FH | JOANNEUM
Elektronik und
Computer Engineering

Balanbot

Model Based Design

Mario Waldherr, Iris Unterkircher

20.02.2020

Content

1 Task description

The Balanbot is in principle an inverted pendulum with its mass above its pivot point placed on a cart. The goal is to keep this construction in an upright position through controlling its drive according to the angle of the pendulum. Therefore, a Model-Based Design approach should lead through the modelling and implementation process on hardware. The system shall in the first place be designed in its non-linear form and later on be linearized by considering its physical operation constraints. By deriving the transfer function out of the system function via Laplace transformation the system´s behaviour and most important its stability shall be analysed. In addition, simulations in Simulink should give a better understanding of the behaviour. In the end, a PID controller with suitable parameters should be implemented on the real hardware in order to balance the pendulum in upright position.

2 Model development

As the Balanbot behaves in its physical principle as an inverted pendulum on a cart it can depicted as a theoretical model like in Figure 1.

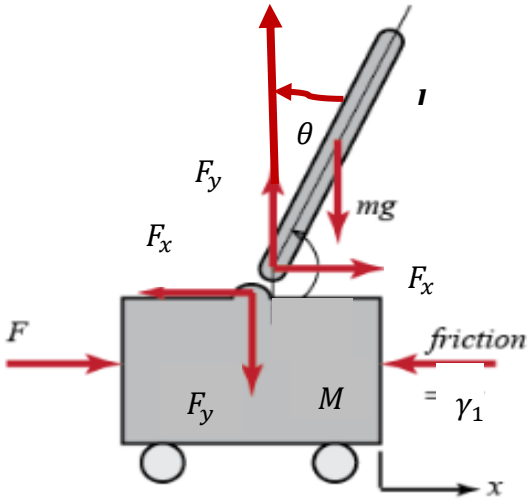


Figure 1: Theoretical model of the system **[1]**-modified

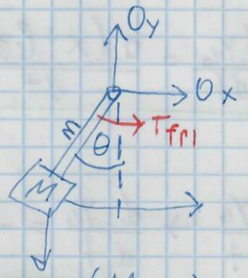
The occurring items in the systems model are shown in Table 1.

Table 1: Items of mathematical system model

$x: cart´s\ position$
$\theta: pendulum´s\ position\ (angular)$
$M: cart´s\ mass$
$m: pendulum´s\ mass$
$\gamma_1: friciton\ coefficient\ of\ cart$
$I, J: moment\ of\ inertia\ of\ pendulum$
$F: external\ force\ (motors)$
$F_x: interaction\ force\ cart-pendulum\ in\ x\ direction$
$F_y: interaction\ force\ cart-pendulum\ in\ y\ direction$
$g: gravitational\ constant$
$l: length\ of\ pendulum$

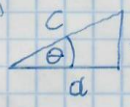
The following four pages show the development process of deriving the system equations from the above model.

Model of pendulum



$F = (M+m)\,g$

Trigonometric



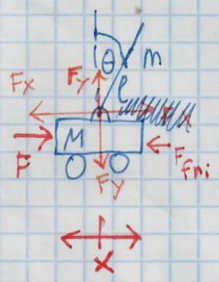
$$T \overset{*}{=} F \cdot r = I \cdot \alpha = I \cdot \ddot{\theta}$$

$$\Sigma T = (M+m)\,g \cdot l \cdot \sin\theta - T_{fr} = I \cdot \ddot{\theta}$$

$$\left[kg \cdot \frac{m}{s^2} \cdot m - Nm\right] = \left[kg\,m^2 \cdot \frac{1}{s}\right]$$

$$[S = Nm - Nm] = [Nm] \checkmark$$

Model of system



3 Freiheitsgrade: θ, x

$$\Sigma F = F - F_{fri} - F_x \qquad F_{fr} = \mu_A \dot{x}$$

$$F = m \cdot a$$

$$\ddot{x} = a = \frac{F}{m} = \frac{1}{M}\left(F - F_x - F_{fri}\right) \quad (1)$$

Pendulum

$$\ddot{\theta} = \alpha = \frac{1}{I}\Sigma \overset{*}{T} = \frac{1}{I}\left(F_x \cdot l \cdot \cos\theta - F_y\, l \cdot \sin\theta\right) \quad (2)$$

F_x? F_y?

Center of gravity

$x_G = x + l \sin\theta$

$y_G = l \cdot \cos\theta$

$F_x = m \cdot \frac{d^2 x_G}{dt^2} \rightarrow \frac{dx_G}{dt} = \frac{d(x + l \cdot \sin\theta)}{dt}$ → solve by chain rule and insert to ① + ②

$F_y - m \cdot g = m \cdot \frac{d^2 y_G}{dt^2} \rightarrow \frac{d}{dt} y_G = \frac{d(l \cos\theta)}{dt}$

(F_x) $\frac{d}{dt}: \dot{x} + l \cos(\theta) \cdot \dot{\theta}$

$\frac{d^2}{dt^2}: \ddot{x} + [-l \sin(\theta) \cdot \dot{\theta}^2 + l \cos(\theta) \cdot \ddot{\theta}]$

$F_x = m\left(\ddot{x} + l\left(-\sin(\theta)\dot{\theta}^2 + \cos(\theta) \cdot \ddot{\theta}\right)\right)$

(F_y) $\frac{d}{dt}: -l \sin(\theta)\dot{\theta}$

$\frac{d^2}{dt^2}: -l \cos(\theta)\dot{\theta}^2 - l \sin(\theta)\ddot{\theta}$

$F_y = -ml\left(\cos(\theta)\dot{\theta}^2 + \sin(\theta)\ddot{\theta}\right) + mg$

(2)

