Documentation Parkassist

Graphische Programmierung und Simulation

at the Cooperative State University Baden-Württemberg Stuttgart

by

Nahku Saidy und Hanna Siegfried

07.04.2020

Time of Project	24.03.2020 - 07.04.2020
Student ID, Course	8540946; XXX, STG-TINF17-ITA
Company	Daimler AG, Stuttgart
Supervisor in the Company	Dr. Kai Pinnow

Contents

Acronyms

NaN Not a Number

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

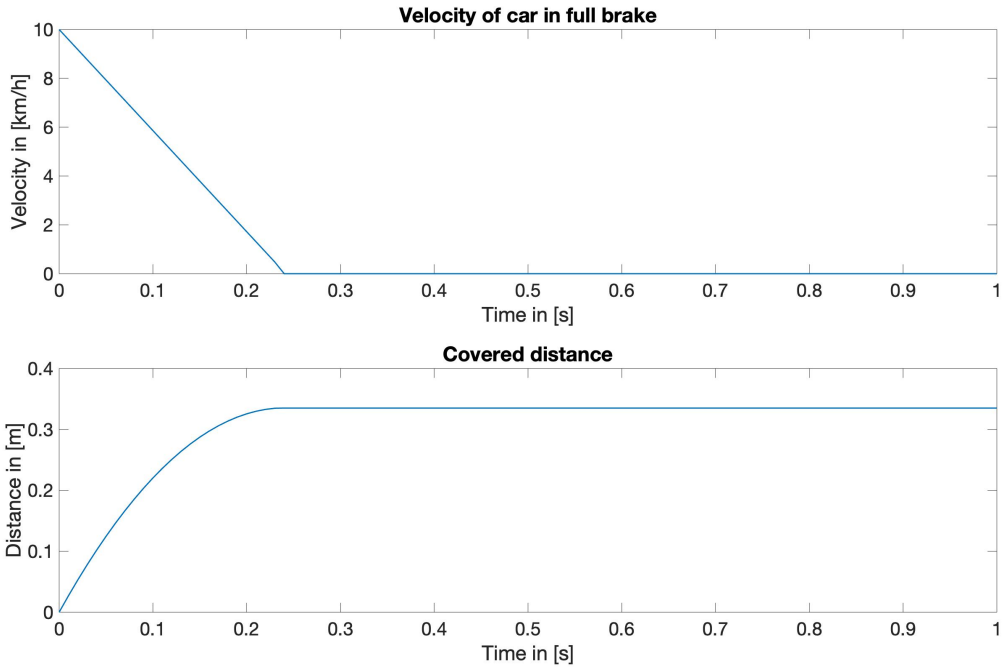

Figure 3.1: UML diagram of the architecture of the software tool

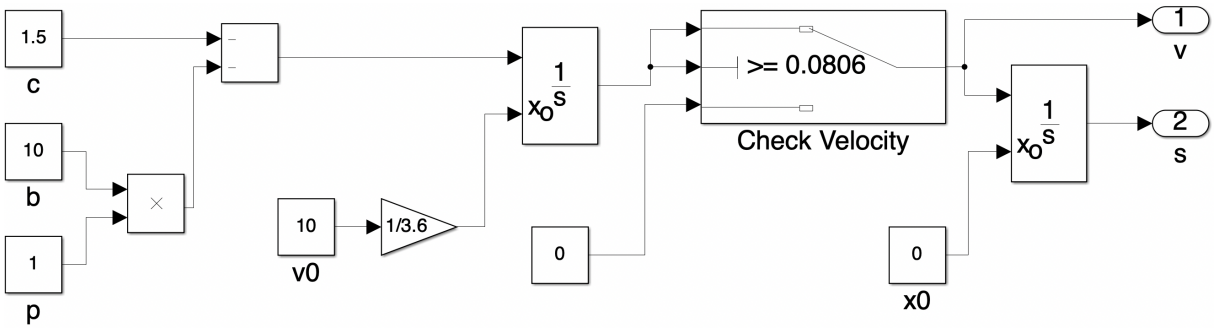

Figure 3.2: Simulink Modell der Differenzialgleichungen

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 four steps.

1. Import the measurement data

The measurement data is imported in Matlab.

```
1 %import velocity data
2 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 1,2).

$$v_{car} = (v_{w1} + v_{w2} + v_{w3} + v_{w4}) * 4 \tag{4.1}$$

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 11). 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 8). 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.

```
1 %compute mean velocity of all 4 wheels
2 velocity_per_wheel = velocity_data(:,2:5);
3 mean_velocity = mean(velocity_per_wheel,2);
4 mean_velocity = mean_velocity/3.6;      %convert velocity from km/h to m/s
5 raw_mean_velocity = mean_velocity;      %for demonstration purposes
6
7 %apply moving average filter to smoothen the data
8 mean_velocity = movmean(mean_velocity, 200);
```

```
9
10 %differentiate velocity to get acceleration
11 acceleration = diff(mean_velocity);
12 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 5). Also, all velocities that are based on a positive acceleration are not considered anymore (line 6). We decided to set those values to Not a Number (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 2) and differentiated (line 3). 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 13). 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 consequent time steps -> one breaking
      sequence has consequent indices
3 diff_notNaN_velocity = diff(notNaN_velocity);          % differentiate indices -> if
      indices are not consequent (unequal 1), a new breaking sequence has begun
4 n = 1;                                                 % set start values
5 start = 1;
6
7 %seperate breaking sequences with previous findings
8 for i=1:length(diff_notNaN_velocity)
9     %for every index check if indices are consequent -> equals 1
10    if diff_notNaN_velocity(i) > 1
11        %if not check if a breaking sequence consists of min 500 time
12        %steps, this way small breaking sequences are sorted out
13        if (i-start) > 500
14            %add breaking sequence to section
```

```
15              %i is end of sequence
16              section{n} = notNaN_velocity(start:i);
17              %start for new sequence is end of old sequence + 1
18              start = i+1;
19              %increment n
20              n = n+1;
21          else
22              %if breaking sequence is to small, set start for new sequence
23              %to end of old sequence + 1 anyway
24              start = i+1;
25          end
26      end
27  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

-diagram

-forces need to be considered (hangabtriebskraft)

-negative slope would increase stopping time and position

-positive slope would make stopping time shorter

-therefore brake power would have to be increased or decreased

-negative slope critical since collision could occur

6 D5: Discussion of inaccuracies in velocity measurement

-in human velocity profile 4 tire velocities recorded -mean used to compute car velocity -car driving around corner 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

-only 2 meter stop considered