postgres_ways_target,
Workspace, "target_for_extract", "", "", "")

# Process: Project Raster
arcpy.ProjectRaster_management(DEM, dem_proj, "GEOGCS['GCS_WGS_1984', DA-
TUM['D_WGS_1984', SPHE-
ROID['WGS_1984', 6378137.0, 298.257223563]], PRIMEM['Green-
wich', 0.0], UNIT['Degree', 0.0174532925199433]]", "NEAREST",
"2.21884308916892E-05 2.21884308916892E-05", "CH1903_To_WGS_1984_1", "",
"PROJCS['CH1903_LV03', GEOGCS['GCS_CH1903', DATUM['D_CH1903', SPHEROID['Bes-
sel_1841', 6377397.155, 299.1528128]], PRIMEM['Greenwich', 0.0], UNIT['De-
gree', 0.0174532925199433]], PROJECTION['Hotine_Oblique_Mercator_Azi-
muth_Center'], PARAMETER['False_Easting', 600000.0], PARAMETER['False_North-
ing', 200000.0], PARAMETER['Scale_Factor', 1.0], PARAMETER['Azi-
muth', 90.0], PARAMETER['Longitude_Of_Center', 7.439583333333333], PARAME-
TER['Latitude_Of_Center', 46.952405555555556], UNIT['Me-
ter', 1.0]], VERTCS['LN_1902', VDATUM['Landesnivellement_1902'], PARAME-
TER['Vertical_Shift', 0.0], PARAMETER['Direction', 1.0], UNIT['Meter', 1.0]]")

# Process: Extract Values to Points
tempEnvironment0 = arcpy.env.workspace

```

```

arcpy.env.workspace = Workspace
arcpy.gp.ExtractValuesToPoints_sa(ebike_postgres_target_for_extract,
dem_proj, ebike_postgres_target_extract, "INTERPOLATE", "VALUE_ONLY")
arcpy.env.workspace = tempEnvironment0

# Process: Add Field (10)
arcpy.AddField_management(ebike_postgres_target_extract, "target_el",
"DOUBLE", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (10)
arcpy.CalculateField_management(ebike_postgres_target_extract__3_, "tar-
get_el", "!rastervalu!", "PYTHON_9.3", "")

# Process: Delete Field (3)
arcpy.DeleteField_management(ebike_postgres_target_extract__2_, "raster-
valu")

# Process: Join Field (5)
arcpy.JoinField_management(ebike_postgres_ways_calculation__2_, "gid",
ebike_postgres_target_extract__4_, "gid", "target_el")

# Process: Make XY Event Layer (2)
arcpy.MakeXYEventLayer_management(ebike_postgres_ways_calculation__2_,
"x1", "y1", ebike_postgres_ways_source, "GEOGCS['GCS_WGS_1984', DA-
TUM['D_WGS_1984', SPHE-
ROID['WGS_1984', 6378137.0, 298.257223563]], PRIMEM['Green-
wich', 0.0], UNIT['Degree', 0.0174532925199433]];-400 -400 1000000000;-
100000 10000;-100000 10000;8.98315284119522E-09;0.001;0.001;IsHighPreci-
sion", "")

# Process: Feature Class to Feature Class (2)
arcpy.FeatureClassToFeatureClass_conversion(ebike_postgres_ways_source,
Workspace, "source_for_extract", "", "", "")

# Process: Extract Values to Points (2)
tempEnvironment0 = arcpy.env.workspace
arcpy.env.workspace = Workspace
arcpy.gp.ExtractValuesToPoints_sa(ebike_postgres_source_for_extract,
dem_proj, ebike_postgres_source_extract, "INTERPOLATE", "VALUE_ONLY")
arcpy.env.workspace = tempEnvironment0

# Process: Add Field (11)
arcpy.AddField_management(ebike_postgres_source_extract, "source_el",
"DOUBLE", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (11)
arcpy.CalculateField_management(ebike_postgres_source_extract__3_,
"source_el", "!rastervalu!", "PYTHON_9.3", "")

# Process: Delete Field (2)
arcpy.DeleteField_management(ebike_postgres_source_extract__2_, "raster-
valu")

# Process: Join Field (4)
arcpy.JoinField_management(ebike_postgres_ways_calculation__2_, "gid",
ebike_postgres_source_extract__4_, "gid", "source_el")

# Process: Add Field (12)

```

```

arcpy.AddField_management(ebike_postgres_ways_calculation__23_, "slope-
perc", "DOUBLE", "", "", "", "slope_percentage", "NULLABLE", "NON_RE-
QUIRED", "")

# Process: Calculate Field (12)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__36_,
"slopeperc", "(!target_el!-!source_el!)/!length_m!*100", "PYTHON_9.3",
"")

# Process: Add Field (5)
arcpy.AddField_management(ebike_postgres_ways_calculation__10_, "slopean-
gle_degree", "DOUBLE", "", "", "", "angle_of_slope_degree", "NULLABLE",
"NON_REQUIRED", "")

# Process: Calculate Field (5)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__33_,
"slopeangle_degree", "math.degrees(math.atan(!target_el!-
!source_el!)/!length_m!))", "PYTHON_9.3", "")

# Process: Add Field (6)
arcpy.AddField_management(ebike_postgres_ways_calculation__46_,
"climbres_N", "DOUBLE", "", "", "", "climbingresistance_N", "NULLABLE",
"NON_REQUIRED", "")

# Process: Calculate Field (6)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__16_,
"climbres_N", "(%weight_driver_kg%+%weight_e-
bike_kg%)*9.806*math.sin(math.radians(!slopeangle_degree!))", "PY-
THON_9.3", "")

# Process: Add Field (13)
arcpy.AddField_management(ebike_postgres_ways_calculation__9_,
"rollres_N", "DOUBLE", "", "", "", "rollingresistance_N", "NULLABLE",
"NON_REQUIRED", "")

# Process: Calculate Field (14)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__8_,
"rollres_N", "%rolling_coefficient%*(%weight_driver_kg%+%weight_e-
bike_kg%)*9.806*math.cos(math.radians(!slopeangle_degree!))", "PY-
THON_9.3", "")

# Process: Add Field (8)
arcpy.AddField_management(ebike_postgres_ways_calculation__12_, "pres-
sure_hPa", "DOUBLE", "", "", "", "ambientairpressure_hPa", "NULLABLE",
"NON_REQUIRED", "")

# Process: Calculate Field (8)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__13_,
"pressure_hPa", "1013.25*math.pow(1-(0.0065*(!tar-
get_el!+!source_el!)/2))/(%temperature_celsius%+273.15),5.255)", "PY-
THON_9.3", "")

# Process: Add Field (25)
arcpy.AddField_management(ebike_postgres_ways_calculation__23_, "slope-
perc_r", "DOUBLE", "", "", "", "slope_percentage_reverse", "NULLABLE",
"NON_REQUIRED", "")

# Process: Calculate Field (25)

```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation__47_,
"slopeperc_r", "(!source_el!-!target_el!)/!length_m!*100", "PYTHON_9.3",
"")

# Process: Add Field (20)
arcpy.AddField_management(ebike_postgres_ways_calculation__15_, "slopean-
gle_r_degree", "DOUBLE", "", "", "", "angle_of_slope_reverse_degree",
"NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (20)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__39_,
"slopeangle_r_degree", "math.degrees(math.atan(!source_el!-!tar-
get_el!)/!length_m!)", "PYTHON_9.3", "")

# Process: Add Field (21)
arcpy.AddField_management(ebike_postgres_ways_calculation__25_,
"climbres_r_N", "DOUBLE", "", "", "", "climbingresistance_reverse_N",
"NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (21)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__5_,
"climbres_r_N", "(%weight_driver_kg%+%weight_e-
bike_kg%)*9.806*math.sin(math.radians(!slopeangle_r_degree!))", "PY-
THON_9.3", "")

# Process: Add Field (28)
arcpy.AddField_management(ebike_postgres_ways_calculation__11_,
"rollres_r_N", "DOUBLE", "", "", "", "rollingresistance_reverse_N", "NUL-
LABLE", "NON_REQUIRED", "")

# Process: Calculate Field (28)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__3_,
"rollres_r_N", "%rolling_coefficient%*(%weight_driver_kg%+%weight_e-
bike_kg%)*9.806*math.cos(math.radians(!slopeangle_degree!))", "PY-
THON_9.3", "")

# Process: Add Field (23)
arcpy.AddField_management(ebike_postgres_ways_calculation__7_, "pres-
sure_r_hPa", "DOUBLE", "", "", "", "ambientairpressure_reverse_hPa",
"NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (23)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__14_,
"pressure_r_hPa", "1013.25*math.pow(1-(0.0065*(!source_el!+!tar-
get_el!)/2))/(%temperature_celsius%+273.15),5.255)", "PYTHON_9.3", "")

# Process: Delete(3)
arcpy.Delete_management(ebike_postgres_target_for_extract, "")

# Process: Delete (2)
arcpy.Delete_management(ebike_postgres_source_for_extract, "")

# Process: Delete
arcpy.Delete_management(temp_osm, "")

```



*energyconsumptionsubmodel\_whiteknight\_2.py*

```

# -*- coding: utf-8 -*-
# -----
# -----
# energyconsumptionsubmodel_whiteknight_2.py
# Created on: 2017-05-29 11:02:10.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionsubmodel_whiteknight_2 <from_velocity_kmh>
<to_velocity_kmh> <by_velocity_kmh> <ebike_postgres_ways_calcula-
tion_55_>
# Description:
# -----
# -----

# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("Model Functions")

# Script arguments
from_velocity_kmh = arcpy.GetParameterAsText(0)
if from_velocity_kmh == '#' or not from_velocity_kmh:
    from_velocity_kmh = "5" # provide a default value if unspecified

to_velocity_kmh = arcpy.GetParameterAsText(1)
if to_velocity_kmh == '#' or not to_velocity_kmh:
    to_velocity_kmh = "25" # provide a default value if unspecified

by_velocity_kmh = arcpy.GetParameterAsText(2)
if by_velocity_kmh == '#' or not by_velocity_kmh:
    by_velocity_kmh = "5" # provide a default value if unspecified

ebike_postgres_ways_calculation_55_ = arcpy.GetParameterAsText(3)
if ebike_postgres_ways_calculation_55_ == '#' or not ebike_post-
gres_ways_calculation_55_ :
    ebike_postgres_ways_calculation_55_ = "Database Connections\\Connec-
tion to localhost.sde\\ebike.postgres.ways_calculation" # provide a de-
fault value if unspecified

# Local variables:
ebike_postgres_ways_calculation_6_ = ebike_postgres_ways_calcula-
tion_55_
ebike_postgres_ways_calculation_35_ = ebike_postgres_ways_calcula-
tion_55_
velocity_kmh = from_velocity_kmh
ebike_postgres_ways_calculation_29_ = ebike_postgres_ways_calcula-
tion_35_
ebike_postgres_ways_calculation_39_ = ebike_postgres_ways_calcula-
tion_29_
ebike_postgres_ways_calculation_4_ = ebike_postgres_ways_calcula-
tion_39_
ebike_postgres_ways_calculation_31_ = ebike_postgres_ways_calcula-
tion_4_
ebike_postgres_ways_calculation_34_ = ebike_postgres_ways_calcula-
tion_31_
ebike_postgres_ways_calculation_3_ = ebike_postgres_ways_calcula-
tion_34_

```

```

ebike_postgres_ways_calculation__30_ = ebike_postgres_ways_calcula-
tion__3_
ebike_postgres_ways_calculation__38_ = ebike_postgres_ways_calcula-
tion__30_
ebike_postgres_ways_calculation__10_ = ebike_postgres_ways_calcula-
tion__38_
ebike_postgres_ways_calculation__9_ = ebike_postgres_ways_calcula-
tion__10_
ebike_postgres_ways_calculation__23_ = ebike_postgres_ways_calcula-
tion__9_
ebike_postgres_ways_calculation__19_ = ebike_postgres_ways_calcula-
tion__23_
ebike_postgres_ways_calculation__5_ = ebike_postgres_ways_calcula-
tion__19_
ebike_postgres_ways_calculation__24_ = ebike_postgres_ways_calcula-
tion__5_
ebike_postgres_ways_calculation__44_ = ebike_postgres_ways_calcula-
tion__24_
ebike_postgres_ways_calculation = ebike_postgres_ways_calculation__6_
ebike_postgres_ways_calculation__37_ = ebike_postgres_ways_calculation
ebike_postgres_ways_calculation__33_ = ebike_postgres_ways_calcula-
tion__37_
ebike_postgres_ways_calculation__32_ = ebike_postgres_ways_calcula-
tion__33_
ebike_postgres_ways_calculation__21_ = ebike_postgres_ways_calcula-
tion__32_
ebike_postgres_ways_calculation__14_ = ebike_postgres_ways_calcula-
tion__21_
ebike_postgres_ways_calculation__27_ = ebike_postgres_ways_calcula-
tion__14_
ebike_postgres_ways_calculation__20_ = ebike_postgres_ways_calcula-
tion__27_
ebike_postgres_ways_calculation__16_ = ebike_postgres_ways_calcula-
tion__20_
ebike_postgres_ways_calculation__22_ = ebike_postgres_ways_calcula-
tion__16_
ebike_postgres_ways_calculation__26_ = ebike_postgres_ways_calcula-
tion__22_
ebike_postgres_ways_calculation__28_ = ebike_postgres_ways_calcula-
tion__26_
ebike_postgres_ways_calculation__25_ = ebike_postgres_ways_calcula-
tion__28_
ebike_postgres_ways_calculation__49_ = ebike_postgres_ways_calcula-
tion__25_
ebike_postgres_ways_calculation__48_ = ebike_postgres_ways_calcula-
tion__49_

```

```
# Set Geoprocessing environments
```

```

arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Doku-
mente\\1_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"

```

```
# Process: For
```

```

arcpy.IterateCount_mb(from_velocity_kmh, to_velocity_kmh, by_veloc-
ity_kmh)

```

```
# Process: Add Field (22)
```



```

arcpy.AddField_management(ebike_postgres_ways_calculation__55_,
"dragresv%velocity_kmh%_r_N", "DOUBLE", "", "", "", "dragresistanceveloc-
ity%velocity_kmh%_reverse_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (22)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__35_,
"dragresv%velocity_kmh%_r_N", "!pressure_r_hPa!*100/(2*287.058*(%tempera-
ture_celsius%+273.15))*1.15*0.55*math.pow((%velocity_kmh%/3.6),2)", "PY-
THON_9.3", "")

# Process: Add Field (16)
arcpy.AddField_management(ebike_postgres_ways_calculation__29_,
"tracforcev%velocity_kmh%_r_N", "DOUBLE", "", "", "", "tractiveforce-
velocity%velocity_kmh%_reverse_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (16)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__39_,
"tracforcev%velocity_kmh%_r_N",
"!climbres_r_N!+!rollres_r_N!+!dragresv%velocity_kmh%_r_N!", "PY-
THON_9.3", "")

# Process: Add Field (17)
arcpy.AddField_management(ebike_postgres_ways_calculation__4_,
"torquev%velocity_kmh%_r_Nm", "DOUBLE", "", "", "", "torquevelocity%ve-
locity_kmh%_reverse_Nm", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (17)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__31_,
"torquev%velocity_kmh%_r_Nm", "factorr(!tracforcev%velocity_kmh%_r_N!,
%factor_human_torque%, %wheel_diameter_inches)", "PYTHON_9.3", "def
factorr(tracforcevvelocity_kmh_r_N, factor_human_torque, wheel_diame-
ter_inches):\n    if tracforcevvelocity_kmh_r_N > 0:\n        return
(tracforcevvelocity_kmh_r_N-(tracforcevvelocity_kmh_r_N*factor_hu-
man_torque))*((wheel_diameter_inches*0.0254)/2)\n    else:\n        re-
turn tracforcevvelocity_kmh_r_N*((wheel_diameter_inches*0.0254)/2)")

# Process: Add Field (18)
arcpy.AddField_management(ebike_postgres_ways_calculation__34_, "angu-
larv%velocity_kmh%_r_s_inverse", "DOUBLE", "", "", "", "angularwheel-
velocity%velocity_kmh%_reverse_s_inverse", "NULLABLE", "NON_REQUIRED",
"")

# Process: Calculate Field (18)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__3_, "an-
gularv%velocity_kmh%_r_s_inverse", "(%velocity_kmh%/3.6)/((%wheel_diame-
ter_inches*0.0254)/2)", "PYTHON_9.3", "")

# Process: Add Field (24)
arcpy.AddField_management(ebike_postgres_ways_calculation__30_, "v%veloc-
ity_kmh%_norec_r_W", "DOUBLE", "", "", "", "velocity%velocity_kmh%_no_re-
cuperation_reverse_W", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (24)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__38_,
"v%velocity_kmh%_norec_r_W", "pemnorecrW(!slopeangle_r_degree!, %gradea-
bility_degree%, !torquev%velocity_kmh%_r_Nm!, !angularv%veloc-
ity_kmh%_r_s_inverse!, %motor_efficiency%, %gearbox_efficiency%, %auxil-
iary_components_W%)", "PYTHON_9.3", "def pemnorecrW(slopeangle_r_degree,

```

```

gradeability_degree, torquevvelocity_kmh_r_Nm, angularvveloc-
ity_kmh_r_s_inverse, motor_efficiency, gearbox_efficiency, auxiliary_com-
ponents_W):\n      if slopeangle_r_degree >= gradeability_degree:\n
return 999999\n      elif slopeangle_r_degree <= -gradeability_de-
gree:\n      return 999999\n      elif torquevvelocity_kmh_r_Nm <
0:\n      return auxiliary_components_W\n      else:\n      return
torquevvelocity_kmh_r_Nm * angularvvelocity_kmh_r_s_inverse / (motor_ef-
ficiency * gearbox_efficiency) + auxiliary_components_W")

```

```
# Process: Add Field (6)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_10_,
"ctempv%velocity_kmh%_nrec_r_W", "DOUBLE", "", "", "", "capacitiytemper-
aturevelocity%velocity_kmh%_no_recuperation_reverse_W", "NULLABLE", "RE-
QUIRED", "")

```

```
# Process: Calculate Field (6)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_9_,
"ctempv%velocity_kmh%_nrec_r_W", "pemctempnrecrW(!v%veloc-
ity_kmh%_nrec_r_W!, %temperature_celsius)", "PYTHON_9.3", "def pem-
ctempnrecrW(vvelocity_kmh_nrec_r_W, temperature_celsius):\n      if vve-
locity_kmh_nrec_r_W == 999999:\n      return 999999\n      if tempera-
ture_celsius >= 25:\n      return vvelocity_kmh_nrec_r_W\n
else:\n      return vvelocity_kmh_nrec_r_W * (1+((25-temperature_cel-
sius)*0.0047))")

```

```
# Process: Add Field (27)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_23_, "v%veloc-
ity_kmh%_nrec_r_Wh", "DOUBLE", "", "", "", "velocity%veloc-
ity_kmh%_no_recuperation_reverse_Wh", "NULLABLE", "NON_REQUIRED", "")

```

```
# Process: Calculate Field (27)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_19_,
"v%velocity_kmh%_nrec_r_Wh", "pemnrecrWh(!ctempv%veloc-
ity_kmh%_nrec_r_W!, !length_m!, %velocity_kmh%)", "PYTHON_9.3", "def
pemnrecrWh(ctempvvelocity_kmh_nrec_r_W, length_m, velocity_kmh):\n
if ctempvvelocity_kmh_nrec_r_W == 999999:\n      return 999999\n
else:\n      return ctempvvelocity_kmh_nrec_r_W*((length_m/1000)/ve-
locity_kmh)")

```

```
# Process: Add Field (12)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_5_, "v%veloc-
ity_kmh%_nrec_r_Whkm", "DOUBLE", "", "", "", "velocity%veloc-
ity_kmh%_nrecuperation_reverse_Wh_per_km", "NULLABLE", "NON_REQUIRED",
"")

```

```
# Process: Calculate Field (12)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_24_,
"v%velocity_kmh%_nrec_r_Whkm", "pemnrecrWhkm(!v%veloc-
ity_kmh%_nrec_r_Wh!, !length_m!)", "PYTHON_9.3", "def pemnrecrWhkm(vve-
locity_kmh_nrec_r_Wh, length_m):\n      if vvelocity_kmh_nrec_r_Wh ==
999999:\n      return None\n      else:\n      return vveloc-
ity_kmh_nrec_r_Wh*1000/ length_m")

```

```
# Process: Add Field (7)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_55_,
"dragresv%velocity_kmh%_N", "DOUBLE", "", "", "", "dragresistanceveloc-
ity%velocity_kmh%_N", "NULLABLE", "NON_REQUIRED", "")

```

```
# Process: Calculate Field (7)
```



```

arcpy.CalculateField_management(ebike_postgres_ways_calculation__6_,
"dragresv%velocity_kmh%_N", "!pressure_hPa!*100/(2*287.058*(%tempera-
ture_celsius%+273.15))*1.15*0.5*math.pow((%velocity_kmh%/3.6),2)", "PY-
THON_9.3", "")

# Process: Add Field
arcpy.AddField_management(ebike_postgres_ways_calculation,
"tracforcev%velocity_kmh%_N", "DOUBLE", "", "", "", "tractiveforceveloc-
ity%velocity_kmh%_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field
arcpy.CalculateField_management(ebike_postgres_ways_calculation__37_,
"tracforcev%velocity_kmh%_N", "!climbres_N!+!rollres_N!+!dragresv%veloc-
ity_kmh%_N!", "PYTHON_9.3", "")

# Process: Add Field (2)
arcpy.AddField_management(ebike_postgres_ways_calculation__33_,
"torquev%velocity_kmh%_Nm", "DOUBLE", "", "", "", "torquevelocity%veloc-
ity_kmh%_Nm", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (2)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__32_,
"torquev%velocity_kmh%_Nm", "factor(!tracforcev%velocity_kmh%_N!, %fac-
tor_human_torque%, %wheel_diameter_inches)", "PYTHON_9.3", "def fact-
tor(tracforcevvelocity_kmh_N, factor_human_torque, wheel_dia-
meter_inches):\n    if tracforcevvelocity_kmh_N > 0:\n        return
(tracforcevvelocity_kmh_N-(tracforcevvelocity_kmh_N*factor_hu-
man_torque))*((wheel_diameter_inches*0.0254)/2)\n    else:\n        re-
turn tracforcevvelocity_kmh_N*((wheel_diameter_inches*0.0254)/2)")

# Process: Add Field (3)
arcpy.AddField_management(ebike_postgres_ways_calculation__21_, "angu-
larv%velocity_kmh%_s_inverse", "DOUBLE", "", "", "", "angularwheelveloc-
ity%velocity_kmh%_s_inverse", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (3)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__14_,
"angularv%velocity_kmh%_s_inverse", "(%velocity_kmh%/3.6)/((%wheel_dia-
meter_inches*0.0254)/2)", "PYTHON_9.3", "")

# Process: Add Field (9)
arcpy.AddField_management(ebike_postgres_ways_calculation__27_, "v%veloc-
ity_kmh%_norec_W", "DOUBLE", "", "", "", "velocity%velocity_kmh%_no_recu-
peration_W", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (9)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__20_,
"v%velocity_kmh%_norec_W", "pennorecW(!slopeangle_degree!, %gradeabil-
ity_degree%, !torquev%velocity_kmh%_Nm!, !angularv%velocity_kmh%_s_in-
verse!, %motor_efficiency%, %gearbox_efficiency%, %auxiliary_compo-
nents_W%)", "PYTHON_9.3", "def pennorecW(slopeangle_degree, gradeabil-
ity_degree, torquevvelocity_kmh_Nm, angularvvelocity_kmh_s_inverse, mo-
tor_efficiency, gearbox_efficiency, auxiliary_components_W):\n    if
slopeangle_degree >= gradeability_degree:\n        return 999999\n
elif slopeangle_degree <= -gradeability_degree:\n        return
999999\n    elif torquevvelocity_kmh_Nm < 0:\n        return auxil-
iary_components_W\n    else:\n        return torquevvelocity_kmh_Nm *
angularvvelocity_kmh_s_inverse / (motor_efficiency * gearbox_efficiency)
+ auxiliary_components_W")