F_x und F_y in (2) einsetzen

$$\ddot{\theta} = \frac{1}{J}\left\{-m\ell\cos\theta\left[\ddot{x}+\ell(-\sin(\theta)\dot{\theta}^2+\cos(\theta)\ddot{\theta})\right]-m\ell\sin\theta\left[-\ell(\cos(\theta)\dot{\theta}^2+\sin(\theta)\ddot{\theta})+g\right]\right\}$$

$$\ddot{\theta} = -\frac{1}{J}m\ell\left\{\ddot{x}\cos\theta+\ell(-\sin(\theta)\cos(\theta)\dot{\theta}^2+\cos^2(\theta)\ddot{\theta})+\left[-\ell(\sin(\theta)\cos(\theta)\dot{\theta}^2+\sin^2(\theta)\ddot{\theta})+g\sin(\theta)\right]\right\}$$

$$-\frac{J}{m\ell}\ddot{\theta} = \ddot{x}\cos\theta+\ell\left[-\frac{1}{2}\sin(2\theta)\dot{\theta}^2+\cos^2(\theta)\ddot{\theta}\right]+\left\{-\ell\left[\frac{1}{2}\sin(2\theta)\dot{\theta}^2+\sin^2(\theta)\ddot{\theta}\right]+g\sin(\theta)\right\}$$

$$-\frac{J}{m\ell}\ddot{\theta} = \ddot{x}\cos\theta+\ell\left[-\frac{1}{2}\sin(2\theta)\dot{\theta}^2+\cos^2(\theta)\ddot{\theta}-\frac{1}{2}\sin(2\theta)\dot{\theta}^2-\sin^2(\theta)\ddot{\theta}\right]+g\sin(\theta)$$

$$\frac{J}{m\ell}\ddot{\theta} = \ddot{x}\cos\theta+\ell\left(\cos^2(\theta)\ddot{\theta}+\sin^2(\theta)\ddot{\theta}\right)-g\sin(\theta) \quad |\cdot m\ell$$

$$J\ddot{\theta} = m\ell\cos(\theta)\ddot{x}+m\ell^2\left(\cos^2(\theta)\ddot{\theta}+\sin^2(\theta)\ddot{\theta}\right)-mg\ell\sin\theta$$

$$J\ddot{\theta} = m\ell\cos(\theta)\ddot{x}+m\ell^2\left[\frac{1}{2}[1+\cos(2\theta)]\ddot{\theta}+\frac{1}{2}[1-\cos(2\theta)]\ddot{\theta}\right]-mg\ell\sin\theta$$

$$J\ddot{\theta} = m\ell\cos(\theta)\ddot{x}+m\ell^2\ddot{\theta}\left[\frac{1}{2}+\frac{1}{2}\cos(2\theta)+\frac{1}{2}-\frac{1}{2}\cos(2\theta)\right]-mg\ell\sin\theta$$

$$J\ddot{\theta} = m\ell\cos(\theta)\ddot{x}-mg\ell\sin\theta+m\ell^2\ddot{\theta}$$

$$\ddot{\theta}(J+m\ell^2) = -m\ell\cos(\theta)\ddot{x}+mg\ell\sin\theta \quad (2)_*$$

✓

F_x in ① einsetzen

$$\ddot{x} = \frac{1}{M}\left(F - m\left(\ddot{x} + l\left(-\sin(\theta)\dot{\theta}^2 + \cos(\theta)\ddot{\theta}\right)\right) - F_{fri}\right) \cdot M$$

$$M\ddot{x} = F - m\ddot{x} + ml\sin(\theta)\dot{\theta}^2 - ml\cos(\theta)\ddot{\theta} - F_{fri} \quad | + m\ddot{x}$$

$$(M+m)\ddot{x} + \gamma_1\dot{x} = F + ml\sin(\theta)\dot{\theta}^2 - ml\cos(\theta)\ddot{\theta}$$

✓

3 Non-linear system equations and parameter

From the above derivations we see that the following equations are valid for the Balanbot´s angle and position

$$\ddot{x} = \left[F + ml \cdot sin(\theta) \cdot \dot{\theta}^2 - ml \cdot \cos(\theta) \cdot \ddot{\theta} - \gamma_1 \dot{x}\right] \frac{1}{(M+m)} \tag{1}$$

$$\ddot{\theta} = [mgl \cdot sin(\theta) - ml \cdot \cos(\theta) \cdot \ddot{x}] \frac{1}{(J + ml^2)} \tag{2}$$

The parameters in the equations above can be found in Table 2:

Table 2: Physical parameters of the system´s hardware setup

$M = 0.716\ kg$
$m = 0.1756\ kg$
$\gamma_1 : 0.1$
$I, J = 0.001\ m^2 kg$
$l = 0.11\ m$
$g = 9.81 \frac{m}{s^2}$

As can be seen, the equations contain non-linear functions like sine and cosine, but our modelling approach and especially the control are restricted to LTI (linear, time-invariant)-systems. Hence, the non-linear must be linearized in order to only contain linear functions.

4 Linearization of the system

The linearization process consists mainly of substituting non-linear functions of an equation by linear approximations in a certain operating point. The advantage is that linear systems are most often easier to model/calculate and also more suitable to deploy on real hardware later on as the calculation of non-linear model equations has a higher memory and calculation power consumption. The disadvantage is that the approximation is only valid for a certain area of the real function (=operating point) and out that leads to an increasing error.

In our case, the linearization process is possible because we make the assumption that the system stays in a small range of its upright balance. So, we assume that the angle of deviation of the equilibrium stays small. As the deviation angle consists of $\phi = \theta + \pi$, we can for small angles assume the non-linear functions of the equations (1) and (2) to be

$$\cos(\theta) = \cos(\phi + \pi) \xrightarrow{\phi \to 0} -1$$

$$\sin(\theta) = \cos(\phi + \pi) \xrightarrow{\phi \to 0} -\phi$$

$$\dot{\theta} = \frac{d(\phi + \pi)}{dt} = \dot{\phi} = 0$$

As the error curve in Figure 2 shows, this assumptions only holds for angle values around $\theta = \pm 0.5\, rad \approx \pm 28\,°$

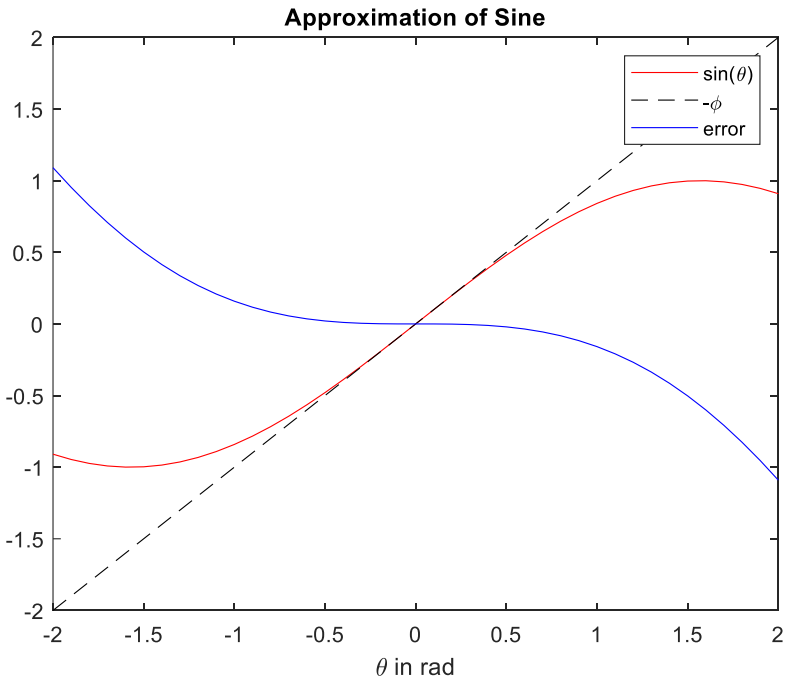


Figure 2: Approximation of Sine according to the linearization assumptions

By using the correlations above and applying them on the equations of (1) and (2) we obtain the linear system equations

