Documentation Parkassist

Graphische Programmierung und Simulation

at the Cooperative State University Baden-Württemberg Stuttgart

by

Nahku Saidy und Hanna Siegfried

07.04.2020

Contents

Acronyms

AABB   Axis-Aligned Bounding Box

List of Figures

List of Tables

Listings

1 Introduction

??

2 D1: Time estimate based on three point estimation

Table 2.1: Three point estimation of effort for meeting requirements

Requirement	Optimistic	Likely	Pessimistic	<T>	sigmahoch2	Actual
D1		.	.	.	.	.

3 D2: Feasibility study

The aim of the feasibility study is to analyse whether the introduced model in section 1 can be implemented based on the given formulas.

$$\frac{\partial v}{\partial t} = -c - b * p \tag{3.1}$$

$$\frac{\partial x}{\partial t} = v \tag{3.2}$$

- Minimale Geschwindigkeit 0,29km/h beachten -> in m/s umrechnen
- Switch -> wenn Geschwindigkeit kleiner 0,29 folgt daraus Geschwindigkeit = 0
- Screenshot Simulink Modell und Ergebnis
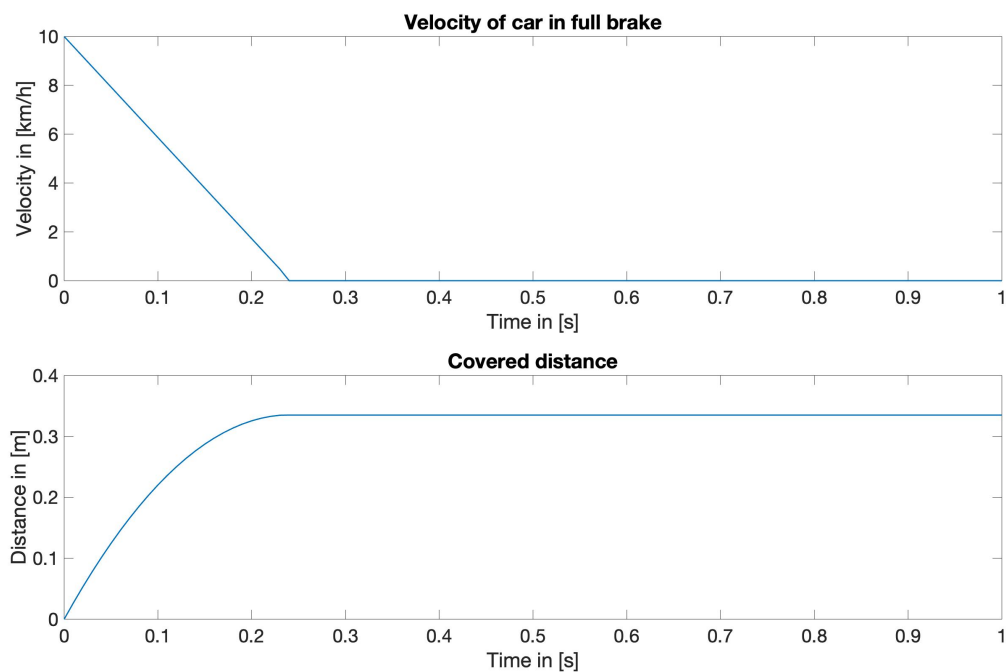- R5 auch beachtet

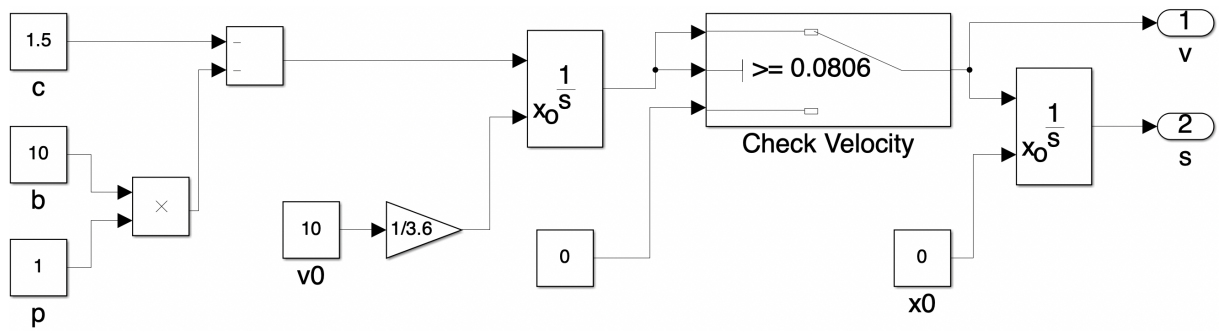Figure 3.1: UML diagram of the architecture of the software tool

Figure 3.2: Simulink Modell der Differenzialgleichungen

4 D3: Analysis of human velocity profile

In this section the provided human velocity profile is analysed in order to find a pattern that shows how a human is breaking a car. This pattern will be used in the following sections and will be adapted to the braking function of the ParkAssist. The analysis is divided into several steps.

