FH JOANNEUM
Elektronik und
Computer Engineering

Balanbot
Model Based Design

REIBENSCHUH Matthias, SCHNABL Maximilian
ECE 17

Contents

Chapter 1

Model in the Loop

1.1 Task Description

In the first laboratory session a complex system, an inverted pendulum (Balanbot) as it is represented in fig. 1.1, is modeled. Therefore the system equations, representing the dynamic behavior of the Balanbot, are derived. The first step is to carry out the system in its non-linear form and afterwards linearize it. The linear model is then converted to its transfer function using the Laplace transformation.
Finally, in order to keep the system in balance, a controller is designed.

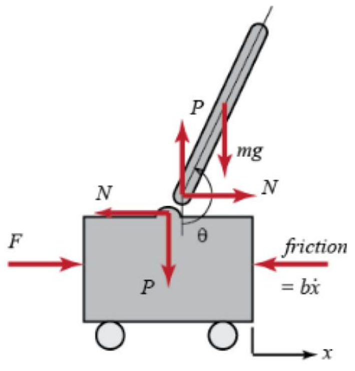


Figure 1.1: System Layout (Source: Balanbot Guidelines)

With following parameters:

x	cart's position
$\dot{x}$	cart's velocity
$\ddot{x}$	cart's acceleration
Θ	pendulum's position (angle)
$\dot{\Theta}$	angular velocity
$\ddot{\Theta}$	angular acceleration
m	mass of pendulum
M	mass of cart
b	car's friction coefficient
l	length to pendulum's center of mass
J	moment of inertia of the pendulum
F	force applied by motors
N	interaction force between cart and pendulum in x direction
P	interaction force between cart and pendulum in y direction
g	gravitational constant

1.2 Model Development

1.2.1 Deriving the system equations

Based on Newton's laws, the system equations are derived. An angle of zero (fig. 1.2) indicates the upright position of the balanbot, which means it is perfectly balanced.

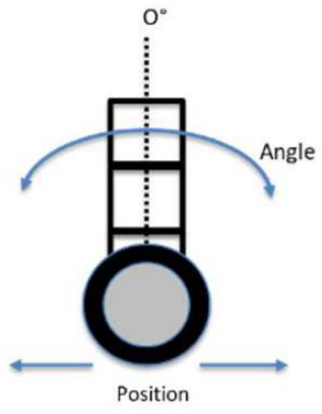


Figure 1.2: Zero Angle Position (Source: Balanbot Guidelines)

The system equaions were derived by the lecturer during the laboratory session to the following:

$$\ddot{x} = \frac{F + ml \cdot \sin(\Theta) \cdot \dot{\Theta}^2 - ml \cdot \cos(\Theta) \cdot \ddot{\Theta} - \mu \dot{x}}{M + m} \tag{1.1}$$

$$\ddot{\Theta} = \frac{mlg \cdot \sin(\Theta) - ml\ddot{x} \cdot \cos(\Theta)}{J + ml^2} \tag{1.2}$$

Derivation missing!

1.2.2 Implementation in MATLAB/SimuLink

Based on the equations 1.1 & 1.2 the SimuLink-model is implemented as it can be seen in fig. 1.3. The parameters are saved in an .m-file and defined as follows:

```
m = 0.1756;
M = 0.7160;
g = 9.810;
J = 0.001;
l = 0.11;
mue = 0.1;
```

Listing 1.1: Listing of the defined parameters of the system

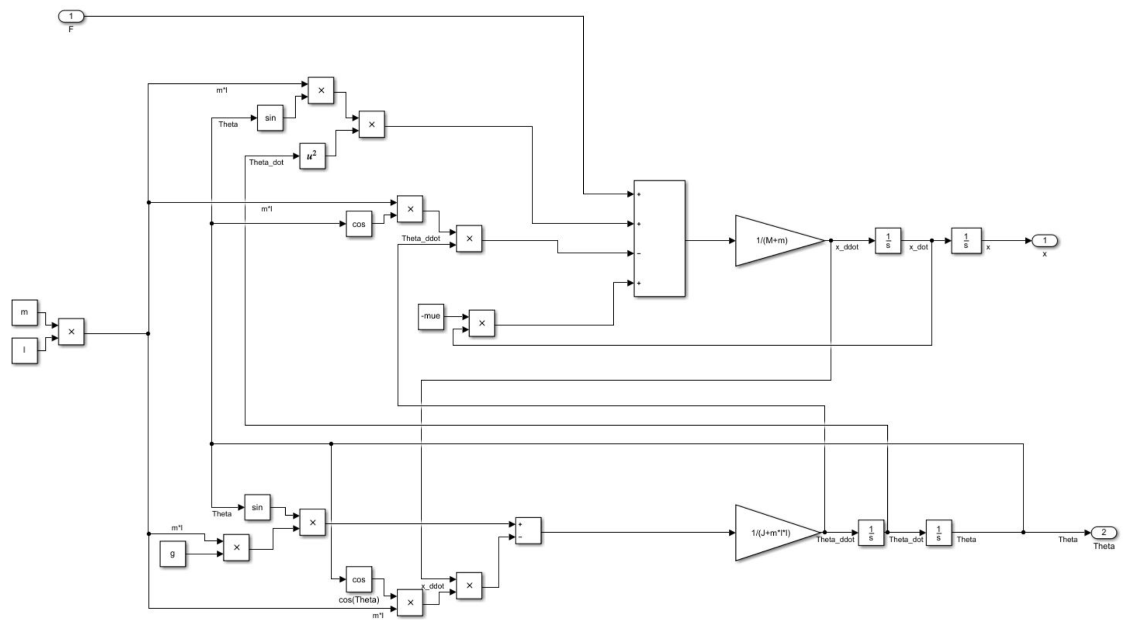


Figure 1.3: SimuLink-model of the system equations

Deriving the transfer functions of the system

To get the transferfunctions for pendulum's position $(\Theta/F(s))$ and the cart's position $(x(s)/F(s))$, the system equations have to be linearized first. This is done by replacing every cos(x) with the constant value of 1 and every sin(x) with x. Thus resulting to following equations: (In the following μ is described as y)

Why is this valid?

$$\ddot{x} = \frac{F + ml\Theta\dot{\Theta}^2 - ml\ddot{\Theta} - y\dot{x}}{M+m} \tag{1.3}$$

$$\ddot{\Theta} = \frac{mlg\Theta - ml\ddot{x}}{J+ml^2} \tag{1.4}$$

Now the transferfunctions can be derived, starting with the Laplace transformation of the pendulums equation and describing Θ afterwards

$$\Theta s^2 = \frac{mgl\Theta}{J+ml^2} - \frac{mlxs^2}{J+ml^2} \tag{1.5}$$

$$\Theta s^2(J+ml^2) - mlg\Theta = -mlxs^2$$

$$\Theta = \frac{-mlxs^2}{s^2(J+ml^2) - mlg} \tag{1.6}$$

Because of the initial condition of $\Theta = 0$ we cann assume that $\dot{\Theta} \cong 0$. This leads to following transformed equation for the cart's position:

$$xs^2 = \frac{F - ml\Theta s^2 - yxs}{M+m} \tag{1.7}$$

Placing Θ in the equation for the cart's position results in:

$$xs^2(M+m) = F - ml\frac{-mlxs^2}{s^2(J+ml^2) - mlg}s^2 - yxs \tag{1.8}$$

then x needs to be isolated

$$x(s^2(M+m)+ys-\frac{m^2l^2xs^4}{s^2(J+ml^2)-mlg})=F \tag{1.9}$$

to get $(x/F(s))$ we divide by F and by the term in the brackets, which is multiplied by x, so we get the following

$$\frac{x}{F}=\frac{1}{\frac{(s^2(M+m)+ys)(s^2(J+ml^2)-mlg)-m^2l^2s^4)}{s^2(J+ml^2)+mlg}} \tag{1.10}$$

solving this, leads to following transferfunction

$$\frac{x}{F(s)}=\frac{s^2(J+ml^2)-mlg}{s^4((M+m)(J+ml^2)-m^2l^2)+s^3(y(J+ml^2))-s^2(mlg(M+m))-s(mlgy)} \quad ✓ \tag{1.11}$$

Then the second transferfunction is derived by exchanging x with the derived term of x

$$xs^2(M+m)+yxs=F-ml\Theta s^2 \tag{1.12}$$
$$x(s^2(M+m)+ys)=F-ml\Theta s^2$$
$$x=\frac{F-ml\Theta s^2}{s^2(M+m)+ys} \tag{1.13}$$

$$\Theta(s^2(J+ml^2)-mlg)=-mls^2\frac{F-ml\Theta s^2}{s^2(M+m)+ys} \tag{1.14}$$

To get the form $\Theta/F(s)$ the equation has to be rearranged:

$$\Theta(s^2(J+ml^2)-mlg)(s^2(M+m)+ys)=m^2l^2\Theta s^4-Fmls^2 \tag{1.15}$$
$$\Theta(s^4((J+ml^2)(M+m)-m^2l2)+s^3(y(J+ml^2))-s^2(mlg(M+m))-s(ymlg)=-Fmls^2 \tag{1.16}$$

$$\frac{\Theta}{F}=\frac{\overset{+}{-}s^2ml}{s^4((J+ml^2)(M+m)-m^2l^2)+s^3(y(J+ml^2))-s^2(mlg(M+m))-s(ymlg)} \quad ✓ \tag{1.17}$$

1.3 System Analysis and Control

1.3.1 Analyzing the open loop response

Applying Zero Force

The first test for the model-in-the-loop is to apply zero force at the input (fig. 1.4), the result can be seen in fig. 1.5.