$$\ddot{x} = \left[F + ml\ddot{\phi} - \gamma_1 \dot{x}\right] \frac{1}{(M+m)} \tag{3}$$

$$\ddot{\phi} = [-mgl\phi + ml\ddot{x}] \frac{1}{(J+ml^2)} \tag{4}$$

In conclusion we can see from the equations (3) and (4) that through the linearization process the non-linear functions sine and sine can be removed but in exchange the restrictions shown in Figure 2 have to be considered.

5 Comparison of non-linear and linear model

At first the linear system is built by means of Simulink according to the equations (3) and (4) as can be seen in Figure 3.

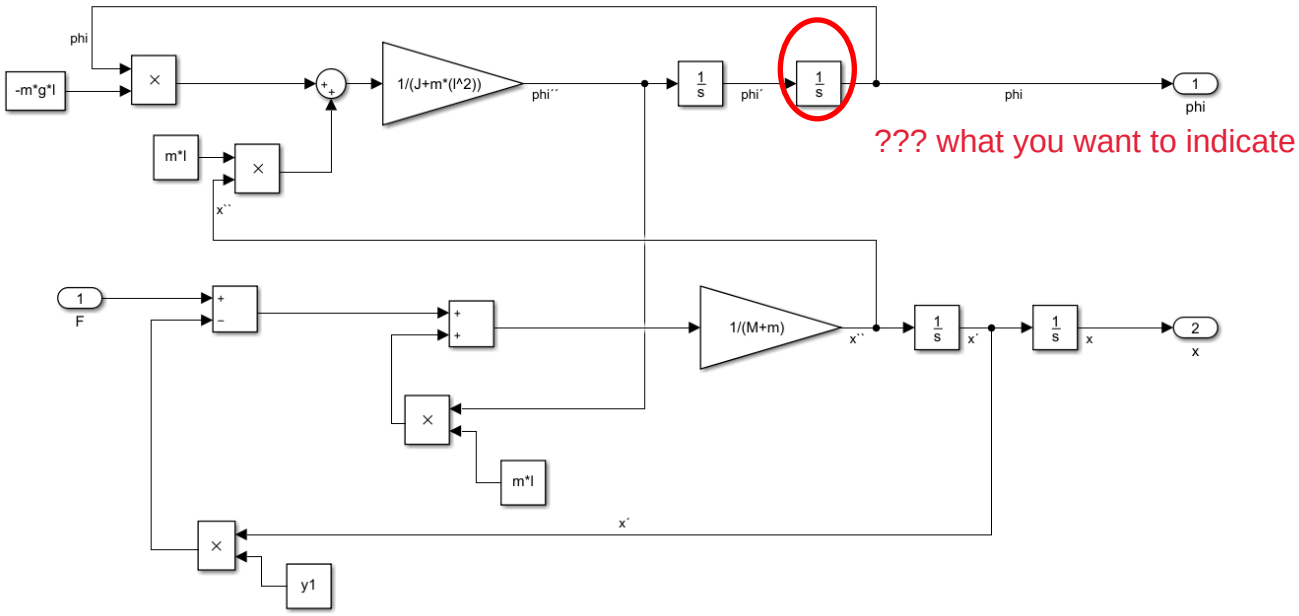


Figure 3: Simulink model of the linear system´s model

The parameters for the model are loaded from the MATLAB workspace as they can thereby be easy alternated and are changed throughout the whole Simulink model.

In the following Figure 4 the non-linear model according to the equations (3) and (4) is implemented analogous to the linear model. For stability reasons it is considered to build the model without Derivate-blocks. So, there are only integrators used for the best system´s performance.

