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

?? matlab version ascet version

todo describe block diagram vs script for every necessary requirement

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 based on the given formulas todo.

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

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

- 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

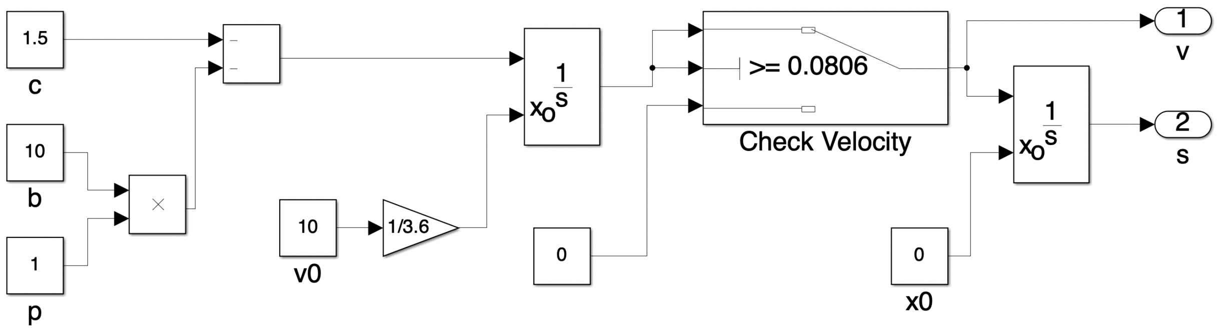

Figure 3.1: Simulink model of the differential equations

-result with constant 5% brake pressure -stops before 2m -constant brake pressure not human like but shows that it is possible to stop before 2 m -brake function can be changed to human like behaviour

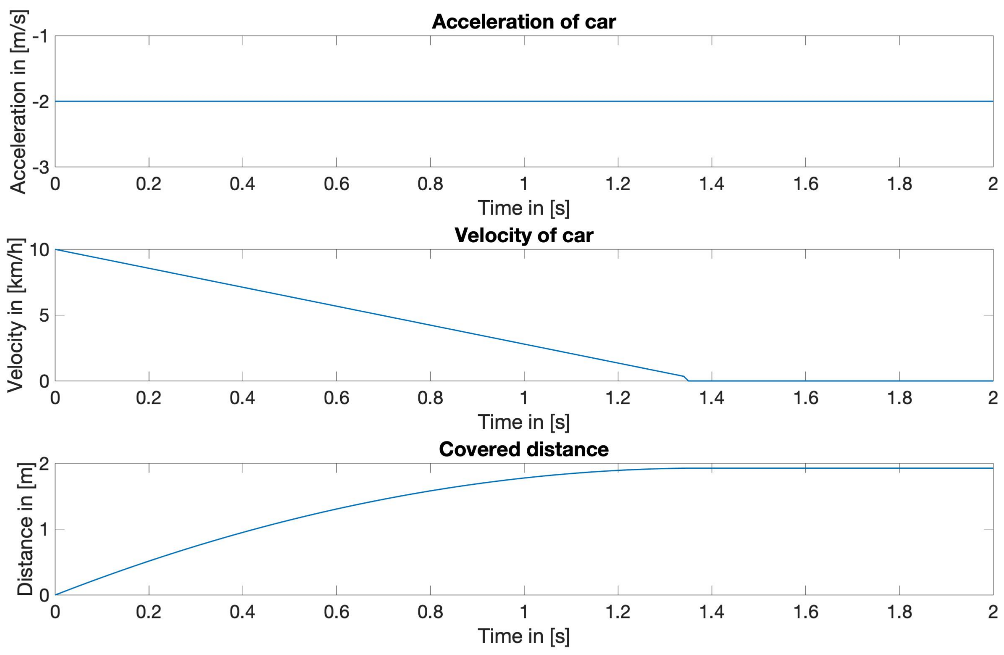

Figure 3.2: Todo

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.

4.1 Import the measurement data

The measurement data is imported in Matlab.

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

Listing 4.1: Import measurement data in Matlab

4.2 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.

```
%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

4.3 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

4.4 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.

