HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

Heilbronn University

Faculty of Technik

Project work in the Autonomous Driving: Path Planning and Control

Documentation

Student:	David Retinski (216921)
	Dominik Bücher (216825)
Professor:	Prof. Dr.-Ing. Frank Tränkle
Date:	19.01.2024
Degree program:	Master of Mechatronics and Robotics
	Master of Automotive Systems Engineering
E-mail:	dretinski@stud.hs-heilbronn.de
	dbuecher@stud.hs-heilbronn.de

1 Table of contents

2 List of Illustrations

3 Task 5.2 Vehicle dynamics simulation

Required lab results:

1. Simulink model *s6_template.slx*
2. Signal-time diagram of the step responses of the vehicle speed $vc1$
3. Signal-time diagram of the step responses of the yaw angle ψ

Elaboration of the tasks:

1. Simulink model *s6_template.slx*
    - The Simulink model *s6_template.slx* can be found in the attached files. However, this is an old file, the new file s9_*template.slx* also contains this functionality and is designed in accordance with the guidelines.
2. Signal-time diagram of the step responses of the vehicle speed $vc1$
    - v_{c1} for Low-speed Dynamics
    - Drive maneuver: $(u_n = 0.1 \quad \delta_n = 1.0)$

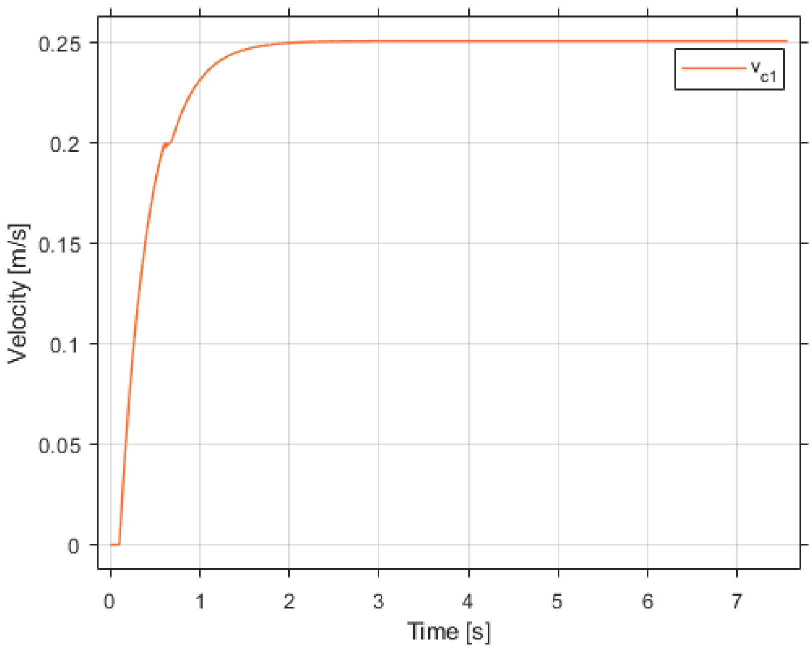


Figure 1: Signal-time diagram of steps response of vehicle speed v_{c1} with Low Speed

- v_{c1} for High-speed Dynamics
- Drive maneuver: $(u_n = 1.0 \quad \delta_n = 1.0)$

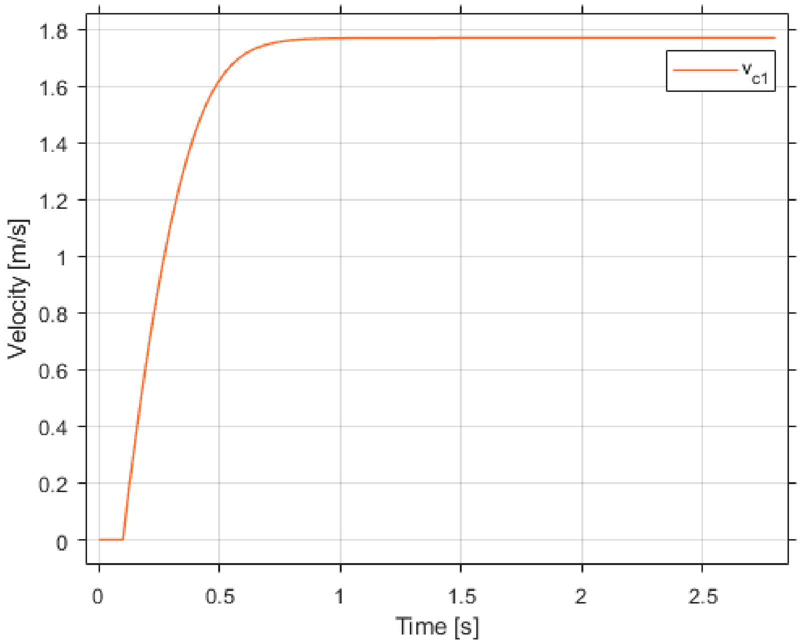


Figure 2: Signal-time diagram of steps response of vehicle speed v_{c1} with High Speed

3. Signal-time diagram of the step responses of the yaw angle ψ
   - ψ for Low-speed Dynamics
   - Drive maneuver: $(u_n \ = \ 0.1 \ \ \delta_n \ = \ 1.0)$:

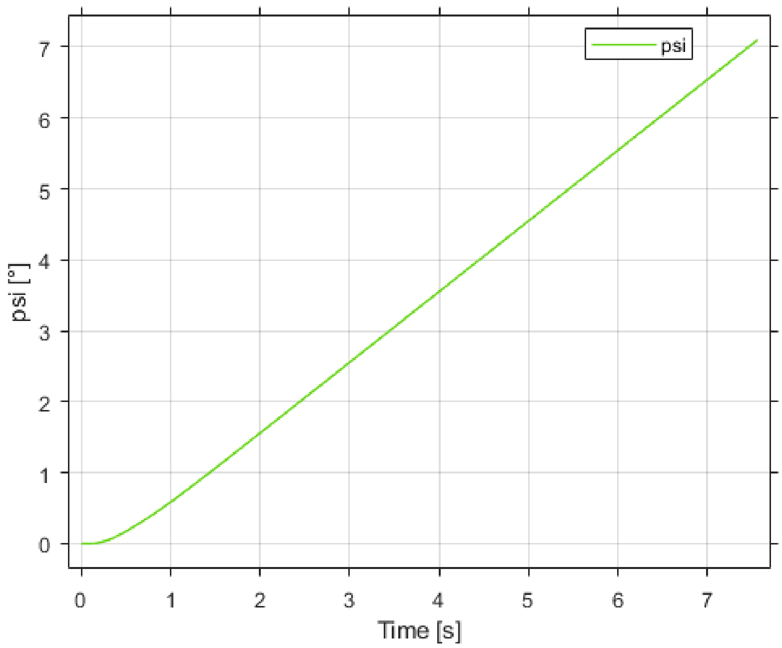


Figure 3: Signal-time diagram of steps response of yaw angle ψ with Low Speed

- ψ für High-speed Dynamics
- Drive maneuver: $(u_n \quad = \ 1.0 \quad \delta_n = \ 1.0)$:

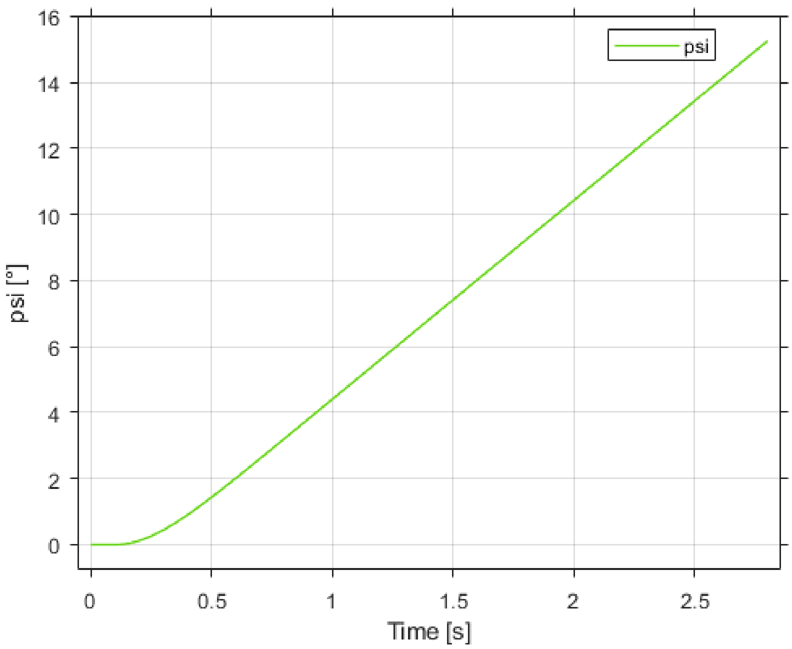


Figure 4: Signal-time diagram of steps response of yaw angle ψ with High Speed

4 Task 6.1 Design of the speed controller

Required lab results:

1. Mathematical expressions and values for T_i and k_r
2. Bode diagram of $G_0(jw)$ including the phase and amplitude edges.
3. Signal-Time-Diagram of the step response of $G_w(s)$
4. Transfer function $G_R^*(z)$ of the discrete-time PI controller
5. Difference equations for calculating the control signal $u_k = u(kT_A)$ and its I component $u_{ik} = u_i(kT_A)$ as a function of the control deviation $e_k = w_k - y_k$
6. MATLAB *script ex6_1.m* for b. and c.