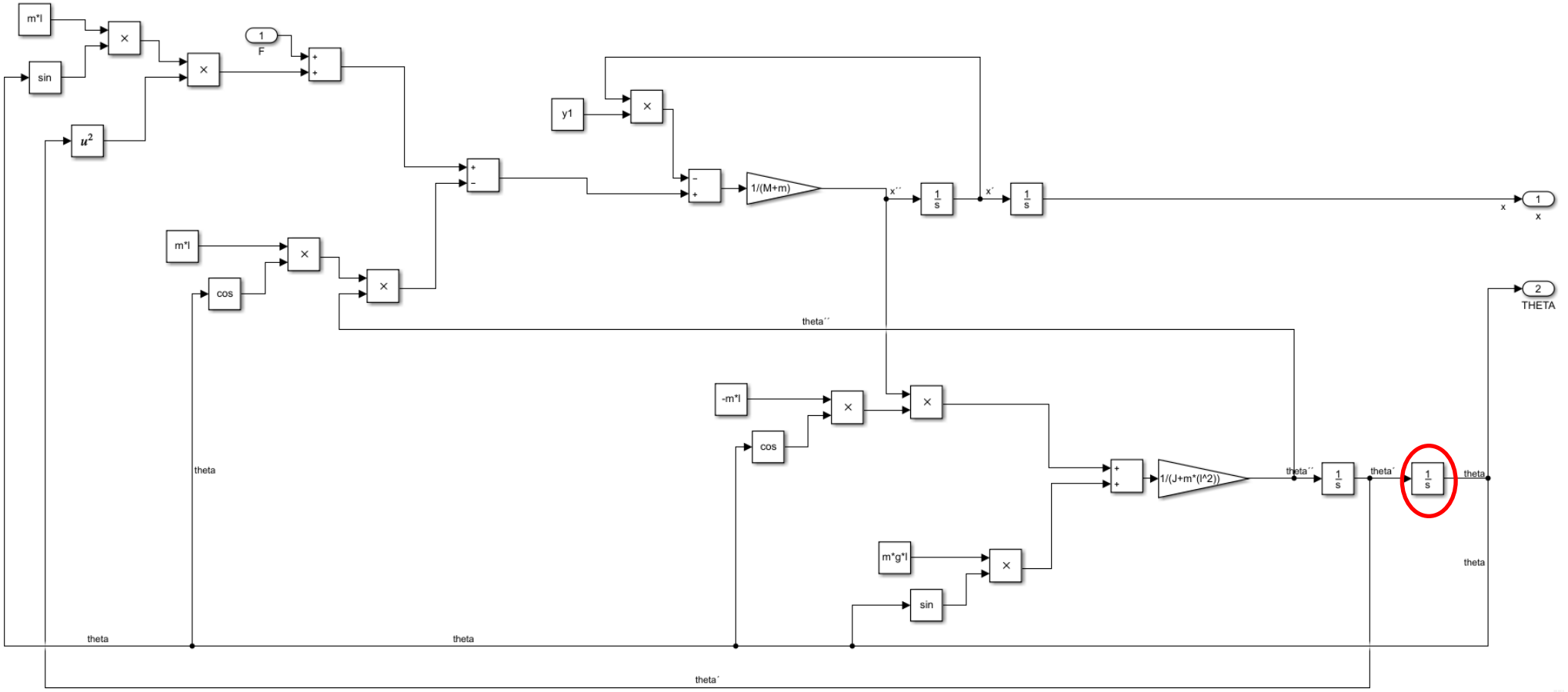


Figure 4: Simulink model of the non-linear system´s model

Then both models are wrapped into subsystems and applied with zero force at the input according to Figure 5.

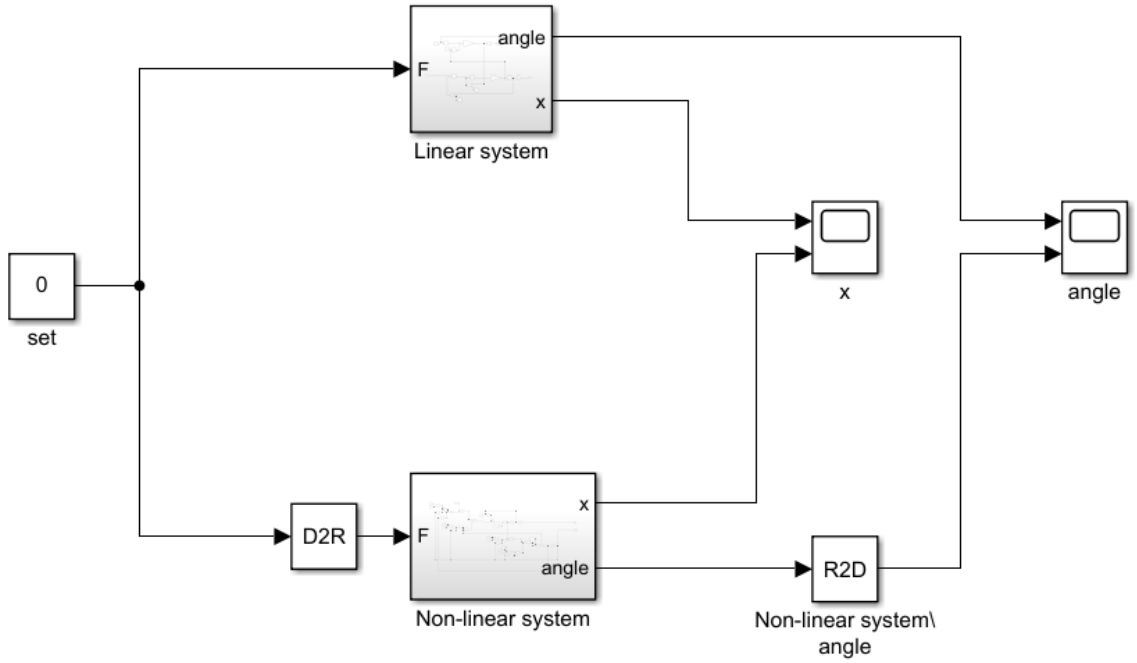
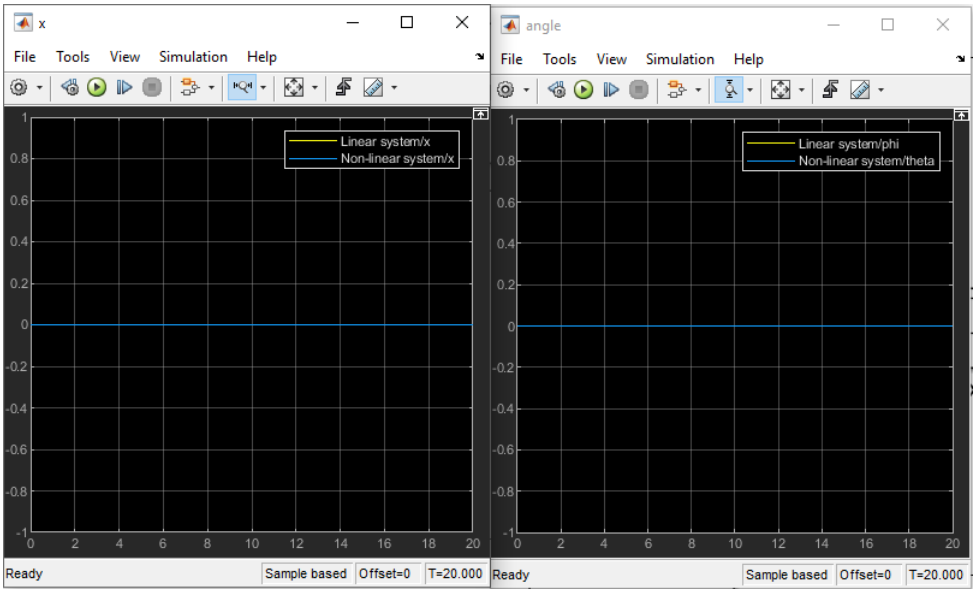


Figure 5: Both system models applied with zero force

5.1 Open loop response with no offset

By simulating both models with solver ODE45 we can see in Figure 6 that both initial values for angle and position are at zero and therefore the systems are stable.



background
axes labels etc.

Figure 6: Simulation result of the both models for solver ODE45 and F=0, r_tol=0.001

5.2 Open loop response with initial condition $\phi = 5°$

In the next step a small initial condition of 5° is applied to the last integrators of the models that deliver the angle. These ones are marked with red circles in Figure 3 and Figure 4. According to Figure 7 the angle has to be converted to radians.