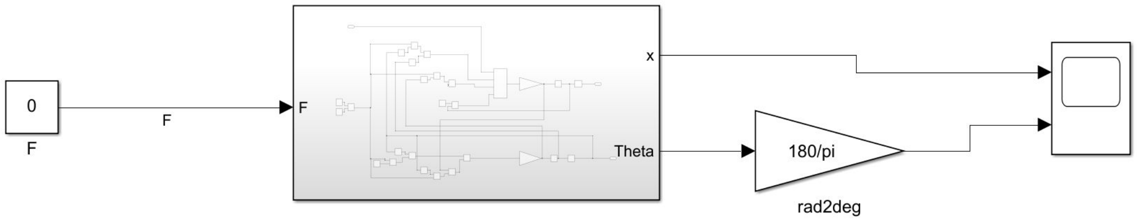
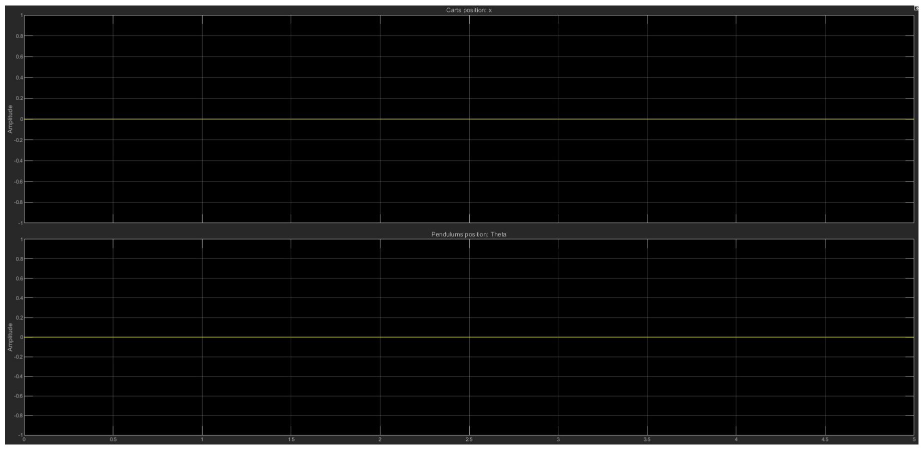


Figure 1.4: Applying zero force at the input



visibility, labels etc.

Figure 1.5: Response of the system while applying zero force

As it can be seen, the result of applying zero force is as expected. The cart's and the pendulum's postition does not change over time, the balanbot keeps it initial angle ($\Theta = 0°$) and position ($x = 0$). The system is stable.

Initial Condition for Θ

In the next step, Θ is initialzied with an angle of $\Theta = 5°$ (fig. 1.6 & fig. 1.7).

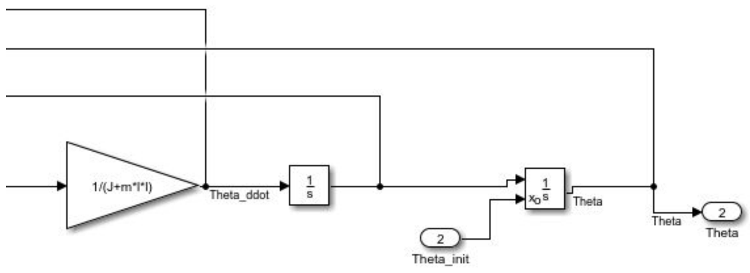


Figure 1.6: Initializing Θ with an angle of $\Theta = 5°$

Optional: The integrator offers a initial condition parameter!

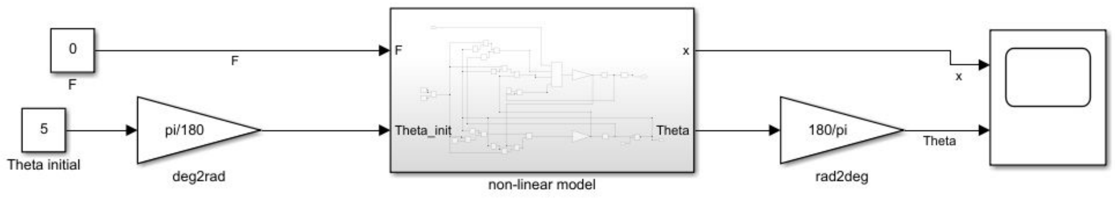


Figure 1.7: Schematic of the sub-model to set an initial parameter for Θ

Since the system is not controlled in any way, it won't stay stable after setting an initial value for Θ. The output oszillates as it can be seen in the following (fig. 1.8).

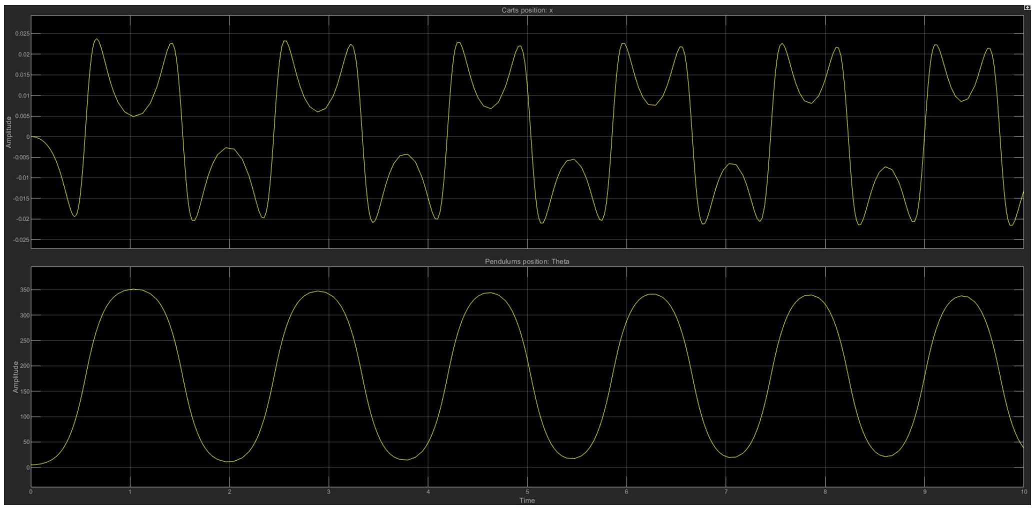


Figure 1.8: Response of the system after setting an initial value for Θ

Analyzing the poles and zeros

The poles of the system are calculated using the MatLab-function „*pole()*"(the result can be seen in table 1.1) and are then drawn using the function „*pzmap()*"(fig. 1.9).

Table 1.1: Poles of the system

0,0000	
8,3591	???
-8,3765	
-0,1122	
0,0000	
8,3591	
-8,3765	
-0,1122	

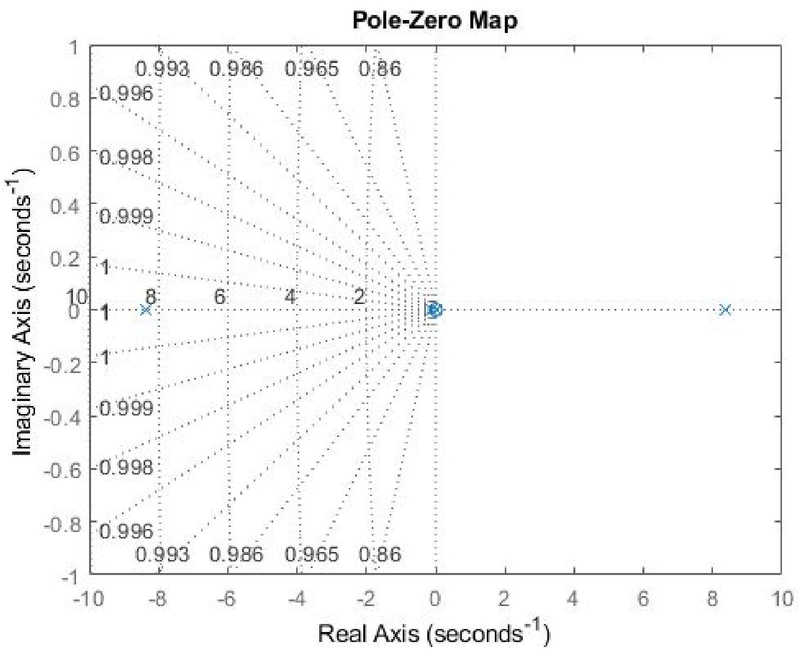


Figure 1.9: Pole-Zero-Map showing the poles of system-transferfunction

What does it mean?