Elaboration of the Task:

1. Mathematical expressions and values for T_i and k_r

$$G_r(s) = k_r \cdot \frac{1 + T_i \cdot s}{T_i \cdot s} \qquad G_s(s) = \left(\frac{k_u}{1 + T \cdot s}\right) \cdot \mathrm{e}^{-T_t \cdot s}$$

$$G_0(s) = G_r(s) \cdot G_s(s) = k_r \cdot k_u \cdot \frac{1 + T_i \cdot s}{T_i \cdot T \cdot s^2 + T_i \cdot s} \cdot e^{-s \cdot T_t}$$

$$s = jw$$

$$\phi_0(\omega) = \arg(G_0\,(j\omega)) = \phi_R(\omega) + \phi_S(\omega)$$

$$\phi_0(\omega) = -\frac{\pi}{2} + \arctan(\omega \cdot T_i) - \arctan(\omega \cdot T) - T_t \cdot w_D$$

$$\boxed{\mathrm{k_r} = \mathrm{k_u} \cdot \frac{\sqrt{1 + (\omega_D \mathrm{T_i})^2}}{((\mathrm{T}\,\omega_D)^2 + 1) \cdot \omega_D \mathrm{T_i}}}$$

1. To calculate k_r and T_i, solve the following equations for k_r and T_i:
   a. Equation 1: $|G_o(jw)| = 1$
   b. Equation 2: $\arg\,(G_o(jw)) = -\pi + \varphi_{Res}$
2. Both equations were solved using MATLAB and the *vpasolve* function:
   a. *vpasolve*: https://de.mathworks.com/help/symbolic/sym.vpasolve.html
3. Values for k_r and T_i:

$$\boxed{k_r = 0.3440 \qquad T_i = 0.2468}$$

2. Bode-Diagram of $G_0(jw)$ including the phase and amplitude edges.

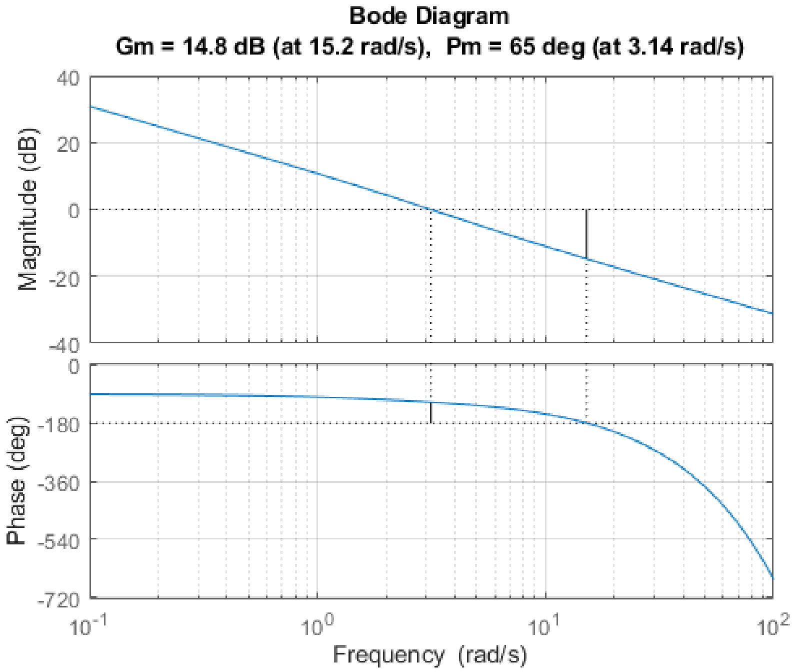


Figure 5: Bode diagram G_0

3. Signal-Time-Diagram of the Step Response of $G_w(s)$

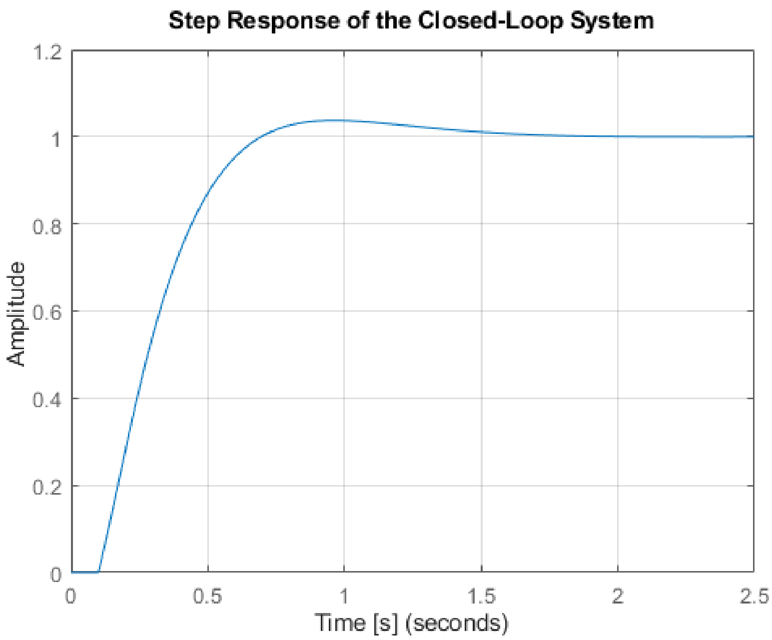


Figure 6: Step response and Overshoot of $G_w(s)$

4. Transfer Function $G_R^*(z)$ oft the time discrete time PI-Controller:

$$G_r(s) = k_r + \frac{k_r}{T_i \cdot s} \qquad s \approx \frac{1 - z^{-1}}{T_A} \qquad T_A = 20\,ms$$

$$G_R^*(z) = \frac{U(z)}{E(z)}$$

$$\boxed{G_R^*(z) = k_r + \frac{k_r \cdot T_A}{T_i \cdot (1 - z^{-1})}}$$

P-Component: $G_{RP}^*(z) = \frac{U(z)}{E(z)} = k_r$

I-Component: $G_{RI}^*(z) = \frac{U(z)}{E(z)} = \frac{k_r \cdot T_A}{T_i \cdot (1 - z^{-1})}$

5. Difference equations for calculating the control signal $u_k = u(kT_A)$ and its I component $u_{ik} = u_i(kT_A)$ as a function of the control deviation $e_k = w_k - y_k$

P-Component: $G_{RP}^*(z) = \frac{U(z)}{E(z)} = k_r$

$$U_p(z) = k_r \cdot E(z) \quad \bullet\!\!-\!\!\circ \quad \boxed{u_{pk} = k_r \cdot e_k}$$

I-Component: $G_{RI}^*(z) = \frac{U(z)}{E(z)} = \frac{k_r \cdot T_A}{T_i \cdot (1 - z^{-1})}$

$$U_i(z) = \frac{k_r \cdot T_A}{T_i} \cdot E(z) + U_i(z) \cdot z^{-1} \quad \bullet\!\!-\!\!\circ \quad \boxed{u_{ik} = \frac{k_r \cdot T_A}{T_i} \cdot e_k + u_{ik-1}}$$

Complete controller:

$$u_k = u_{pk} + u_{ik} \qquad e_k = w_k - y_k$$

$$\boxed{u_k = k_r \cdot e_k + k_r \cdot \frac{T_A}{T_i} \cdot e_k + u_{ik-1}}$$

4. MATLAB-Script *ex6_1.m* for *b*. and *c.*

   - The corresponding MATLAB script *ex6_1.m* can be found in the attached files.