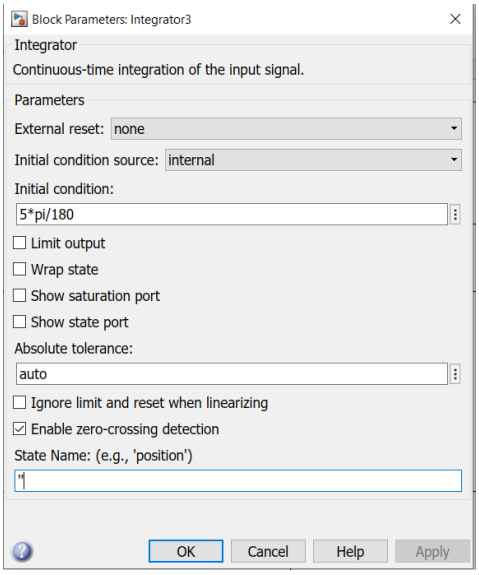


Figure 7: Applying the initial condition of $\phi = 5°$ to the integrators

The simulation results in Figure 8 indicate that the non-linear system is getting unstable with an initial condition for the angle. This is because without feedback/controlling the Balanbot returns to the next stable position, which would be in that case hanging down at 180° (=π) -see the red line in Figure 8. For a long simulation time like in Figure 9 the pendulum nearly stabilizes around that point of angle and also the position is left/right of its initial value.

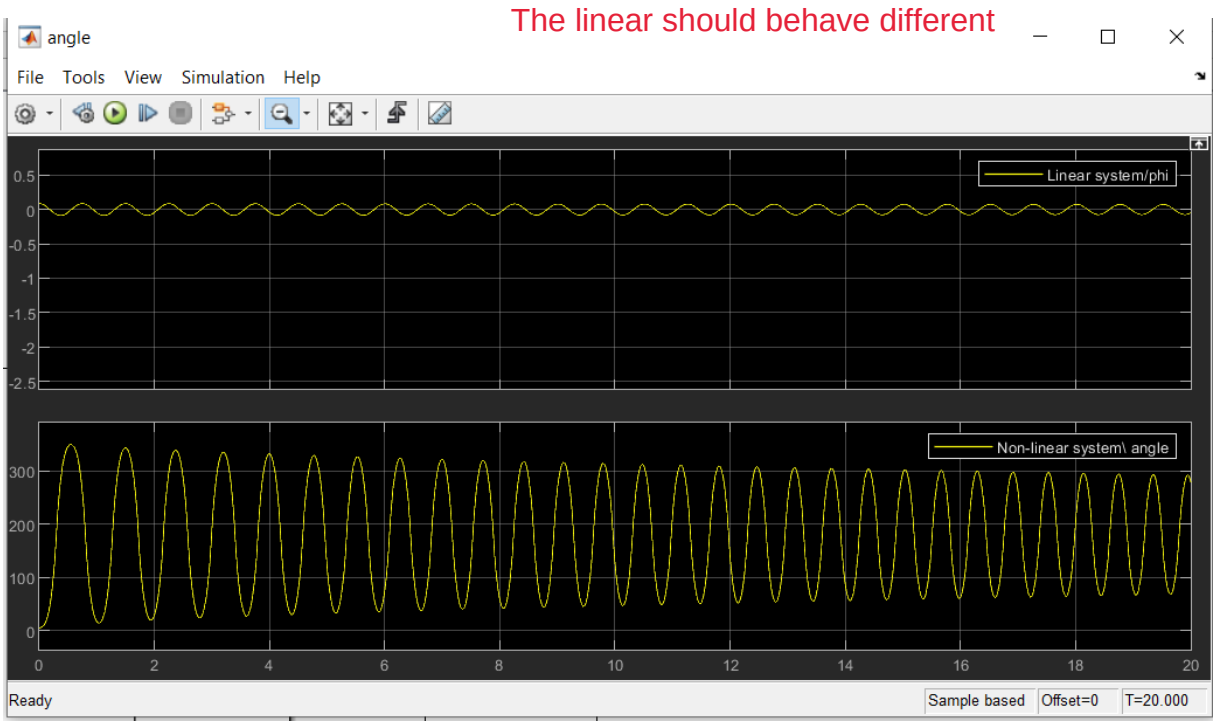


Figure 8: Simulation result of the both models for automatic solver selection and F=0, $\phi = 5°$, r_tol=0.001

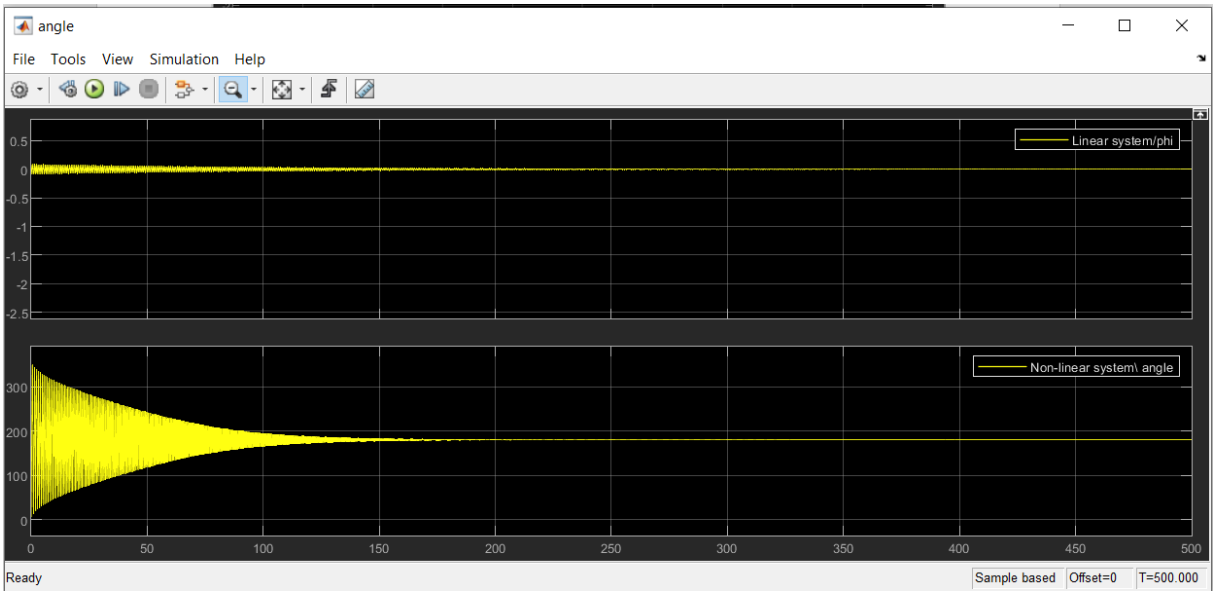


Figure 9: Simulation result of the both models for F=0, $\phi = 5°$, r_tol=0.001 and a simulation time of 500 seconds

5.3 Proportional feedback control

In the next step a proportional control should be added to the model in the loop in order to analyse its oscillation behaviour. Therefore, different values for a proportional factor are applied to the feedback loop and the oscillation period as well as the stability of the response are documented. This is only done for the non-linear model as the above results – especially Figure 9 – indicate that the linear model is anyway stable.

