TECHNISCHE UNIVERSITÄT
CHEMNITZ

University of Technology Chemnitz
Faculty of Electrical Engineering and Information Technology
Professorship of Circuit Design
Prof. Dr.-Ing. Heinkel

ADVANCED RANGE ESTIMATION OF BATTERY ELECTRIC VEHICLES

lab task for the course "System Reliability in Electromobility"

Steffen Weichold
October 2nd, 2014

Contents

1 Introduction **3**
1.1 Theory . . . 3

2 Template Folder **5**

3 Preparing the track data **6**

4 Running the simulation **7**

5 Extending the Simulink model **8**
5.1 Completing the physical model . . . 8
5.2 Parameterizing the model . . . 8
5.3 Exporting results to the workspace . . . 8

6 Analysing results of multiple simulations **9**

7 Further exercise **9**

8 Model submission **10**

1 Introduction

In this lab you will learn how to create and simulate a Simulink model based on a physical model for the range estimation of BEV vehicles. Part of this task is also the preparation of the data for the simulation and the handling of the simulation results using MATLAB. Both of which are also covered in this lab.

The task is divided into the following steps:

- read a track of GPS coordinates from a gpx file and convert it into a list of points with metric coordinates, compute the slope for each track segment
- set a vehicle velocity for each track segment based on some basic assumtions and compute the derived quantities
  - segment time
  - total time
  - acceleration
- create a set of plots for conveniently reviewing the loaded and computed track data
- get familiar with the Simulink environment and simulate the partial model supplied by the template using Simulink
- complete the model according to the subtasks specified in this document
- create a MATLAB script to automate the process of running the simulation using different input data and model parameters and create a report that compares the results

1.1 Theory

A route, driven by a vehicle, is represented by a track T that is composed of n_t ordered trackpoints p_i.

$$T = \{p_i \mid i = 1, ..., n_t\}$$

A track may be also divided into a set S of segments s_i, which are represented by pairs of consecutive track points.

$$S = \{s_i = (p_i, p_{i+1}) \mid i = 1, ..., n_t - 1\}$$

For a given track, the power $P_{Pedelec}$ that is needed to drive along a single track segment s_i is given as

$$P_{Pedelec}(s_i) = P_{Acceleration}(s_i) + P_{RollingFriction}(s_i) + P_{DownhillSlope}(s_i) + P_{AirDrag}(s_i). \quad (1)$$

The individual components of $P_{Pedelec}$ are composed as follows.

$$P_{Acceleration}(s_i) = a_i \cdot (m_{Pedelec} + m_{Driver}) \cdot v_i \tag{2}$$

$$P_{RollingFriction}(s_i) = c_r \cdot cos(\alpha_i) \cdot g \cdot (m_{Pedelec} + m_{Driver}) \cdot v_i \tag{3}$$

$$P_{DownhillSlope}(s_i) = sin(\alpha_i) \cdot g \cdot (m_{Pedelec} + m_{Driver}) \cdot v_i \tag{4}$$

$$P_{AirDrag}(s_i) = \frac{\rho \cdot c_w \cdot A \cdot v_i^2}{2} \cdot v_i \tag{5}$$

Where the components

- v_i: the velocity over Segment s_i
- a_i: the acceleration over Segment s_i
- α_i: the slope over Segment s_i

are implicitly given by the GPX data and a few assumptions.
The remaining components are either physical constants or specific to the chosen bike, the driver and his or her driving style.

- g: gravity of earth (can be assumed to be the constant standard gravity, 9.8 m/s^2):
- ρ: density of air (can be assumed constant, 1.225 kg/m^3)
- c_r: rolling friction coefficient (racing tires: 0.003, mtb-tire 0.006)
- c_w: air drag coefficient (racing bike 0.4, "standard" bike: 1.1)
- A: frontal area of driver and bike (racing bike: 0.38 m^2, "standard" bike: 0.5 m^2)
- $m_{Pedelec}$: mass of the bike
- m_{Driver}: mass of the driver.