5 Task 6.2 Simulink subsystem for speed control

Required lab results:

1. Simulink Model *s7_template.slx*
2. MATLAB-Script *s6_data.m*
3. Signal-Time-Diagram of the Step Response of the vehicle speed v_r (from Simulink-MiL simulations and on the real MAD system)

Elaboration of the Task:

1. Simulink Model *s7_template.slx*
   - *T*he Simulink model *s7_template.slx* can be found in the attached files. However, this is an old file, the new file s9_*template.slx* also contains this functionality and is designed in accordance with the guidelines.
2. MATLAB Script *s6_data.m*
   - The MATLAB script *s6_data.m* can be found in the attached files.
3. Signal-time diagrams of step responses of the vehicle speed v_r (from Simulink-MiL simulations and on the real MAD system)
   - Drive Maneuver: $v_{max} = 0.5\,\frac{m}{s} \quad \delta_n = \ 0°$

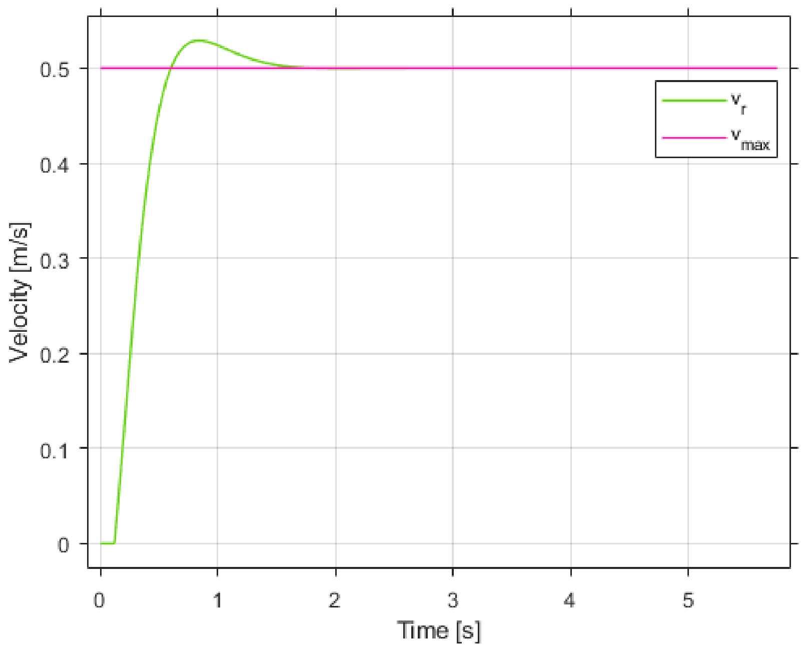


Figure 7: Signal-time diagram of step responses $v_{max} = 0.5\frac{m}{s}$ of the vehicle speed v_r

- Drive Maneuver: $v_{max} = 1.0\,\frac{m}{s} \qquad \delta_n = 0°$

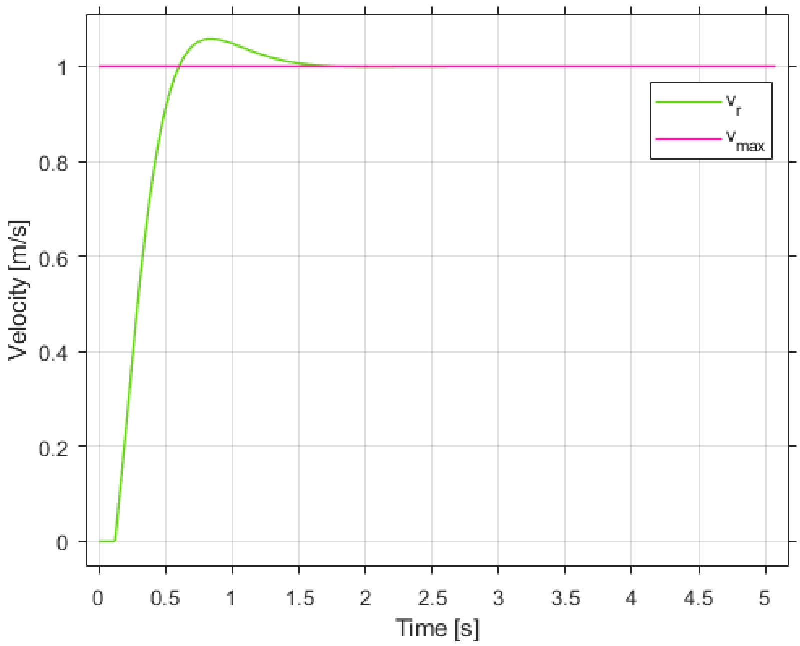


Figure 8: Signal-time diagram of step responses $v_{max} = 1.0\,\frac{m}{s}$ of the vehicle speed v_r

6 Task 7.1 Design of the longitudinal position control

Required lab results:

1. Mathematical expressions for$Y_p(s),\ E_p(s),\ e_y,\ k_p$
2. Value and Unit of k_p
3. Signal-Time-Diagram of the ramp response $y_p(t)$ to $w_p(t) = v \cdot t \cdot h(t)$
4. Root-Locus curves of the control loop as a function of k_p
5. MATLAB-Functions *cd_refpoly_vmax* and *cd_refpoly_ff*
6. Signal Time-Diagrams for $w_p(t),\ y_p(t),\ \dot{w}_p(t),\ \ddot{w}_p(t)$ and $u_{Vp1}(t)$
7. Discrete-time transfer function $G^*_{Vp1}(z)$ for the high-pass component of the feedforward control.
8. Difference equation for the drive signal component u_{Vpk} as a function of u_{Vp1k}
9. Extended MATLAB Script *ex6_1.m* for solving the subtasks and testing the developed MATLAB function.

Elaboration of the Task

1. Mathematical expressions for $Y_p(s),\ E_p(s),\ e_y,\ k_p$

$$Y_p(s) = G_{wp}(s) \cdot W_p(s)$$

$$G_{wp}(s) = \frac{G_{0p}(s)}{1 + G_{0p}(s)}$$

$$E_p(s) = W_p(s) - Y_p(s) \qquad \rightarrow \qquad E_p(s) = \left(1 - G_{wp}(s)\right) \cdot W_p(s)$$

$$w_p(t) = v^* \cdot t \cdot h(t) \;\bullet\!\!-\!\!\circ\; W_p(s) = v^* \cdot \frac{1}{s^2}$$

$$E_{p(s)} = \frac{\left(\frac{T_i T T_t}{k_r k_u} s^4 + \frac{(T T_i + T_t T_i)}{k_r k_u} s^3 + \left(T_i + \frac{T_i}{k_r k_u}\right) s^2 + s\right)}{\left(\frac{T_i T\, T_t}{k_r k_u} s^4 + \frac{(T_i T + T_i T_t)}{k_r k_u} s^3 + \left(T_i + \frac{T_i}{k_r k_u}\right) s^2 + s + (T_i s k_p + k_p)\right)} \cdot \frac{v^*}{s^2}$$

$$e_y = \lim_{t \to \infty} \left(e_p(t)\right) = 10\ cm$$

$$v^* = 0.1 \frac{m}{s}$$

$$e_y = \frac{v^*}{k_p} = \frac{0.1 \frac{m}{s}}{k_p} \rightarrow \boxed{k_p = \frac{v^*}{e_y} = \frac{0.1 \frac{m}{s}}{0.1\ m} = 1 \frac{1}{s}}$$