```
%find individual breaking sequences
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
diff_notNaN_velocity = diff(notNaN_velocity);           % differentiate indices -> if
    indices are not consequent (unequal 1), a new breaking sequence has begun
n = 1;                                                  % set start values
start = 1;

%seperate breaking sequences with previous findings
for i=1:length(diff_notNaN_velocity)
    %for every index check if indices are consequent -> equals 1
    if diff_notNaN_velocity(i) > 1
        %if not check if a breaking sequence consists of min 500 time
        %steps, this way small breaking sequences are sorted out
        if (i-start) > 500
            %add breaking sequence to section
            %i is end of sequence
            section{n} = notNaN_velocity(start:i);
            %start for new sequence is end of old sequence + 1
            start = i+1;
            %increment n
            n = n+1;
        else
            %if breaking sequence is to small, set start for new sequence
            %to end of old sequence + 1 anyway
            start = i+1;
        end
    end
end
```

Listing 4.4: Separating breaking sequences

4.5 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

-negative slope critical since collision could occur

-therefore brake power would have to be increased or decreased

-physical borders need to be considered

-also when the car is stopped it should not start to roll forwards or backwards. On a plain surface, the brake can be released, after the car is stopped. A brake assist on a positive or negative slope would need to keep holding the brake or engaging the handbrake.

In the model without a minimum velocity, the velocity would become negative on braking instead of the car coming to a full stop.

6 D5: Discussion of inaccuracies in velocity measurement

-inaccuracy of velocity +- 0.1 km/h -also minimal velocity is 0.29 km/h which is not realistic, because velocity does not drop from 0.29 km/h to zero -in human velocity profile 4 tire velocities recorded -mean used to compute car velocity -because of inaccuracy in velocity computed acceleration also has inaccuracy -car driving around corner validate findings by numbers from simulation

7 D6: Implementation of pulse signal in Simulink

In D6 a pulsing information signal is described. This chapter documents the implementation of that signal. Both signals are needed for the frequency computation. The pulsing signal should only be present if the velocity of the car is ≤ 1 m/s and the position of the car is between 1 and 2 meters. From a frequency of 1 Hz at 1 meter it should rise up to 9 Hz at 1.9 meters. If the traveled distance is greater than 1.9 meters, the signal should become continuous.
This requirement can be divided into two separate compontents with separate responsibilities.
In the first component it is determined in which state the signal is (off, pulsing, continuous) and in the case of a pulsing signal the frequency of that pulse is computed.
The second component is responsible for outputting the pulse signal corresponing to the output of the first component.