```

```

# Process: Add Field (10)
arcpy.AddField_management(ebike_postgres_ways_calculation_16_,
"ctempv%velocity_kmh%_norec_W", "DOUBLE", "", "", "", "capacitiytempera-
turevelocity%velocity_kmh%_no_recuperation_W", "NULLABLE", "REQUIRED",
"")

# Process: Calculate Field (10)
arcpy.CalculateField_management(ebike_postgres_ways_calculation_22_,
"ctempv%velocity_kmh%_norec_W", "pemctempnorecW(!v%veloc-
ity_kmh%_norec_W!, %temperature_celsius)", "PYTHON_9.3", "def pemctemp-
norecW(vvelocity_kmh_norec_W, temperature_celsius):\n    if vveloc-
ity_kmh_norec_W == 999999:\n        return 999999\n    if tempera-
ture_celsius >= 25:\n        return vvelocity_kmh_norec_W\n    else:\n
return vvelocity_kmh_norec_W * (1+((25-temperature_celsius)*0.0047))")

# Process: Add Field (15)
arcpy.AddField_management(ebike_postgres_ways_calculation_26_, "v%veloc-
ity_kmh%_norec_Wh", "DOUBLE", "", "", "", "velocity%velocity_kmh%_no_re-
cuperation_Wh", "NULLABLE", "REQUIRED", "")

# Process: Calculate Field (15)
arcpy.CalculateField_management(ebike_postgres_ways_calculation_28_,
"v%velocity_kmh%_norec_Wh", "pemnorecWh(!ctempv%velocity_kmh%_norec_W!,
!length_m!, %velocity_kmh%)", "PYTHON_9.3", "def pemnorecWh(ctempvveloc-
ity_kmh_norec_W, length_m, velocity_kmh):\n    if ctempvveloc-
ity_kmh_norec_W == 999999:\n        return 999999\n    else:\n
return ctempvvelocity_kmh_norec_W*((length_m/1000)/velocity_kmh)")

# Process: Add Field (20)
arcpy.AddField_management(ebike_postgres_ways_calculation_25_, "v%veloc-
ity_kmh%_norec_Whkm", "DOUBLE", "", "", "", "velocity%velocity_kmh%_nore-
cuperation_Wh_per_km", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (20)
arcpy.CalculateField_management(ebike_postgres_ways_calculation_49_,
"v%velocity_kmh%_norec_Whkm", "pemnorecWhkm(!v%velocity_kmh%_norec_Wh!,
!length_m!)", "PYTHON_9.3", "def pemnorecWhkm(vvelocity_kmh_norec_Wh,
length_m):\n    if vvelocity_kmh_norec_Wh == 999999:\n        return
None\n    else:\n        return vvelocity_kmh_norec_Wh*1000/ length_m")

```

*energyconsumptionmodel\_stromer\_st2.py*

```

# -*- coding: utf-8 -*-
# -----
# -----
# energyconsumptionmodel_stromer_st2.py
# Created on: 2017-05-29 11:02:31.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionmodel_stromer_st2 <factor_human_torque> <auxiliary_components_W> <rolling_coefficient> <gradeability_degree> <gearbox_efficiency> <wheel_diameter_inches> <temperature_celsius> <weight_driver_kg> <weight_e_bike_kg> <DEM> <mapconfig_for_bicycles_xml> <osm2pgrouting_exe> <Connection_to_localhost_sde> <Scratch> <ebike_postgres_ways_calculation_15_>
# Description:
# -----
# -----

# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("C:/Users/sim_h/OneDrive/Dokumente/1_Uni_Wien/16_Mas-
terarbeit/4_Daten/Daten/Workspace/databaseconnectionfile.sde/ebike.post-
gres.energy_consumption_model")

# Script arguments
factor_human_torque = arcpy.GetParameterAsText(0)
if factor_human_torque == '#' or not factor_human_torque:
    factor_human_torque = "0.1" # provide a default value if unspecified

auxiliary_components_W = arcpy.GetParameterAsText(1)
if auxiliary_components_W == '#' or not auxiliary_components_W:
    auxiliary_components_W = "8.37" # provide a default value if unspeci-
fied

rolling_coefficient = arcpy.GetParameterAsText(2)
if rolling_coefficient == '#' or not rolling_coefficient:
    rolling_coefficient = "0.003" # provide a default value if unspeci-
fied

gradeability_degree = arcpy.GetParameterAsText(3)
if gradeability_degree == '#' or not gradeability_degree:
    gradeability_degree = "15" # provide a default value if unspecified

gearbox_efficiency = arcpy.GetParameterAsText(4)
if gearbox_efficiency == '#' or not gearbox_efficiency:
    gearbox_efficiency = "0.98" # provide a default value if unspecified

wheel_diameter_inches = arcpy.GetParameterAsText(5)
if wheel_diameter_inches == '#' or not wheel_diameter_inches:
    wheel_diameter_inches = "26" # provide a default value if unspecified

temperature_celsius = arcpy.GetParameterAsText(6)
if temperature_celsius == '#' or not temperature_celsius:
    temperature_celsius = "3" # provide a default value if unspecified

weight_driver_kg = arcpy.GetParameterAsText(7)
if weight_driver_kg == '#' or not weight_driver_kg:

```



```

weight_driver_kg = "100" # provide a default value if unspecified

weight_e_bike_kg = arcpy.GetParameterAsText(8)
if weight_e_bike_kg == '#' or not weight_e_bike_kg:
    weight_e_bike_kg = "27" # provide a default value if unspecified

DEM = arcpy.GetParameterAsText(9)
if DEM == '#' or not DEM:
    DEM = "%Scratch%\DEM.tif" # provide a default value if unspecified

mapconfig_for_bicycles_xml = arcpy.GetParameterAsText(10)
if mapconfig_for_bicycles_xml == '#' or not mapconfig_for_bicycles_xml:
    mapconfig_for_bicycles_xml = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\mapconfig_for_bicycles.xml" # provide a default value
if unspecified

osm2pgrouting_exe = arcpy.GetParameterAsText(11)
if osm2pgrouting_exe == '#' or not osm2pgrouting_exe:
    osm2pgrouting_exe = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\osm2pgrouting.exe" # provide a default value if unspec-
ified

Connection_to_localhost_sde = arcpy.GetParameterAsText(12)
if Connection_to_localhost_sde == '#' or not Connection_to_localhost_sde:
    Connection_to_localhost_sde = "Database Connections\\Connection to
localhost.sde" # provide a default value if unspecified

Scratch = arcpy.GetParameterAsText(13)
if Scratch == '#' or not Scratch:
    Scratch = "C:\\Users\\sim_h\\OneDrive\\Dokumente\\1_Uni_Wien\\16_Mas-
terarbeit\\4_Daten\\Daten\\Workspace\\Scratch" # provide a default value
if unspecified

ebike_postgres_ways_calculation_15_ = arcpy.GetParameterAsText(14)
if ebike_postgres_ways_calculation_15_ == '#' or not ebike_post-
gres_ways_calculation_15_:
    ebike_postgres_ways_calculation_15_ = "%Workspace%\ebike.post-
gres.ways_calculation" # provide a default value if unspecified

# Local variables:
temp_osm = "%Scratch%\temp.osm"
ebike_postgres_ways_calculation_17_ = "%Workspace%\ebike.post-
gres.ways_calculation"
ebike_postgres_ways_calculation_16_ = ebike_postgres_ways_calcula-
tion_17_
ebike_postgres_ways_calculation_29_ = ebike_postgres_ways_calcula-
tion_16_
ebike_postgres_ways_calculation_31_ = ebike_postgres_ways_calcula-
tion_16_
ebike_postgres_ways_calculation = ebike_postgres_ways_calculation_31_
ebike_postgres_ways_calculation_49_ = ebike_postgres_ways_calculation
ebike_postgres_ways_calculation_10_ = ebike_postgres_ways_calcula-
tion_49_
ebike_postgres_ways_calculation_51_ = ebike_postgres_ways_calcula-
tion_10_
ebike_postgres_ways_calculation_11_ = ebike_postgres_ways_calcula-
tion_51_
ebike_postgres_ways_calculation_53_ = ebike_postgres_ways_calcula-
tion_11_

```



```

ebike_postgres_ways_calculation__13_ = ebike_postgres_ways_calcula-
tion__53_
ebike_postgres_ways_calculation__55_ = ebike_postgres_ways_calcula-
tion__13_
ebike_postgres_ways_calculation__12_ = ebike_postgres_ways_calcula-
tion__55_
ebike_postgres_ways_calculation__57_ = ebike_postgres_ways_calcula-
tion__12_
ebike_postgres_ways_calculation__14_ = ebike_postgres_ways_calcula-
tion__57_
ebike_postgres_ways_calculation__59_ = ebike_postgres_ways_calcula-
tion__14_
ebike_postgres_ways_calculation__3_ = ebike_postgres_ways_calcula-
tion__29_
ebike_postgres_ways_calculation__32_ = ebike_postgres_ways_calcula-
tion__3_
ebike_postgres_ways_calculation__4_ = ebike_postgres_ways_calcula-
tion__32_
ebike_postgres_ways_calculation__35_ = ebike_postgres_ways_calcula-
tion__4_
ebike_postgres_ways_calculation__5_ = ebike_postgres_ways_calcula-
tion__35_
ebike_postgres_ways_calculation__39_ = ebike_postgres_ways_calcula-
tion__5_
ebike_postgres_ways_calculation__6_ = ebike_postgres_ways_calcula-
tion__39_
ebike_postgres_ways_calculation__41_ = ebike_postgres_ways_calcula-
tion__6_
ebike_postgres_ways_calculation__7_ = ebike_postgres_ways_calcula-
tion__41_
ebike_postgres_ways_calculation__43_ = ebike_postgres_ways_calcula-
tion__7_
ebike_postgres_ways_calculation__8_ = ebike_postgres_ways_calcula-
tion__43_
ebike_postgres_ways_calculation__46_ = ebike_postgres_ways_calcula-
tion__8_
ebike_postgres_ways_calculation__9_ = ebike_postgres_ways_calcula-
tion__46_

```

#### # Set Geoprocessing environments

```

arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Doku-
mente\\11_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"

```

#### # Process: energyconsumptionsubmodel\_st2\_1

```

arcpy.energyconsumptionsubmodelst21_energy_consumption_model(Scratch,
Connection_to_localhost_sde, "8.485 47.35 8.57 47.42", temp_osm,
osm2pgrouting_exe, mapconfig_for_bicycles_xml, DEM, weight_e_bike_kg,
weight_driver_kg, temperature_celsius, wheel_diameter_inches, gearbox_ef-
ficiency, gradeability_degree, rolling_coefficient, ebike_post-
gres_ways_calculation__17_, "http://www.overpass-api.de/api/xapi_meta?",
auxiliary_components_W, factor_human_torque)

```

#### # Process: energyconsumptionsubmodel\_st2\_2

```

arcpy.energyconsumptionsubmodelst22_energy_consumption_model("5", "35",
"5", ebike_postgres_ways_calculation__17_)

```

#### # Process: Add Field (12)



```
arcpy.AddField_management(ebike_postgres_ways_calculation__16_, "motorefficiencyv5_r", "DOUBLE", "", "", "", "motorefficiencyvelocity5_reverse", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (22)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__31_, "motorefficiencyv5_r", "meff5(!torquev5_r_Nm!)", "PYTHON_9.3", "def meff5(torquev5_r_Nm):\n    if -12.5 >= torquev5_r_Nm:\n        return 0.1686509334044\n    elif -7.5 >= torquev5_r_Nm > -12.5:\n        re- turn 0.400272727088039\n    elif 0 >= torquev5_r_Nm > -7.5:\n        re- turn 0.53365978248219\n    elif 0 < torquev5_r_Nm < 7.5:\n        re- turn 0.623834964702797\n    elif 7.5 <= torquev5_r_Nm < 12.5:\n        re- turn 0.579450836444724\n    elif 12.5 <= torquev5_r_Nm < 17.5:\n        re- turn 0.514837509676393\n    elif 17.5 <= torquev5_r_Nm < 22.5:\n        re- turn 0.472091740648567\n    elif 22.5 <= torquev5_r_Nm < 27.5:\n        re- turn 0.409172192178427\n    elif 27.5 <= torquev5_r_Nm < 32.5:\n        re- turn 0.358790286998961\n    elif 32.5 <= torquev5_r_Nm < 37.5:\n        re- turn 0.32394748442385\n    elif 37.5 <= torquev5_r_Nm:\n        re- turn 0.192834086191931")
```

```
# Process: Add Field (21)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation, "motorefficiencyv10_r", "DOUBLE", "", "", "", "motorefficiencyvelocity10_reverse", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (23)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__49_, "motorefficiencyv10_r", "meff10(!torquev10_r_Nm!)", "PYTHON_9.3", "def meff10(torquev10_r_Nm):\n    if -27.5 >= torquev10_r_Nm:\n        re- turn 0.137684115952318\n    elif -22.5 >= torquev10_r_Nm > -27.5:\n        re- turn 0.312279619894597\n    elif -17.5 >= torquev10_r_Nm > -22.5:\n        re- turn 0.459405191207038\n    elif -12.5 >= torquev10_r_Nm > -17.5:\n        re- turn 0.556210442620491\n    elif -7.5 >= torquev10_r_Nm > -12.5:\n        re- turn 0.660850232379088\n    elif 0 >= torquev10_r_Nm > -7.5:\n        re- turn 0.68784176628815\n    elif 0 < torquev10_r_Nm < 7.5:\n        re- turn 0.715972669192336\n    elif 7.5 <= torquev10_r_Nm < 12.5:\n        re- turn 0.701007821443731\n    elif 12.5 <= torquev10_r_Nm < 17.5:\n        re- turn 0.654257533958745\n    elif 17.5 <= torquev10_r_Nm < 22.5:\n        re- turn 0.622696398091954\n    elif 22.5 <= torquev10_r_Nm < 27.5:\n        re- turn 0.56559006689693\n    elif 27.5 <= torquev10_r_Nm < 32.5:\n        re- turn 0.514923576679544\n    elif 32.5 <= torquev10_r_Nm < 37.5:\n        re- turn 0.476584799195279\n    elif 37.5 <= torquev10_r_Nm:\n        re- turn 0.305739781527573")
```

```
# Process: Add Field (22)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation__10_, "motorefficiencyv15_r", "DOUBLE", "", "", "", "motorefficiencyvelocity15_reverse", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (25)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__51_, "motorefficiencyv15_r", "meff15(!torquev15_r_Nm!)", "PYTHON_9.3", "def meff15(torquev15_r_Nm):\n    if -27.5 >= torquev15_r_Nm:\n        re- turn 0.137684115952318\n    elif -22.5 >= torquev15_r_Nm > -27.5:\n        re- turn 0.312279619894597\n    elif -17.5 >= torquev15_r_Nm > -22.5:\n        re- turn 0.459405191207038\n    elif -12.5 >= torquev15_r_Nm > -17.5:\n        re- turn 0.556210442620491\n    elif -7.5 >= torquev15_r_Nm > -12.5:\n        re- turn 0.660850232379088\n    elif 0 >= torquev15_r_Nm > -7.5:\n        re- turn 0.68784176628815\n    elif 0 < torquev15_r_Nm < 7.5:\n        re- turn 0.715972669192336\n    elif 7.5 <= torquev15_r_Nm < 12.5:\n        re- turn 0.701007821443731\n    elif 12.5 <= torquev15_r_Nm < 17.5:\n        re- turn 0.654257533958745\n    elif 17.5 <= torquev15_r_Nm < 22.5:\n        re- turn 0.622696398091954\n    elif 22.5 <= torquev15_r_Nm < 27.5:\n        re- turn 0.56559006689693\n    elif 27.5 <= torquev15_r_Nm < 32.5:\n        re- turn 0.514923576679544\n    elif 32.5 <= torquev15_r_Nm < 37.5:\n        re- turn 0.476584799195279\n    elif 37.5 <= torquev15_r_Nm:\n        re- turn 0.305739781527573")
```



```

return 0.715972669192336\\n      elif 7.5 <= torquev15_r_Nm < 12.5:\\n
return 0.701007821443731\\n      elif 12.5 <= torquev15_r_Nm < 17.5:\\n
return 0.654257533958745\\n      elif 17.5 <= torquev15_r_Nm < 22.5:\\n
return 0.622696398091954\\n      elif 22.5 <= torquev15_r_Nm < 27.5:\\n
return 0.56559006689693\\n      elif 27.5 <= torquev15_r_Nm < 32.5:\\n
return 0.514923576679544\\n      elif 32.5 <= torquev15_r_Nm < 37.5:\\n
return 0.476584799195279\\n      elif 37.5 <= torquev15_r_Nm:\\n      re-
turn 0.305739781527573")

```

# Process: Add Field (23)

```

arcpy.AddField_management(ebike_postgres_ways_calculation__11_, "motoreff-
ficiencyv20_r", "DOUBLE", "", "", "", "motorefficiencyvelocity20_re-
verse", "NULLABLE", "NON_REQUIRED", "")

```

# Process: Calculate Field (28)

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation__53_,
"motorefficiencyv20_r", "meff20(!torquev20_r_Nm!)", "PYTHON_9.3", "def
meff20(torquev20_r_Nm):\\n      if -37.5 >= torquev20_r_Nm:\\n      re-
turn 0.242985328431138\\n      elif -32.5 >= torquev20_r_Nm > -37.5:\\n
return 0.326487747096875\\n      elif -27.5 >= torquev20_r_Nm > -32.5:\\n
return 0.406298826260984\\n      elif -22.5 >= torquev20_r_Nm > -27.5:\\n
return 0.53371216133064\\n      elif -17.5 >= torquev20_r_Nm > -22.5:\\n
return 0.61970983060857\\n      elif -12.5 >= torquev20_r_Nm > -17.5:\\n
return 0.684670290041407\\n      elif -7.5 >= torquev20_r_Nm > -12.5:\\n
return 0.751571154518068\\n      elif 0 >= torquev20_r_Nm > -7.5:\\n
return 0.722428702257428\\n      elif 0 < torquev20_r_Nm < 7.5:\\n
return 0.749823207947012\\n      elif 7.5 <= torquev20_r_Nm < 12.5:\\n
return 0.756144166249551\\n      elif 12.5 <= torquev20_r_Nm < 17.5:\\n
return 0.727493370155266\\n      elif 17.5 <= torquev20_r_Nm < 22.5:\\n
return 0.704447090103871\\n      elif 22.5 <= torquev20_r_Nm < 27.5:\\n
return 0.653827051136146\\n      elif 27.5 <= torquev20_r_Nm < 32.5:\\n
return 0.610366351999488\\n      elif 32.5 <= torquev20_r_Nm < 37.5:\\n
return 0.570148382769345\\n      elif 37.5 <= torquev20_r_Nm:\\n      re-
turn 0.411535127628699")

```

# Process: Add Field (25)

```

arcpy.AddField_management(ebike_postgres_ways_calculation__13_, "motoreff-
ficiencyv25_r", "DOUBLE", "", "", "", "motorefficiencyvelocity25_re-
verse", "NULLABLE", "NON_REQUIRED", "")

```

# Process: Calculate Field (29)

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation__55_,
"motorefficiencyv25_r", "meff25(!torquev25_r_Nm!)", "PYTHON_9.3", "def
meff25(torquev25_r_Nm):\\n      if -37.5 >= torquev25_r_Nm:\\n      re-
turn 0.409268259437775\\n      elif -32.5 >= torquev25_r_Nm > -37.5:\\n
return 0.481313215219602\\n      elif -27.5 >= torquev25_r_Nm > -32.5:\\n
return 0.54783588540812\\n      elif -22.5 >= torquev25_r_Nm > -27.5:\\n
return 0.641936048794025\\n      elif -17.5 >= torquev25_r_Nm > -22.5:\\n
return 0.707092957349041\\n      elif -12.5 >= torquev25_r_Nm > -17.5:\\n
return 0.749965174434069\\n      elif -7.5 >= torquev25_r_Nm > -12.5:\\n
return 0.793345434388403\\n      elif 0 >= torquev25_r_Nm > -7.5:\\n
return 0.77195594049356\\n      elif 0 < torquev25_r_Nm < 7.5:\\n
return 0.775650893643593\\n      elif 7.5 <= torquev25_r_Nm < 12.5:\\n
return 0.783856109033499\\n      elif 12.5 <= torquev25_r_Nm < 17.5:\\n
return 0.764640691523341\\n      elif 17.5 <= torquev25_r_Nm < 22.5:\\n
return 0.748987636783427\\n      elif 22.5 <= torquev25_r_Nm < 27.5:\\n
return 0.705186831992523\\n      elif 27.5 <= torquev25_r_Nm < 32.5:\\n
return 0.666858058750972\\n      elif 32.5 <= torquev25_r_Nm:\\n      re-
turn 0.640896012607356")