2. Value and Unit of k_p

$$\boxed{k_p = 1\frac{1}{s}}$$

3. Signal Time Diagram of the Ramp response $y_p(t)$ to $w_p(t) = v \cdot t \cdot h(t)$

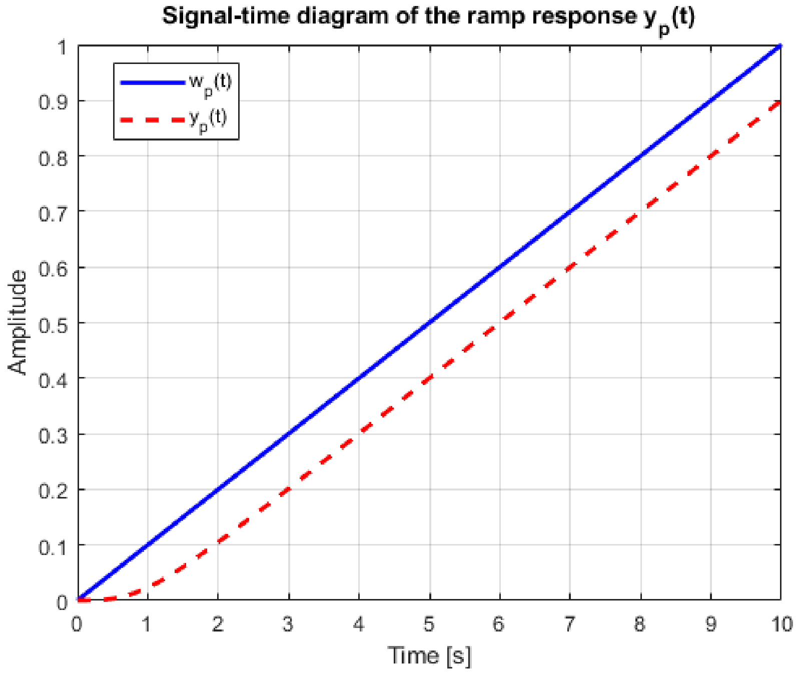


Figure 9: Signal-time diagram of the ramp response $y_p(t)$

4. Root-Locus curves of the control loop as a function of k_p
   - $k_p = 1$:

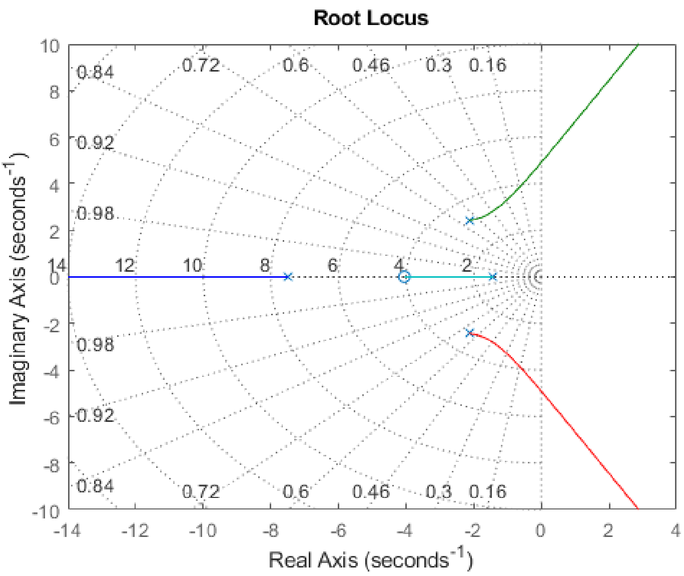


Figure 10: Root. Locus curve with $k_p = 1$

- $k_p = 7.5$:

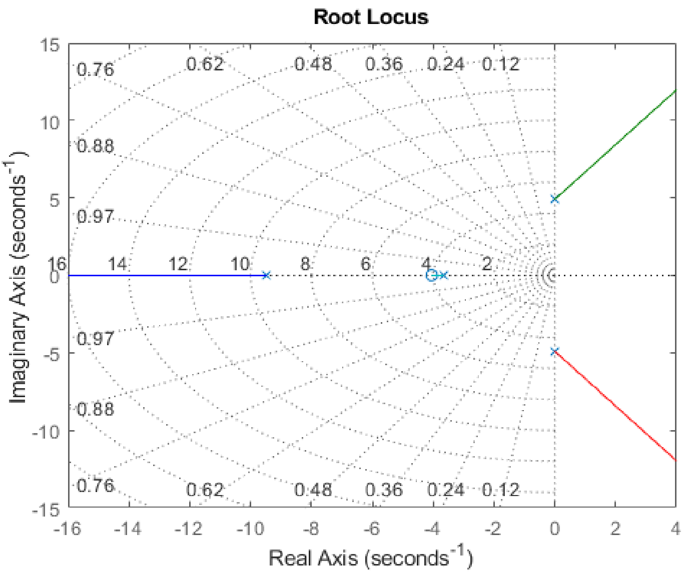
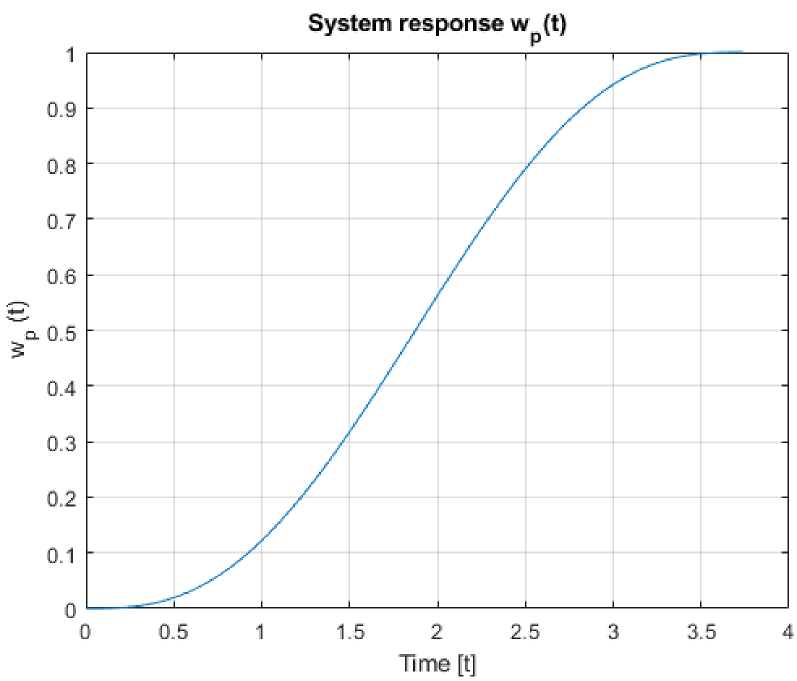


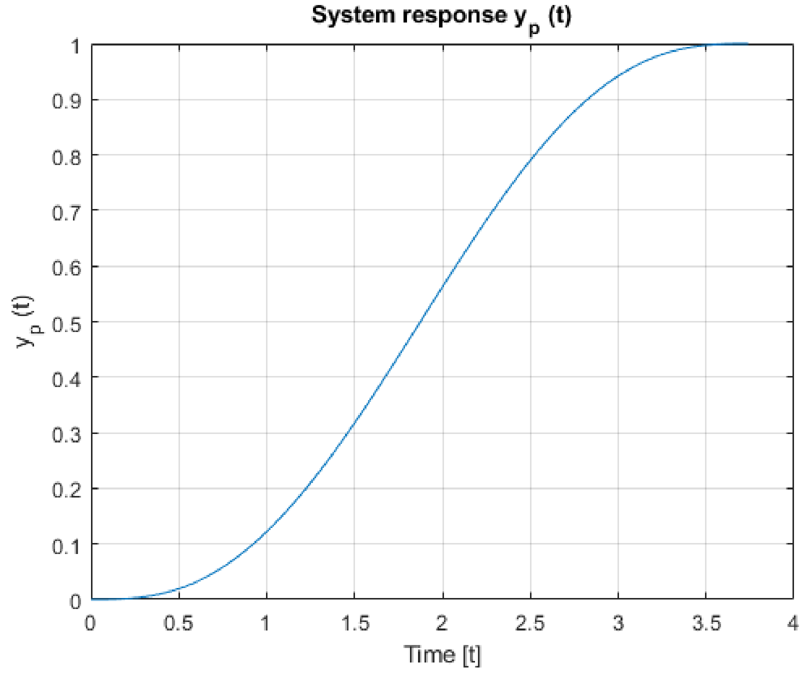
Figure 11: Root Locus curve with $k_p = 7.5$

5. MATLAB-Functions *cd_refpoly_vmax* and *cd_refpoly_ff*
    - The MATLAB scripts for the functions *cd_refpoly_vmax* and *cd_refpoly_ff* can be found in the attached files.