The additional a proportional factor is added to the model according to Figure 10.

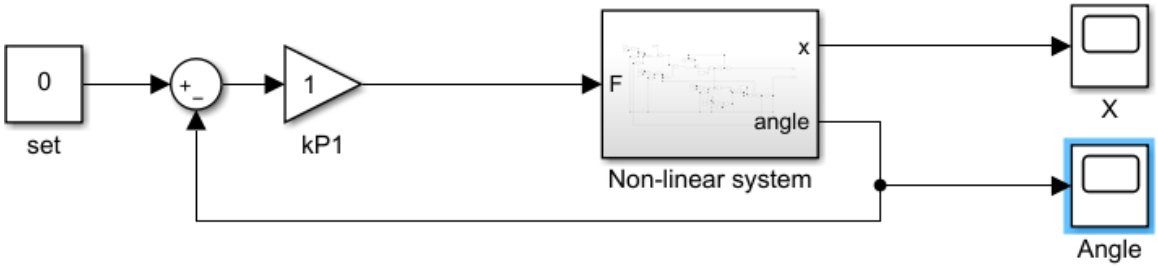


Figure 10: Non-linear model in the loop with proportional feedback control

For the analysis of the stable time it is defined that the deviation angle should be around five degrees. With the cursors the oscillation period, the stable time and the boundary angle at which this stable time ends are measured for different values of *kp*. The system is defined stable if its stable time lies above 10 seconds. The results with various proportional factors are shown in Table 3.

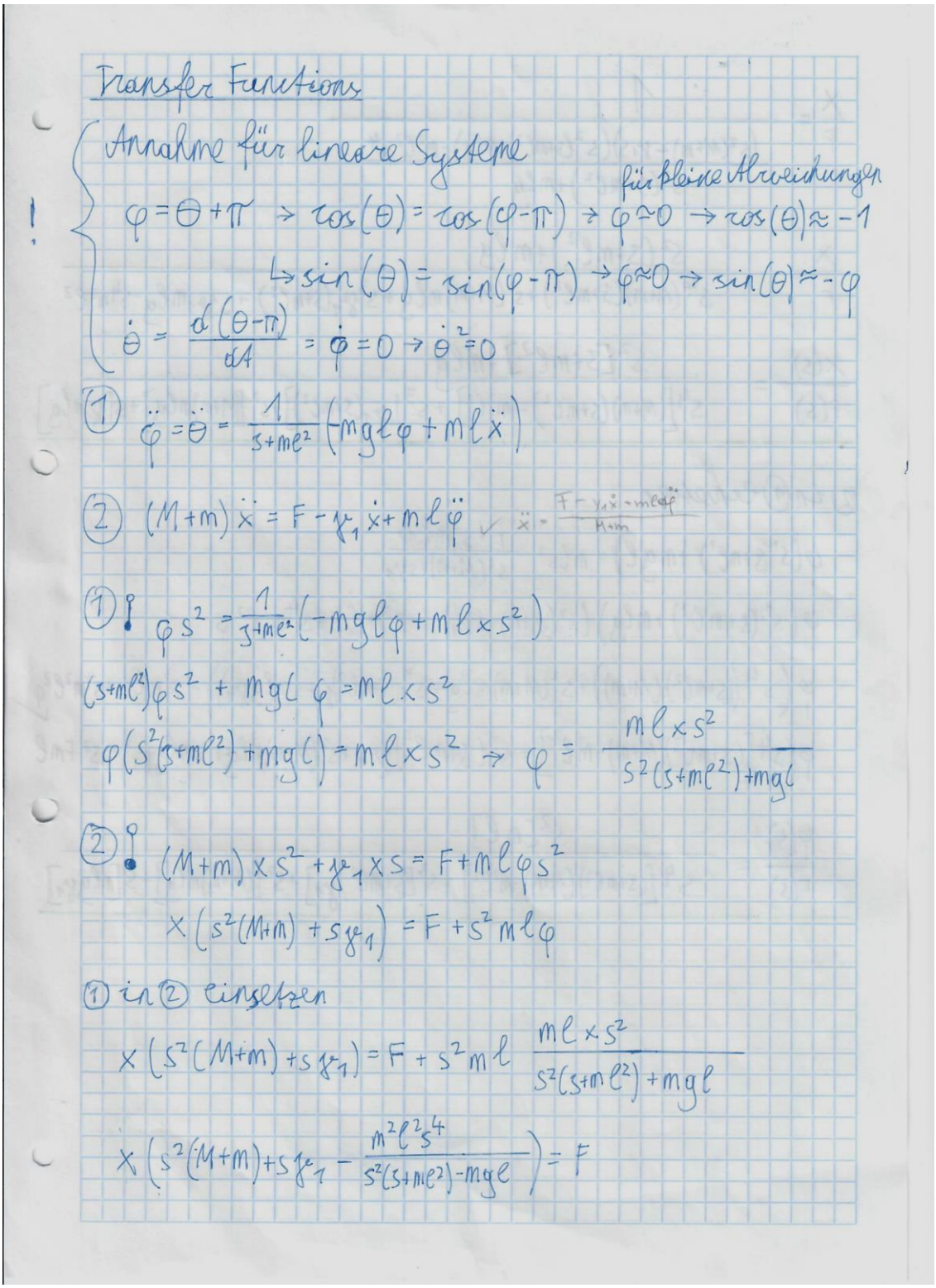
Table 3: Analysis of the proportional feedback stability

k_P	$oscillation\ period\ [s]$	$stable\ time\ [s]$	$until\ angle[°]$	$Stable?\ (more\ than\ 10\ s)$
1	2.2654	0.705	4.8757	NO
5	1.45	1.678	4.8506	NO
10	0.984	52.34	5.0237	YES
20	0.654	289.534	4.9962	YES
30	0.576	496.291	5.0184	YES

As can be seen in Table 3 with a higher proportional factor, the system is getting more stable and the oscillation period is decreasing.