The components are implemented using simulink subsystems which enables re-usability, encapsulation and also creates a better overview when looking at the main simulink model.

7.1 Computation of pulse signal frequency

Figure 7.1 shows the simulink subsystem for the computation of the frequency of the pulse signal. The inputs are the velocity and position of the car. The if-condition determines in which state the signal is.
In the first case the car velocity is ≤ 1 m/s and the position of the car is > 1.9 meters. In this case, the pulsing signal should provide a continuous output as demanded by requirement D6. The subsystem outputs a value of 10 in that case. This value is the indication for a continuous signal.
In the second case of the if condition the car velocity is also ≤ 1 m/s but the position of the car is between 1 and 1.9 meters. In this case the frequency of the pulsing signal should be between 1 and 9 Hz depending on the location. For that a lookup table is used, that provides an output frequency that linearly increases from 1 to 9 Hz depending on the position from 1 to 1.9 meters. A linear increase is used, because then the driver could

estimate the position linearly by the signal.
When none of the above mentioned conditions are the case, the subsystem outputs 0, which indicates that the pulsing signal is not present.

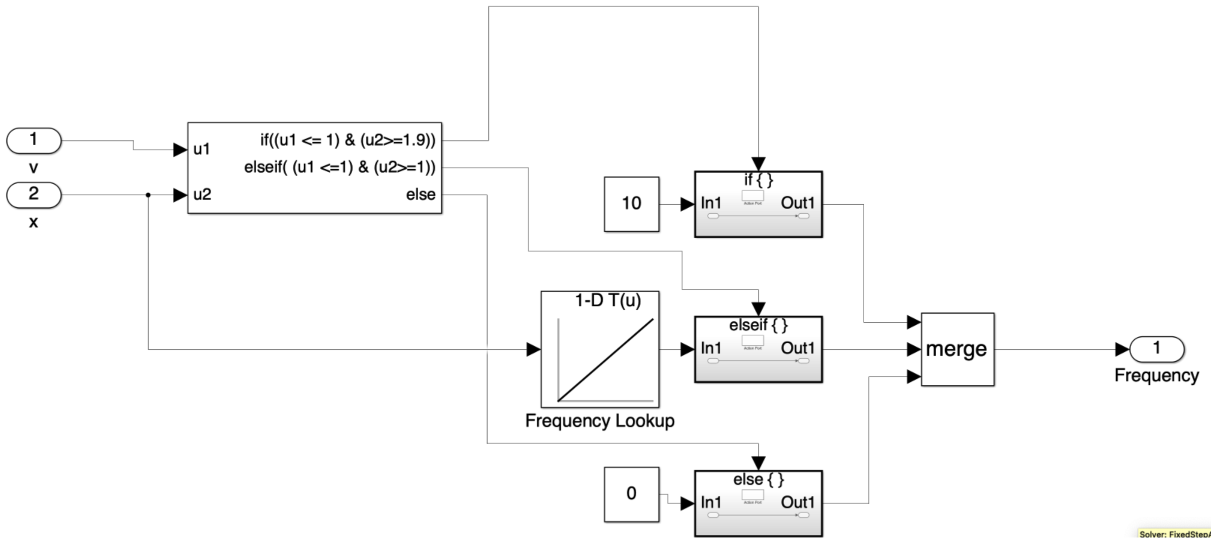

Figure 7.1: Simulink model of the frequency computation subsystem

todo output plot

7.2 Pulse signal generation

Figure 7.2 shows the simulink model of the pulse signal generator. This component provides a pulse signal with a variable frequency input. The input is the output of the before described component. Therefore if the input is 10, a continuous signal should be output. This is realised by the if-condition, which, if the input signal is 10 provides a continuous high output.
If that is not the case ... todo

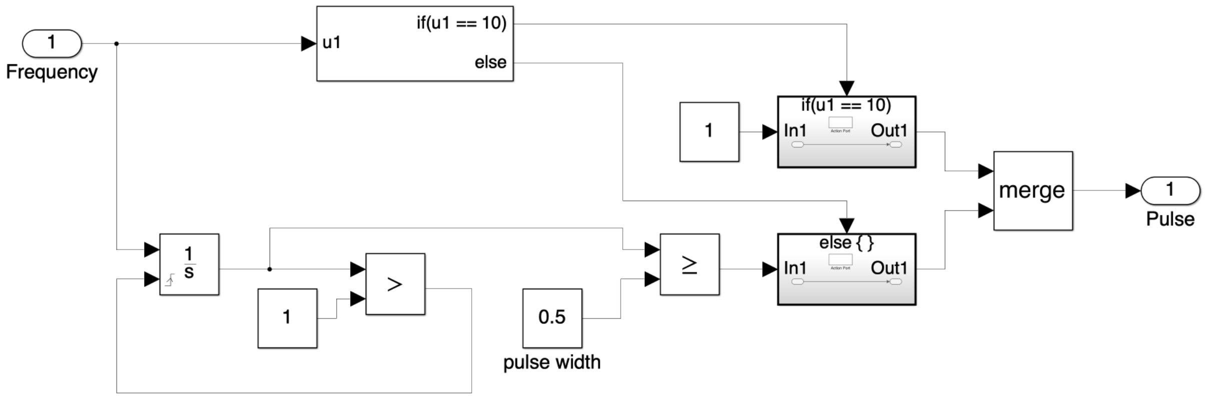

Figure 7.2: Simulink model of the pulse generation subsystem

7.3 Integration into car model

Using subsystems for the developed components makes it easy to include into and extend the existing model, which can be seen in figure 7.3

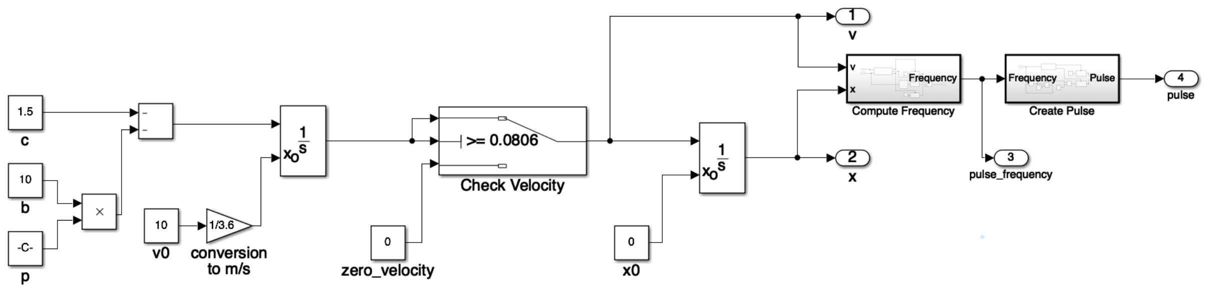

Figure 7.3: Simulink model of the car including the pulse signal

8 D7: Transfer of Simulink model to ASCET

- integration block as combination of multiplication and addition

9 D8: Implementation of pulse signal in ASCET

10 D9: Implementation of unit tests for ASCET model parts

- modular design for unit tests
- pulse signal, drive model not possible to unit test because of time component

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

- wait 5 seconds until starting because ascet experiment environment does not display the first seconds
- in first 5 seconds brake = -0.15 to counter c and b, v = 0
- after five seconds set v to 10 km/h and start experiment

12 D11*: Plausibility check comparing measured velocities and distances

-statistically independent ultrasonic distance and velocity -if distance to next object for ultrasonic sensor is known

-al

13 D13*: Impact of inaccuracies

-ultrasonic measurement: what if object is round? -

14 D14*: Reflection

-only 2 meter stop considered -model not realistic