1.3.2 Applying a proportional control

For a proportional control, the model is extended by a feedback loop with a proportional gain (kP, shown in fig. 1.10). Then a suitable kP shall be found by changing its value (starting with a kP = 1) and observing the system's response, this can be seen in fig. 1.11 - 1.15. The oscillation period is measeured at the first wave.

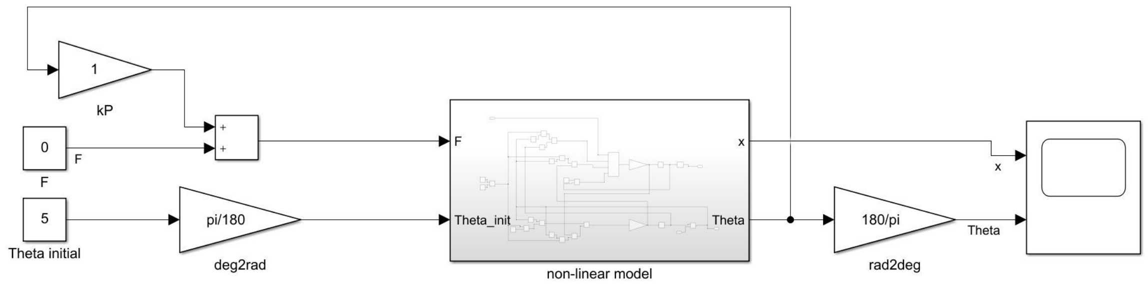


Figure 1.10: Using a proportional gain in a feedback loop

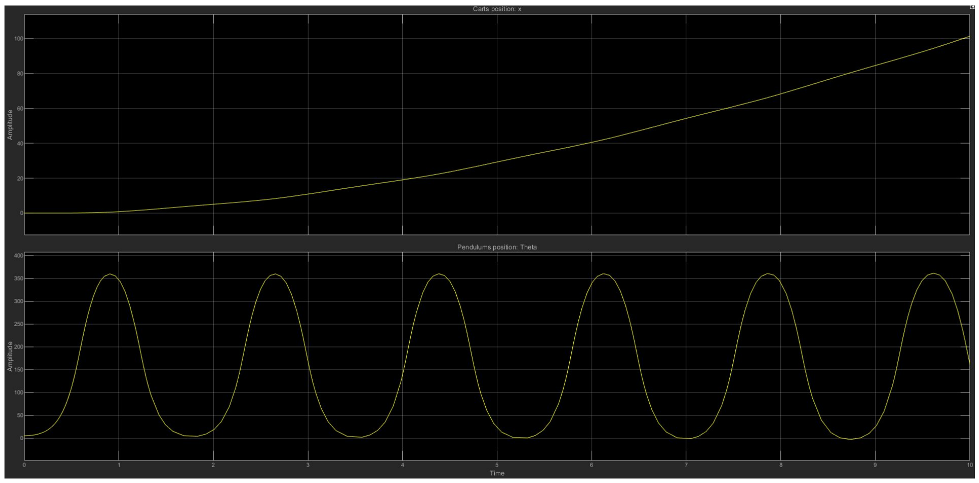


Figure 1.11: Systems response with a gain in the feedback loop of kP = 1

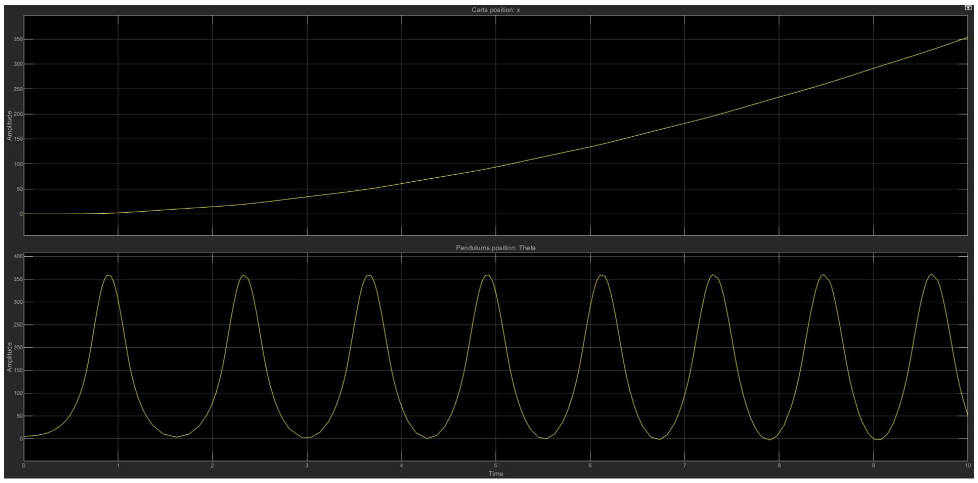


Figure 1.12: Systems response with a gain in the feedback loop of kP = 4

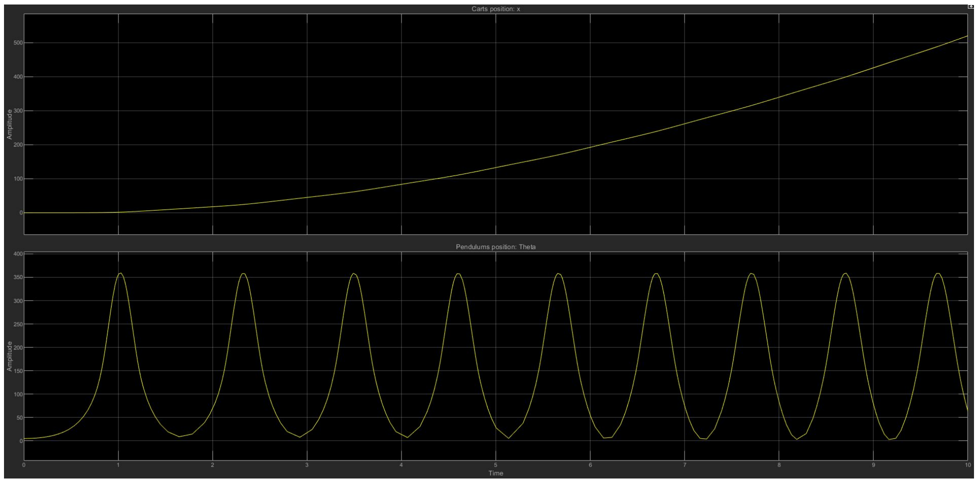


Figure 1.13: Systems response with a gain in the feedback loop of kP = 6

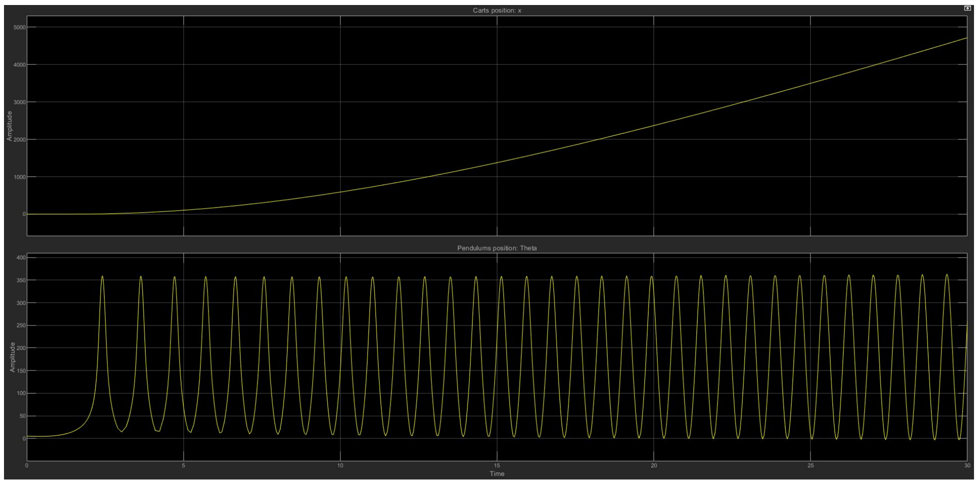


Figure 1.14: Systems response with a gain in the feedback loop of kP = 9

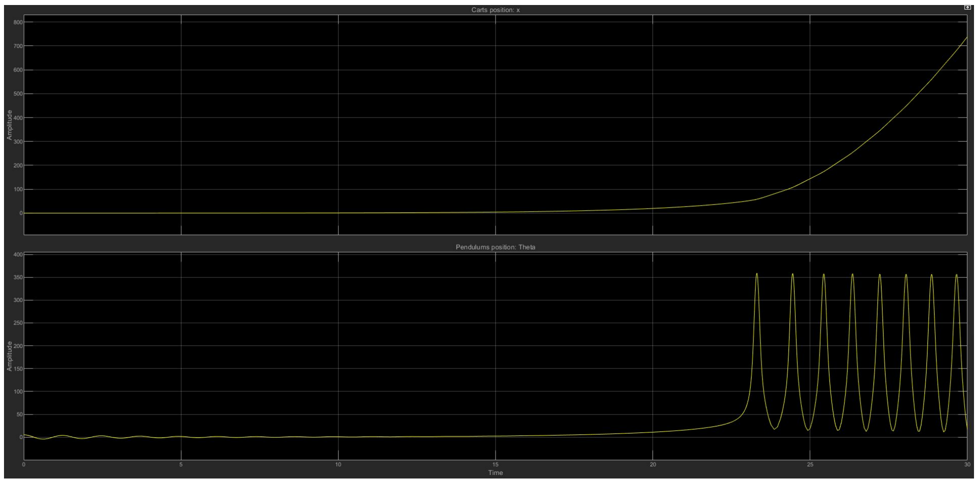


Figure 1.15: Systems response with a gain in the feedback loop of kP = 12

Table 1.2: Different values for kP

kP	Oscillation period(sec)	stable response (Y/N)
1	1,749	N
4	1,427	N
6	1,301	N
9	1,236	N
12	1,136	N
12,5	1,117	N

As it can be seen in the previous figures, the system won't stay stable with a proportional constant only, but it is possible to keep the system stable for a certain period of time.