Finally, the energy W_{Track} that is used to drive along the track can be computed by creating the sum of the products for each segments power and time:

$$W_{Track} = \sum_{i=1}^{n_t-1} W(s_i) = \sum_{i=1}^{n_t-1} P(s_i) \cdot t_i. \tag{6}$$

E-bike ist a general term that refers to different kinds of bicycles that can be powered either by an electric motor, the drivers own muscle-power or a combination of both. For this course the pedelec ("Pedal Electric Bike") is chosen to be simulated. This kind of e-bike can be considered as a hybrid electric vehicle and is motor and human powered. By statutory rule it is only allowed to assist the driver while he is actively pedaling and only up to a velocity of 25 km/h. Above this velocity the full power has to be provided by the driver alone and the motor has to be automatically shut off.

The simulation has to use a method to distribute the power between driver and bike to consider those legal requirements. For this the template model uses a fixed minimal driver power. If the power p_i over a segment is below this parameter value, the driver has to provide all the power. Should p_i exceed the minimal driver power, then the motor will support up to its maximum power. Other methods of balancing the power between driver and bike are left as an exercise.

Finally, to draw a conclusion after the simulation, the total energy used by the motor has to be compared to the current capacity of the bikes battery. If the computed total energy (see equation 6) is larger than the capacity of the bikes battery, the track can not be driven with the motor support distribution used by the simulation. The simulation template includes a simple virtual battery subsystem that continually drains energy from a initial battery capacity.

2 Template Folder

The supplied template folder contains a basic skeleton of scripts, functions and a partial e-bike model to get you started. Some of the steps, such as loading the GPX data, for creating the full range prediction model are already implemented. The template contains the following files.

- "`const.m`":
  Defines basic constants that are used to enhance the readability of the source code.
- "`loadgpx.m`":
  Contains the function, that reads the GPX data from the supplied filename and computes the track coordinates in metric units as well as the slope over each track segment. Returns a matrix containing the track data.
- "`assign_speed.m`":
  Contains functions to assign a velocity to each segment based on a few basic assumptions. It has to be extended to also compute derived values like the time spent in each segment, total time and the acceleration.
- "`plot_track.m`":
  Creates a matlab figure with different plots to visualize the track data. This will be extended to conveniently check all the computed values for correctness.
- "`prepare_sim.m`":
  This script combines the afforementioned functions to prepare the workspace for the simulation.

- ”`run_sim.m`”:
  A script that will run a simulation and captures the results. During the course it will be extended to run multiple simulations with different parameters and create a basic report to show the results in a compressed form.
- ”`track_01.gpx`”:
  The GPX file containing the raw track data. See figure 1.
- ”`ebike.mdl`”:
  The Simulink model with the partial range prediction algorithm. It will be completed during the course.
- ”`editors.txt`”:
  Please fill in the names and matriculation numbers of the editors (max. two) in this file before submiting.