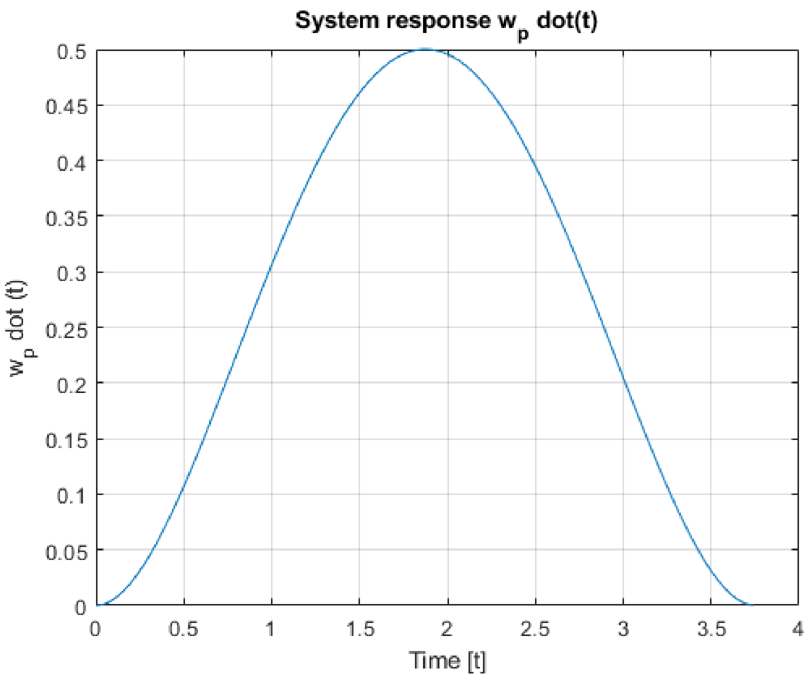
6. Signal Time Diagram for $w_p(t),\ y_p(t),\ \dot{w}_p(t),\ \ddot{w}_p(t)$ and $u_{Vp1}(t)$
    - $w_p(t)$:



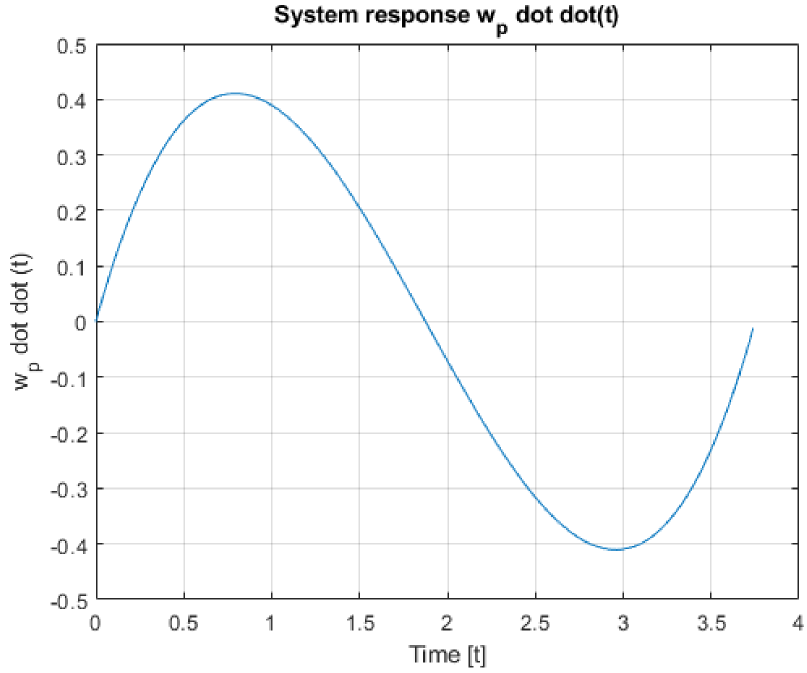
- $y_p(t)$:


System response y p (t)
y p (t)
Time [t]
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
2
2.5
3
3.5
4

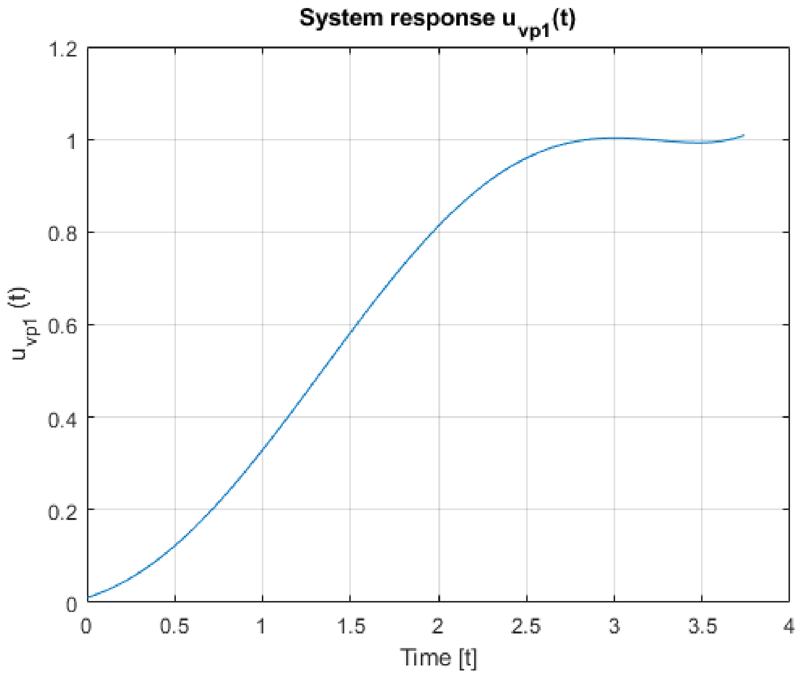
- $\dot{w}_p(t)$:


System response w p dot(t)
w p dot (t)
Time [t]
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.5
1
1.5
2
2.5
3
3.5
4

- $\ddot{w}_p(t)$:



- $u_{Vp1}(t)$:



7. Discrete-time transfer function $G^*_{Vp1}(z)$ for the high-pass component of the feedforward control.

$$G_{Vp1}(s) = \frac{s}{T_i \cdot s + 1}$$

Trapezoidal rule: $s = \frac{2}{T_A} \cdot \frac{z-1}{z+1}$

$$G^*_{Vp1}(z) = \frac{\left(\frac{2}{T_A} \cdot \frac{z-1}{z+1}\right)}{Ti \cdot \left(\frac{2}{T_A} \cdot \frac{z-1}{z+1}\right) + 1}$$

$$\boxed{G^*_{Vp1}(z) = \frac{2 \cdot (z-1)}{T_A \cdot (z+1) + 2T_i \cdot (z-1)}}$$

8. Difference equation for the drive signal component u_{Vpk} as a function of u_{Vp1k}

$$G(z) = \frac{A(z)}{B(z)} = \frac{a_0 + a_1\,(z-1) + a_2(z-2) + \cdots}{b_0 + b_1(z-1) + b_2(z-2) + \cdots}$$

$$G^*_{Vp1}(z) = \frac{2z - 2}{T_A \cdot z + T_A + 2T_i \cdot z - 2T_i}$$

$$U_{vp}(z) = U_{vp1}(z) \cdot G_{vp1}(z)$$

$$G^*_{Vp1}(z) = \frac{\frac{2}{T_A} \cdot (z-1)}{(z+1) + \frac{2T_i}{T_A}(z-1)} = \frac{U_{vp}(z)}{U_{vp1}(z)}$$

$$\frac{2}{T_A} \cdot (z-1) \;\cdot U_{vp1}(z) = U_{vp}(z) \cdot \left[(z+1) + \frac{2T_i}{T_A}(z-1)\right] \qquad | \cdot z^{-1}$$

$$\left(\frac{2}{T_A} - z^{-1} \cdot \frac{2}{T_A}\right) \cdot U_{vp1}(z) = U_{vp}(z) \cdot \left(1 + \frac{2T_i}{T_A}\right) + z^{-1} \cdot U_{vp}(z) - z^{-1} \cdot \frac{2T_i}{T_A} \cdot U_{vp}(z)$$

$$\bullet\!\!-\!\!\circ$$

$$\frac{2}{T_A} \cdot u_{vp1k} - \frac{2}{T_A} \cdot u_{vp1k-1} \;= u_{vpk} + u_{vpk} \cdot \frac{2T_i}{T_A} \;+ u_{vp1k-1} - \frac{2T_i}{T_A} \cdot u_{vp1k-1}$$

$$\frac{2}{T_A} u_{vp1k} - \left(\frac{2}{T_A} + 1 - \frac{2T_i}{T_A}\right) \cdot u_{vp1k-1} \;= u_{vpk}\left(1 + \frac{2T_i}{T_A}\right)$$

$$\boxed{u_{vpk} = \left[\frac{2}{\mathrm{T_A}} \cdot \mathrm{u_{vp1k}} - \left(1 + \frac{2}{\mathrm{T_A}} - \frac{2\mathrm{T_i}}{\mathrm{T_A}}\right) \cdot \mathrm{u_{vp1k-1}}\right] \cdot \frac{1}{1 + \frac{2T_i}{T_A}}}$$

9. Extended MATLAB script *ex6_1.m* for solving the subtasks and testing the developed MATLAB functions.
    - The MATLAB scripts *ex7_1.m* can be found in the attached files.