```

```

# Process: Add Field (28)
arcpy.AddField_management(ebike_postgres_ways_calculation_12_, "motorefficiencyv30_r", "DOUBLE", "", "", "", "motorefficiencyvelocity30_reverse", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (30)
arcpy.CalculateField_management(ebike_postgres_ways_calculation_57_, "motorefficiencyv30_r", "meff30(!torquev30_r_Nm!)", "PYTHON_9.3", "def meff30(torquev30_r_Nm):\n    if -37.5 >= torquev30_r_Nm:\n        re-
turn 0.503961290702052\n    elif -32.5 >= torquev30_r_Nm > -37.5:\n
return 0.574071561001055\n    elif -27.5 >= torquev30_r_Nm > -32.5:\n
return 0.62418619075502\n    elif -22.5 >= torquev30_r_Nm > -27.5:\n
return 0.692363882400556\n    elif -17.5 >= torquev30_r_Nm > -22.5:\n
return 0.748143465737966\n    elif -12.5 >= torquev30_r_Nm > -17.5:\n
return 0.785475525219002\n    elif -7.5 >= torquev30_r_Nm > -12.5:\n
return 0.810884982216235\n    elif 0 >= torquev30_r_Nm > -7.5:\n
return 0.767802941434139\n    elif 0 < torquev30_r_Nm < 7.5:\n
return 0.766445865111972\n    elif 7.5 <= torquev30_r_Nm < 12.5:\n
return 0.802113888658795\n    elif 12.5 <= torquev30_r_Nm < 17.5:\n
return 0.787104493659772\n    elif 17.5 <= torquev30_r_Nm < 22.5:\n
return 0.775961036216906\n    elif 22.5 <= torquev30_r_Nm < 27.5:\n
return 0.742906970272747\n    elif 27.5 <= torquev30_r_Nm:\n        re-
turn 0.709593390721885")

# Process: Add Field (29)
arcpy.AddField_management(ebike_postgres_ways_calculation_14_, "motorefficiencyv35_r", "DOUBLE", "", "", "", "motorefficiencyvelocity35_reverse", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (31)
arcpy.CalculateField_management(ebike_postgres_ways_calculation_59_, "motorefficiencyv35_r", "meff35(!torquev35_r_Nm!)", "PYTHON_9.3", "def meff35(torquev35_r_Nm):\n    if -37.5 >= torquev35_r_Nm:\n        re-
turn 0.572518348372519\n    elif -32.5 >= torquev35_r_Nm > -37.5:\n
return 0.628062859898562\n    elif -27.5 >= torquev35_r_Nm > -32.5:\n
return 0.680406586919219\n    elif -22.5 >= torquev35_r_Nm > -27.5:\n
return 0.737424308689985\n    elif -17.5 >= torquev35_r_Nm > -22.5:\n
return 0.781179207736998\n    elif -12.5 >= torquev35_r_Nm > -17.5:\n
return 0.809803596145166\n    elif -7.5 >= torquev35_r_Nm > -12.5:\n
return 0.818725408364537\n    elif 0 >= torquev35_r_Nm > -7.5:\n
return 0.759422563052545\n    elif 0 < torquev35_r_Nm < 7.5:\n
return 0.772907061038415\n    elif 7.5 <= torquev35_r_Nm < 12.5:\n
return 0.820490814499513\n    elif 12.5 <= torquev35_r_Nm < 17.5:\n
return 0.811626316454911\n    elif 17.5 <= torquev35_r_Nm < 22.5:\n
return 0.805890003387917\n    elif 22.5 <= torquev35_r_Nm < 27.5:\n
return 0.773830012847367\n    elif 27.5 <= torquev35_r_Nm:\n        re-
turn 0.756120670347907")

# Process: Add Field (7)
arcpy.AddField_management(ebike_postgres_ways_calculation_16_, "motorefficiencyv5", "DOUBLE", "", "", "", "motorefficiencyvelocity5", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (2)
arcpy.CalculateField_management(ebike_postgres_ways_calculation_29_, "motorefficiencyv5", "meff5(!torquev5_Nm!)", "PYTHON_9.3", "def meff5(torquev5_Nm):\n    if -12.5 >= torquev5_Nm:\n        return
0.1686509334044\n    elif -7.5 >= torquev5_Nm > -12.5:\n        return

```



```

0.400272727088039\\n      elif 0 >= torquev5_Nm > -7.5:\\n          return
0.53365978248219\\n      elif 0 < torquev5_Nm < 7.5:\\n          return
0.623834964702797\\n      elif 7.5 <= torquev5_Nm < 12.5:\\n          return
0.579450836444724\\n      elif 12.5 <= torquev5_Nm < 17.5:\\n          return
0.514837509676393\\n      elif 17.5 <= torquev5_Nm < 22.5:\\n          return
0.472091740648567\\n      elif 22.5 <= torquev5_Nm < 27.5:\\n          return
0.409172192178427\\n      elif 27.5 <= torquev5_Nm < 32.5:\\n          return
0.358790286998961\\n      elif 32.5 <= torquev5_Nm < 37.5:\\n          return
0.32394748442385\\n      elif 37.5 <= torquev5_Nm:\\n          return
0.192834086191931")

```

# Process: Add Field (2)

```

arcpy.AddField_management(ebike_postgres_ways_calculation_3, "motorefficiencyv10", "DOUBLE", "", "", "", "motorefficiencyvelocity10", "NULLABLE", "NON_REQUIRED", "")

```

# Process: Calculate Field (11)

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_32, "motorefficiencyv10", "meff10(!torquev10_Nm!)", "PYTHON_9.3", "def meff10(torquev10_Nm):\\n    if -27.5 >= torquev10_Nm:\\n        return 0.137684115952318\\n    elif -22.5 >= torquev10_Nm > -27.5:\\n        re- turn 0.312279619894597\\n    elif -17.5 >= torquev10_Nm > -22.5:\\n        return 0.459405191207038\\n    elif -12.5 >= torquev10_Nm > -17.5:\\n        return 0.556210442620491\\n    elif -7.5 >= torquev10_Nm > -12.5:\\n        return 0.660850232379088\\n    elif 0 >= torquev10_Nm > -7.5:\\n        return 0.68784176628815\\n    elif 0 < torquev10_Nm < 7.5:\\n        re- turn 0.715972669192336\\n    elif 7.5 <= torquev10_Nm < 12.5:\\n        return 0.701007821443731\\n    elif 12.5 <= torquev10_Nm < 17.5:\\n        return 0.654257533958745\\n    elif 17.5 <= torquev10_Nm < 22.5:\\n        return 0.622696398091954\\n    elif 22.5 <= torquev10_Nm < 27.5:\\n        return 0.56559006689693\\n    elif 27.5 <= torquev10_Nm < 32.5:\\n        return 0.514923576679544\\n    elif 32.5 <= torquev10_Nm < 37.5:\\n        return 0.476584799195279\\n    elif 37.5 <= torquev10_Nm:\\n        re- turn 0.305739781527573")

```

# Process: Add Field (11)

```

arcpy.AddField_management(ebike_postgres_ways_calculation_4, "motorefficiencyv15", "DOUBLE", "", "", "", "motorefficiencyvelocity15", "NULLABLE", "NON_REQUIRED", "")

```

# Process: Calculate Field (14)

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_35, "motorefficiencyv15", "meff15(!torquev15_Nm!)", "PYTHON_9.3", "def meff15(torquev15_Nm):\\n    if -27.5 >= torquev15_Nm:\\n        return 0.137684115952318\\n    elif -22.5 >= torquev15_Nm > -27.5:\\n        re- turn 0.312279619894597\\n    elif -17.5 >= torquev15_Nm > -22.5:\\n        return 0.459405191207038\\n    elif -12.5 >= torquev15_Nm > -17.5:\\n        return 0.556210442620491\\n    elif -7.5 >= torquev15_Nm > -12.5:\\n        return 0.660850232379088\\n    elif 0 >= torquev15_Nm > -7.5:\\n        return 0.68784176628815\\n    elif 0 < torquev15_Nm < 7.5:\\n        re- turn 0.715972669192336\\n    elif 7.5 <= torquev15_Nm < 12.5:\\n        return 0.701007821443731\\n    elif 12.5 <= torquev15_Nm < 17.5:\\n        return 0.654257533958745\\n    elif 17.5 <= torquev15_Nm < 22.5:\\n        return 0.622696398091954\\n    elif 22.5 <= torquev15_Nm < 27.5:\\n        return 0.56559006689693\\n    elif 27.5 <= torquev15_Nm < 32.5:\\n        return 0.514923576679544\\n    elif 32.5 <= torquev15_Nm < 37.5:\\n        return 0.476584799195279\\n    elif 37.5 <= torquev15_Nm:\\n        re- turn 0.305739781527573")

```



```

# Process: Add Field (13)
arcpy.AddField_management(ebike_postgres_ways_calculation__5_, "motorefficiencyv20", "DOUBLE", "", "", "", "motorefficiencyvelocity20", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (16)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__39_, "motorefficiencyv20", "meff20(!torquev20_Nm!)", "PYTHON_9.3", "def meff20(torquev20_Nm):\n    if -37.5 >= torquev20_Nm:\n        return 0.242985328431138\n    elif -32.5 >= torquev20_Nm > -37.5:\n        re- turn 0.326487747096875\n    elif -27.5 >= torquev20_Nm > -32.5:\n        return 0.406298826260984\n    elif -22.5 >= torquev20_Nm > -27.5:\n        return 0.53371216133064\n    elif -17.5 >= torquev20_Nm > -22.5:\n        return 0.61970983060857\n    elif -12.5 >= torquev20_Nm > -17.5:\n        return 0.684670290041407\n    elif -7.5 >= torquev20_Nm > -12.5:\n        return 0.751571154518068\n    elif 0 >= torquev20_Nm > -7.5:\n        return 0.722428702257428\n    elif 0 < torquev20_Nm < 7.5:\n        re- turn 0.749823207947012\n    elif 7.5 <= torquev20_Nm < 12.5:\n        return 0.756144166249551\n    elif 12.5 <= torquev20_Nm < 17.5:\n        return 0.727493370155266\n    elif 17.5 <= torquev20_Nm < 22.5:\n        return 0.704447090103871\n    elif 22.5 <= torquev20_Nm < 27.5:\n        return 0.653827051136146\n    elif 27.5 <= torquev20_Nm < 32.5:\n        return 0.610366351999488\n    elif 32.5 <= torquev20_Nm < 37.5:\n        re- turn 0.570148382769345\n    elif 37.5 <= torquev20_Nm:\n        re- turn 0.411535127628699")

# Process: Add Field (16)
arcpy.AddField_management(ebike_postgres_ways_calculation__6_, "motorefficiencyv25", "DOUBLE", "", "", "", "motorefficiencyvelocity25", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (17)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__41_, "motorefficiencyv25", "meff25(!torquev25_Nm!)", "PYTHON_9.3", "def meff25(torquev25_Nm):\n    if -37.5 >= torquev25_Nm:\n        return 0.409268259437775\n    elif -32.5 >= torquev25_Nm > -37.5:\n        re- turn 0.481313215219602\n    elif -27.5 >= torquev25_Nm > -32.5:\n        return 0.54783588540812\n    elif -22.5 >= torquev25_Nm > -27.5:\n        return 0.641936048794025\n    elif -17.5 >= torquev25_Nm > -22.5:\n        return 0.707092957349041\n    elif -12.5 >= torquev25_Nm > -17.5:\n        return 0.749965174434069\n    elif -7.5 >= torquev25_Nm > -12.5:\n        return 0.793345434388403\n    elif 0 >= torquev25_Nm > -7.5:\n        return 0.77195594049356\n    elif 0 < torquev25_Nm < 7.5:\n        re- turn 0.775650893643593\n    elif 7.5 <= torquev25_Nm < 12.5:\n        return 0.783856109033499\n    elif 12.5 <= torquev25_Nm < 17.5:\n        return 0.764640691523341\n    elif 17.5 <= torquev25_Nm < 22.5:\n        return 0.748987636783427\n    elif 22.5 <= torquev25_Nm < 27.5:\n        return 0.705186831992523\n    elif 27.5 <= torquev25_Nm < 32.5:\n        re- turn 0.666858058750972\n    elif 32.5 <= torquev25_Nm:\n        re- turn 0.640896012607356")

# Process: Add Field (17)
arcpy.AddField_management(ebike_postgres_ways_calculation__7_, "motorefficiencyv30", "DOUBLE", "", "", "", "motorefficiencyvelocity30", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (20)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__43_, "motorefficiencyv30", "meff30(!torquev30_Nm!)", "PYTHON_9.3", "def

```





```

meff30(torquev30_Nm):\n      if -37.5 >= torquev30_Nm:\n          return
0.503961290702052\n      elif -32.5 >= torquev30_Nm > -37.5:\n          re-
turn 0.574071561001055\n      elif -27.5 >= torquev30_Nm > -32.5:\n
return 0.62418619075502\n      elif -22.5 >= torquev30_Nm > -27.5:\n
return 0.692363882400556\n      elif -17.5 >= torquev30_Nm > -22.5:\n
return 0.748143465737966\n      elif -12.5 >= torquev30_Nm > -17.5:\n
return 0.785475525219002\n      elif -7.5 >= torquev30_Nm > -12.5:\n
return 0.810884982216235\n      elif 0 >= torquev30_Nm > -7.5:\n
return 0.767802941434139\n      elif 0 < torquev30_Nm < 7.5:\n          re-
turn 0.766445865111972\n      elif 7.5 <= torquev30_Nm < 12.5:\n
return 0.802113888658795\n      elif 12.5 <= torquev30_Nm < 17.5:\n
return 0.787104493659772\n      elif 17.5 <= torquev30_Nm < 22.5:\n
return 0.775961036216906\n      elif 22.5 <= torquev30_Nm < 27.5:\n
return 0.742906970272747\n      elif 27.5 <= torquev30_Nm:\n          re-
turn 0.709593390721885")

```

```
# Process: Add Field (20)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation__8_, "motoreff-
iciencyv35", "DOUBLE", "", "", "", "motorefficiencyvelocity35", "NULLA-
BLE", "NON_REQUIRED", "")

```

```
# Process: Calculate Field (21)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation__46_,
"motorefficiencyv35", "meff35(!torquev35_Nm!)", "PYTHON_9.3", "def
meff35(torquev35_Nm):\n      if -37.5 >= torquev35_Nm:\n          return
0.572518348372519\n      elif -32.5 >= torquev35_Nm > -37.5:\n          re-
turn 0.628062859898562\n      elif -27.5 >= torquev35_Nm > -32.5:\n
return 0.680406586919219\n      elif -22.5 >= torquev35_Nm > -27.5:\n
return 0.737424308689985\n      elif -17.5 >= torquev35_Nm > -22.5:\n
return 0.781179207736998\n      elif -12.5 >= torquev35_Nm > -17.5:\n
return 0.809803596145166\n      elif -7.5 >= torquev35_Nm > -12.5:\n
return 0.818725408364537\n      elif 0 >= torquev35_Nm > -7.5:\n
return 0.759422563052545\n      elif 0 < torquev35_Nm < 7.5:\n          re-
turn 0.772907061038415\n      elif 7.5 <= torquev35_Nm < 12.5:\n
return 0.820490814499513\n      elif 12.5 <= torquev35_Nm < 17.5:\n
return 0.811626316454911\n      elif 17.5 <= torquev35_Nm < 22.5:\n
return 0.805890003387917\n      elif 22.5 <= torquev35_Nm < 27.5:\n
return 0.773830012847367\n      elif 27.5 <= torquev35_Nm:\n          re-
turn 0.756120670347907")

```

```
# Process: energyconsumptionsubmodel_st2_3
```

```

arcpy.energyconsumptionsubmodelst23_energy_consumption_model("5", "35",
"5", ebike_postgres_ways_calculation__15_)

```

*energyconsumptionsubmodel\_st2\_1.py*

```

# -*- coding: utf-8 -*-
# -----
#
# energyconsumptionsubmodel_st2_1.py
# Created on: 2017-05-29 11:03:27.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionsubmodel_st2_1 <Scratch> <Workspace> <Extent>
<temp_osm> <osm2pgrouting_exe> <mapconfig_for_bicycles_xml> <DEM>
<weight_e_bike_kg> <weight_driver_kg> <temperature_celsius> <wheel_diame-
ter_inches> <gearbox_efficiency> <gradeability_degree> <rolling_coeffi-
cient> <ebike_postgres_ways_calculation__55_> <Download_URL> <auxil-
iary_components_W> <factor_human_torque>
# Description:
# -----
#
# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("C:/Users/sim_h/OneDrive/Dokumente/1_Uni_Wien/16_Mas-
terarbeit/4_Daten/Daten/Workspace/databaseconnectionfile.sde/ebike.post-
gres.energy_consumption_model")

# Script arguments
Scratch = arcpy.GetParameterAsText(0)
if Scratch == '#' or not Scratch:
    Scratch = "C:\\Users\\sim_h\\OneDrive\\Dokumente\\1_Uni_Wien\\16_Mas-
terarbeit\\4_Daten\\Daten\\Workspace\\Scratch" # provide a default value
if unspecified

Workspace = arcpy.GetParameterAsText(1)
if Workspace == '#' or not Workspace:
    Workspace = "Database Connections\\Connection to localhost.sde" #
provide a default value if unspecified

Extent = arcpy.GetParameterAsText(2)
if Extent == '#' or not Extent:
    Extent = "8.485 47.35 8.57 47.42" # provide a default value if un-
specified

temp_osm = arcpy.GetParameterAsText(3)
if temp_osm == '#' or not temp_osm:
    temp_osm = "%Scratch%\\temp.osm" # provide a default value if unspec-
ified

osm2pgrouting_exe = arcpy.GetParameterAsText(4)
if osm2pgrouting_exe == '#' or not osm2pgrouting_exe:
    osm2pgrouting_exe = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\osm2pgrouting.exe" # provide a default value if unspec-
ified

mapconfig_for_bicycles_xml = arcpy.GetParameterAsText(5)
if mapconfig_for_bicycles_xml == '#' or not mapconfig_for_bicycles_xml:
    mapconfig_for_bicycles_xml = "C:\\Program Files\\Post-
greSQL\\9.4\\bin\\mapconfig_for_bicycles.xml" # provide a default value
if unspecified

```



```

DEM = arcpy.GetParameterAsText(6)
if DEM == '#' or not DEM:
    DEM = "%Scratch%\\DEM.tif" # provide a default value if unspecified

weight_e_bike_kg = arcpy.GetParameterAsText(7)
if weight_e_bike_kg == '#' or not weight_e_bike_kg:
    weight_e_bike_kg = "27" # provide a default value if unspecified

weight_driver_kg = arcpy.GetParameterAsText(8)
if weight_driver_kg == '#' or not weight_driver_kg:
    weight_driver_kg = "100" # provide a default value if unspecified

temperature_celsius = arcpy.GetParameterAsText(9)
if temperature_celsius == '#' or not temperature_celsius:
    temperature_celsius = "3" # provide a default value if unspecified

wheel_diameter_inches = arcpy.GetParameterAsText(10)
if wheel_diameter_inches == '#' or not wheel_diameter_inches:
    wheel_diameter_inches = "26" # provide a default value if unspecified

gearbox_efficiency = arcpy.GetParameterAsText(11)
if gearbox_efficiency == '#' or not gearbox_efficiency:
    gearbox_efficiency = "0.98" # provide a default value if unspecified

gradeability_degree = arcpy.GetParameterAsText(12)
if gradeability_degree == '#' or not gradeability_degree:
    gradeability_degree = "15" # provide a default value if unspecified

rolling_coefficient = arcpy.GetParameterAsText(13)
if rolling_coefficient == '#' or not rolling_coefficient:
    rolling_coefficient = "0.003" # provide a default value if unspecified

ebike_postgres_ways_calculation__55_ = arcpy.GetParameterAsText(14)
if ebike_postgres_ways_calculation__55_ == '#' or not ebike_postgres_ways_calculation__55_:
    ebike_postgres_ways_calculation__55_ = "%Workspace%\\ebike.postgres.ways_calculation" # provide a default value if unspecified

Download_URL = arcpy.GetParameterAsText(15)
if Download_URL == '#' or not Download_URL:
    Download_URL = "http://www.overpass-api.de/api/xapi_meta?" # provide a default value if unspecified

auxiliary_components_W = arcpy.GetParameterAsText(16)
if auxiliary_components_W == '#' or not auxiliary_components_W:
    auxiliary_components_W = "8.37" # provide a default value if unspecified

factor_human_torque = arcpy.GetParameterAsText(17)
if factor_human_torque == '#' or not factor_human_torque:
    factor_human_torque = "0.1" # provide a default value if unspecified

# Local variables:
ebike_postgres_ways = "%Workspace%\\ebike.postgres.ways"
successful = "true"
Delete_succeeded__3_ = successful

```

```

ebike_postgres_ways_calculation_2_ = "%Workspace%\ebike.post-
gres.ways_calculation"
ebike_postgres_ways_calculation_4_ = ebike_postgres_ways_calculation_2_
ebike_postgres_ways_calculation_23_ = ebike_postgres_ways_calcula-
tion_2_
ebike_postgres_ways_target = "%Workspace%\ebike.postgres.ways_target"
ebike_postgres_target_for_extract = "Database Connections\Connection to
localhost.sde\ebike.postgres.target_for_extract"
dem_proj = "%Workspace%\ebike.postgres.dem_proj"
ebike_postgres_target_extract = "%Workspace%\ebike.postgres.target_ex-
tract"
ebike_postgres_target_extract_3_ = ebike_postgres_target_extract
ebike_postgres_target_extract_2_ = ebike_postgres_target_extract_3_
ebike_postgres_target_extract_4_ = ebike_postgres_target_extract_2_
ebike_postgres_ways_calculation_36_ = ebike_postgres_ways_calcula-
tion_23_
ebike_postgres_ways_calculation_47_ = ebike_postgres_ways_calcula-
tion_23_
ebike_postgres_ways_source = "%Workspace%\ebike.postgres.ways_source"
ebike_postgres_source_for_extract = "Database Connections\Connection to
localhost.sde\ebike.postgres.source_for_extract"
ebike_postgres_source_extract = "%Workspace%\ebike.postgres.source_ex-
tract"
ebike_postgres_source_extract_3_ = ebike_postgres_source_extract
ebike_postgres_source_extract_2_ = ebike_postgres_source_extract_3_
ebike_postgres_source_extract_4_ = ebike_postgres_source_extract_2_
ebike_postgres_ways_calculation_10_ = ebike_postgres_ways_calcula-
tion_36_
ebike_postgres_ways_calculation_33_ = ebike_postgres_ways_calcula-
tion_10_
ebike_postgres_ways_calculation_46_ = ebike_postgres_ways_calcula-
tion_33_
ebike_postgres_ways_calculation_16_ = ebike_postgres_ways_calcula-
tion_46_
ebike_postgres_ways_calculation_9_ = ebike_postgres_ways_calcula-
tion_16_
ebike_postgres_ways_calculation_8_ = ebike_postgres_ways_calculation_9_
ebike_postgres_ways_calculation_12_ = ebike_postgres_ways_calcula-
tion_8_
ebike_postgres_ways_calculation_13_ = ebike_postgres_ways_calcula-
tion_12_
ebike_postgres_ways_calculation = ebike_postgres_ways_calculation_13_
unsuccessful = "false"
ebike_postgres_ways_calculation_15_ = ebike_postgres_ways_calcula-
tion_47_
ebike_postgres_ways_calculation_39_ = ebike_postgres_ways_calcula-
tion_15_
ebike_postgres_ways_calculation_25_ = ebike_postgres_ways_calcula-
tion_39_
ebike_postgres_ways_calculation_5_ = ebike_postgres_ways_calcula-
tion_25_
ebike_postgres_ways_calculation_11_ = ebike_postgres_ways_calcula-
tion_5_
ebike_postgres_ways_calculation_3_ = ebike_postgres_ways_calcula-
tion_11_
ebike_postgres_ways_calculation_7_ = ebike_postgres_ways_calculation_3_
ebike_postgres_ways_calculation_14_ = ebike_postgres_ways_calcula-
tion_7_
Delete_succeeded = "false"