6 System analysis via transfer functions

In order to analyse the system regarding to its poles and zeros, the linearized model functions of (3) and (4) are derived into continuous transfer functions in the following two pages.



$$\frac{X}{F} = \frac{1}{\frac{(s^2(M+m)+\gamma_1 s)(s^2(\Theta+me^2)+mlg)-m^2l^2s^4}{s^2(\Theta+me^2)+mlg}}$$

$$\frac{X}{F} = \frac{s^2(\Theta+me^2)+mlg}{s^4(M+m)(\Theta+me^2)+s^2(M+m)mlg+s^3\gamma_1(\Theta+me^2)+s\gamma_1 mlg-s^4m^2l^2}$$

$$\frac{x(s)}{F(s)} = \frac{s^2[\Theta+me^2]+mlg}{s^4[(M+m)(\Theta+me^2)-m^2l^2]+s^3[\gamma_1(\Theta+me^2)]+s^2[(M+m)mlg]+s[\gamma_1 mlg]}$$

② in ① einsetzen

$$\varphi(s^2(\Theta+me^2)+mgl) = mls^2\,\frac{F+s^2ml\varphi}{s^2(M+m)+s\gamma_1}$$

$$\varphi(s^2(\Theta+me^2)+mlg)(s^2(M+m)+s\gamma_1) = s^2mlF+s^4m^2l^2\varphi$$

$$\varphi\left(s^4((\Theta+me^2)(M+m))+s^2(M+m)mlg+s^3(\Theta+me^2)\gamma_1+sml g\gamma_1\right) = s^2mlF+s^4m^2l^2\varphi$$

$$\varphi\left(s^4[(\Theta+me^2)(M+m)-m^2l^2]+s^3(\Theta+me^2)\gamma_1+s^2(M+m)mlg+sml g\gamma_1\right) = s^2Fml$$

$$\frac{\varphi(s)}{F(s)} = \frac{s^2[ml]}{s^4[(\Theta+me^2)(M+m)-m^2l^2]+s^3[(\Theta+me^2)\gamma_1]+s^2[(M+m)mlg]+s[mlg\gamma_1]}$$

From the above calculations the transfer functions result in

$$\frac{X(s)}{F(s)} = \frac{s^2[J + ml^2] + mlg}{s^4\,[(M+m)(J+ml^2) - m^2l^2] + s^3[\gamma_1(J+ml^2)] + s^2[(M+m)\cdot mlg] + s[\gamma_1 mlg]} \quad (5)$$

$$\frac{\phi(s)}{F(s)} = \frac{s^2[ml]}{s^4\,[(M+m)(J+ml^2) - m^2l^2] + s^3[\gamma_1(J+ml^2)] + s^2[(M+m)\cdot mlg] + s[\gamma_1 mlg]} \quad (6)$$

These equations (5) and (6) can be used now to specify the numerator and denominator of system transfer model in MATLAB – see Code 2.

```
%Transfer Functions
s = tf('s');
% For Transfer functions the coefficients of the numerator and
% denumerator have to be entered like tf(Numerator, Denominator)
% with i.e Numerator = [2]s^3 [3]s^2 [4]s^1 [0]s^0

X_tf = tf([(J+m*(l^2)) 0 (m*l*g)], ...
          [((M+m)*(J+m*l^2)-((m^2)*(l^2))) (y1*(J+m*(l^2)))
((M+m)*m*l*g) (y1*m*l*g) 0])

PHI_tf = tf([(m*l) 0 0],...
    [(((J+m*(l^2))*(M*m))-((m^2)*(l^2))) ((J+m*(l^2))*y1)
((M+m)*m*l*g) (m*l*g*y1) 0])

SYSTEM_tf = [X_tf;PHI_tf]

inputs = {'F'};
outputs = {'X'; 'PHI'};

set(SYSTEM_tf,'InputName',inputs)
set(SYSTEM_tf,'OutputName',outputs)

SYSTEM_tf
```

Code 1: System declaration via transfer functions in MATLAB

The system´s transfer functions in MATLAB result to:

```
SYSTEM_tf =

  From input "F" 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
   PHI:  ---------------------------------------------------
         1.977e-05 s^4 + 0.0003125 s^3 + 0.1689 s^2 + 0.01895 s

Continuous-time transfer function.
```

6.1 Open loop

With the MATLAB function *pzmap()* the allocation of the poles and zeros of the system transfer function based on the linear equations are shown according to Figure 11.

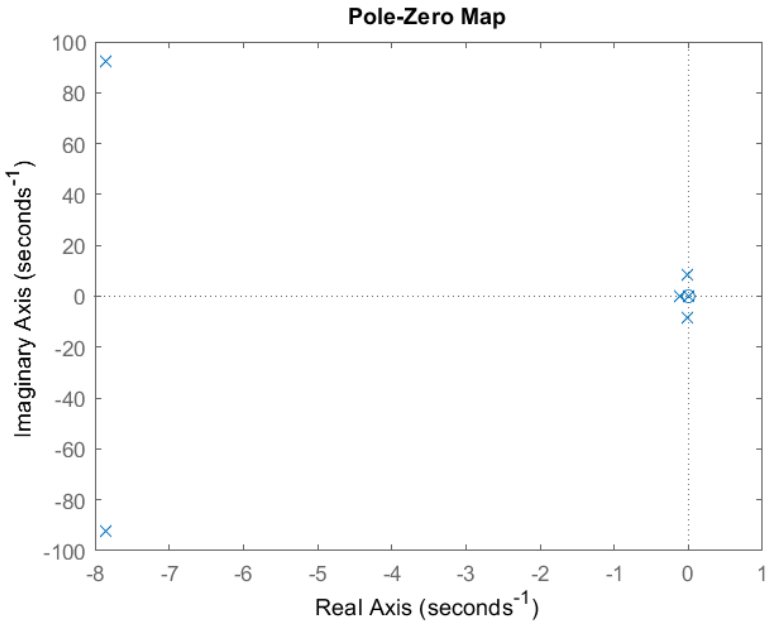


Figure 11: Pole-Zero Mapping of the continuous transfer model – open loop

```
poles = 8×1 complex
 0.0000 + 0.0000i
-0.0087 + 8.3676i
-0.0087 - 8.3676i
-0.1122 + 0.0000i
 0.0000 + 0.0000i
-7.8479 +92.1070i
-7.8479 -92.1070i
-0.1122 + 0.0000i
zeros = 1.1635e-16
```

The mapping indicates that all the poles (x) are on the left half plane and the system is in principle stable. As there is a pole with a real part of zero the system is only marginally stable. It also shows that there are nearly cancelling poles and zeros, which means that the system could be simplified by eliminating them without affecting the model response. The complex conjugate poles at -7.8479 ± 92.1070i and $-0.0087 \pm 8.3676i$ prove that the system is capable of oscillating, but because they are on the left half plane it means that these are decaying oscillations. The system is also minimum phase, because there are no poles and zeros in the right half plane.