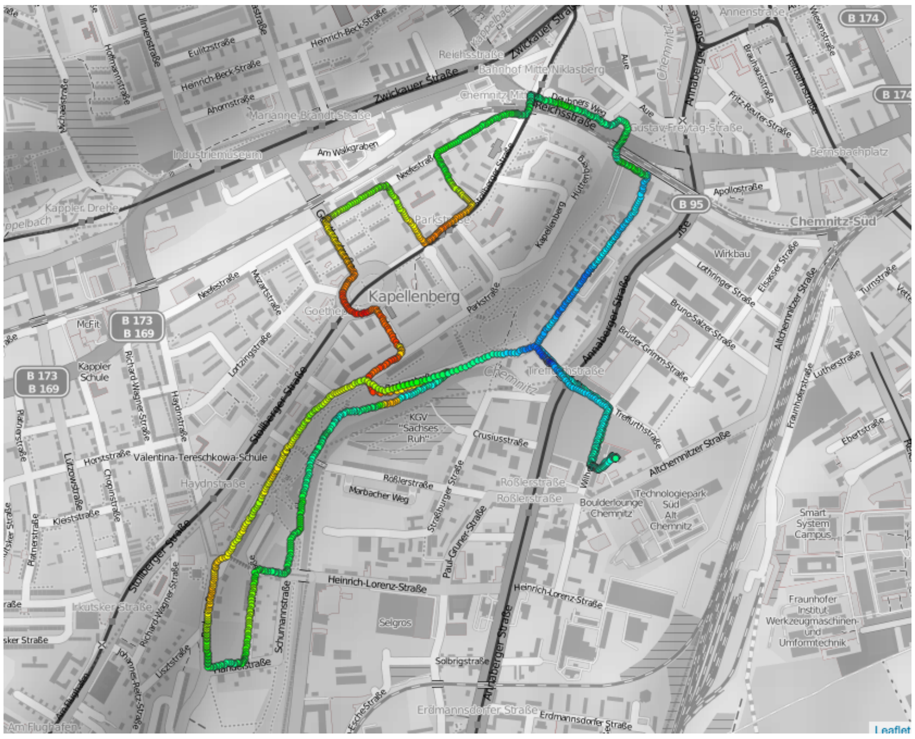

Figure 1: The example track with its height ploted in a gradient from blue (lowest) to red (highest).

3 Preparing the track data

The function ”`loadgpx`” already supplies the track data as a series of trackpoints in metric coordinates, the distance and the slope between consecutive trackpoints and the accumulated track distance.

To compute the power necessary to drive along the track, additional data is needed. The first step is to assume a velocity for each track segment, that the rider might choose. As an example, a constant speed is already assigned to each segment by the "`assign_speed`" function.
Next, the computation of values that can be derived after this assignment has to be implemented. This includes

- the time needed to traverse each segment,
- the accumulated time for each track point,
- the acceleration in each segment.

The "`assign_speed`" function already contains placeholders for these computations.

To check them for correctness the function "`plot_track`" takes the track data as an argument and creates a MATLAB figure with multiple plots to visualize the complete trackdata. Plots for the height profile of the track and a 3D-plot of the track itself are already supplied. The plots for all remaining quantities have to be created as an exercise.

4 Running the simulation

After all the necessary precomputations are implemented, the supplied model (figure 2) can be simulated using Simulink. To do this, open the file "`ebike.mdl`" and click the "Run" button in the Simulink environment. If any errors are reported, go back and revise your previous steps according to the error message.

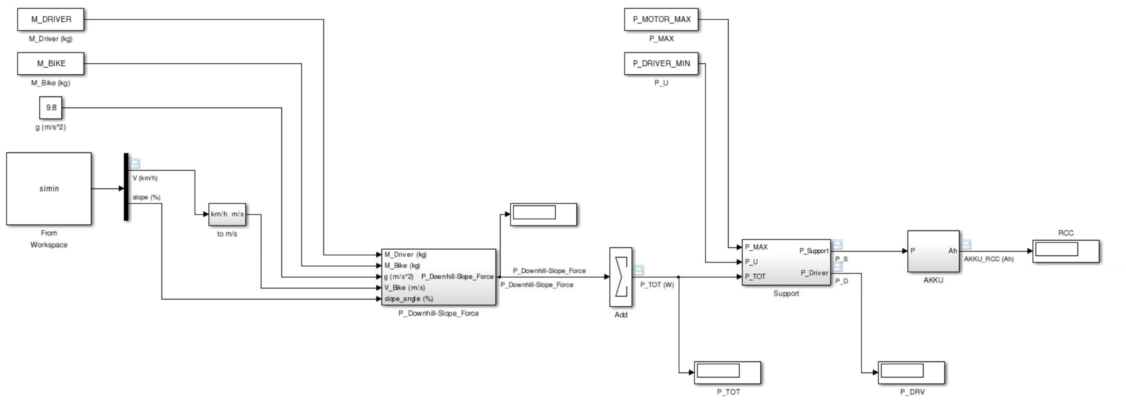

Figure 2: Simulink Model supplied with the template.