```



```

Delete_succeeded__2_ = "false"

# Set Geoprocessing environments
arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Dokumente\\1_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"

# Process: Download OSM Data (XAPI)
tempEnvironment0 = arcpy.env.scratchWorkspace
arcpy.env.scratchWorkspace = Scratch
arcpy.XAPIDownload_osmtools(Download_URL, Extent, "*", "highway=primary|primary_link|secondary|tertiary|residential|living_street|track|pedestrian|path|cycleway|footway|byway|unclassified|secondary_link|tertiary_link|lane|track|opposite_lane|opposite_lane|track|opposite_lane|opposite_lane|grade1|grade2|grade3|grade4|grade5|roundabout", temp_osm)
arcpy.env.scratchWorkspace = tempEnvironment0

# Process: osm2pgrouting
arcpy.osm2pgrouting_energy_consumption_model(osm2pgrouting_exe, temp_osm, mapconfig_for_bicycles_xml)

# Process: Copy
arcpy.Copy_management(ebike_postgres_ways, ebike_postgres_ways_calculation__2_, "%Workspace\\ebike.postgres.ways")

# Process: Make XY Event Layer (4)
arcpy.MakeXYEventLayer_management(ebike_postgres_ways_calculation__2_, "x2", "y2", ebike_postgres_ways_target, "GEOGCS['GCS_WGS_1984', DATUM['D_WGS_1984', SPHEROID['WGS_1984', 6378137.0, 298.257223563]], PRIMEM['Greenwich', 0.0], UNIT['Degree', 0.0174532925199433]];-400 -400 1000000000;-100000 10000;-100000 10000;8.98315284119522E-09;0.001;0.001;IsHighPrecision", "")

# Process: Feature Class to Feature Class
arcpy.FeatureClassToFeatureClass_conversion(ebike_postgres_ways_target, Workspace, "target_for_extract", "", "", "", "")

# Process: Project Raster
arcpy.ProjectRaster_management(DEM, dem_proj, "GEOGCS['GCS_WGS_1984', DATUM['D_WGS_1984', SPHEROID['WGS_1984', 6378137.0, 298.257223563]], PRIMEM['Greenwich', 0.0], UNIT['Degree', 0.0174532925199433]]", "NEAREST", "2.21884308916892E-05 2.21884308916892E-05", "CH1903_To_WGS_1984_1", "", "PROJCS['CH1903_LV03', GEOGCS['GCS_CH1903', DATUM['D_CH1903', SPHEROID['Bessel_1841', 6377397.155, 299.1528128]], PRIMEM['Greenwich', 0.0], UNIT['Degree', 0.0174532925199433]], PROJECTION['Hotine_Oblique_Mercator_Azimuth_Center'], PARAMETER['False_Easting', 600000.0], PARAMETER['False_Northing', 200000.0], PARAMETER['Scale_Factor', 1.0], PARAMETER['Azimuth', 90.0], PARAMETER['Longitude_Of_Center', 7.439583333333333], PARAMETER['Latitude_Of_Center', 46.952405555555556], UNIT['Meter', 1.0]], VERTCS['LN_1902', VDATUM['Landesnivellement_1902'], PARAMETER['Vertical_Shift', 0.0], PARAMETER['Direction', 1.0], UNIT['Meter', 1.0]]")

# Process: Extract Values to Points
tempEnvironment0 = arcpy.env.workspace
arcpy.env.workspace = Workspace
arcpy.gp.ExtractValuesToPoints_sa(ebike_postgres_target_for_extract, dem_proj, ebike_postgres_target_extract, "INTERPOLATE", "VALUE_ONLY")

```

```

arcpy.env.workspace = tempEnvironment0

# Process: Add Field (10)
arcpy.AddField_management(ebike_postgres_target_extract, "target_el",
"DOUBLE", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (10)
arcpy.CalculateField_management(ebike_postgres_target_extract__3_, "tar-
get_el", "!rastervalu!", "PYTHON_9.3", "")

# Process: Delete Field (3)
arcpy.DeleteField_management(ebike_postgres_target_extract__2_, "raster-
valu")

# Process: Join Field (5)
arcpy.JoinField_management(ebike_postgres_ways_calculation__2_, "gid",
ebike_postgres_target_extract__4_, "gid", "target_el")

# Process: Make XY Event Layer (3)
arcpy.MakeXYEventLayer_management(ebike_postgres_ways_calculation__2_,
"x1", "y1", ebike_postgres_ways_source, "GEOGCS['GCS_WGS_1984',DA-
TUM['D_WGS_1984',SPHE-
ROID['WGS_1984',6378137.0,298.257223563]],PRIMEM['Green-
wich',0.0],UNIT['Degree',0.0174532925199433]];-400 -400 1000000000;-
100000 10000;-100000 10000;8.98315284119522E-09;0.001;0.001;IsHighPreci-
sion", "")

# Process: Feature Class to Feature Class (2)
arcpy.FeatureClassToFeatureClass_conversion(ebike_postgres_ways_source,
Workspace, "source_for_extract", "", "", "")

# Process: Extract Values to Points (2)
tempEnvironment0 = arcpy.env.workspace
arcpy.env.workspace = Workspace
arcpy.gp.ExtractValuesToPoints_sa(ebike_postgres_source_for_extract,
dem_proj, ebike_postgres_source_extract, "INTERPOLATE", "VALUE_ONLY")
arcpy.env.workspace = tempEnvironment0

# Process: Add Field (11)
arcpy.AddField_management(ebike_postgres_source_extract, "source_el",
"DOUBLE", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (11)
arcpy.CalculateField_management(ebike_postgres_source_extract__3_,
"source_el", "!rastervalu!", "PYTHON_9.3", "")

# Process: Delete Field (2)
arcpy.DeleteField_management(ebike_postgres_source_extract__2_, "raster-
valu")

# Process: Join Field (4)
arcpy.JoinField_management(ebike_postgres_ways_calculation__2_, "gid",
ebike_postgres_source_extract__4_, "gid", "source_el")

# Process: Add Field (12)
arcpy.AddField_management(ebike_postgres_ways_calculation__23_, "slope-
perc", "DOUBLE", "", "", "", "slope_percentage", "NULLABLE", "NON_RE-
QUIRED", "")

```



```

# Process: Calculate Field (12)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__36_,
"slopeperc", "(!target_el!-!source_el!)/!length_m!*100", "PYTHON_9.3",
"")

# Process: Add Field (5)
arcpy.AddField_management(ebike_postgres_ways_calculation__10_, "slopeangle_degree", "DOUBLE", "", "", "", "angle_of_slope_degree", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (5)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__33_,
"slopeangle_degree", "math.degrees(math.atan((!target_el!-!source_el!)/!length_m!))", "PYTHON_9.3", "")

# Process: Add Field (6)
arcpy.AddField_management(ebike_postgres_ways_calculation__46_,
"climbres_N", "DOUBLE", "", "", "", "climbingresistance_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (6)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__16_,
"climbres_N", "(%weight_driver_kg%+%weight_ebike_kg%)*9.806*math.sin(math.radians(!slopeangle_degree!))", "PYTHON_9.3", "")

# Process: Add Field (13)
arcpy.AddField_management(ebike_postgres_ways_calculation__9_,
"rollres_N", "DOUBLE", "", "", "", "rollingresistance_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (14)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__8_,
"rollres_N", "%rolling_coefficient%*(%weight_driver_kg%+%weight_ebike_kg%)*9.806*math.cos(math.radians(!slopeangle_degree!))", "PYTHON_9.3", "")

# Process: Add Field (8)
arcpy.AddField_management(ebike_postgres_ways_calculation__12_, "pressure_hPa", "DOUBLE", "", "", "", "ambientairpressure_hPa", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (8)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__13_,
"pressure_hPa", "1013.25*math.pow(1-(0.0065*(!target_el!+!source_el!)/2))/(%temperature_celsius%+273.15),5.255)", "PYTHON_9.3", "")

# Process: Add Field (25)
arcpy.AddField_management(ebike_postgres_ways_calculation__23_, "slopeperc_r", "DOUBLE", "", "", "", "slope_percentage_reverse", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (25)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__47_,
"slopeperc_r", "(!source_el!-!target_el!)/!length_m!*100", "PYTHON_9.3",
"")

# Process: Add Field (20)

```



```

arcpy.AddField_management(ebike_postgres_ways_calculation__15_, "slopeangle_r_degree", "DOUBLE", "", "", "", "angle_of_slope_reverse_degree", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (20)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__39_, "slopeangle_r_degree", "math.degrees(math.atan(!source_el!-!target_el!)/!length_m!)", "PYTHON_9.3", "")

# Process: Add Field (21)
arcpy.AddField_management(ebike_postgres_ways_calculation__25_, "climbres_r_N", "DOUBLE", "", "", "", "climbingresistance_reverse_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (21)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__5_, "climbres_r_N", "(%weight_driver_kg%+%weight_ebike_kg%)*9.806*math.sin(math.radians(!slopeangle_r_degree!))", "PYTHON_9.3", "")

# Process: Add Field (28)
arcpy.AddField_management(ebike_postgres_ways_calculation__11_, "rollres_r_N", "DOUBLE", "", "", "", "rollingresistance_reverse_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (28)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__3_, "rollres_r_N", "%rolling_coefficient%*(%weight_driver_kg%+%weight_ebike_kg%)*9.806*math.cos(math.radians(!slopeangle_degree!))", "PYTHON_9.3", "")

# Process: Add Field (23)
arcpy.AddField_management(ebike_postgres_ways_calculation__7_, "pressure_r_hPa", "DOUBLE", "", "", "", "ambientairpressure_reverse_hPa", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (23)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__14_, "pressure_r_hPa", "1013.25*math.pow(1-(0.0065*(!source_el!+!target_el!)/2))/(%temperature_celsius%+273.15),5.255)", "PYTHON_9.3", "")

# Process: Delete (3)
arcpy.Delete_management(ebike_postgres_target_for_extract, "")

# Process: Delete (2)
arcpy.Delete_management(ebike_postgres_source_for_extract, "")

# Process: Delete
arcpy.Delete_management(temp_osm, "")

```

*energyconsumptionsubmodel\_st2\_2.py*

```

# -*- coding: utf-8 -*-
# -----
# -----
# energyconsumptionsubmodel_st2_2.py
# Created on: 2017-05-29 11:03:41.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionsubmodel_st2_2 <from_velocity_kmh> <to_velocity_kmh> <by_velocity_kmh> <ebike_postgres_ways_calculation__55_> <ebike_postgres_ways_calculation__2_>
# Description:
# -----
# -----

# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("Model Functions")

# Script arguments
from_velocity_kmh = arcpy.GetParameterAsText(0)
if from_velocity_kmh == '#' or not from_velocity_kmh:
    from_velocity_kmh = "5" # provide a default value if unspecified

to_velocity_kmh = arcpy.GetParameterAsText(1)
if to_velocity_kmh == '#' or not to_velocity_kmh:
    to_velocity_kmh = "35" # provide a default value if unspecified

by_velocity_kmh = arcpy.GetParameterAsText(2)
if by_velocity_kmh == '#' or not by_velocity_kmh:
    by_velocity_kmh = "5" # provide a default value if unspecified

ebike_postgres_ways_calculation__55_ = arcpy.GetParameterAsText(3)
if ebike_postgres_ways_calculation__55_ == '#' or not ebike_postgres_ways_calculation__55_:
    ebike_postgres_ways_calculation__55_ = "Database Connections\\Connection to localhost.sde\\ebike.postgres.ways_calculation" # provide a default value if unspecified

ebike_postgres_ways_calculation__2_ = arcpy.GetParameterAsText(4)
if ebike_postgres_ways_calculation__2_ == '#' or not ebike_postgres_ways_calculation__2_:
    ebike_postgres_ways_calculation__2_ = "Database Connections\\Connection to localhost.sde\\ebike.postgres.ways_calculation" # provide a default value if unspecified

# Local variables:
ebike_postgres_ways_calculation__6_ = ebike_postgres_ways_calculation__55_
ebike_postgres_ways_calculation__35_ = ebike_postgres_ways_calculation__55_
velocity_kmh = from_velocity_kmh
ebike_postgres_ways_calculation = ebike_postgres_ways_calculation__6_
ebike_postgres_ways_calculation__37_ = ebike_postgres_ways_calculation
ebike_postgres_ways_calculation__33_ = ebike_postgres_ways_calculation__37_

```

```

ebike_postgres_ways_calculation__3_ = ebike_postgres_ways_calcula-
tion__33_
ebike_postgres_ways_calculation__29_ = ebike_postgres_ways_calcula-
tion__35_
ebike_postgres_ways_calculation__39_ = ebike_postgres_ways_calcula-
tion__29_
ebike_postgres_ways_calculation__4_ = ebike_postgres_ways_calcula-
tion__39_
ebike_postgres_ways_calculation__5_ = ebike_postgres_ways_calculation__4_
ebike_postgres_ways_calculation__7_ = ebike_postgres_ways_calculation__5_

# Set Geoprocessing environments
arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Doku-
mente\\1_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"

# Process: For
arcpy.IterateCount_mb(from_velocity_kmh, to_velocity_kmh, by_veloc-
ity_kmh)

# Process: Add Field (7)
arcpy.AddField_management(ebike_postgres_ways_calculation__55_,
"dragresv%velocity_kmh%_N", "DOUBLE", "", "", "", "dragresistanceveloc-
ity%velocity_kmh%_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (7)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__6_,
"dragresv%velocity_kmh%_N", "!pressure_hPa!*100/(2*287.058*(%tempera-
ture_celsius%+273.15))*1.15*0.5*math.pow((%velocity_kmh%/3.6),2)", "PY-
THON_9.3", "")

# Process: Add Field
arcpy.AddField_management(ebike_postgres_ways_calculation,
"tracforcev%velocity_kmh%_N", "DOUBLE", "", "", "", "tractiveforceveloc-
ity%velocity_kmh%_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field
arcpy.CalculateField_management(ebike_postgres_ways_calculation__37_,
"tracforcev%velocity_kmh%_N", "!climbres_N!+!rollres_N!+!dragresv%veloc-
ity_kmh%_N!", "PYTHON_9.3", "")

# Process: Add Field (2)
arcpy.AddField_management(ebike_postgres_ways_calculation__33_,
"torquev%velocity_kmh%_Nm", "DOUBLE", "", "", "", "torquevelocity%veloc-
ity_kmh%_Nm", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (2)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__3_,
"torquev%velocity_kmh%_Nm", "fachtor(!tracforcev%velocity_kmh%_N!, %fac-
tor_human_torque%, %wheel_diameter_inches)", "PYTHON_9.3", "def facht-
or(tracforcevvelocity_kmh_N, factor_human_torque, wheel_dia-
meter_inches):\\n    if tracforcevvelocity_kmh_N > 0:\\n        return
(tracforcevvelocity_kmh_N-(tracforcevvelocity_kmh_N*factor_hu-
man_torque))*((wheel_diameter_inches*0.0254)/2)\\n    else:\\n        re-
turn tracforcevvelocity_kmh_N*((wheel_diameter_inches*0.0254)/2)")

# Process: Add Field (22)

```



```

arcpy.AddField_management(ebike_postgres_ways_calculation__55_,
"dragresv%velocity_kmh%_r_N", "DOUBLE", "", "", "", "dragresistanceveloc-
ity%velocity_kmh%_reverse_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (22)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__35_,
"dragresv%velocity_kmh%_r_N", "!pressure_r_hPa!*100/(2*287.058*(%tempera-
ture_celsius%+273.15))*1.15*0.55*math.pow((%velocity_kmh%/3.6),2)", "PY-
THON_9.3", "")

# Process: Add Field (16)
arcpy.AddField_management(ebike_postgres_ways_calculation__29_,
"tracforcev%velocity_kmh%_r_N", "DOUBLE", "", "", "", "tractiveforce-
velocity%velocity_kmh%_reverse_N", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (16)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__39_,
"tracforcev%velocity_kmh%_r_N",
"!climbres_r_N!+!rollres_r_N!+!dragresv%velocity_kmh%_r_N!", "PY-
THON_9.3", "")

# Process: Add Field (17)
arcpy.AddField_management(ebike_postgres_ways_calculation__4_,
"torquev%velocity_kmh%_r_Nm", "DOUBLE", "", "", "", "torquevelocity%ve-
locity_kmh%_reverse_Nm", "NULLABLE", "NON_REQUIRED", "")

# Process: Calculate Field (17)
arcpy.CalculateField_management(ebike_postgres_ways_calculation__5_,
"torquev%velocity_kmh%_r_Nm", "factorr(!tracforcev%velocity_kmh%_r_N!,
%factor_human_torque%, %wheel_diameter_inches%)", "PYTHON_9.3", "def
factorr(tracforcevvelocity_kmh_r_N, factor_human_torque, wheel_diame-
ter_inches):\n    if tracforcevvelocity_kmh_r_N > 0:\n        return
(tracforcevvelocity_kmh_r_N-(tracforcevvelocity_kmh_r_N*factor_hu-
man_torque))*((wheel_diameter_inches*0.0254)/2)\n    else:\n        re-
turn tracforcevvelocity_kmh_r_N*((wheel_diameter_inches*0.0254)/2)")

```

*energyconsumptionsubmodel\_st2\_3.py*

```

# -*- coding: utf-8 -*-
# -----
#
# energyconsumptionsubmodel_st2_3.py
# Created on: 2017-05-29 11:03:55.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: energyconsumptionsubmodel_st2_3 <from_velocity_kmh> <to_velocity_kmh> <by_velocity_kmh> <ebike_postgres_ways_calculation__55_>
# Description:
# -----
#
# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("Model Functions")

# Script arguments
from_velocity_kmh = arcpy.GetParameterAsText(0)
if from_velocity_kmh == '#' or not from_velocity_kmh:
    from_velocity_kmh = "5" # provide a default value if unspecified

to_velocity_kmh = arcpy.GetParameterAsText(1)
if to_velocity_kmh == '#' or not to_velocity_kmh:
    to_velocity_kmh = "35" # provide a default value if unspecified

by_velocity_kmh = arcpy.GetParameterAsText(2)
if by_velocity_kmh == '#' or not by_velocity_kmh:
    by_velocity_kmh = "5" # provide a default value if unspecified

ebike_postgres_ways_calculation__55_ = arcpy.GetParameterAsText(3)
if ebike_postgres_ways_calculation__55_ == '#' or not ebike_postgres_ways_calculation__55_:
    ebike_postgres_ways_calculation__55_ = "Database Connections\\Connection to localhost.sde\\ebike.postgres.ways_calculation" # provide a default value if unspecified

# Local variables:
velocity_kmh = from_velocity_kmh
ebike_postgres_ways_calculation = ebike_postgres_ways_calculation__55_
ebike_postgres_ways_calculation__2_ = ebike_postgres_ways_calculation__55_
ebike_postgres_ways_calculation__27_ = ebike_postgres_ways_calculation__55_
ebike_postgres_ways_calculation__9_ = ebike_postgres_ways_calculation__27_
ebike_postgres_ways_calculation__20_ = ebike_postgres_ways_calculation__27_
ebike_postgres_ways_calculation__11_ = ebike_postgres_ways_calculation__9_
ebike_postgres_ways_calculation__3_ = ebike_postgres_ways_calculation__11_
ebike_postgres_ways_calculation__31_ = ebike_postgres_ways_calculation__3_
ebike_postgres_ways_calculation__24_ = ebike_postgres_ways_calculation__31_

```



```

ebike_postgres_ways_calculation__13_ = ebike_postgres_ways_calcula-
tion__24_
ebike_postgres_ways_calculation__14_ = ebike_postgres_ways_calcula-
tion__20_
ebike_postgres_ways_calculation__4_ = ebike_postgres_ways_calcula-
tion__14_
ebike_postgres_ways_calculation__34_ = ebike_postgres_ways_calcula-
tion__4_
ebike_postgres_ways_calculation__28_ = ebike_postgres_ways_calcula-
tion__34_
ebike_postgres_ways_calculation__25_ = ebike_postgres_ways_calcula-
tion__28_
ebike_postgres_ways_calculation__30_ = ebike_postgres_ways_calcula-
tion__2_
ebike_postgres_ways_calculation__45_ = ebike_postgres_ways_calcula-
tion__30_
ebike_postgres_ways_calculation__38_ = ebike_postgres_ways_calcula-
tion__30_
ebike_postgres_ways_calculation__12_ = ebike_postgres_ways_calcula-
tion__38_
ebike_postgres_ways_calculation__6_ = ebike_postgres_ways_calcula-
tion__12_
ebike_postgres_ways_calculation__23_ = ebike_postgres_ways_calcula-
tion__6_
ebike_postgres_ways_calculation__19_ = ebike_postgres_ways_calcula-
tion__23_
ebike_postgres_ways_calculation__15_ = ebike_postgres_ways_calcula-
tion__19_
ebike_postgres_ways_calculation__26_ = ebike_postgres_ways_calcula-
tion__45_
ebike_postgres_ways_calculation__5_ = ebike_postgres_ways_calcula-
tion__26_
ebike_postgres_ways_calculation__18_ = ebike_postgres_ways_calcula-
tion__5_
ebike_postgres_ways_calculation__17_ = ebike_postgres_ways_calcula-
tion__18_
ebike_postgres_ways_calculation__16_ = ebike_postgres_ways_calcula-
tion__17_