7 Task 7.2 Extension of the Simulink subsystem Control Software for

Required Lab Results:

1. Extended Simulink model *s7_template.slx*
2. Extended model data file *s6_data.m*
3. Signal Time diagram of the target position $w_p(t)$ and the actual position $y_p(t) = x(t)$ from Simulink-MiL simulations and on the real MAD system

---

Elaboration of the Task:

1. Extended Simulink model *s7_template.slx*
    - The required file *s7_template.slx* can be found in the attached files. However, this is an old file, the new file s9_*template.slx* also contains this functionality and is designed in accordance with the guidelines.
2. Extended model data file *s6_data.m*
    - The required file *s6_data*.m ca be found in the attached files.
3. Signal Time diagram of the target position $w_p(t)$ and the actual position $y_p(t) = x(t)$ from Simulink-MiL simulations and on the real MAD System
    - Drive maneuver: $v_{max} = 0.2\frac{m}{s}$ $\quad \delta_n = auto \quad xManeuverEnd = 1m$
    - $y_p(t)$ and $w_p(t)$ are identical.

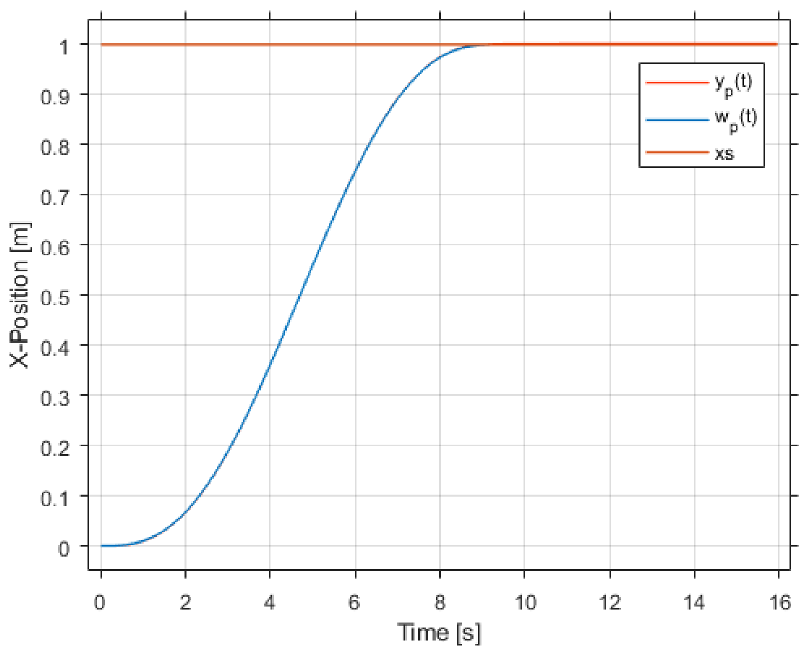


Figure 12: Signal-time diagram of the target position $w_p(t)$ and the actial position $y_p(t)$

8 Task 8.1 Straight track curve

Required Lab Task:

1. $\vec{s}(x),\ \vec{t}(x),\ \vec{n}(x),\ K(x)$

Elaboration of the Task:

1. $\vec{s}(x),\ \vec{t}(x),\ \vec{n}(x),\ K(x)$

$$\vec{s}(0) = \vec{s}_0 = \begin{pmatrix} s_{01} \\ s_{02} \end{pmatrix}$$

$$\psi(0) = \psi_0$$

$$\vec{s}(x) = \begin{pmatrix} s_1(x) \\ s_2(x) \end{pmatrix} = \begin{pmatrix} s_{01} - x \cdot \sin(\psi_0 - 90°) \\ s_{02} - x \cdot \cos(\psi_0 - 90°) \end{pmatrix}$$

$$\boxed{\vec{s}(x) = \begin{pmatrix} s_{01} + x \cdot \cos(\psi_0) \\ s_{02} + x \cdot \sin(\psi_0) \end{pmatrix}}$$

t is the tangential vector indicating the direction of the straight line.

The tangential vector $t(x)$ is given by:

$$\vec{t}(x) = \vec{s}(x)\frac{d}{dx} = \begin{pmatrix} s_1{}'(x) \\ s_2{}'(x) \\ 0 \end{pmatrix}$$

$$\vec{t}(x) = \vec{s'}(x) = \begin{pmatrix} \cos(\psi_0) \\ \sin(\psi_0) \\ 0 \end{pmatrix}$$

$$\boxed{\vec{t}(x) = \begin{pmatrix} \cos(\psi_0) \\ \sin(\psi_0) \\ 0 \end{pmatrix}}$$

As this is a straight path curve, the tangential vector is constant, and the normal vector n(x) is perpendicular to t(x). The following applies to the normal vector:

$$\vec{n}(x) = \vec{t}(x)\frac{d}{dx} = \vec{s}(x)\frac{d^2}{dx^2} = \begin{pmatrix} s_1{}''(x) \\ s_2{}''(x) \\ 0 \end{pmatrix}$$

$$\vec{n}(x) = \vec{s''}(x) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\boxed{\vec{n}(x) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}}$$

In general, the following must apply: $t(x) \times n(x) = 0$

The curvature κ of a straight line is zero everywhere, as the line is not curved.

$$r(x) = 0 \quad \rightarrow \quad \boxed{\kappa(x) = \frac{1}{r} = \sqrt{s_1''(x)^2 + s_2''(x)^2} = \sqrt{0 + 0} = 0}$$

9 Task 8.2 MODBAS-CAR-Function for Clothoid

Required Lab Task:

1. MATLAB-Script *mbc_clothoid_create.m*
2. MATLAB-Script *mbc_clothoid_get_points.m*
3. Modified MATLAB-Script *s6_data.m*

Elaboration of the Task:

1. MATLAB-Script *mbc_clothoid_create.m*

   The script mbc_clothoid_create.m can be found in the attached files.

2. MATLAB-Script *mbc_clothoid_get_points.m*

   The script *mbc_clothoid_get_points.m* can be found in the attached files.

3. modified MATLAB-Script *s6_data.m*

   The Script *s6_data.m* can be found in the attached files.

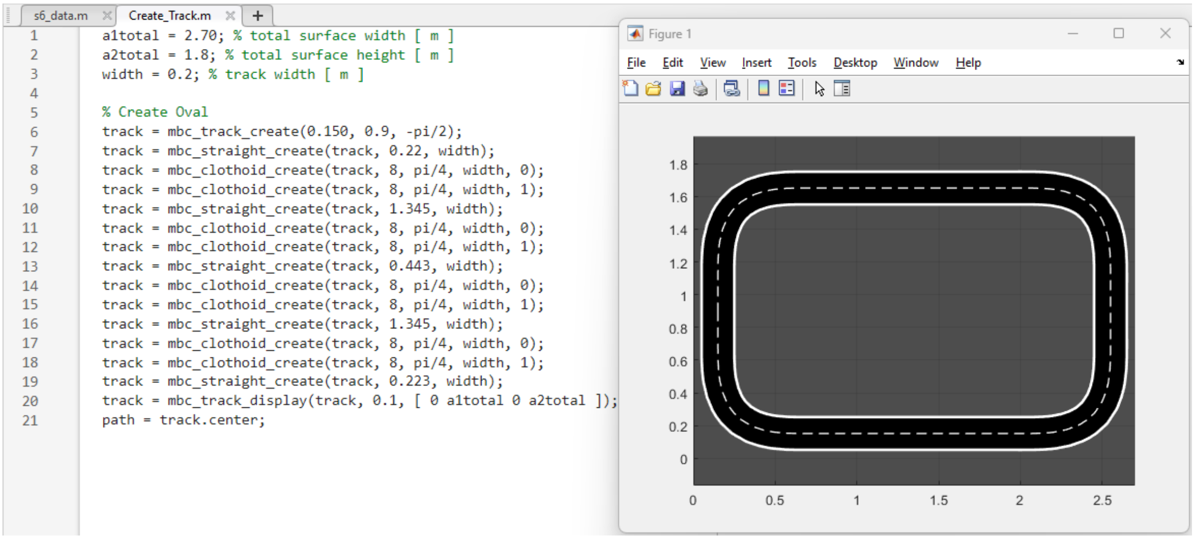


Figure 13: Created Oval track and code.