1.4 Linearization

Linearization process

It is required to linearize the system equations, since our approaches for modelling and control are restricted to linear time invariant systems. For this, every sin(Θ)-function is replaced by Θ and every cos(Θ)-function is replaced by 1, as described in section1.2.2 and can be used to represent the model as long as the system values do not differ much from the chosen linerisation points (angle of zero degrees). The resulting model can be seen in fig. 1.16.

Implementation of the linear model

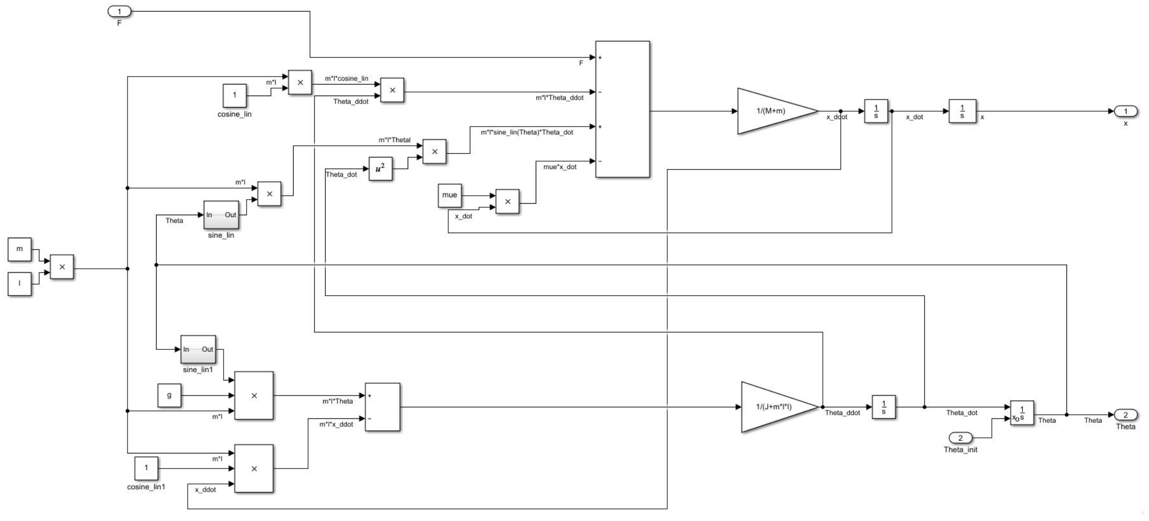


Figure 1.16: SimuLink model of the linear system

For the comparison between the linear and the non-linear model, a submodel of the linear model is created and then copied to the non-linear to use the same inputs (fig. 1.17) parallel for both models.

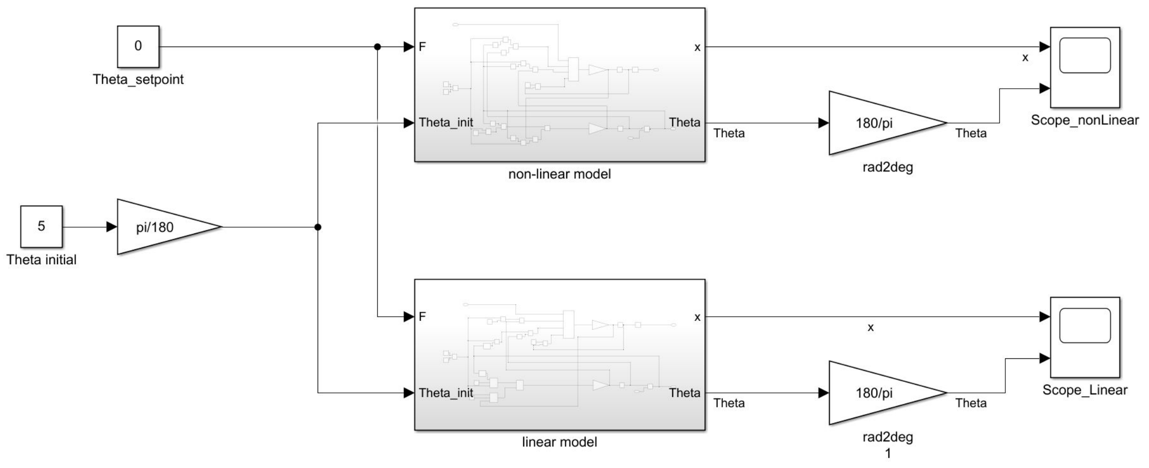


Figure 1.17: Comparison between the linear and the non-linear model

Simulating these models results for the non-linear submodel to the same as in section 1.3.1 (fig. 1.8). The result of the linear model can be seen in fig. 1.18.

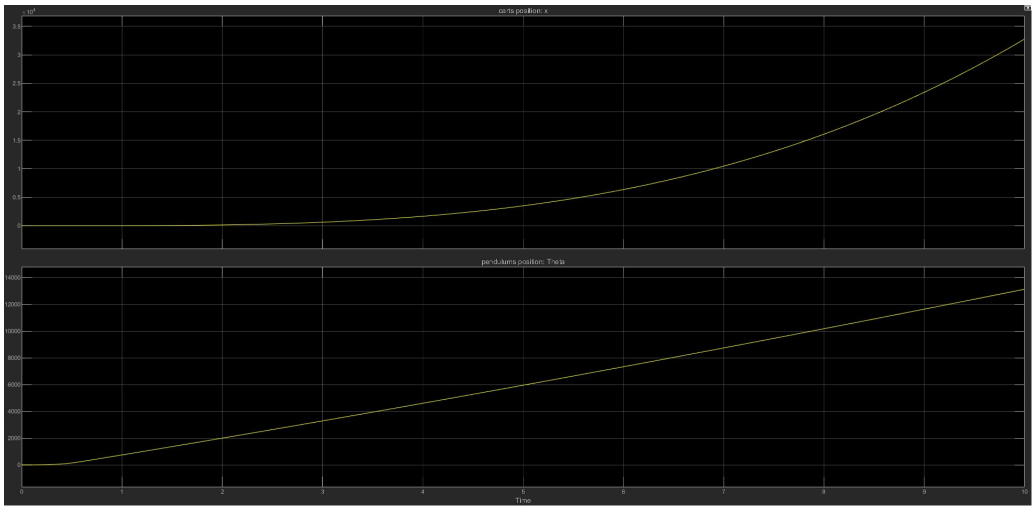


Figure 1.18: Scope of the linear model

What can be seen here?

The two scopes differ, because the system was linearized around 0°.