```

```
# Set Geoprocessing environments
```

```
arcpy.env.scratchWorkspace = "C:\\Users\\sim_h\\OneDrive\\Doku-
mente\\11_Uni_Wien\\16_Masterarbeit\\4_Daten\\Daten\\Workspace\\Scratch"
arcpy.env.workspace = "Database Connections\\Connection to localhost.sde"
```

```
# Process: For
```

```
arcpy.IterateCount_mb(from_velocity_kmh, to_velocity_kmh, by_veloc-
ity_kmh)
```

```
# Process: Add Field (3)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation__55_, "angu-
larv%velocity_kmh%_s_inverse", "DOUBLE", "", "", "", "angularwheelveloc-
ity%velocity_kmh%_s_inverse", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (3)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation, "angu-
larv%velocity_kmh%_s_inverse", "(%velocity_kmh%/3.6)/((%wheel_diame-
ter_inches%*0.0254)/2)", "PYTHON_9.3", "")
```

```
# Process: Add Field (4)
```



```
arcpy.AddField_management(ebike_postgres_ways_calculation__27_, "v%velocity_kmh%_rec_W", "DOUBLE", "", "", "", "velocity%velocity_kmh%_recuperation_W", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (4)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__9_, "v%velocity_kmh%_rec_W", "pemrecW(!slopeangle_degree!, %gradeability_degree%, !torquev%velocity_kmh%_Nm!, !angularv%velocity_kmh%_s_inverse!, !motorefficiencyv%velocity_kmh%!, %gearbox_efficiency%, %auxiliary_components_W)", "PYTHON_9.3", "def pemrecW(slopeangle_degree, gradeability_degree, torquevvelocity_kmh_Nm, angularvvelocity_kmh_s_inverse, motorefficiencyvvelocity_kmh, gearbox_efficiency, auxiliary_components_W):\n\n if slopeangle_degree >= gradeability_degree:\n return 999999\n\n elif slopeangle_degree <= -gradeability_degree:\n\n return 999999\n\n elif torquevvelocity_kmh_Nm < 0:\n\n return torquevvelocity_kmh_Nm * angularvvelocity_kmh_s_inverse * motorefficiencyvvelocity_kmh * gearbox_efficiency + auxiliary_components_W\n\n else:\n\n return torquevvelocity_kmh_Nm * angularvvelocity_kmh_s_inverse / (motorefficiencyvvelocity_kmh * gearbox_efficiency) + auxiliary_components_W")
```

```
# Process: Add Field (8)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation__11_, "ctempv%velocity_kmh%_rec_W", "DOUBLE", "", "", "", "capacitiytemperaturevelocity%velocity_kmh%_recuperation_W", "NULLABLE", "REQUIRED", "")
```

```
# Process: Calculate Field (8)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__3_, "ctempv%velocity_kmh%_rec_W", "pemctemprecW(!v%velocity_kmh%_rec_W!, %temperature_celsius%)", "PYTHON_9.3", "def pemctemprecW(vvelocity_kmh_rec_W, temperature_celsius):\n\n if vvelocity_kmh_rec_W == 999999:\n\n return 999999\n\n if temperature_celsius >= 25:\n\n return vvelocity_kmh_rec_W\n\n else:\n\n return vvelocity_kmh_rec_W * (1+((25-temperature_celsius)*0.0047))")
```

```
# Process: Add Field (14)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation__31_, "v%velocity_kmh%_rec_Wh", "DOUBLE", "", "", "", "velocity%velocity_kmh%_recuperation_Wh", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (13)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__24_, "v%velocity_kmh%_rec_Wh", "pemrecWh(!ctempv%velocity_kmh%_rec_W!, !length_m!, %velocity_kmh%)", "PYTHON_9.3", "def pemrecWh(ctempvvelocity_kmh_rec_W, length_m, velocity_kmh):\n\n if ctempvvelocity_kmh_rec_W == 999999:\n\n return 999999\n\n else:\n\n return ctempvvelocity_kmh_rec_W*((length_m/1000)/velocity_kmh)")
```

```
# Process: Add Field (9)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation__27_, "v%velocity_kmh%_norec_W", "DOUBLE", "", "", "", "velocity%velocity_kmh%_no_recuperation_W", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (9)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__20_, "v%velocity_kmh%_norec_W", "pemnorecW(!slopeangle_degree!, %gradeability_degree%, !torquev%velocity_kmh%_Nm!, !angularv%velocity_kmh%_s_inverse!, !motorefficiencyv%velocity_kmh%!, %gearbox_efficiency%, %auxiliary_components_W)", "PYTHON_9.3", "def pemnorecW(slopeangle_degree,
```



```

gradeability_degree, torquevvelocity_kmh_Nm, angularvvelocity_kmh_s_in-
verse, motorefficiencyvvelocity_kmh, gearbox_efficiency, auxiliary_compo-
nents_W):\n\n if slopeangle_degree >= gradeability_degree:\n\n
return 999999\n\n         elif slopeangle_degree <= -gradeability_de-
gree:\n\n         return 999999\n\n         elif torquevvelocity_kmh_Nm <
0:\n\n         return auxiliary_components_W\n\n         else:\n\n         return
torquevvelocity_kmh_Nm * angularvvelocity_kmh_s_inverse / (motorefficien-
cyvvelocity_kmh * gearbox_efficiency) + auxiliary_components_W")

```

```
# Process: Add Field (10)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_14_,
"ctempv%velocity_kmh%_nrec_W", "DOUBLE", "", "", "", "capacitiytempera-
turevelocity%velocity_kmh%_no_recuperation_W", "NULLABLE", "REQUIRED",
"")

```

```
# Process: Calculate Field (10)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_4_,
"ctempv%velocity_kmh%_nrec_W", "pemctempnrecW(!v%veloc-
ity_kmh%_nrec_W!, %temperature_celsius)", "PYTHON_9.3", "def pemctemp-
nrecW(vvelocity_kmh_nrec_W, temperature_celsius):\n\n if vveloc-
ity_kmh_nrec_W == 999999:\n\n         return 999999\n\n         if tempera-
ture_celsius >= 25:\n\n         return vvelocity_kmh_nrec_W\n\n         else:\n\n
return vvelocity_kmh_nrec_W * (1+((25-temperature_celsius)*0.0047))")

```

```
# Process: Add Field (15)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_34_, "v%veloc-
ity_kmh%_nrec_Wh", "DOUBLE", "", "", "", "velocity%velocity_kmh%_no_re-
cuperation_Wh", "NULLABLE", "REQUIRED", "")

```

```
# Process: Calculate Field (15)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_28_,
"v%velocity_kmh%_nrec_Wh", "pemnrecWh(!ctempv%velocity_kmh%_nrec_W!,
!length_m!, %velocity_kmh%)", "PYTHON_9.3", "def pemnrecWh(ctempvveloc-
ity_kmh_nrec_W, length_m, velocity_kmh):\n\n if ctempvveloc-
ity_kmh_nrec_W == 999999:\n\n         return 999999\n\n         else:\n\n
return ctempvvelocity_kmh_nrec_W*((length_m/1000)/velocity_kmh)")

```

```
# Process: Add Field (18)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_55_, "angu-
larv%velocity_kmh%_r_s_inverse", "DOUBLE", "", "", "", "angularwheel-
velocity%velocity_kmh%_reverse_s_inverse", "NULLABLE", "NON_REQUIRED",
"")

```

```
# Process: Calculate Field (18)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_2_, "an-
gularv%velocity_kmh%_r_s_inverse", "(%velocity_kmh%/3.6)/((%wheel_diame-
ter_inches%*0.0254)/2)", "PYTHON_9.3", "")

```

```
# Process: Add Field (24)
```

```

arcpy.AddField_management(ebike_postgres_ways_calculation_30_, "v%veloc-
ity_kmh%_nrec_r_W", "DOUBLE", "", "", "", "velocity%velocity_kmh%_no_re-
cuperation_reverse_W", "NULLABLE", "NON_REQUIRED", "")

```

```
# Process: Calculate Field (24)
```

```

arcpy.CalculateField_management(ebike_postgres_ways_calculation_38_,
"v%velocity_kmh%_nrec_r_W", "pemnrecrW(!slopeangle_r_degree!, %gradea-
bility_degree%, !torquev%velocity_kmh%_r_Nm!, !angularv%veloc-
ity_kmh%_r_s_inverse!, !motorefficiencyv%velocity_kmh%_r!, %gearbox_effi-

```

```
ciency%, %auxiliary_components_W)", "PYTHON_9.3", "def pem-
nocrW(slopeangle_r_degree, gradeability_degree, torquevveloc-
ity_kmh_r_Nm, angularvvelocity_kmh_r_s_inverse, motorefficiencyvveloc-
ity_kmh_r, gearbox_efficiency, auxiliary_components_W):\n if slopean-
gle_r_degree >= gradeability_degree:\n         return 999999\n
elif slopeangle_r_degree <= -gradeability_degree:\n         return
999999\n         elif torquevvelocity_kmh_r_Nm < 0:\n         return auxil-
iary_components_W\n         else:\n         return torquevvelocity_kmh_r_Nm *
angularvvelocity_kmh_r_s_inverse / (motorefficiencyvvelocity_kmh_r *
gearbox_efficiency) + auxiliary_components_W")
```

```
# Process: Add Field (6)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation_12_,
"ctempv%velocity_kmh%_nrec_r_W", "DOUBLE", "", "", "", "capacitiytemper-
aturevelocity%velocity_kmh%_no_recuperation_reverse_W", "NULLABLE", "RE-
QUIRED", "")
```

```
# Process: Calculate Field (6)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation__6_,
"ctempv%velocity_kmh%_nrec_r_W", "pemctempnocrW(!v%veloc-
ity_kmh%_nrec_r_W!, %temperature_celsius)", "PYTHON_9.3", "def pem-
ctempnocrW(vvelocity_kmh_nrec_r_W, temperature_celsius):\n if vve-
locity_kmh_nrec_r_W == 999999:\n         return 999999\n if tempera-
ture_celsius >= 25:\n         return vvelocity_kmh_nrec_r_W\n
else:\n         return vvelocity_kmh_nrec_r_W * (1+((25-temperature_cel-
sius)*0.0047))")
```

```
# Process: Add Field (27)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation_23_, "v%veloc-
ity_kmh%_nrec_r_Wh", "DOUBLE", "", "", "", "velocity%veloc-
ity_kmh%_no_recuperation_reverse_Wh", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (27)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation_19_,
"v%velocity_kmh%_nrec_r_Wh", "pemnocrWh(!ctempv%veloc-
ity_kmh%_nrec_r_W!, !length_m!, %velocity_kmh)", "PYTHON_9.3", "def
pemnocrWh(ctempvvelocity_kmh_nrec_r_W, length_m, velocity_kmh):\n
if ctempvvelocity_kmh_nrec_r_W == 999999:\n         return 999999\n
else:\n         return ctempvvelocity_kmh_nrec_r_W*((length_m/1000)/ve-
locity_kmh)")
```

```
# Process: Add Field (19)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation_30_, "v%veloc-
ity_kmh%_rec_r_W", "DOUBLE", "", "", "", "velocity%velocity_kmh%_recuper-
ation_reverse_W", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (19)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation_45_,
"v%velocity_kmh%_rec_r_W", "pemrecrW(!slopeangle_r_degree!, %gradeabil-
ity_degree%, !torquev%velocity_kmh%_r_Nm!, !angularv%velocity_kmh%_r_s_in-
verse!, !motorefficiencyv%velocity_kmh%_r!, %gearbox_efficiency%, %auxil-
iary_components_W)", "PYTHON_9.3", "def pemrecrW(slopeangle_r_degree,
gradeability_degree, torquevvelocity_kmh_r_Nm, angularvveloc-
ity_kmh_r_s_inverse, motorefficiencyvvelocity_kmh_r, gearbox_efficiency,
auxiliary_components_W):\n if slopeangle_r_degree >= gradeability_de-
gree:\n         return 999999\n         elif slopeangle_r_degree <=
-gradeability_degree:\n         return 999999\n         elif torquevve-
locity_kmh_r_Nm < 0:\n         return torquevvelocity_kmh_r_Nm * angular-
```



```
vvelocity_kmh_r_s_inverse * motorefficiencyvvelocity_kmh_r * gearbox_efficiency + auxiliary_components_W\n    else:\n        return torquevvelocity_kmh_r_Nm * angularvvelocity_kmh_r_s_inverse / (motorefficiencyvvelocity_kmh_r * gearbox_efficiency) +auxiliary_components_W")
```

```
# Process: Add Field (5)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation_26_, "ctempv%velocity_kmh%_rec_r_W", "DOUBLE", "", "", "", "capacitiytemperaturevelocity%velocity_kmh%_recuperation_reverse_W", "NULLABLE", "REQUIRED", "")
```

```
# Process: Calculate Field (5)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation_5_, "ctempv%velocity_kmh%_rec_r_W", "pemctemprecrW(!v%velocity_kmh%_rec_r_W!, %temperature_celsius%)", "PYTHON_9.3", "def pemctemprecrW(vvelocity_kmh_rec_r_W, temperature_celsius):\n    if vvelocity_kmh_rec_r_W == 999999:\n        return 999999\n    if temperature_celsius >= 25:\n        return vvelocity_kmh_rec_r_W\n    else:\n        return vvelocity_kmh_rec_r_W * (1+((25-temperature_celsius)*0.0047))")
```

```
# Process: Add Field (26)
```

```
arcpy.AddField_management(ebike_postgres_ways_calculation_18_, "v%velocity_kmh%_rec_r_Wh", "DOUBLE", "", "", "", "velocity%velocity_kmh%_recuperation_reverse_Wh", "NULLABLE", "NON_REQUIRED", "")
```

```
# Process: Calculate Field (26)
```

```
arcpy.CalculateField_management(ebike_postgres_ways_calculation_17_, "v%velocity_kmh%_rec_r_Wh", "pemrecrWh(!ctempv%velocity_kmh%_rec_r_W!, !length_m!, %velocity_kmh%)", "PYTHON_9.3", "def pemrecrWh(ctempvvelocity_kmh_rec_r_W, length_m, velocity_kmh):\n    if ctempvvelocity_kmh_rec_r_W == 999999:\n        return 999999\n    else:\n        return ctempvvelocity_kmh_rec_r_W*((length_m/1000)/velocity_kmh)")
```

## Programming Code Dijkstra Application

*index\_v20\_norec\_Wh.html*

```
<!DOCTYPE
html>

<html>
<head>

<title>e-Bike Routing</title>

<!-- Inspired by
http://workshop.pgrouting.org/2.1.0-dev/en/index.html. -->

<meta charset="utf-8">
<link href="C:\Users\sim_h\OneDrive\Dokumente\1_Uni_Wien\16_Mas-
terarbeit\5_Applikation\Dijkstra\src\v3.18.2-dist\ol.css" rel="styles-
heet">
<style>
#map {
width: 100%;
height: 500px;
}
</style>
</head>
<body>
<div id="map"></div>
<button id="clear">clear</button>
<script src="C:\Users\sim_h\OneDrive\Dokumente\1_Uni_Wien\16_Mas-
terarbeit\5_Applikation\Dijkstra\src\v3.18.2-dist\ol.js"></script>
<script type="text/javascript">
// The map on which we add all elements.
var map = new ol.Map({
  target: 'map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.OSM()
    })
  ],
  view: new ol.View({
    center: ol.proj.transform([8.54226, 47.37174],
'EPSG:4326', 'EPSG:3857'),
    zoom: 13
```



```

    }),
    controls: ol.control.defaults({
      attributionOptions: {
        collapsible: false
      }
    })
  });
  // Variable that calls the required layer from Geoserver.
  var params = {
    LAYERS: 'pgrouting:v20_norec_wh',
    FORMAT: 'image/png'
  }
  // The "start" and "destination" features.
  var startPoint = new ol.Feature();
  var destPoint = new ol.Feature();
  // The vector layer used to display the "start" and "destination"
  features.
  var vectorLayer = new ol.layer.Vector({
    source: new ol.source.Vector({
      features: [startPoint, destPoint]
    })
  });
  map.addLayer(vectorLayer);

  // A transform function to convert coordinates from EPSG:3857 to
  EPSG:4326.
  var transform = ol.proj.getTransform('EPSG:3857', 'EPSG:4326');
  // Register a map click listener.
  map.on('click', function(event) {
    if (startPoint.getGeometry() == null) {
      /**
       * First click.
       */
      startPoint.setGeometry(new ol.geom.Point(event.coordinate));
    } else if (destPoint.getGeometry() == null) {
      /**
       * Second click.
       */
      destPoint.setGeometry(new ol.geom.Point(event.coordinate));
      /**
       * Transform the coordinates from the map projection
       * (EPSG:3857) to the server projection (EPSG:4326).
       */
    }
  });

```

```

        var startCoord = transform(startPoint.getGeometry().getCoordinates());
        var destCoord = transform(destPoint.getGeometry().getCoordinates());
    /**
     * Retrieval, matching and displaying of the WMS Image
     * from Geoserver
     */
    var viewparams = [
        'x1:' + startCoord[0], 'y1:' + startCoord[1],
        'x2:' + destCoord[0], 'y2:' + destCoord[1]
    ];
    params.viewparams = viewparams.join(';');
    result = new ol.layer.Image({
        source: new ol.source.ImageWMS({
            url: 'http://localhost:8080/geoserver/pgrouting/wms',
            params: params
        })
    });
    map.addLayer(result);
}
});
//Function that removes all elements from the map.
var clearButton = document.getElementById('clear');
clearButton.addEventListener('click', function(event) {
    /**
     * Reset the "start" and "destination" features.
     */
    startPoint.setGeometry(null);
    destPoint.setGeometry(null);
    /**
     * Remove the result layer.
     */
    map.removeLayer(result);
});
</script>
</body>
</html>

```



## Programming Code Bellman-Ford Application

### *Graph.rs*

```

use std;
use std::fs::File;
use std::io::{BufReader, Seek, SeekFrom};
use byteorder::{LittleEndian, ReadBytesExt};
use pbr::ProgressBar;
use spade::RTree;
use cgmath::Vector2;
use postgres::{Connection, TlsMode};
use spatialpoint::SpatialPoint;
// Inspired by http://codegists.com/snippet/rust/bellmanrs_tristrang_rust.
/// Holds a single node, containing the OSM id, longitude, and latitude.
#[derive(Debug)]
struct Node {
    /// The OSM id associated with this node.
    id: u64,
    /// The longitude of this node.
    lon: f32,
    /// The latitude of this node.
    lat: f32,
}
/// Holds a single edge, containing the source node, the target node,
/// and the edge weight.
#[derive(Debug)]
struct Edge {
    /// Where this edge starts.
    source: usize,
    /// Where this edge ends.
    target: usize,
    /// The weight of this edge.
    weight: f32,
}
/// Contains a whole graph.
pub struct Graph {
    /// All the edges contained in the graph.
    edges: Vec<Edge>,
    /// All the nodes contained in this graph.
    nodes: Vec<Node>,
    /// An R tree for quick access to the nodes, given a longitude and latitude.
    rtree: RTree<SpatialPoint, SpatialPoint>
}
/// Implementation of node.

```



```

impl Node {
    /// Reads a node from an OSRM file.
    fn from_osrm(reader: &mut BufReader<&File>) -> Node {
        let lon = reader.read_i32::

```



```

for i in 0..nodes_count {
    let node = Node::from_osrm(&mut reader);
    nodes.push(node);
    if i % 1000 == 0 {
        n_pb.add(1000);
    }
}
// Then, we continue with all edges.
let edges_count = reader.read_u32::<LittleEndian>().unwrap() as usize;
println!("  - Reading {:?} edges", edges_count);
let mut edges = Vec::with_capacity(edges_count);
let mut e_pb = ProgressBar::new(edges_count as u64);
for i in 0..edges_count {
    edges.push(Edge::from_osrm(&mut reader));
    if i % 1000 == 0 {
        e_pb.add(1000);
    }
}
// Finally, we build an R tree for quick access.
let mut rtree = RTree::new();
for n in nodes.iter() {
    let p = SpatialPoint::new(Vector2::new(n.lon, n.lat), n.id);
    rtree.insert(p);
}
Graph { edges: edges, nodes: nodes, rtree: rtree }
}

/// Loads a graph from a Postgres database.
pub fn new_from_db(uname: &String, pw: &String, db: &String,
    ways_vert_table: &String, ways_table: &String,
    weight: &String, weight_rev: &String) -> Graph {
    let conn_str = format!("postgres://{}:{}@localhost/{}", uname, pw, db);
    let conn = Connection::connect(conn_str, TlsMode::None).unwrap();
    let mut nodes = Vec::with_capacity(1000);
    let select_str_vert = format!("SELECT id, lon, lat FROM {} ORDER BY id",
ways_vert_table);
    for row in &conn.query(&select_str_vert, &[]).unwrap() {
        let osm_id: i64 = row.get(0);
        let lon_raw: f64 = row.get(1);
        let lat_raw: f64 = row.get(2);
        let node = Node {
            id: osm_id as u64,
            lon: lon_raw as f32,
            lat: lat_raw as f32
        };
    }
};

```

```

        nodes.push(node);
    }
    let mut edges = Vec::with_capacity(1000);
    let select_str = format!("SELECT source, target, {}, {} FROM {}", weight,
weight_rev, ways_table);
    for row in &conn.query(&select_str, &[]).unwrap() {
        let source_id: i64 = row.get(0);
        let target_id: i64 = row.get(1);
        let weight_raw: f64 = row.get(2);
        let weight_raw_rev: f64 = row.get(3);
        // When inserting, we simply subtract 1, so that the IDs map to those of
        // the nodes.
        // This comes from the fact that the Rust vector is 0-indexed, but in
        // Postgres,
        // the IDs start with 1.
        let edge = Edge {
            source: source_id as usize - 1,
            target: target_id as usize - 1,
            weight: weight_raw as f32
        };
        edges.push(edge);
        // We also insert edges for every backward edge.
        let edge = Edge {
            source: target_id as usize - 1,
            target: source_id as usize - 1,
            weight: weight_raw_rev as f32
        };
        edges.push(edge);
    }
    // Finally, we build an R tree for quick access.
    let mut rtree = RTree::new();
    for n in nodes.iter() {
        let p = SpatialPoint::new(Vector2::new(n.lon, n.lat), n.id);
        rtree.insert(p);
    }
    Graph { edges: edges, nodes: nodes, rtree: rtree }
}

/// Gets the node IDs from a longitude and latitude.
pub fn get_id_from_lon_lat(&self, lon: f32, lat: f32) -> u64 {
    let nearest = self.rtree.nearest_neighbor(&Vector2::new(lon, lat)).unwrap();
    nearest.id
}

/// Gets the internal ID from an OSM id.
fn get_id_from_osm(&self, osm_id: usize) -> usize {