10 Task 9.1 Simulink subsystem control software for speed and trajectory control

Required laboratory results:

1. Signal-time diagrams for yaw angle $\psi(t)$
2. tar.gz or zip file containing the following files.
   a. *s7_template.slx*
   b. *s6_data.m*

Elaboration of the tasks:

1. Signal-time diagrams for yaw angles $\psi(t)$
   a. $v_{max} = 0.5\ \frac{m}{s}$ Car drove two Rounds.

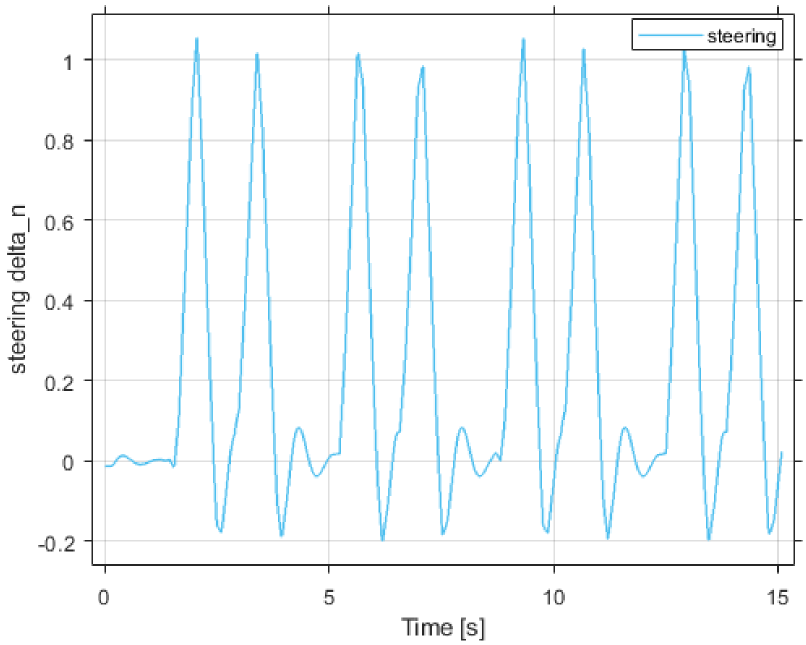


Figure 14: Time-signal diagram of the steering angle, Car drove two rounds with $v_{max} = 1.0\frac{m}{s}$

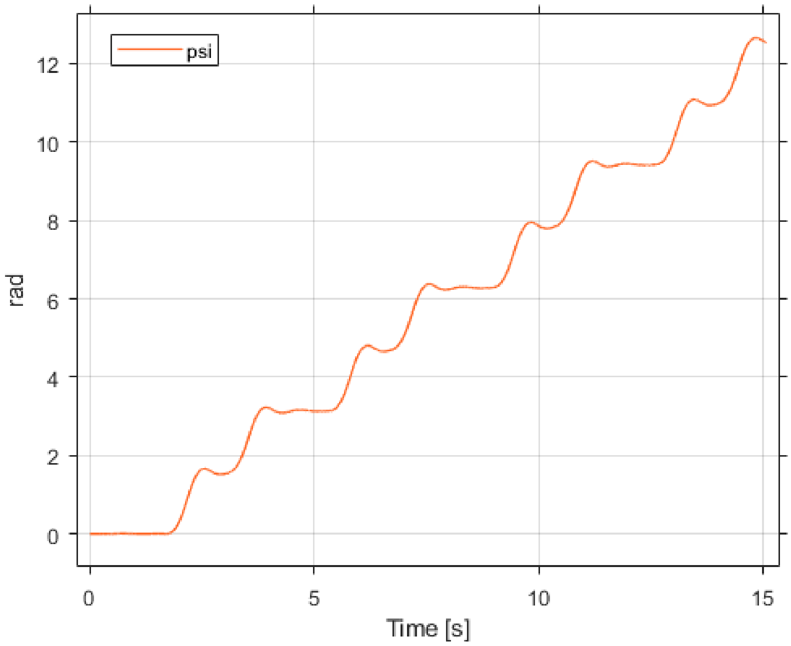


Figure 15: Singal-time diagram of the yaw angle $\psi(t)$. Car drove two rounds with $v_{max} = 1.0\,\frac{m}{s}$

2. tar.gz or zip file containing the following files: *s7_template.slx* and *s6_data.m*
   - All files can be found in the zip, the s9_template is the final project, the other files are only previous versions that have been attached for completeness.

11 Exercise 20.1 Safety Halt

Requirements

1. In case of critical camera or image processing or self-localization faults, the ego car shall stop with minimal braking distance.

2. As soon as the ego car moves outside of the pre-defined, safe driving area, the ego car shall stop with minimal braking distance.

3. The ego car shall not leave this safe driving area under any circumstances.

4. These safety methods shall be tested in SiL and in real driving.

Rationale

- Critical faults are faults that may lead to intolerable and not manageable dangers for ego car or environment.
- Critical faults must be detected and reacted on within pre-defined time intervals.
  - ISO26262: (fault handling time interval FHTI) = (fault detection time interval FDTI) + (fault reaction time interval FRTI) < (fault tolerance time interval FTTI)
- Under presence of critical faults, the ego car must be transferred to a safe state.

Possible Concepts

- Critical faults can be detected by monitoring topic /mad/caroutputsext.
- Either when messages on /mad/caroutputsext are lost or no message has been received with carid 0 for greater than the jitter tolerance time interval $3\ T_A$, critical faults in camera / image processing or self-localization are present.
- The safe driving area is defined as the complete rectangular surface minus a safe boundary of 50mm.
- To initiate a safety halt, the driving command CMD_HALT may be applied.

Solution:

- To ensure that the car stops as soon as it leaves the prescribed area, we compare the x and y position of the car with the limits of the track. If the car exceeds the limit or does not exceed the lower limit, the car is stopped using CMD_HALT.
  This can be recognised by the following image, in which the car stops as soon as it has crossed the safety area.

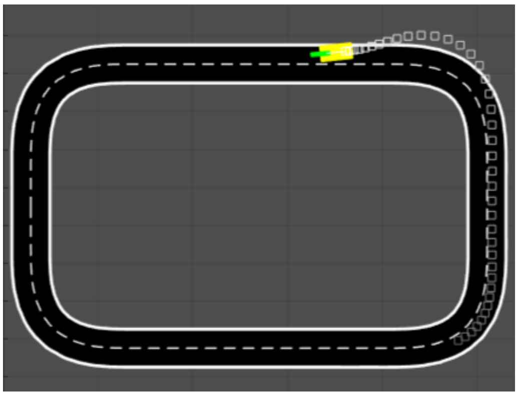


Figure 16: Example of Car stop after Safety halt.

- We also have a query that checks whether the correct CARID has been selected, if not the car is stopped using CMD_HALT. For this we have built in a manual switch which can be set to one to check if the car stops when the wrong CARID is received.
- The last point we have realized is that the car receives a steering impulse just before it leaves the track. This is to prevent the car from leaving the track in the first place. In addition, the speed is reduced as soon as the specified area is reached to make it easier to steer away from the end of the lane. However, this works only partially well, as the track is very thin and therefore there is hardly any room for a buffer to ensure this maneuver.
  The area is defined in such a way that as soon as the car comes closer than 5 cm to the end of the road, the car receives an offset of 0.4 on the steering so that it drives further away from the end of the road. In addition, the pedal position is set to 0.05 to slow the car down.
  To be able to perform this maneuver, we need the pedals and steering to actively adjust the car's ride.

In the following illustrations, the steering and the pedals are first displayed without the safty_halt, then the same values are displayed again while the safety_halt was activated.

- Steering without safety_halt ($v_{max} = 1.35$):

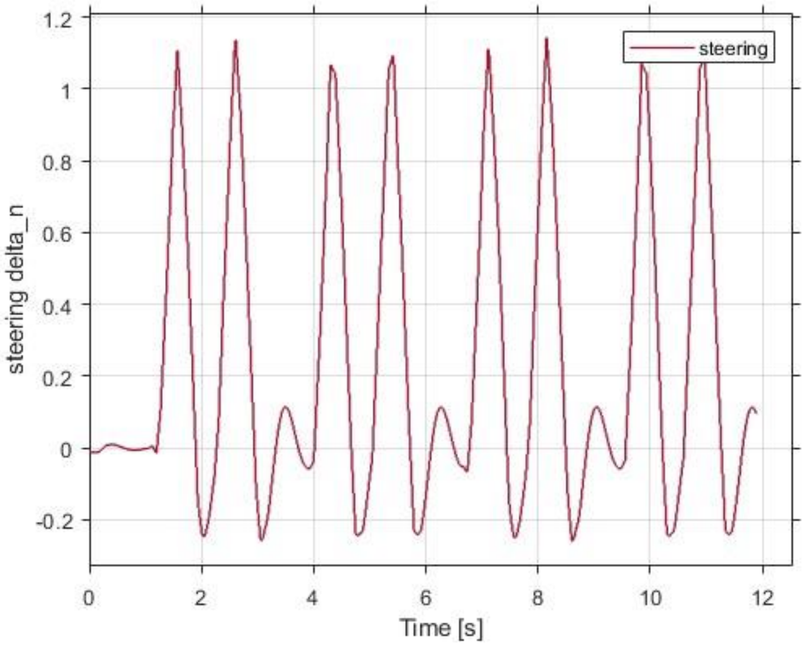


Figure 17: Time-signal diagram of δ_n. Car drove one round without Safety Halt.