1.5 PID Control

Requirements

The control function shall meet following requirements:

- **Functional Requirement 1:** In front of the PID-controller, a Kalman-filter shall be implemented to get a more accurate actual position coming from the gyrosensor. The controller itself calculates the necessary torque the motor shall provide in order to keep the 0° vertical position.
- **Functional Requirement 2:** The control function shall operate in the range between ±40°. If the actual position exceeds these limits, the motor shall stop immediately.
- **Non-functional Requirement 1:** The entire control function shall be implemented as a triggered sub-system model in MatLab/SimuLink with a sampling time of 0.01 sec.
- **Non-functional Requirement 2:** The Kalman filter script (provided by the lecturer) shall be included in a MatLab function within the control function.
- **Non-functional Requirement 3:** The PID controller shall be implemented by means of SimuLink-blocks.

Implementation

Using SimuLink-Blocks, the schematic of the model of the PID controller is shown in fig. 1.19. In order to convert the controller output to torque, a factor of 0.005 is needed (this was empirically found).

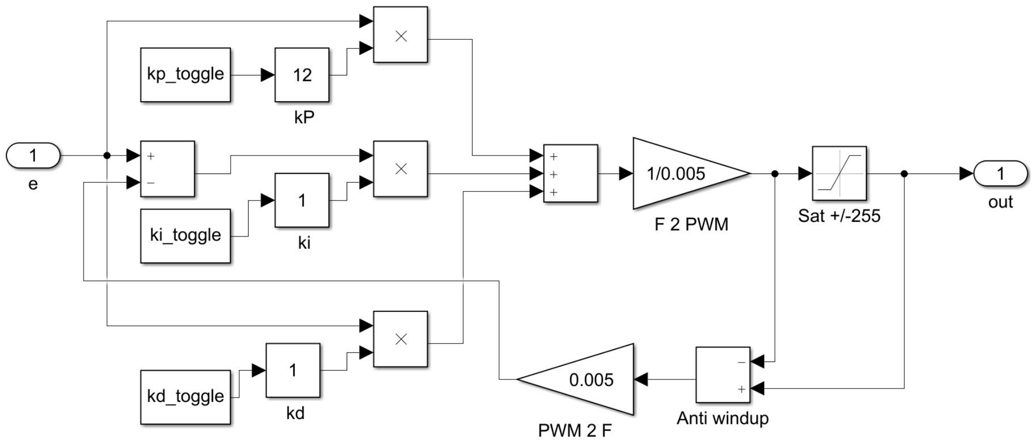


Figure 1.19: SimuLink-model of the PID controller

To change the controller parameter during run a slider gain is needed. The constants *kp_toggle, ki_toggle* and *kd_toggle* are used to de- and activate the parts of the controller in the MatLab script (changing them to zero deactivates them).

Applying the PID control to both systems

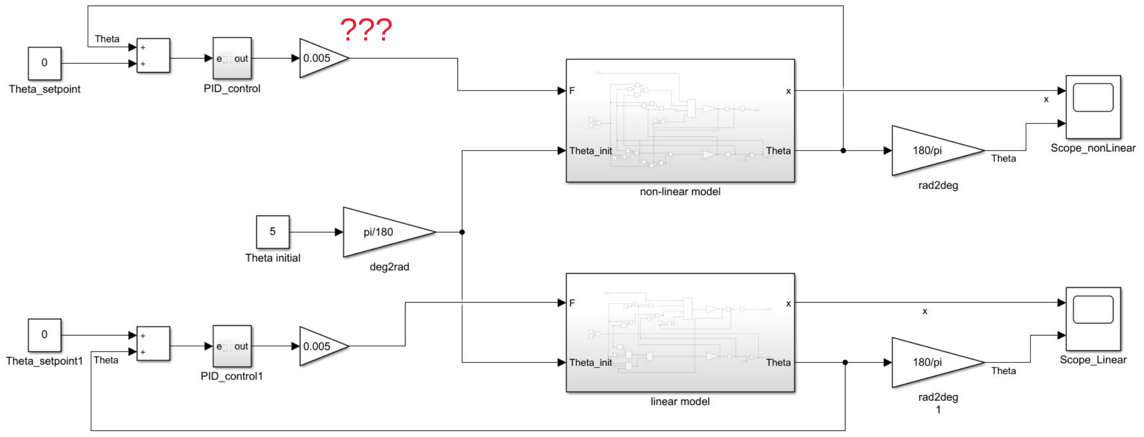


Figure 1.20: SimuLink of the PID controlled models

Using the empirically found value for kP from section 1.3.2 (kP = 12) where the system is stable for ∼20sec., the simulation results in following outputs:

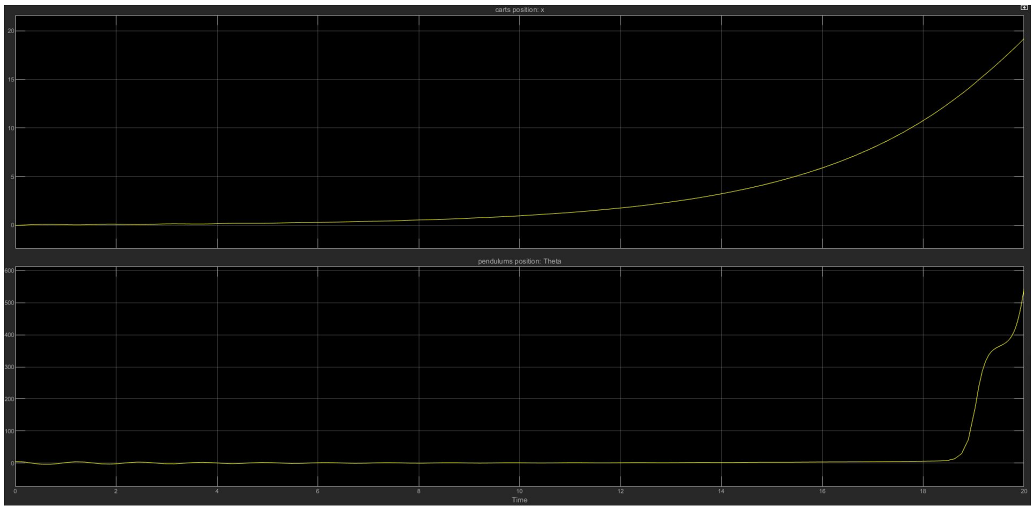


Figure 1.21: Output of the kP-controlled non-linear model

What can be seen?

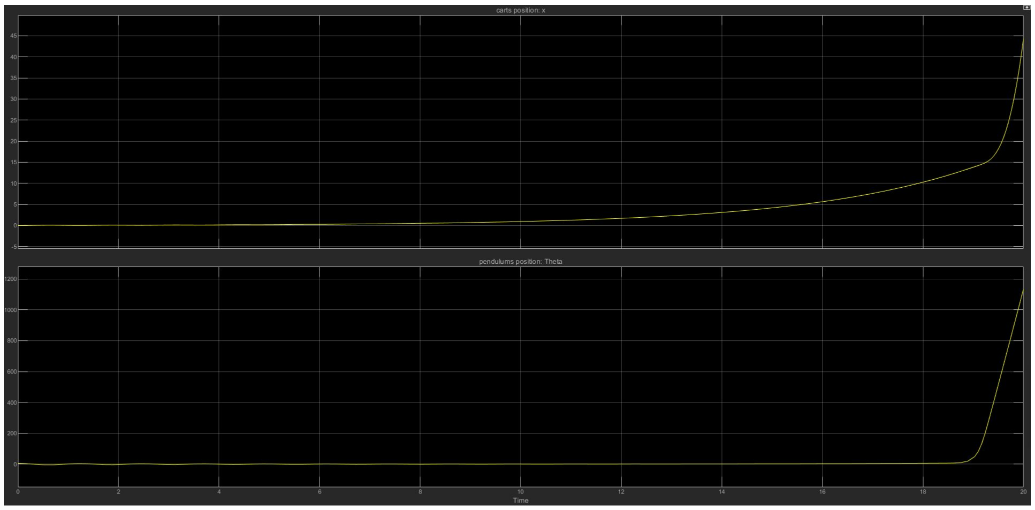


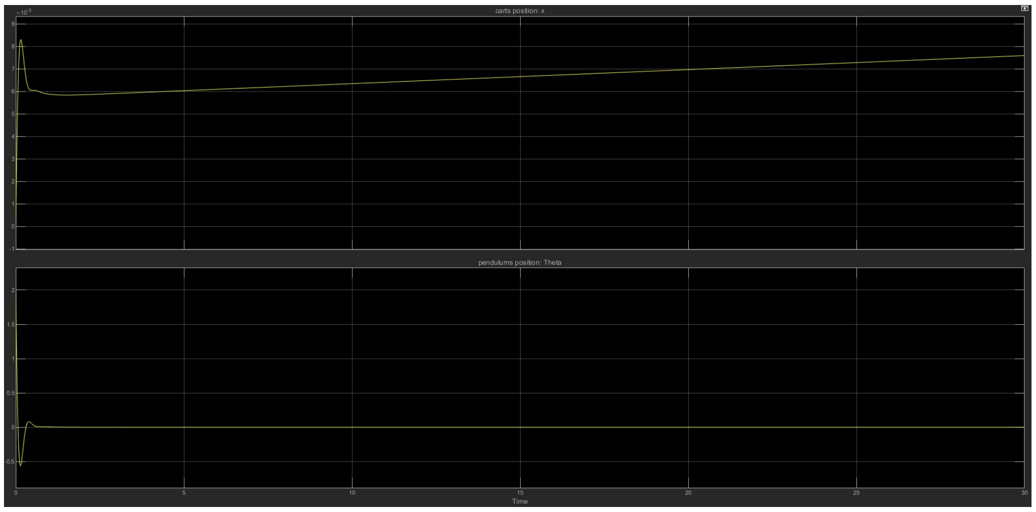
Figure 1.22: Output of the kP-controlled linear model