```



```

        self.nodes.iter().position(|r| r.id == osm_id as u64).unwrap()
    }
    /// Gets the location from an internal id. Returns a vector containing
    /// longitude and latitude.
    fn get_loc_from_id(&self, id: usize) -> Vec<f32> {
        vec![self.nodes[id].lon, self.nodes[id].lat]
    }
    /// Performs a routing request from source to target.
    pub fn route(&self, source: usize, target: usize) -> (Vec<Vec<f32>>, f32) {
        let source_id = self.get_id_from_osm(source);
        let target_id = self.get_id_from_osm(target);
        let (pred, dist) = self.bellman(source_id);
        let max_length = self.edges.len();
        println!(" ← Backtracking from {}, having {} edges. Total cost: {}. ",
            target_id, max_length, dist[target_id]);
        let mut trace = Vec::new();
        let mut current_node = target_id;
        trace.push(self.get_loc_from_id(current_node));
        let mut count = 0;
        while current_node != source_id {
            current_node = pred[current_node];
            trace.push(self.get_loc_from_id(current_node));
            count = count + 1;
            // Make sure this doesn't run forever.
            if count > max_length {
                current_node = source_id;
            }
        }
        (trace, dist[target_id])
    }
    /// Computes the reachability of all nodes in the graph, and returns those which
    /// are reachable. Returns a vector of vectors, where the coordinates are as
    /// follows:
    /// longitude, latitude, remaining_energy.
    pub fn reachability(&self, source: usize, capacity: f32) -> Vec<Vec<f32>> {
        let source_id = self.get_id_from_osm(source);
        let (pred, dist) = self.bellman(source_id);
        let max_length = self.nodes.len();
        println!(" ← Assessing all {} nodes to select feasible ones.", max_length);
        let mut trace = Vec::new();
        for (i, node) in dist.iter().enumerate() {
            if capacity - node >= 0.0 {
                let mut loc = self.get_loc_from_id(i);
                loc.push(capacity - node);
            }
        }
    }

```

```

        trace.push(loc);
    }
}
trace
}
// Runs the Bellman Ford algorithm on the graph. Returns a tuple, containing a
// vector of
// predecessors and a vector of distances to the source node.
fn bellman(&self, source: usize) -> (Vec<usize>, Vec<f32>) {
    let nodes_count = self.nodes.len();
    let max_length = self.edges.len();
    println!(" ← Starting from {}, having {} nodes.", source, nodes_count);
    let mut pred = (0..nodes_count).collect::<Vec<_>>();
    let mut dist = std::iter::repeat(std::f32::MAX).take(nodes_count).col-
lect::<Vec<_>>();
    dist[source] = 0.0;
    let mut count = 0;
    let mut improvement = true;
    while improvement {
        improvement = false;
        for edge in &self.edges {
            let source_dist = dist[edge.source];
            let target_dist = dist[edge.target];
            if source_dist != std::f32::MAX && source_dist + edge.weight < tar-
get_dist {
                dist[edge.target] = source_dist + edge.weight;
                pred[edge.target] = edge.source;
                improvement = true;
            }
            // This would be needed for undirected edges, as we'd have to follow
            // every
            // edge both ways in that case.
            // if target_dist != std::f32::MAX && target_dist + edge.weight <
            // source_dist {
            //     dist[edge.source] = target_dist + edge.weight;
            //     pred[edge.source] = edge.target;
            //     improvement = true;
            // }
        }
        count = count + 1;
        // Make sure this doesn't run forever.
        if count > max_length {
            improvement = false;
        }
    }
}

```



```

    }
    println!(" ← Bellman iterations: {}", count);
    (pred, dist)
  }
}

```

### *Spatialpoints.rs*

```

use cgmath::Vector2;
use num::zero;
use spade::SpatialObject;
use spade::BoundingRect;
/// A spatial point, to be stored in an R tree from the spade crate.
#[derive(Debug)]
pub struct SpatialPoint {
    /// The point's coordinates.
    pub center: Vector2<f32>,
    /// The associated OSM id.
    pub id: u64,
}
impl SpatialPoint {
    /// Create a new point.
    pub fn new(center: Vector2<f32>, id: u64) -> SpatialPoint {
        SpatialPoint {
            center: center,
            id: id,
        }
    }
}
impl SpatialObject for SpatialPoint {
    type Vector = Vector2<f32>;
    fn mbr(&self) -> BoundingRect<Vector2<f32>> {
        BoundingRect::from_corners(&(self.center.clone()), &(self.center.clone()))
    }
    fn distance2(&self, point: &Vector2<f32>) -> f32 {
        let dx = self.center[0] - point[0];
        let dy = self.center[1] - point[1];
        let dist = (dx * dx + dy * dy).sqrt().max(zero());
        dist * dist
    }
    // Nothing is contained within a point.
    fn contains(&self, point: &Vector2<f32>) -> bool {
        false
    }
}

```

*Main.rs*

```

extern crate byteorder;
extern crate time;
extern crate pbr;
extern crate iron;
extern crate params;
extern crate router;
extern crate mount;
extern crate persistent;
extern crate cgmsh;
extern crate spade;
extern crate num;
extern crate staticfile;
extern crate postgres;
extern crate geojson;
extern crate rustc_serialize;
use std::env;
use iron::prelude::*;
use router::Router;
use mount::Mount;
use persistent::Read;
use staticfile::Static;
use std::path::Path;
mod graph;
mod spatialpoint;
mod endpoints;
use graph::Graph;
use endpoints::GraphPool;
/// Main function and entry point to the program.
fn main() {
    let args: Vec<_> = env::args().collect();
    let start = time::now();
    // Loading the graph data.
    println!("Loading the data");
    // let graph = Graph::new(&args[1]);
    let graph = Graph::new_from_db(&args[1], &args[2], &args[3], &args[4], &args[5],
                                  &args[6], &args[7]);
    println!("    duration: {}s\n", (time::now() - start).num_seconds());
    // Setting up the router for the web server.
    let mut router = Router::new();
    router.get("/route", endpoints::route_lat_lon, "route");
    router.get("/route-using-ids", endpoints::route_ids, "routeIds");
    router.get("/reachability", endpoints::reachability, "reachability");
    let mut mount = Mount::new();

```





```

mount.mount("/api", router);
mount.mount("/", Static::new(Path::new("./src/static/")));
let mut chain = Chain::new(mount);
chain.link_before(Read::<GraphPool>::one(graph));
Iron::new(chain).http("127.0.0.1:9000").unwrap();
}

```

### Endpoint.rs

```

extern crate time;
extern crate iron;
extern crate geojson;
use iron::prelude::*;
use iron::typemap::Key;
use persistent::Read;
use graph::Graph;
use std::collections::BTreeMap;
use rustc_serialize::json::ToJson;
use geojson::{Feature, FeatureCollection, GeoJson, Geometry};
// A pool that abstracts over the graph, and makes it available to all requests.
pub struct GraphPool;
impl Key for GraphPool { type Value = Graph; }
// Transforms the result of a route calculation into a GeoJSON, convenient for sending
// over the Internet.
fn route_res_to_geojson(lat_lons: Vec<Vec<f32>>, cost: f32) -> String {
    let geometry = Geometry::new(
        geojson::Value::LineString(lat_lons.iter().map(|x|
            x.iter().map(|&y| y as f64).collect::<Vec<_>>()
        ).collect::<Vec<_>>()
    );
    let mut properties = BTreeMap::new();
    properties.insert(
        String::from("total_cost"),
        cost.to_json(),
    );
    let geojson = GeoJson::Feature(Feature {
        crs: None,
        bbox: None,
        geometry: Some(geometry),
        id: None,
        properties: Some(properties),
    });
    geojson.to_string()
}

```

```

/// Transforms the result of a reachability calculation to a GeoJSON string, ready
/// to be processed in the frontend.
fn reachability_res_to_geojson(lat_lon_caps: Vec<Vec<f32>>) -> String {
    let mut features = Vec::new();
    for lat_lon in lat_lon_caps {
        let mut props = BTreeMap::new();
        props.insert(
            String::from("capacity_remaining"),
            lat_lon[2].to_json(),
        );
        features.push(Feature {
            crs: None,
            bbox: None,
            geometry: Some(Geometry::new(
                geojson::Value::Point(lat_lon[0..2].iter().map(|&y|
                    y as f64).collect:::<Vec<_>())
            )),
            id: None,
            properties: Some(props)
        });
    }
    let geojson = GeoJson::FeatureCollection(FeatureCollection {
        crs: None,
        bbox: None,
        features: features,
    });
    geojson.to_string()
}

/// Computes a route, given a start and end latitude and longitude.
pub fn route_lat_lon(req: &mut Request) -> IronResult<Response> {
    let graph = req.get:::<Read<GraphPool>>().unwrap();
    use params::{Params, Value};
    let map = req.get_ref:::<Params>().unwrap();
    match (map.find(&["source-lon"]), map.find(&["source-lat"]),
        map.find(&["target-lon"]), map.find(&["target-lat"])) {
        (Some(&Value::String(ref source_lon)), Some(&Value::String(ref source_lat)),
         Some(&Value::String(ref target_lon)), Some(&Value::String(ref target_lat)))
    => {
        let bellman_start = time::now();
        println!("Starting Bellman-Ford ...");
        let source_id = graph.get_id_from_lon_lat(source_lon.parse:::<f32>().un-
wrap(),
  source_lat.parse:::<f32>().un-
wrap());

```



```

        let target_id = graph.get_id_from_lon_lat(target_lon.parse::().un-
wrap(),
  target_lat.parse::().un-
wrap());
        let res = graph.route(source_id as usize, target_id as usize);
        println!("  ↳ duration: {}s\n", (time::now() - bellman_start).num_sec-
onds());
        Ok(Response::with((iron::status::Ok, route_res_to_geojson(res.0, res.1))))
    },
    _ => Ok(Response::with(iron::status::NotFound))
}
}
}
// Computes a route, given a start and end OSM ID.
pub fn route_ids(req: &mut Request) -> IronResult<Response> {
    let graph = req.get::().unwrap() as usize,
                                target_id.parse::().unwrap() as usize);
        println!("  ↳ duration: {}s\n", (time::now() - bellman_start).num_sec-
onds());
        Ok(Response::with((iron::status::Ok, route_res_to_geojson(res.0, res.1))))
    },
    _ => Ok(Response::with(iron::status::NotFound))
    }
}
}
// Returns all reachable nodes in a vicinity. This can be a lot, so take care!
pub fn reachability(req: &mut Request) -> IronResult<Response> {
    let graph = req.get::().un-
wrap(),

```

```

source_lat.parse::

```

### Index.html

```

<!DOCTYPE                                     html>
<html>
<head>
  <title>e-Bike Routing</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="shortcut icon" type="image/x-icon" href="docs/images/favicon.ico" />
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.0.2/dist/leaflet.css" />
  <script src="https://unpkg.com/leaflet@1.0.2/dist/leaflet.js"></script>
  <script
src="https://code.jquery.com/jquery-3.1.1.min.js"
integrity="sha256-hVVnYaiADRT02PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
crossorigin="anonymous"></script>
  <script src="https://npmcdn.com/@turf/turf@3.5.1/turf.js"></script>
  <style media="screen">
html * {
  font-size: 1em !important;
  color: #000 !important;
  font-family: Arial !important;
}
.ebike-overlay {
  position: absolute;
  right: 20px;
  top: 20px;
  z-index: 999;
}
#cost-box {
  background-color: white;
  padding: 16px 24px;
  -webkit-box-shadow: 0px 0px 5px 0px rgba(0,0,0,0.5);

```



```

        -moz-box-shadow: 0px 0px 5px 0px rgba(0,0,0,0.5);
        box-shadow: 0px 0px 5px 0px rgba(0,0,0,0.5);
        border-radius: 3px;
    }
</style>
</head>
<body style="width: 100%; height: 100%; position: absolute; margin: 0;">
    <div class="ebike-overlay">
        <div id="cost-box">
            Select method:
            <form id="method-form" action="">
                <input type="radio" id="method-route" name="method"
value="route" checked><label for="method-route">Route</label><br>
                <input type="radio" id="method-reachability" name="method"
value="reachability"><label for="method-reachability">Reachability</label>
            </form>
            <label for="capacity">Battery Capacity: </label><input type="number"
id="capacity" name="capacity" value="50.0">
            <br><br>
            <span id="cost-box-explanation">Please compute a route by clicking on the
map!</span>
        </div>
    </div>
    <div id="mapid" style="width: 100%; height: 100%;"></div>
    <script>
// The map on which we add all elements.
var mymap = L.map('mapid').setView([47.3673, 8.55], 13);
var mymapelements = [];
/**
 * Function that removes all elements from the map.
 */
function removeAllMapElements() {
    for(i = 0; i < mymapelements.length; i++) {
        mymap.removeLayer(mymapelements[i]);
    }
    mymapelements = [];
}
/**
 * Function that adds an element to the map.
 */
function addToMap(element) {
    mymapelements.push(element);
    element.addTo(mymap);
}
    </script>

```

```

/**
 * Helper function to convert hsv color ramps to rgb.
 */
var hsv2rgb = function(hsv) {
    var h = hsv.hue, s = hsv.sat, v = hsv.val;
    var rgb, i, data = [];
    if (s === 0) {
        rgb = [v,v,v];
    } else {
        h = h / 60;
        i = Math.floor(h);
        data = [v*(1-s), v*(1-s*(h-i)), v*(1-s*(1-(h-i)))]];
        switch(i) {
            case 0:
                rgb = [v, data[2], data[0]];
                break;
            case 1:
                rgb = [data[1], v, data[0]];
                break;
            case 2:
                rgb = [data[0], v, data[2]];
                break;
            case 3:
                rgb = [data[0], data[1], v];
                break;
            case 4:
                rgb = [data[2], data[0], v];
                break;
            default:
                rgb = [v, data[0], data[1]];
                break;
        }
    }
    return '#' + rgb.map(function(x){
        return ("0" + Math.round(x*255).toString(16)).slice(-2);
    }).join('');
};

L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_to-
ken=pk.eyJ1IjoibWFWYm94IiwiaSI6ImNpandmbXliNDdjZDdm2x6bDk3c2ZtOTkifQ._QA7i5Mpkd_m30IGE1Hziw', {
    maxZoom: 18,
    attribution: 'Map data &copy; <a href="http://openstreetmap.org">Open-
StreetMap</a> contributors, ' +
        '<a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +
        '<a href="http://mapbox.com">Mapbox</a>',

```



```

        id: 'mapbox.streets'
    }).addTo(mymap);
    var clickCount = 0;
    var start = [];
    var end = [];
    // Specifies what happens when someone clicks on the map.
    function onMapClick(e) {
        // Either, we are in routing mode, where we simply compute and display
        // routes.
        if ($('#input[name=method]:checked', '#method-form').val() == "route") {
            if (clickCount == 0) {
                removeAllMapElements();
                start = [e.latlng.lng, e.latlng.lat];
                addToMap(L.marker([e.latlng.lat, e.latlng.lng]));
                clickCount = 1;
            } else {
                end = [e.latlng.lng, e.latlng.lat];
                addToMap(L.marker([e.latlng.lat, e.latlng.lng]));
                $.get("http://127.0.0.1:9000/api/route?source-lon=" + start[0] +
                    "&source-lat=" + start[1] +
                    "&target-lon=" + end[0] + "&target-lat=" + end[1], function(data)
                {
                    geoJson = JSON.parse(data);
                    var geoJSONStyle = {
                        color: 'red',
                        weight: 3,
                        opacity: 0.5,
                        smoothFactor: 1
                    };
                    addToMap(L.geoJSON(geoJson, { style: geoJSONStyle }));
                    $('#cost-box-explanation').text("Energy cost of route: "
                    + geoJson.properties.total_cost);
                });
                clickCount = 0;
            }
            // Or we are in the reachability mode, where we draw a contour plot
            // of reachable nodes.
        } else if ($('#input[name=method]:checked', '#method-form').val() == "reachabil-
        ity") {
            removeAllMapElements();
            start = [e.latlng.lng, e.latlng.lat];
            addToMap(L.marker([e.latlng.lat, e.latlng.lng]));
            var capacity = $('#capacity').val();
            $.get("http://127.0.0.1:9000/api/reachability?source-lon=" + start[0] +

```



```

"&source-lat=" + start[1] +
"&capacity=" + capacity, function(data) {
    geoJson = JSON.parse(data);
    // Determine max and min capacity, used for coloring later.
    var maxCapacity = 0;
    var minCapacity = Infinity;
    for (i in geoJson.features) {
        var feature = geoJson.features[i];
        if (feature.properties.capacity_remaining > maxCapacity)
            maxCapacity = feature.properties.capacity_re-
maining;
        if (feature.properties.capacity_remaining < minCapacity)
            minCapacity = feature.properties.capacity_re-
maining;
    }
    // Create 10 equally spaced breaks.
    var breaks = Array.apply(null, Array(10)).map(function (_, i)
{return (minCapacity + i) * (maxCapacity - minCapacity) / 10;});
    var resolution = 50;
    var isobands = turf.isolines(geoJson, 'capacity_remaining', res-
olution, breaks);
    isobands.features.forEach(function (feature) {
        var cap_diff = 100 - 100 * (feature.properties.capac-
ity_remaining - minCapacity) / (maxCapacity - minCapacity);
        var h = Math.floor((100 - cap_diff) * 120 / 100);
        var s = 1; //Math.abs(cap_diff - 50) / 50;
        var v = 1;
        feature.properties["stroke"] = hsv2rgb({hue: h, sat: s,
val: v});
        feature.properties["stroke-width"] = 10;
        feature.properties["stroke-opacity"] = .5;
    });
    addToMap(L.geoJSON(isobands, {
        style: function(feature) {
            return {
                color: feature.properties['stroke'],
                width: feature.properties['stroke-
width'],
                opacity: feature.properties['opacity']
            };
        }
    }));

```

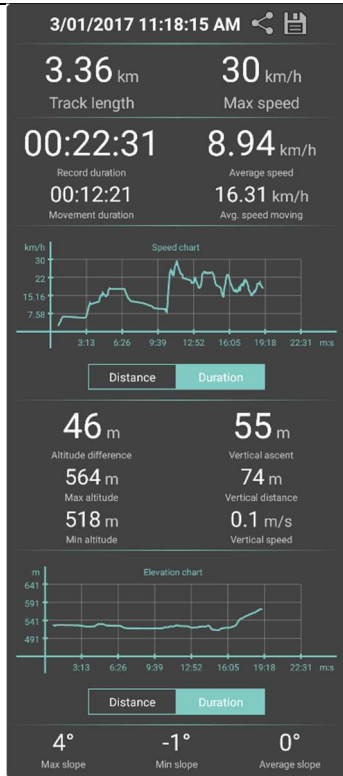
```
        }
    });
});
}
}
mymap.on('click', onMapClick);
</script>

</body>
</html>
```

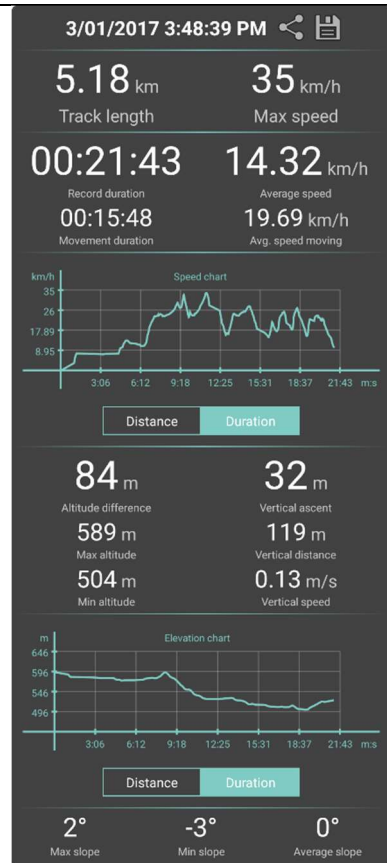
## Test Session A

03.01.2017

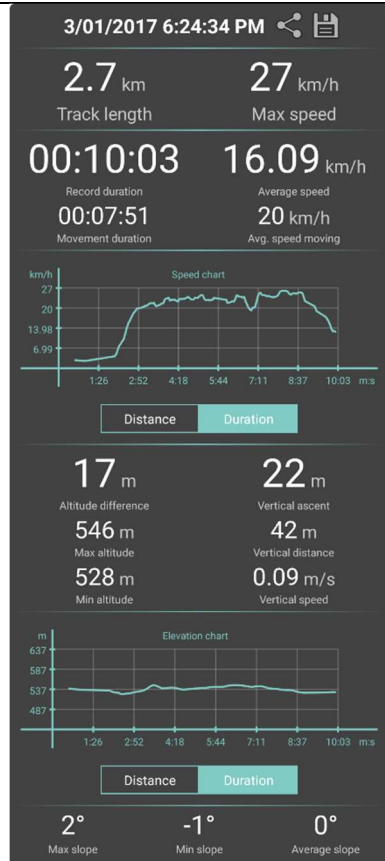
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	15
	Temperature [° C]	3
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	73.91877696
	Measured [Wh]	71



<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	46.35903523
	Measured [Wh]	49



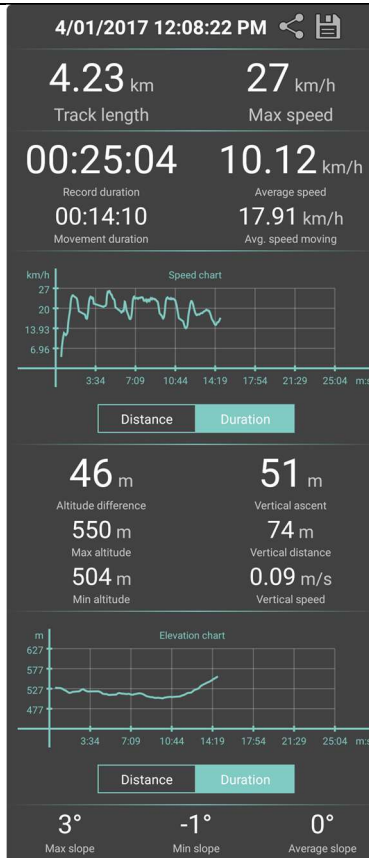
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	39.96724041
	Measured [Wh]	40