1. Import the measurement data in Matlab.

```matlab
%import velocity data
velocity_data = importdata('MeasuredVelocities.txt');
```

Listing 4.1: Import measurement data in Matlab

2. Data preparation: Before analysing the data it is necessary to preprocess it. First the four velocities of each wheel are extracted from the measurement data and a mean velocity is calculated (see line). The mean velocity is converted to [m/s] for further calculations. For analysing the braking behaviour the acceleration is calculated. This can be done by differentiating the velocity (see line). As can be seen in Figure REF in the upper plot the calculated acceleration is ?? as cause of minor variations in the velocity. Due to the ?? it would be hard to analyse the braking behaviour. Therefore, we used a moving average filter to smoothen the data (line). A moving average filter is calculating a mean value of neighbourhood elements within a sliding window of a predefined size. Applying a moving average filter results in a smoother graph as can bee seen in the lower plot in figure REF.

```matlab
%compute mean velocity of all 4 wheels
velocity_per_wheel = velocity_data(:,2:5);
mean_velocity = mean(velocity_per_wheel,2);
mean_velocity = mean_velocity/3.6;        %convert velocity from km/h to m/s
raw_mean_velocity = mean_velocity;        %for demonstration purposes

%apply moving average filter to smoothen the data
mean_velocity = movmean(mean_velocity, 200);

%differentiate velocity to get acceleration
acceleration = diff(mean_velocity);
raw_acceleration = diff(raw_mean_velocity);      %for demonstration purposes
```

Listing 4.2: Preprocessing measurement data

3. Extracting negative acceleration: As the goal is to analyse the braking behaviour only the negative acceleration is relevant and is thus being extracted from the overall

acceleration (line). Also, all velocities that are based on a positive acceleration are not considered anymore. We decided to set those values to NaN because this way we are still able to plot the velocity and the acceleration over time without cut-outs. The result can be seen in plot REF.

```
1 %search for negative acceleration and set positive acceleration to NaN
2 %set all velocities that have a positive acceleration to NaN
3 neg_acceleration = acceleration;
4 decreasing_velocity = mean_velocity;
5 neg_acceleration(neg_acceleration>0) = nan;
6 decreasing_velocity(isnan(neg_acceleration)) = nan;
```

Listing 4.3: Extracting negative acceleration

4. Separating breaking sequences: For improved visualisation the individual breaking sequences that can be seen in plot REF are stored separately. One breaking sequence consists of multiple consequent velocities. This means that the breaking sequences can be separated by searching for a gap in the velocities. Therefore, the indices of all velocities that are set not NaN are extracted (line) and differentiated (line). Considered the differentiation, every differentiation that is greater than 1 marks a gap between velocities because indices of one braking sequence are consequent (differentiation equals 1). Furthermore, small breaking sequences are not considered because they are most likely not relevant for recognizing a breaking pattern (line). Two individual breaking sequences can be seen in plot REF and plot REF.

```
1  %find individual breaking sequences
2  notNaN_velocity = find(~isnan(decreasing_velocity));    % find index of every velocity
        that is not NaN -> one breaking sequence has
3                                                          % consequent time steps -> one
                                                                breaking sequence has
                                                                consequent indices
4  diff_notNaN_velocity = diff(notNaN_velocity);           % differentiate indices -> if
        indices are not consequent (unequal 1), a new
5                                                          % breaking sequence has begun
6  n = 1;                                                  % set start values
7  start = 1;
8
9  %seperate breaking sequences with previous findings
10 for i=1:length(diff_notNaN_velocity)
11     %for every index check if indices are consequent -> equals 1
12     if diff_notNaN_velocity(i) > 1
13         %if not check if a breaking sequence consists of min 500 time
14         %steps, this way small breaking sequences are sorted out
15         if (i-start) > 500
16             %add breaking sequence to section
17             %i is end of sequence
18             section{n} = notNaN_velocity(start:i);
19             %start for new sequence is end of old sequence + 1
20             start = i+1;
21             %increment n
22             n = n+1;
23         else
24             %if breaking sequence is to small, set start for new sequence
```

```
25              %to end of old sequence + 1 anyway
26                start = i+1;
27           end
28       end
29  end
```

Listing 4.4: Separating breaking sequences

Result

Considering Figures REF, REF and REF, it can be seen that the acceleration is approximately in the shape of a parable with the minimal turning point at XX. For further implementation of the ParkAssist, we will start the breaking process with an acceleration and increase it until we reach approximately REF and will then decrease it.

5 D4*: Consideration of uneven parking spaces

6 D5: Discussion of inaccuracies in velocity measurement

validate findings by numbers from simulation

7 D6: Implementation of pulse signal in Simulink

8 D7: Transfer of Simulink model to ASCET

9 D8: Implementation of pule signal in ASCET

10 D9: Implementation of unit tests for ASCET model parts

11 D10: Development and implementation of a system test environment for ASCET simulation

12 D11*: Plausibility check comparing measured velocities and distances

13 D13*: Impact of inaccuracies

14 D14*: Reflection