In order to tune the PID control, the oscillation ($t_{osz} = 1.211\text{sec}$) of the system response is measured in SimuLink. The other parameters can then be calculated using the Ziegler-Nichols method:

reference

$$\begin{align*}
t_n &= 0.5 \cdot t_{osz} = 0.6055 \\
t_i &= \frac{t_n}{kP} = 0.0505 \\
kI &= \frac{1}{t_i} = 19.8183 \tag{1.18} \\
t_v &= 0.125 \cdot t_{osz} = 0.1514 \\
kD &= t_v \cdot kP = 1.8165 \tag{1.19}
\end{align*}$$

The following scopes are showing the response of the model when using the calculated parameters.



visibility

Figure 1.23: Response of the non-linear system PID Controller

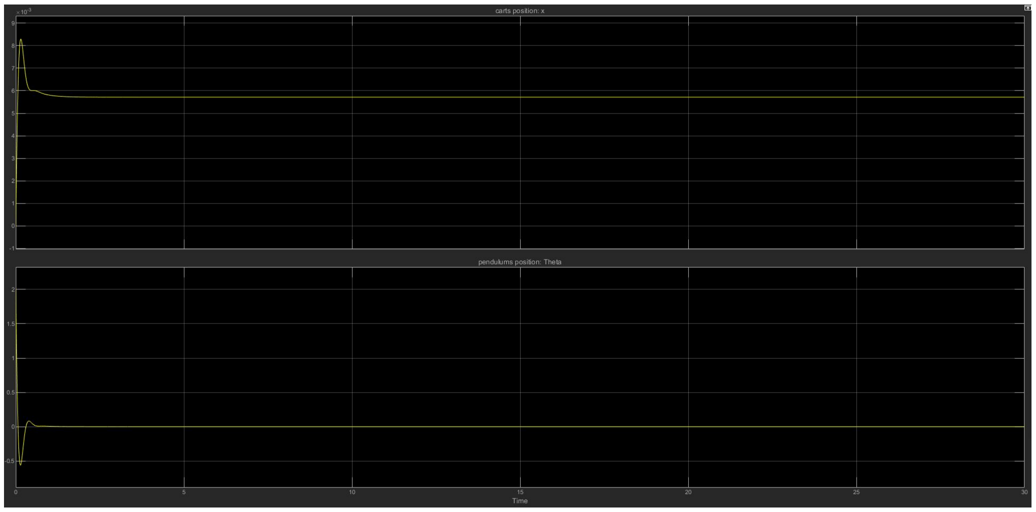


Figure 1.24: Response of the linear system using the PID Controller

The found parameters fit for the use of keeping the system stable at $\Theta = 0°$ but wont stabilize it to another angle. Therefore the SimuLink-block *PID Controller* and its autotuning mode is used, which calculates the PID Parameter as follows:

$$P = 8875.833$$
$$I = 16523.056$$
$$D = 528.303$$

With this parameters it is even possible to keep an stable angle of e.g. $\Theta = 5°$.

Kalman filter

As said before, the Kalman filter is provided as an annex by the lecturer and is implemented as a MatLab-function in SimuLink (fig. 1.25).

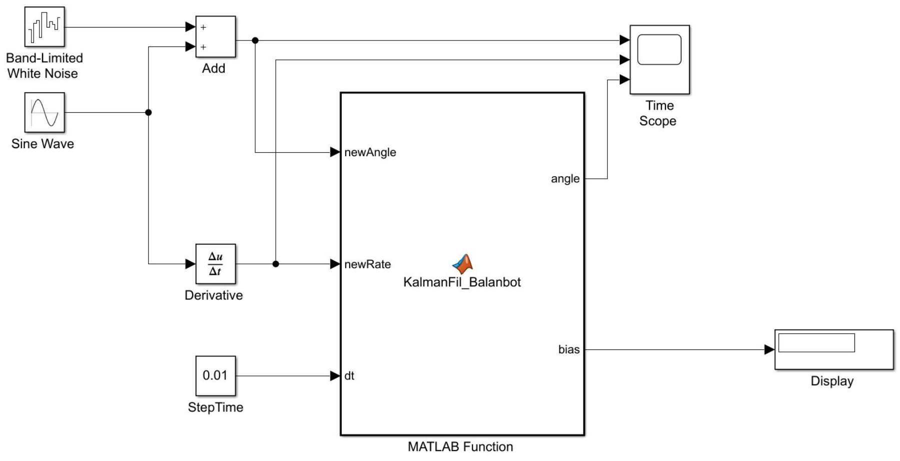


Figure 1.25: SimuLink-Model of the Kalman filter to test its behavior

The Kalman filter is tested on its behavior using a sine and adding some white-noise. As it is shown in fig. 1.26, the filter smoothens the noisy sine (scope on top) pretty good (the output of the filter is the scope on the bottom).

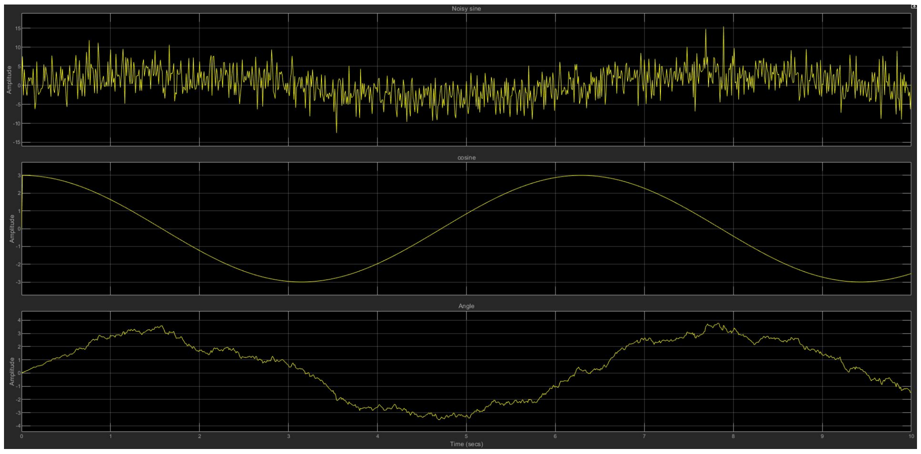


Figure 1.26: In- and outputs of the Kalmanfilter

1.6 Discretization

The transferfunction is then discretized using the zero-order-hold, first-order-hold and tustin discretization functions in MatLAB. The code and its output (discretization progress, calculation of poles) can be seen in the appendix in section 3.1. The calculated poles of the discretized functions can be seen in (table 1.3)). The poles are then drawn using the „*pzmap()*“-function and are shown in fig. 1.28 - 1.29.