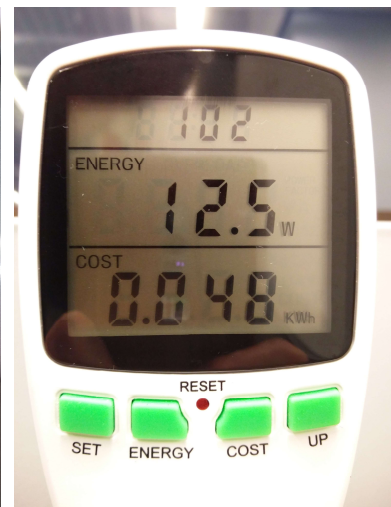
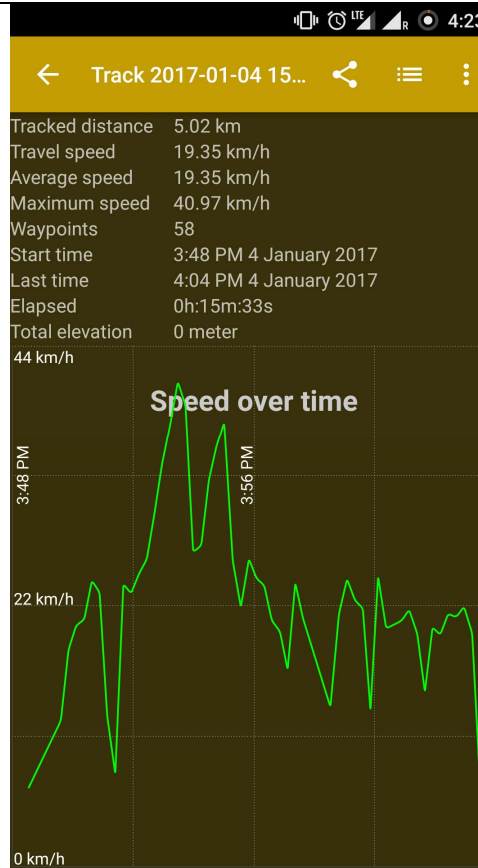
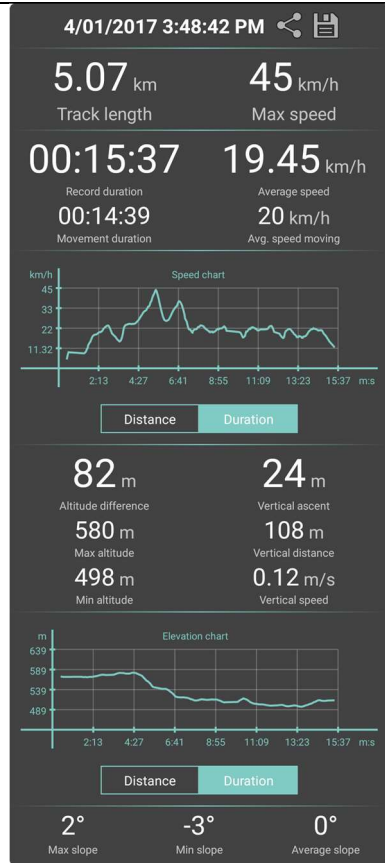


04.01.2017

<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	85.91187674
	Measured [Wh]	85

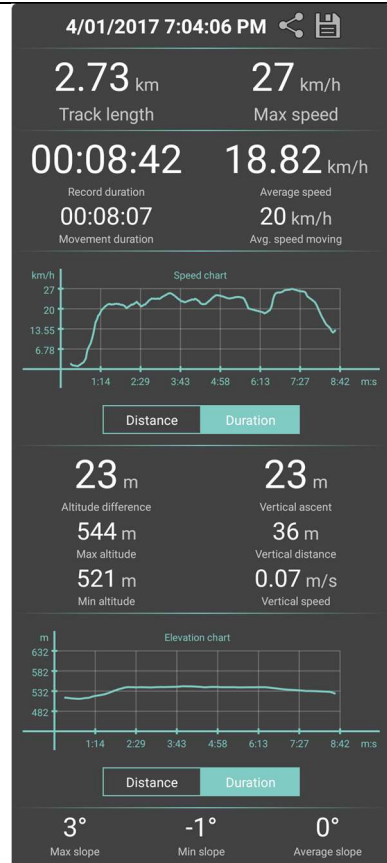


<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	ETH Höggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	46.35903523
	Measured [Wh]	48



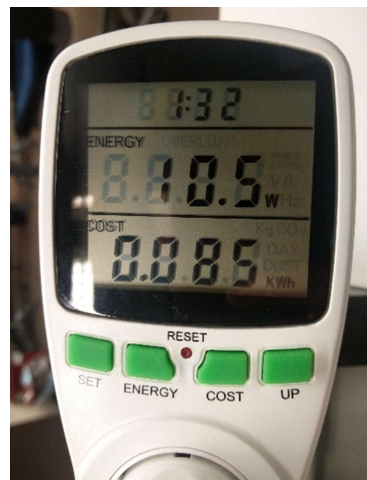
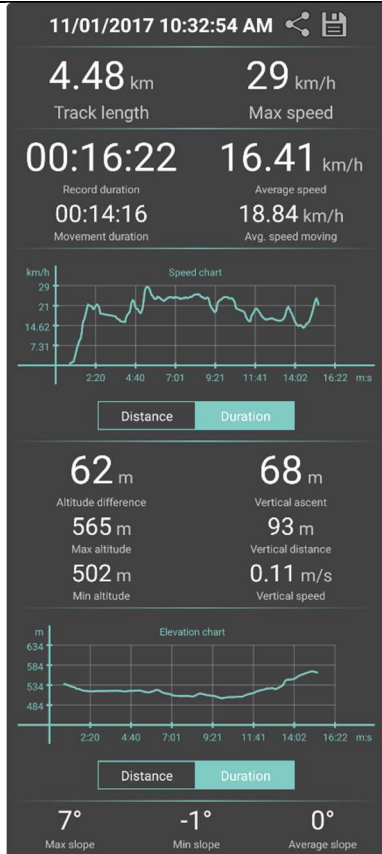


<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	39.96724041
	Measured [Wh]	41

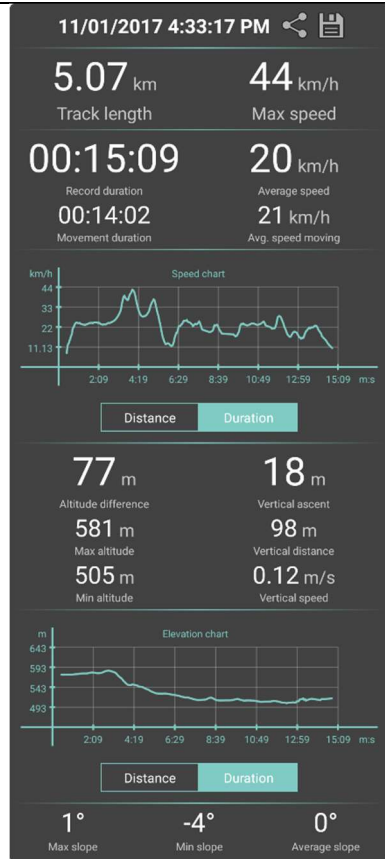


11.01.2017

<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	85.91187674
	Measured [Wh]	85



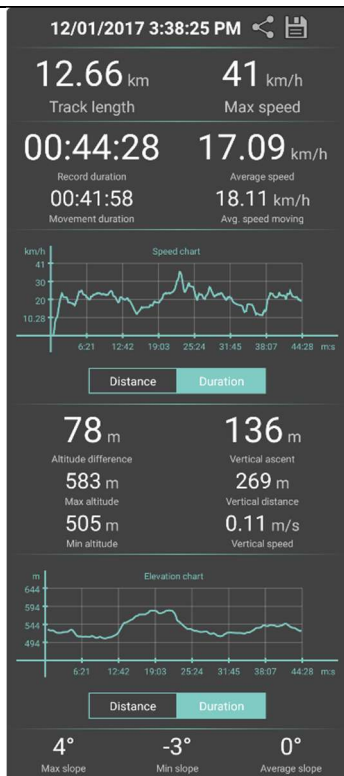
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	3
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	46.35903523
	Measured [Wh]	54





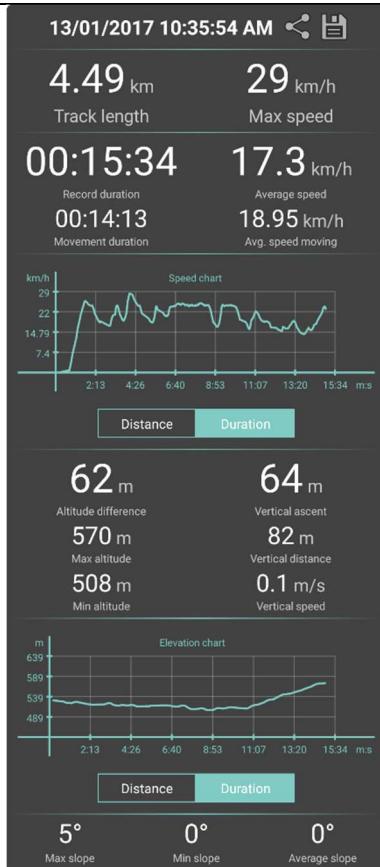
12.01.2017

<b>Model Parameter</b>	Weight [kg]	78
	Velocity [km/h]	20
	Temperature [° C]	0
<b>Track</b>	Source	Bülachhof (- ETH Höggerberg)
	Target	(- ETH Center -) Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	135.16504582
	Measured [Wh]	111

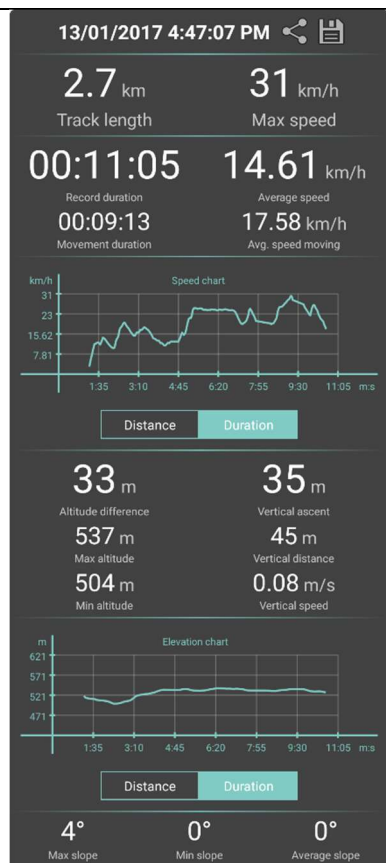


13.01.2017

<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	0
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	87.31076172
	Measured [Wh]	89



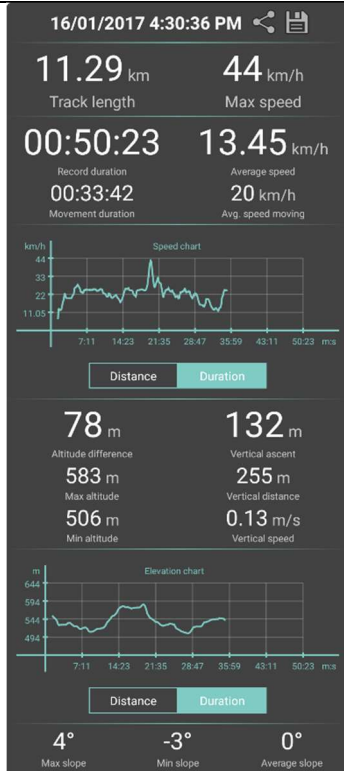
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	0
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	40.62863265
	Measured [Wh]	42





16.01.2017

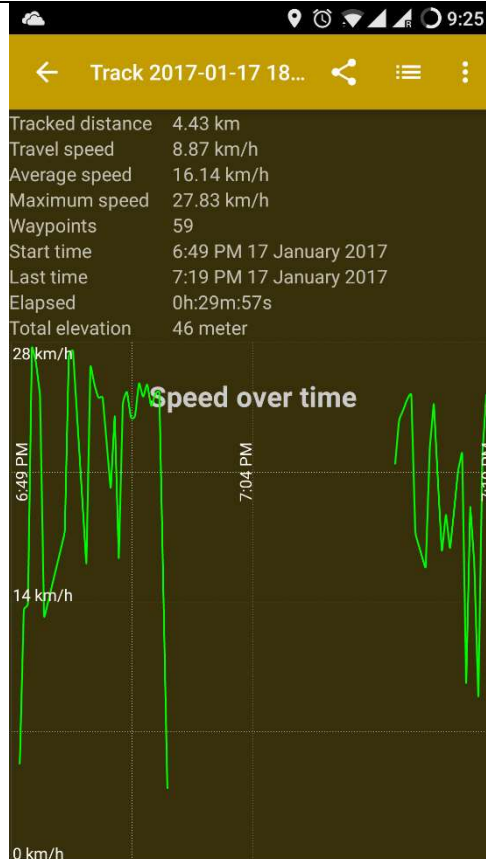
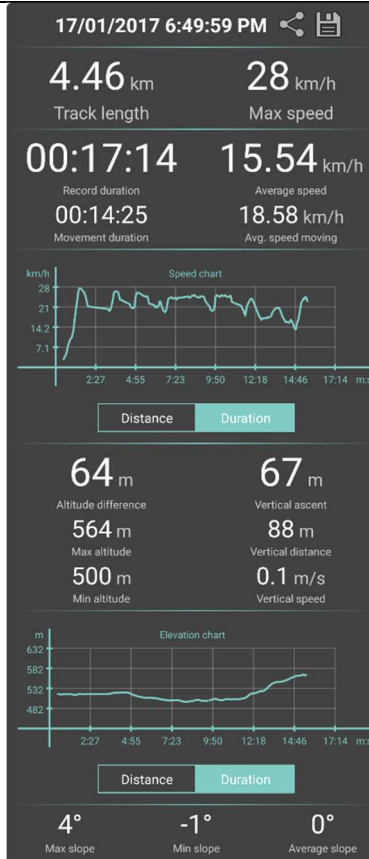
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	-1
<b>Track</b>	Source	Bülachhof (- ETH Hönggerberg)
	Target	(- ETH Center -) Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	176.05760205
	Measured [Wh]	170



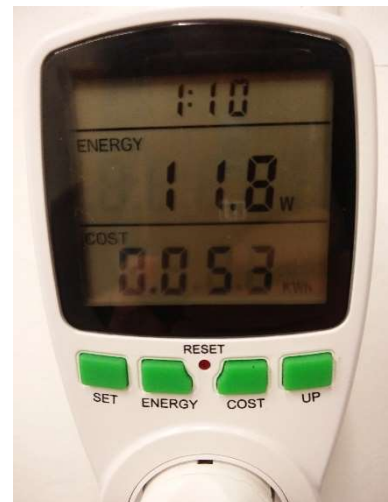
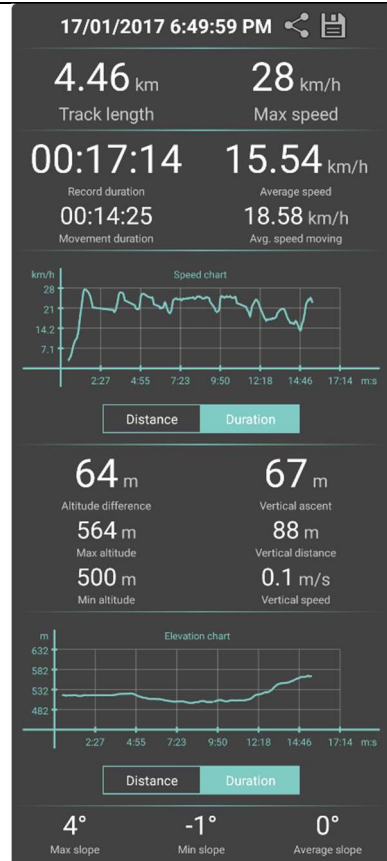


17.01.2017

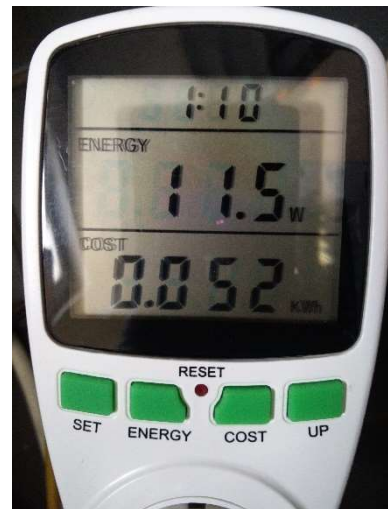
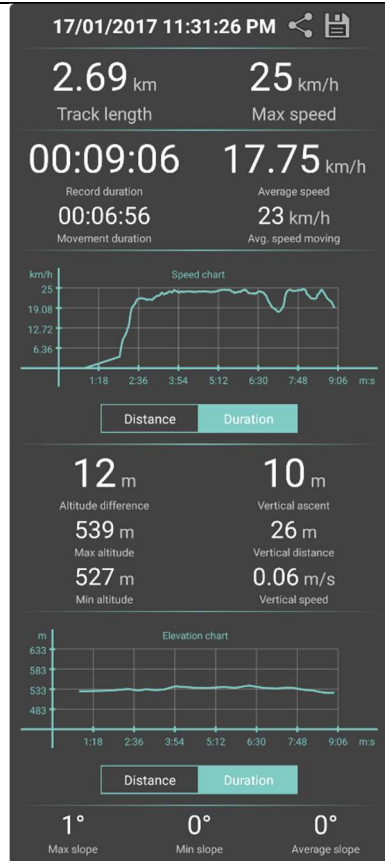
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	-4
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	89.20051469
	Measured [Wh]	83



<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	-4
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	50.13277719
	Measured [Wh]	52



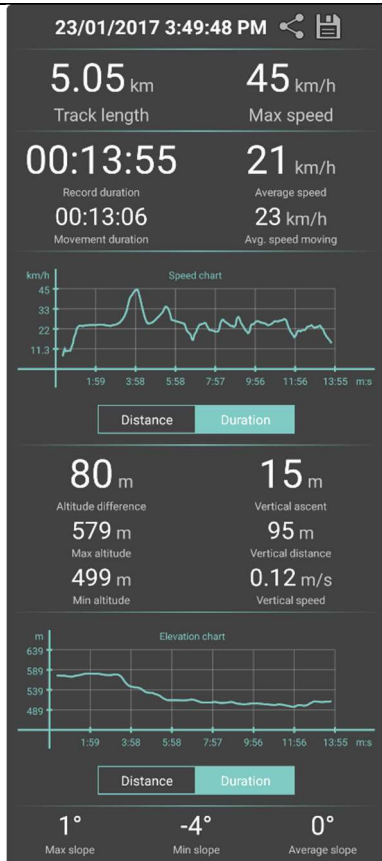
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	25
	Temperature [° C]	-4
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	50.13277719
	Measured [Wh]	53



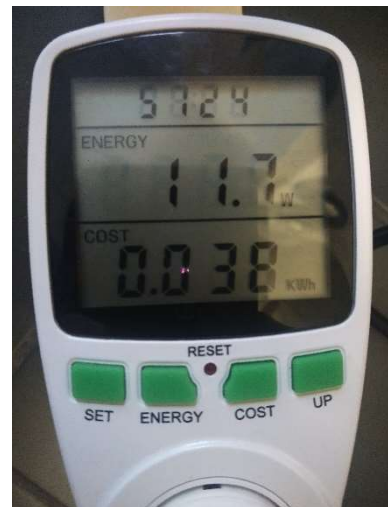
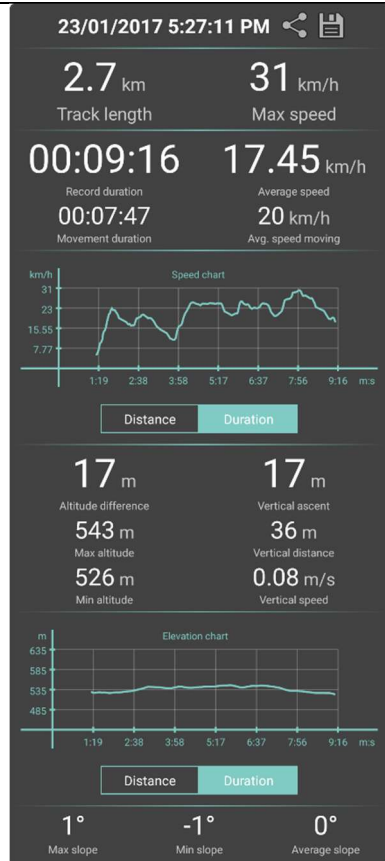


23.01.2017

<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	25
	Temperature [° C]	-4
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	60.19935606
	Measured [Wh]	42



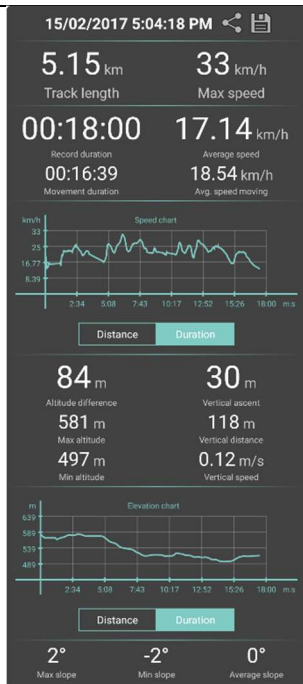
<b>Model Parameter</b>	Weight [kg]	123
	Velocity [km/h]	20
	Temperature [° C]	-4
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	41.52137251
	Measured [Wh]	38



Test Session B

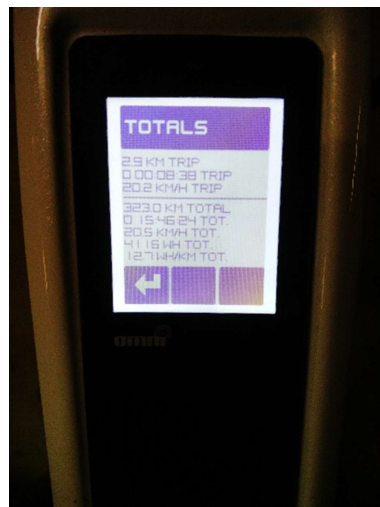
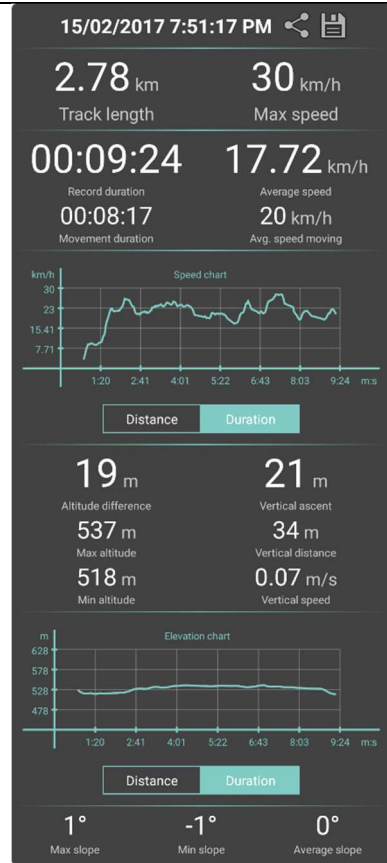
15.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	20
	Temperature [° C]	10
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	16.8643035888671
	Measured [Wh]	42





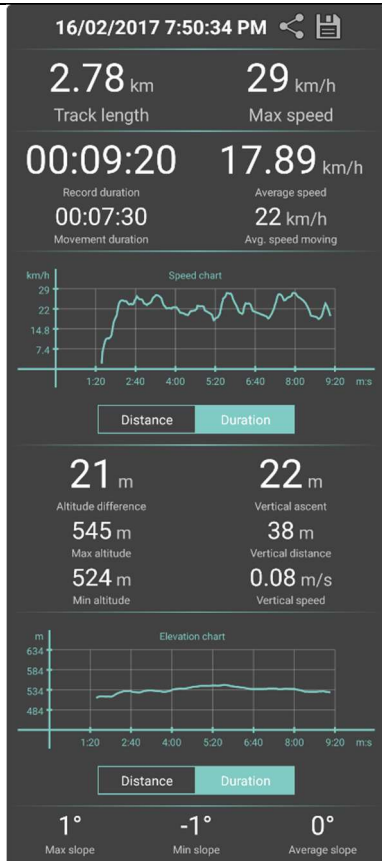
<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	20
	Temperature [° C]	10
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	24.5377445220947
	Measured [Wh]	35