- pedals without safety_halt ($v_{max} = 1.35$):

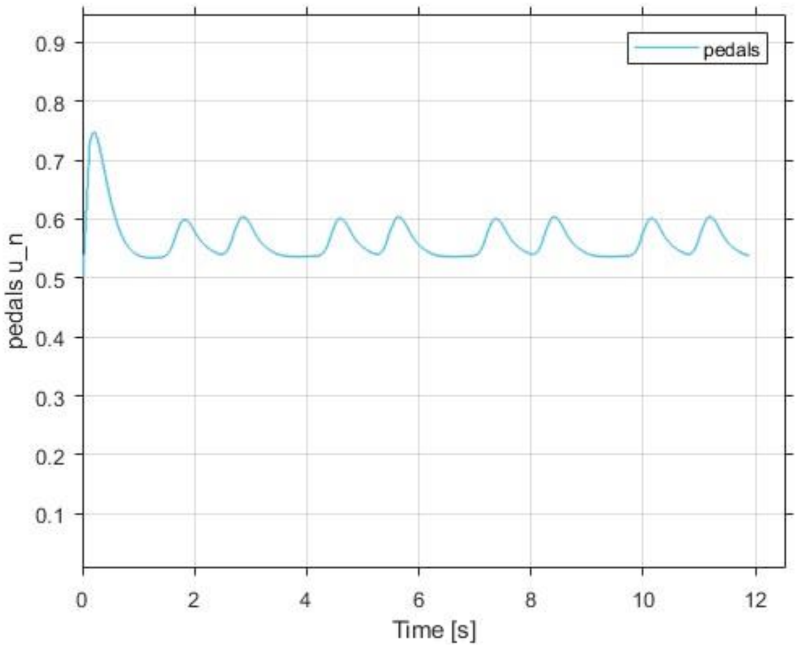


Figure 18: Time-signal diagram of u_n. Car drove one round without Safety Halt.

- Steering with safety_halt activated ($v_{max} = 1.35$):

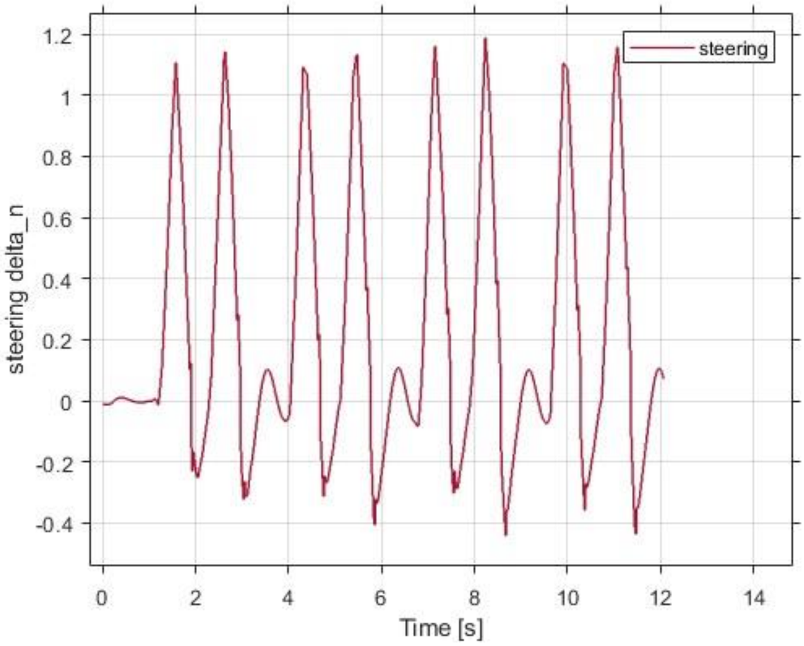


Figure 19: Time-signal diagram of δ_n. Car drove one round with Safety halt.

- pedals with safety_halt activated ($v_{max} = 1.35$):

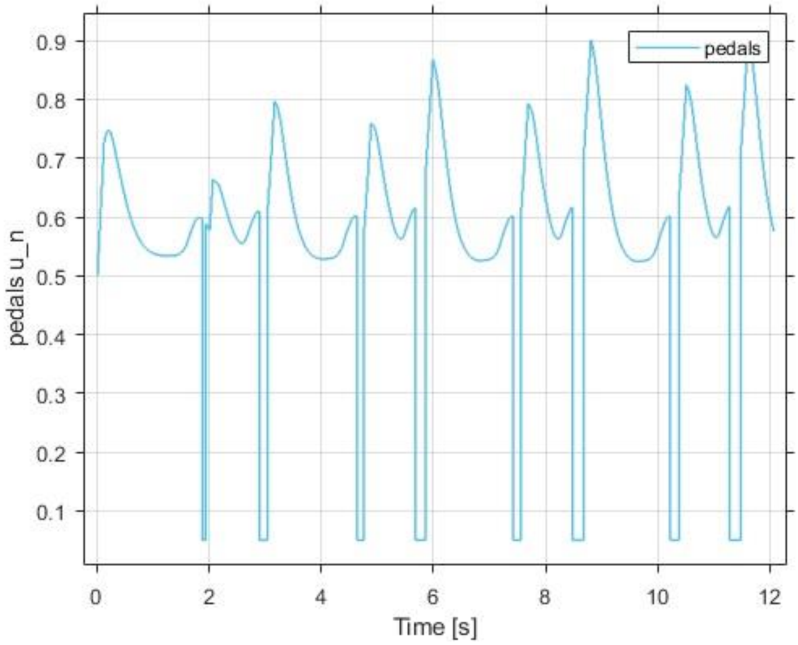


Figure 20: Time-signal diagram of u_n. Car drove one round with Safety halt.

- We have also built in a manual switch that can be used to activate or deactivate the entire Safety halt.

- If you compare the plots, you can clearly see that the pedals are reduced several times, this is because the car enters the area where the steering and speed are adjusted. This is to prevent the car from leaving the track. The car drives two complete laps here and enters the described area once in each of the curves. The difference between the steering and the original plot is not so obvious, but it can be recognised that the steering goes more into minus than before, this is due to the offset that we set on the suitable steering. However, as the Path following control immediately counteracts this, the counter-steering of our function is immediately cancelled out.

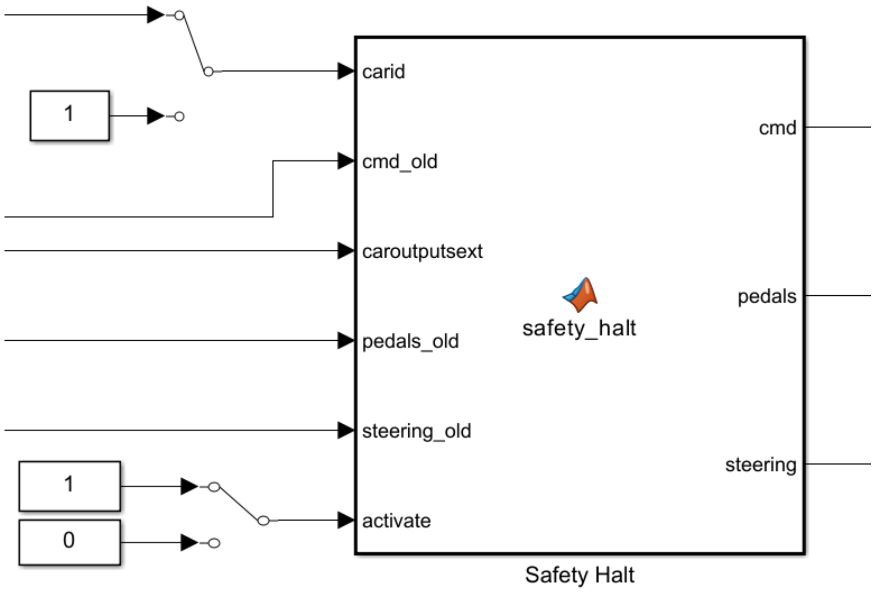


Figure 21: Safety Halt MATLAB Function Block