Table 1.3: Poles of the discretized transfer functions

first-order-hold	zero-order-hold	tustin
1.0084	1.0084	1.0084
1.0000	1.0000	1.0000
0.9999	0.9999	0.9999
0.9917	0.9917	0.9917
1.0084	1.0084	
0.9999	0.9999	
0.9917	0.9917	

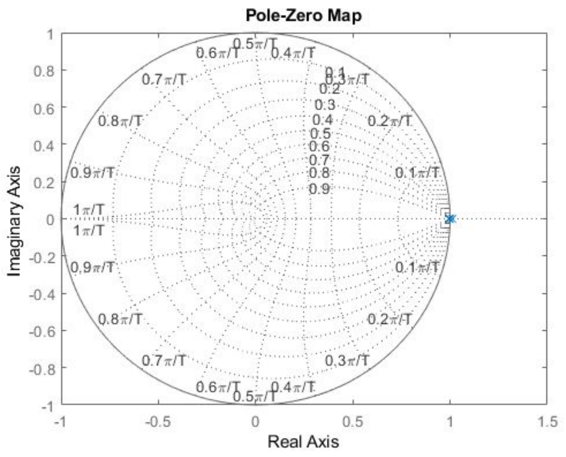


Figure 1.27: Poles of the transferfunction after being discretized using zero-order-hold

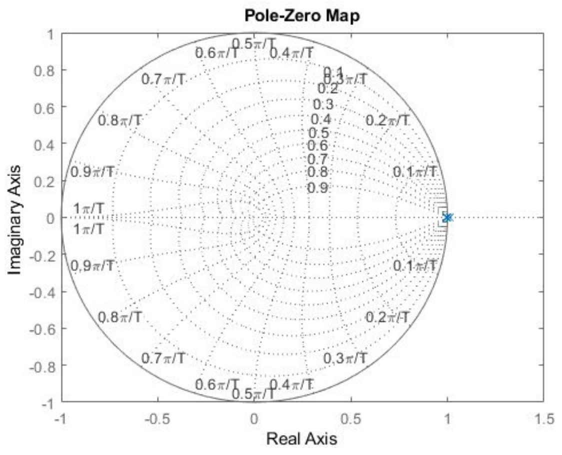


Figure 1.28: Poles of the transferfunction after being discretized using first-order-hold

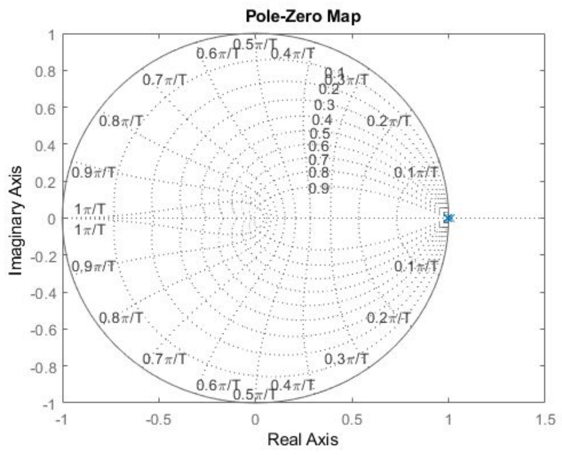


Figure 1.29: Poles of the transferfunction after being discretized using tustin

All the discretization models have at least one pole outside of the unit circle, meaning that all these methods are not stable.

Info: The task of the controller is to bring this poles into the left half plane (or within the unit circle)
As the poles are on the x-axis => it means exponential grow or decay

Chapter 2

Rapid Control Prototyping

2.1 Model Development

In order to test the behavior of the balanbot hardware the following model is built:

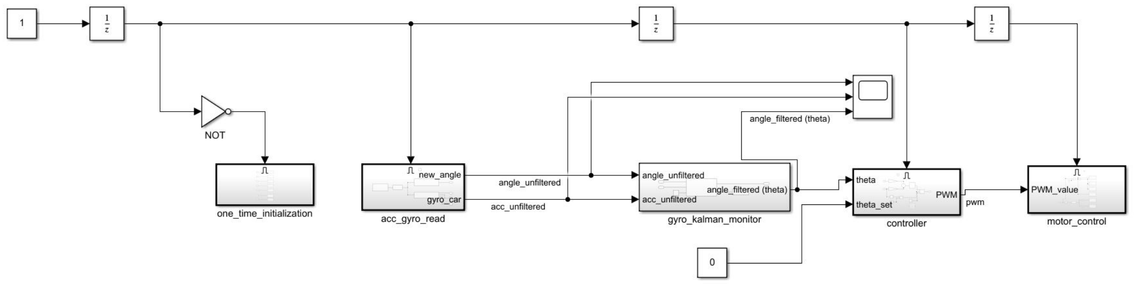


Figure 2.1: SimuLink model to obey the hardwares behavior

To meet the non-functional requirement 1, the solver is set to fixed-step with a step time of 10ms. After the balanbot is connected via USB. In the *Configuration Parameters Panel* the target is set to the Arduino Mega 2560.
The enabled subsystems represent a consecutive execution of the implemented functions.

2.1.1 One Time Initialization

The first subsystem (fig. 2.2) is the intitalization of the sensors and outputs (the registers of the board are set using I2C and Digital output blocks from the Simulink Arduino library) and is executed only once.

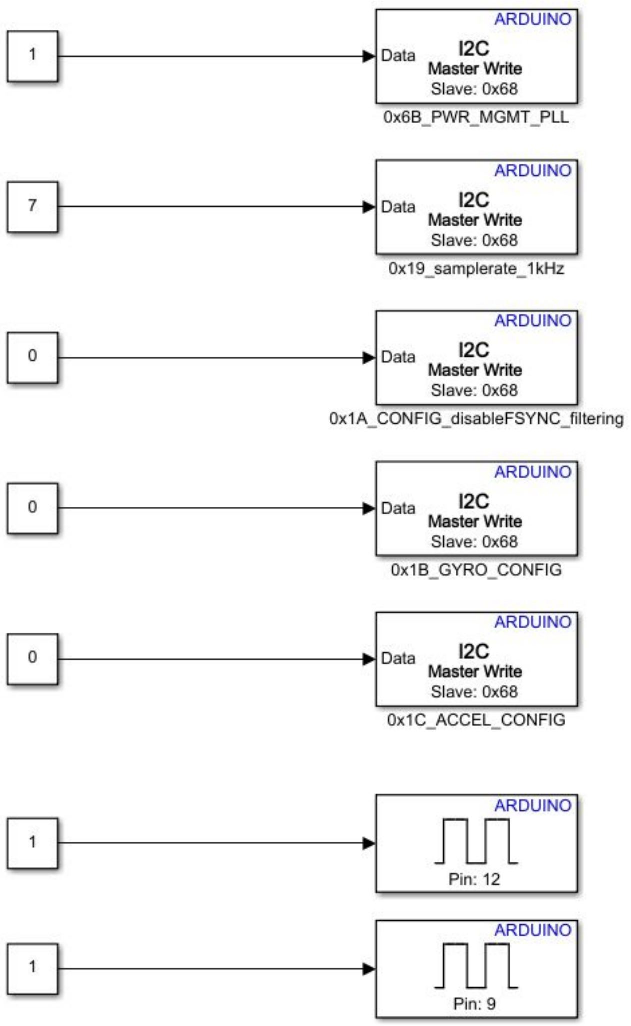


Figure 2.2: Model of one_time_initialization subsystem

2.1.2 Read Gyro Sensor Data

In the second subsystem (fig. 2.3) the sensor values are read from the target with an I2C block at a 10ms rate. Furthermore the angular speed and accelerations are computed. The functions for gathering angle and accelartion are provided.

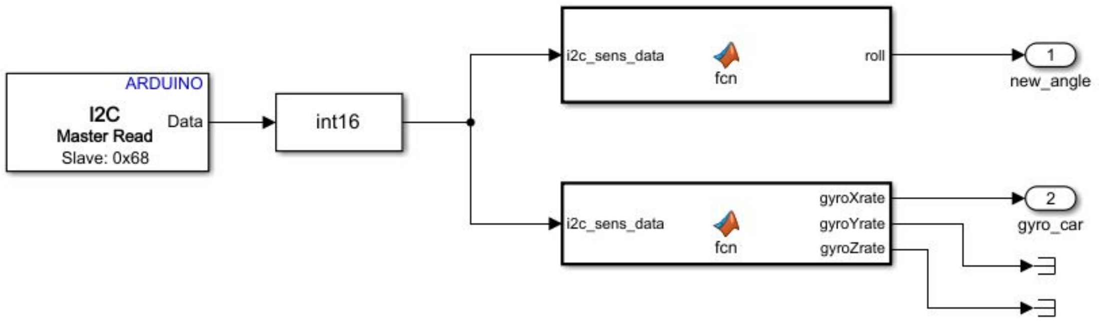


Figure 2.3: Model of the acc_gyro_read subsystem

The gathered angle and acceleration are then filtered with the Kalman filter. The result is shown in below figure (fig. 2.4), where the first scope is the unfiltered angle, the scope in the middle the unfiltered acceleration and the last the filtered angle.