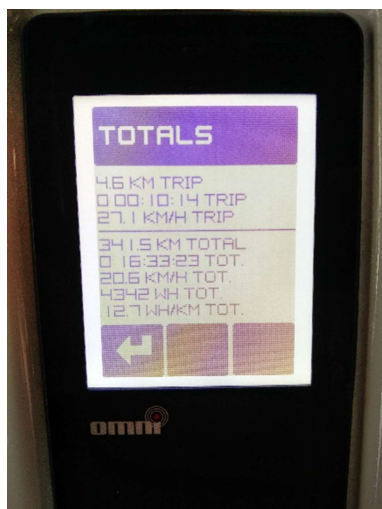
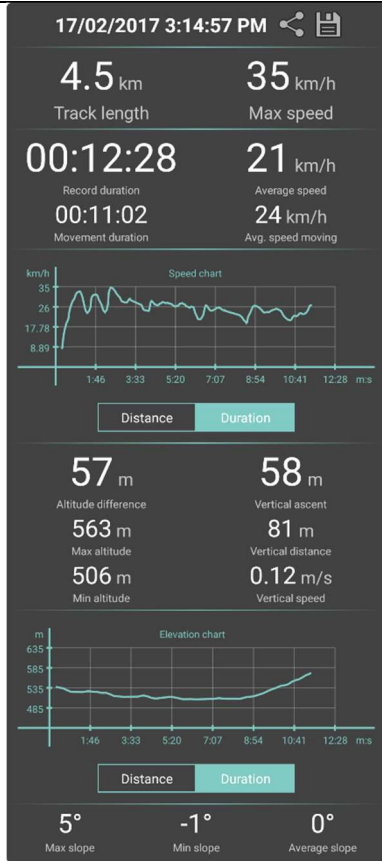
16.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	20
	Temperature [° C]	10
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	24.5377445220947
	Measured [Wh]	28



17.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	25
	Temperature [° C]	5
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	64.861569404602
	Measured [Wh]	85



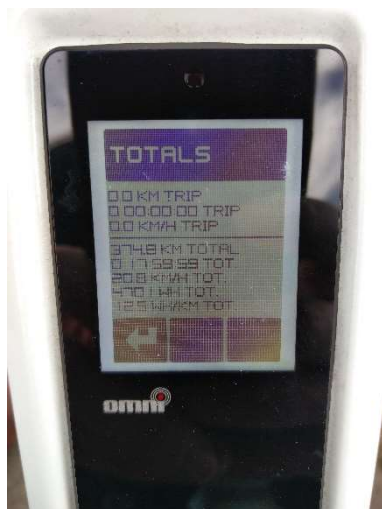
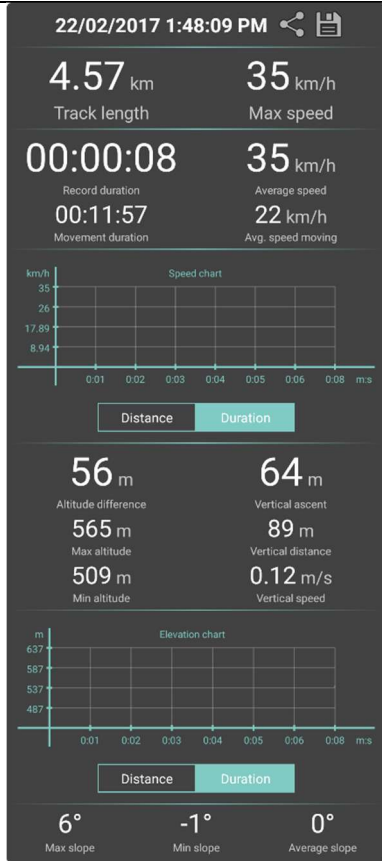
19.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	20
	Temperature [° C]	10
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	56.6393508911132
	Measured [Wh]	64



22.02.2017

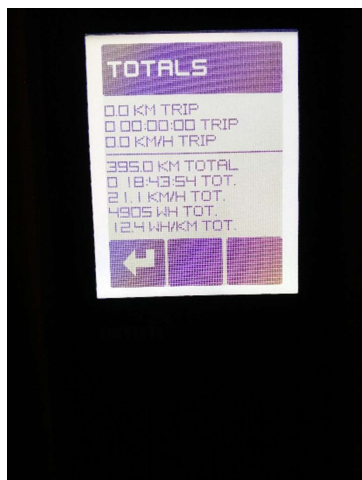
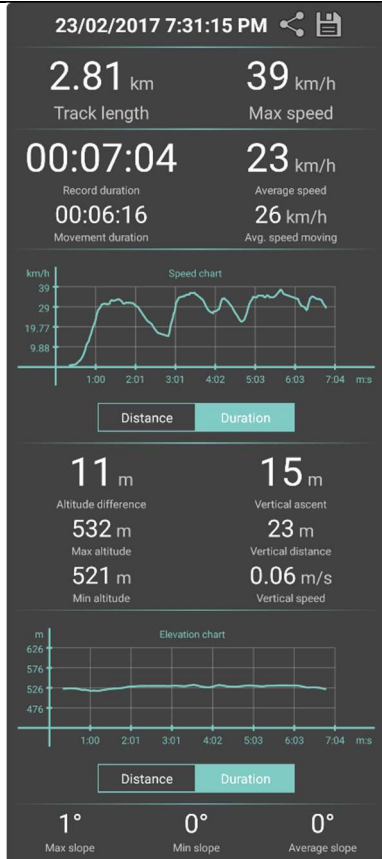
<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	20
	Temperature [° C]	13
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	55.7145195007324
	Measured [Wh]	50





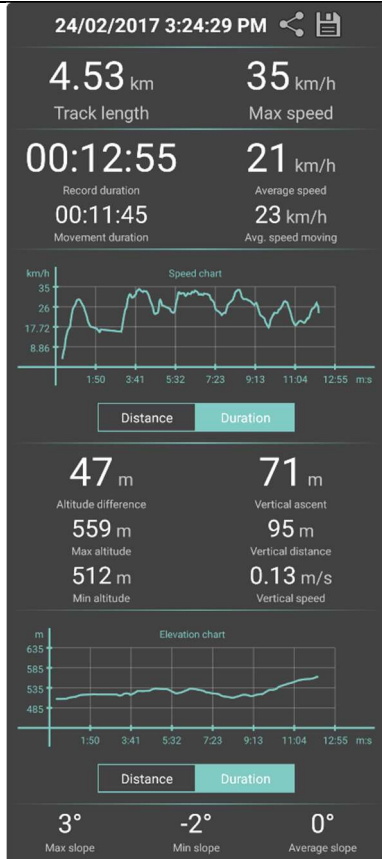
23.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	25
	Temperature [° C]	10
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	27.8884544372558
	Measured [Wh]	14



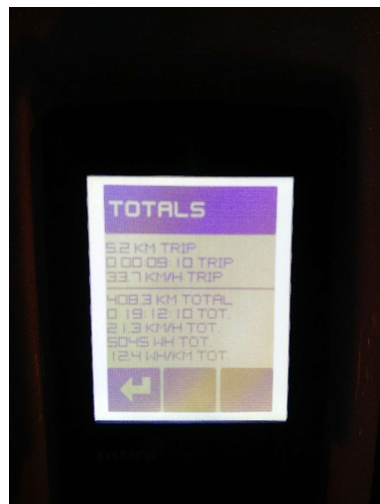
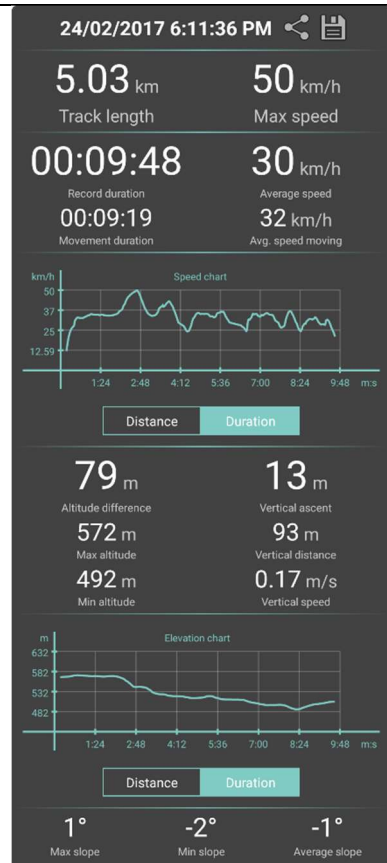
24.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	25
	Temperature [° C]	7
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	64.11506557464587
	Measured [Wh]	70



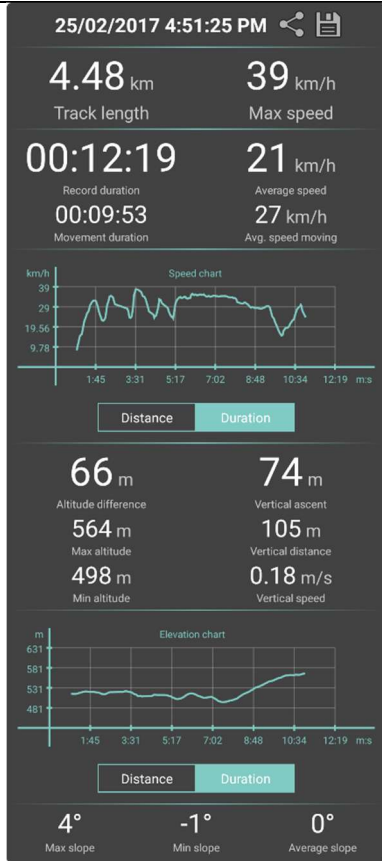


<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	30
	Temperature [° C]	5
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	33.6142349243164
	Measured [Wh]	35

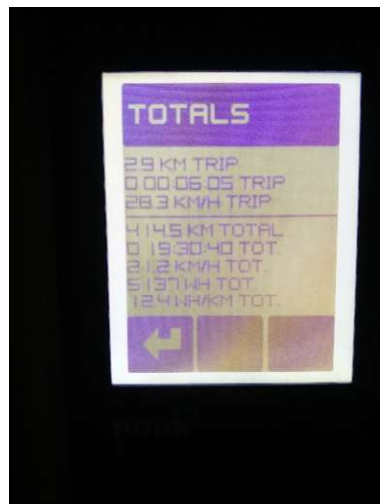
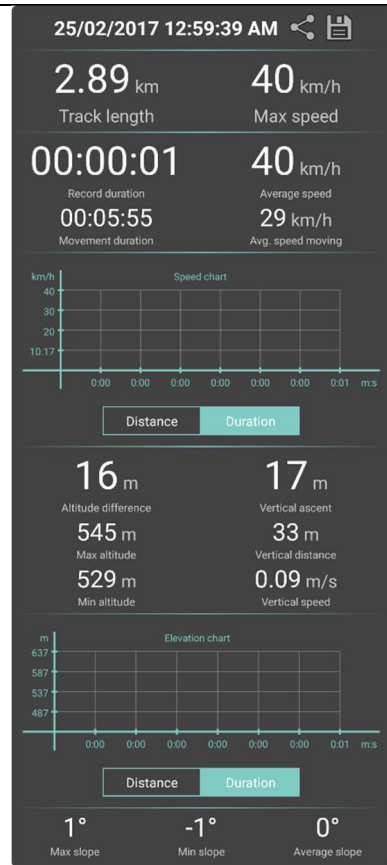


25.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	25
	Temperature [° C]	7
<b>Track</b>	Source	Bülachhof
	Target	ETH Hönggerberg
<b>Energy Consumption</b>	Modelled [Wh]	64.11506557464587
	Measured [Wh]	63

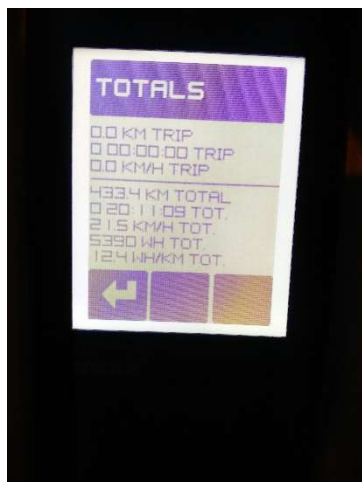
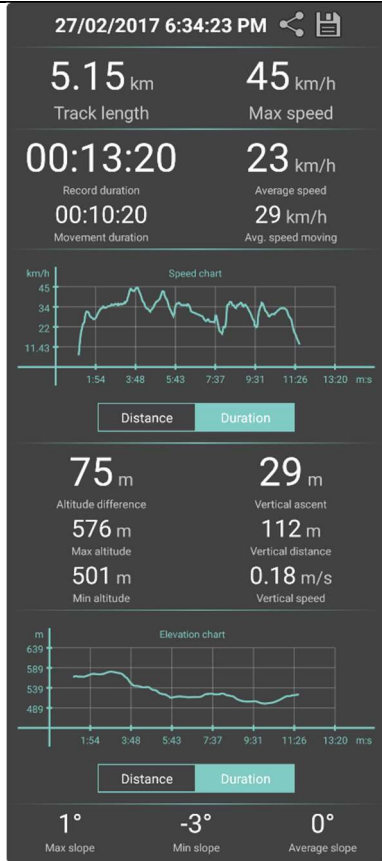


<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	30
	Temperature [° C]	3
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	
	Measured [Wh]	42



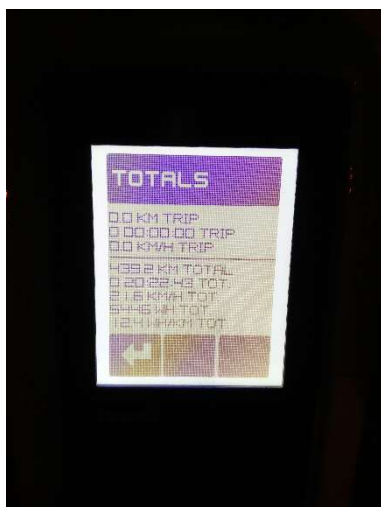
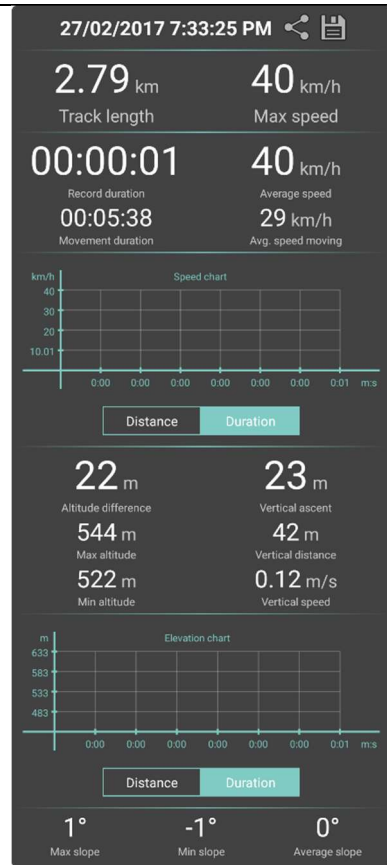
27.02.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	30
	Temperature [° C]	10
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	32.2482948303222
	Measured [Wh]	35



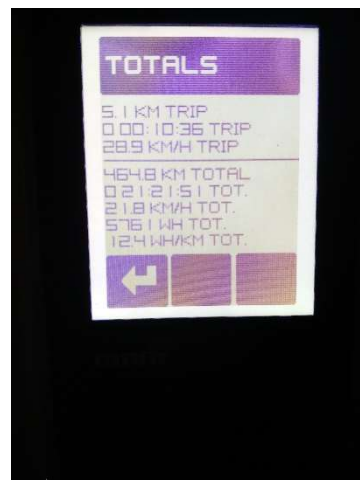
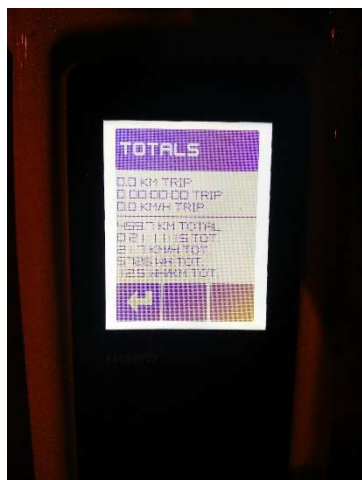
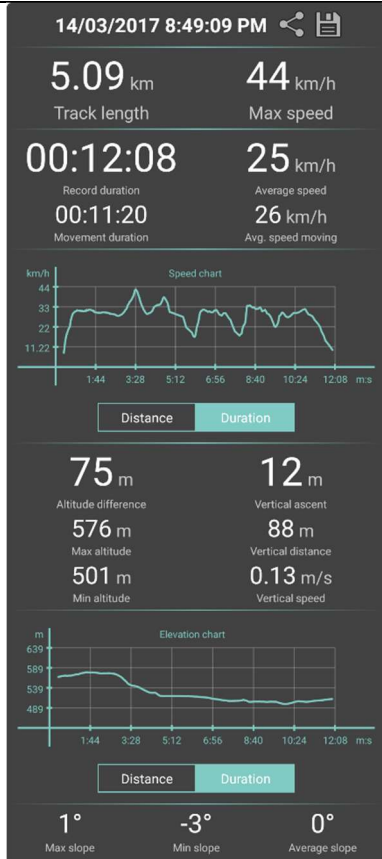


<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	30
	Temperature [° C]	10
<b>Track</b>	Source	ETH Center
	Target	Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	33.6458396911621
	Measured [Wh]	28



14.03.2017

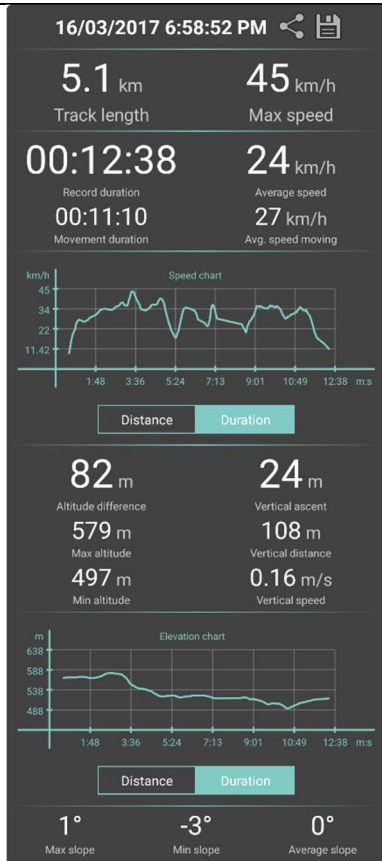
<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	25
	Temperature [° C]	10
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	22.691593170166
	Measured [Wh]	35





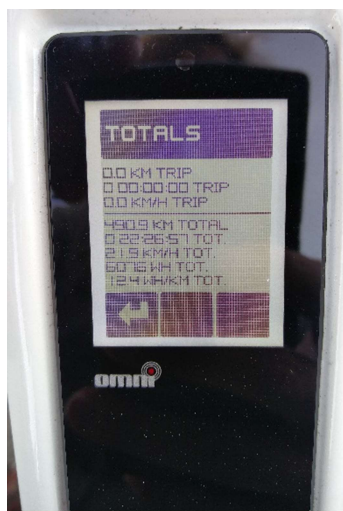
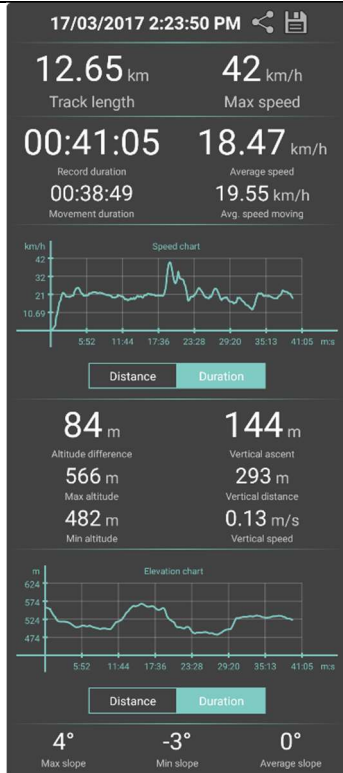
16.03.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	25
	Temperature [° C]	10
<b>Track</b>	Source	ETH Hönggerberg
	Target	ETH Center
<b>Energy Consumption</b>	Modelled [Wh]	22.691593170166
	Measured [Wh]	35



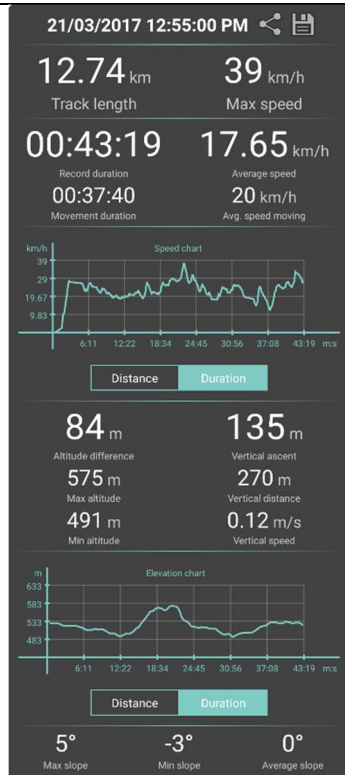
17.03.2017

<b>Model Parameter</b>	Weight [kg]	82
	Velocity [km/h]	20
	Temperature [° C]	20
<b>Track</b>	Source	Bülachhof (- ETH Hönggerberg)
	Target	(- ETH Center -) Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	70.6653275489805
	Measured [Wh]	56



21.03.2017

<b>Model Parameter</b>	Weight [kg]	127
	Velocity [km/h]	20
	Temperature [° C]	13
<b>Track</b>	Source	Bülachhof (- ETH Hönggerberg)
	Target	(- ETH Center -) Bülachhof
<b>Energy Consumption</b>	Modelled [Wh]	96.3204441070556
	Measured [Wh]	149



## Affidavit

Ich versichere:

- dass ich die Masterarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- dass alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Publikationen entnommen sind, als solche kenntlich gemacht sind.
- dass ich dieses Masterarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum

24.05.2017

Unterschrift





Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

### Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

**Titel der Arbeit (in Druckschrift):**

Optimizing the Operation Range of E-Bikes in Routing Systems

**Verfasst von (in Druckschrift):**

*Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.*

**Name(n):**

Haumann

**Vorname(n):**

Simon

Ich bestätige mit meiner Unterschrift:

- Ich habe keine im Merkblatt ["Zitier-Knigge"](#) beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

**Ort, Datum**

Zürich, 30.05.2017

**Unterschrift(en)**

*Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.*



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



universität  
wien