Once the simulation has been successfully run, select one of the "scope" symbols to open the prepared visualization of the simulation results to check them for plausibility.

5 Extending the Simulink model

The model supplied with the template is incomplete and has to be extended in several points.

5.1 Completing the physical model

Only one of the components from equation 1 is already modeled as a subsystem. The three missing components have to be added as individual subsystems. This includes the necessary constants at the top level of the model. See the subsystem for the computation of the downhill-slope force for comparison. To complete this step, the acceleration computed in function "`assign_speed`" has to be supplied as an additional input to the model.

The simulation should now compute the power $P_{Pedelec}$ over all segments and subtract the total used energy from the battery. Using the Signal & Scope Manager, additional scopes should be attached to some of the new signals to check the results for plausibility.

5.2 Parameterizing the model

When running the simulation with different setups (e.g. for other bikes, riders, etc.), the constants used to compute the power have to be changed by hand directly within the model. To automate this process all constant sources in the model have to be set up to take parameters directly from the MATLAB workspace. These should be defined in the script file "`run_sim.m`".

5.3 Exporting results to the workspace

To further support the automation of running simulations with the pedelec model, all relevant signals have to be output from the simulation when run from MATLAB code. A method should be chosen, that allows direct access to the data from the MATLAB workspace (no file should be used). Besides the time values the signals that should be exported include

- the total energy used by the driver,
- the total energy used by the motor,
- the distribution of the total energy over the four components from equation 1,
- the state of charge / rest capacity of the battery.

If further extensions were made to the model, the corresponding signals can be exported as well.

6 Analysing results of multiple simulations

To understand the effects of varying simulation parameters and input data, the empty script "`run_sim.m`" will be completed to automate the creation of a report from the results of two or more simulations. Using the extensions made so far, the script should perform the following steps for each of the simulations to run:

- compute the input data,
- adjust the parameters,
- run the simulation and capture the results,
- process the results to create statistical numbers that show the differences between the simulations,
- create a printed report to the Command Window and a MATLAB figure visualizing the results.

For more realistic simulation results, revise the function "`assign_speed`". So far, the template assigns a fixed speed to all the track segments, which ist not very likely to be done by a real driver. Adapt the function to use both the fixed velocity method and a more realistic version. One possible approach would be to compute a velocity based on the slope of each segment. For this, the script "`assign_speed.m`" already contains an empty function body.

7 Further exercise

- Consider the efficiency distribution of the motor over its rpm-range. Include a basic subsystem, that computes the energy drained from the battery based on the velocity of the pedelec. For simplicity, it can be assumed that the bike has no gearshift and the motor rpm is linear to the velocity.
- Implement another distribution method for the driver/motor power. For example a percentaged distribution and the inclusion of other parameters like the velocity would be possible.
- Extend the model by a termination condition, in case the battery is drained before the end of the track ist reached. Also include a way to signal this to the user when presenting the report for the simulation results.

- Adapt the model to consider the rule, that above a velocity of 25 km/h the motor support will be shut of. As a comfort function, the motor should not be switched off immediately but its power output should be reduced gradually over a small velocity range (e.g. between 22 an 25 km/h).

8 Model submission

The project folder has to be uploaded via the OPAL platform at least three (working) days before the presentation is scheduled. The presentation dates will be assigned individually after the course.

The m-files and the model have to contain all the extensions from section 3 to 6. In addition one selected extension taken from the list under section 7 has to be implemented. Own suggestions for extending the model or the scripts are welcome but have to be discussed with the instructor first. The MATLAB code and the simulation of the model have to run without errors.