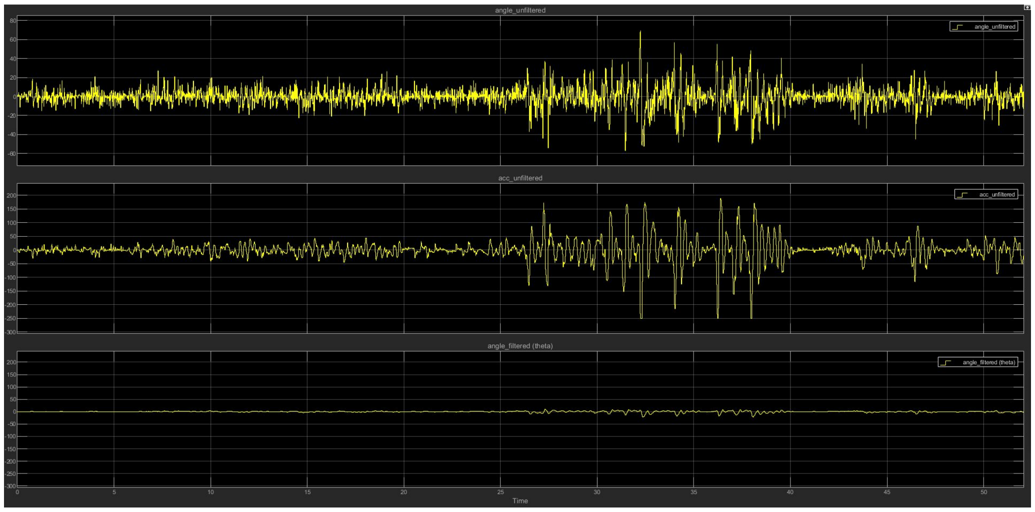


Figure 2.4: Comparison between the unfiltered and filtered angle read from the gyro sensor

2.1.3 Controller

The Controller is taken from the model-in-the-loop part and extended by a switch, in order to turn the output off if the angle exceeds its limit of $\pm 40°$.
The values for the hardware differ a little from the ones found during the simulation process and were empirically found. The parameters shown below suited best:

$$kP = -12$$
$$kI = -6$$
$$kD = -1.5$$

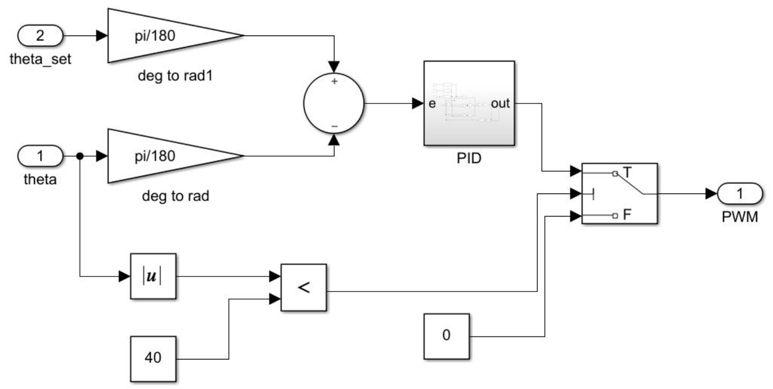


Figure 2.5: Model of the controller subsystem

2.1.4 Motor Control

The last subsystem writes digital signals to the motors, in order to control them. The speed of the motors is configured via PWM blocks and the rotational direction is configured via different digital pins.

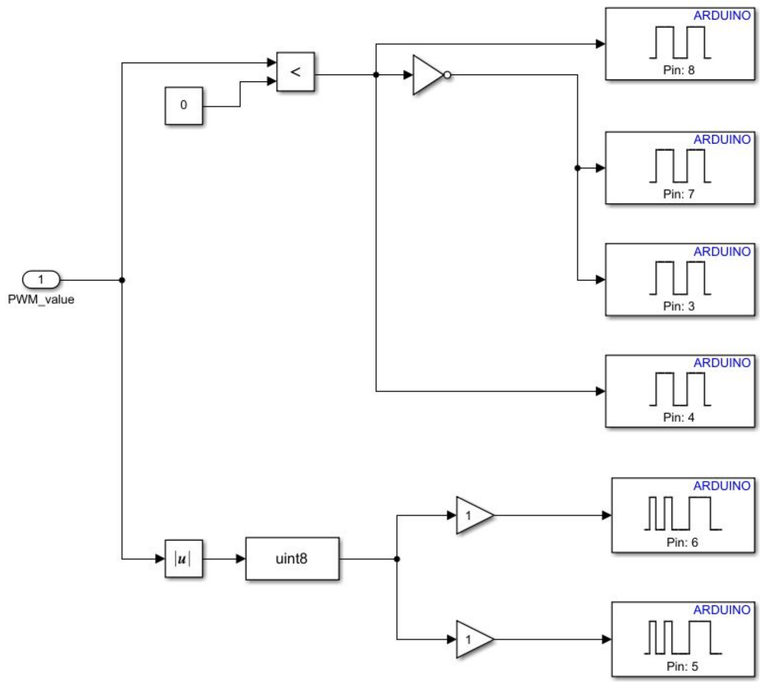


Figure 2.6: Model of the motor_control subsystem

With the described steps and settings, it is possible for the balanbot to keep its upward position for at least 10 seconds.

Chapter 3

Anhang

3.1 Discretization Appendix

Could you use the PID parameters for the final implementation?

Getting the Transferfunction of the system

```
s = tf('s');

Pos_tf = (s^2*(J+m*l^2) - m*g*l) / (s^4*((M+m)*(J+m*l^2) - m^2 * l^2) + ...
    s^3*(J*mue+m*l^2*mue) - s^2*(m*g*l*(M+m)) - s*(m*g*l*mue));
Ang_tf = -(s^2*m*l) / (s^4*((J+m*l^2)*(M+m)-m^2*l^2) + s^3*(mue*(J+m*l^2)) - ...
    s^2*(m*g*l*(M+m)) - s*(mue*m*g*l));

sys_tf = [Pos_tf ; Ang_tf];

inputs = {'u'};
outputs = {'x'; 'Theta'};
set(sys_tf,'InputName',inputs);
set(sys_tf,'OutputName',outputs);
sys_tf
```

```
sys_tf =

  From input "u" to output...
                     0.003125 s^2 - 0.1895
   x:  -----------------------------------------------------
       0.002413 s^4 + 0.0003125 s^3 - 0.1689 s^2 - 0.01895 s

                             -0.01932 s^2
   Theta:  -----------------------------------------------------
           0.002413 s^4 + 0.0003125 s^3 - 0.1689 s^2 - 0.01895 s

Continuous-time transfer function.
```

Discretization

```
stime = 0.001;
sys_tf_zoh = c2d(sys_tf, stime, 'zoh')
```

```
sys_tf_zoh =

  From input "u" to output...
       6.475e-07 z^3 - 6.475e-07 z^2 - 6.475e-07 z + 6.474e-07
   x:  -------------------------------------------------------
                 z^4 - 4 z^3 + 6 z^2 - 4 z + 0.9999

           -4.002e-06 z^2 + 1.728e-10 z + 4.002e-06
   Theta:  ----------------------------------------
                  z^3 - 3 z^2 + 3 z - 0.9999

Sample time: 0.001 seconds
Discrete-time transfer function.
```

```
sys_tf_foh = c2d(sys_tf, stime, 'foh')
```

```
sys_tf_foh =

  From input "u" to output...
       2.158e-07 z^4 + 4.316e-07 z^3 - 1.295e-06 z^2 + 4.316e-07 z + 2.158e-07
   x:  -----------------------------------------------------------------------
                         z^4 - 4 z^3 + 6 z^2 - 4 z + 0.9999

           -1.334e-06 z^3 - 4.002e-06 z^2 + 4.002e-06 z + 1.334e-06
   Theta:  --------------------------------------------------------
```

```
                           z^3 - 3 z^2 + 3 z - 0.9999

Sample time: 0.001 seconds
Discrete-time transfer function.
```

```
sys_tf_tustin = c2d(sys_tf, stime, 'tustin')
```

```
sys_tf_tustin =

  From input "u" to output...
       3.237e-07 z^4 - 1.963e-11 z^3 - 6.475e-07 z^2 - 1.963e-11 z + 3.237e-07
   x:  -----------------------------------------------------------------------
                         z^4 - 4 z^3 + 6 z^2 - 4 z + 0.9999

           -2.001e-06 z^4 + 4.002e-06 z^2 - 2.001e-06
   Theta:  ------------------------------------------
               z^4 - 4 z^3 + 6 z^2 - 4 z + 0.9999

Sample time: 0.001 seconds
Discrete-time transfer function.
```

getting the Poles

```
pole(sys_tf_zoh)
```

```
ans = 7×1
    1.0084
    1.0000
    0.9999
    0.9917
    1.0084
    0.9999
    0.9917
```

```
pole(sys_tf_foh)
```

```
ans = 7×1
    1.0084
    1.0000
    0.9999
    0.9917
    1.0084
    0.9999
    0.9917
```

```
pole(sys_tf_tustin)
```

```
ans = 4×1
    1.0084
    1.0000
    0.9999
    0.9917
```

3.2 Richtigkeitsbeglaubigung

Die beteiligten Personen bestätigen mit ihrer Unterschrift die Richtigkeit der Angaben.

[illegible]

REIBENSCHUH Matthias

Graz, am March 8, 2020

[illegible]

SCHNABL Maximilian