The poles and zeros of the linear system shall be purely on the x axis!
If everything is right

6.2 Closed loop

The closed loop is derived from the system part of the angle of the open loop, because this part is being controlled afterwards. In the following the closed loop system transfer functions is depicted and again the analysis of the pole and zero allocation.

```
closed_loop = feedback(PHI_tf,1)
closed_loop =

                       0.01932 s^2
  ------------------------------------------------------
  1.977e-05 s^4 + 0.0003125 s^3 + 0.1883 s^2 + 0.01895 s

Continuous-time transfer function.
```

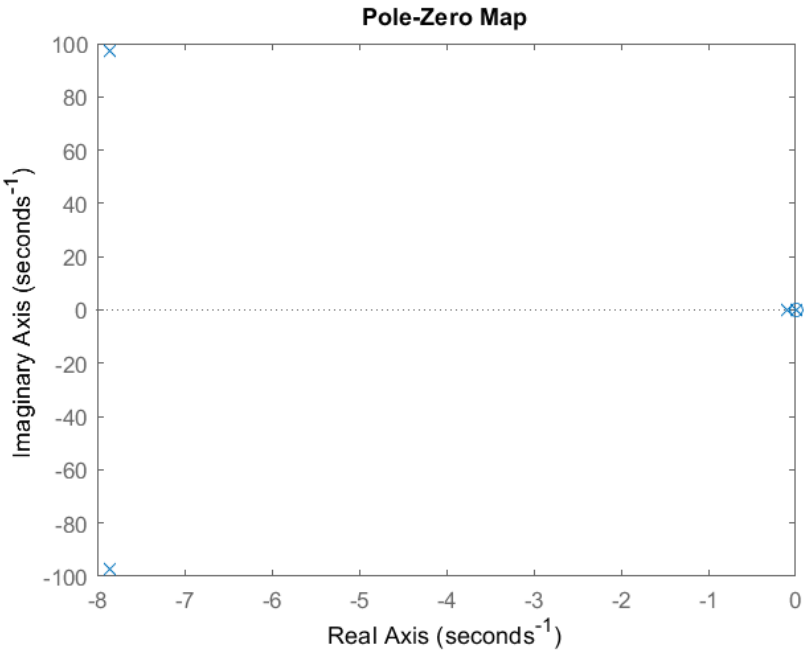


Figure 12: Pole-Zero Mapping of the continuous transfer function of angle – closed loop with kp=1

```
 [poles,zeros]=pzmap(closed_loop)
poles = 4×1 complex
   0.0000 + 0.0000i
  -7.8537 +97.2675i
  -7.8537 -97.2675i
  -0.1007 + 0.0000i
zeros = 2×1
     0
     0
```

The closed loop with a proportional factor of one is analysed according to his poles and zeros allocation in Figure 12. As in the open loop, all the poles are on the left half plane and the closed loop is therefore stable. The complex conjugate poles at $-7.8537 \pm 97.2675\mathrm{i}$ prove that the system is capable of oscillating. The closed loop is also minimum phase, because there are no poles and zeros in the right half plane.

The plant itself is unstable but together with the controller it is stable!
YOu had some sign mistakes

7 PID Control

In order to deploy the above model principle to a real hardware a PID controller is needed for regulating the force of the motors according to the actual angle of the Balanbot. The following requirements have to be met:

Functional Requirement

FR1. The control function for the Balanbot requires a Kalman filter followed by a PID controller.

FR2. The control function shall operate only in the range between -40° and 40°. In case that the actual position exceeds the operating range then the motors shall stop immediately.

Non-functional requirements

NF1. The entire control function shall be implemented as a triggered sub-system model in MATLAB/Simulink with a sampling time of 0.01 sec.

NF2. The Kalman filter script (provided as an annex) shall be included in a MATLAB function within the control function.

NF3. The PID controller shall be implemented by means of Simulink blocks, i.e. don’t use the predefined PID block.

7.2 Kalman Filter

In the first step the behaviour of the Kalman Filter is tested in a separate model by applying a Sine Wave with a noise in order to simulate the signal coming from the gyroscope. The sample time of the simulation T=0.01s is used as dt.

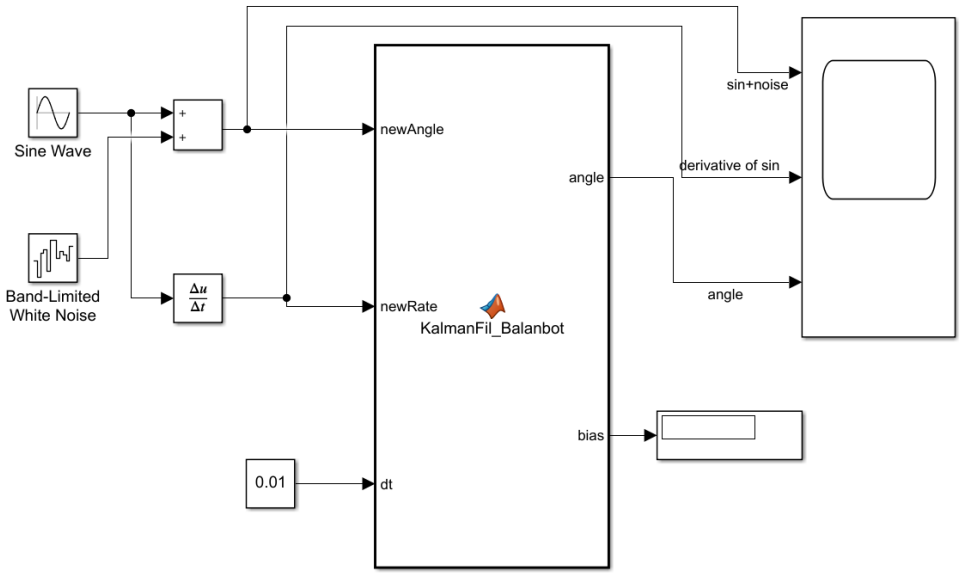


Figure 13: Setup of the Kalman Filter as a standalone version for testing purpose

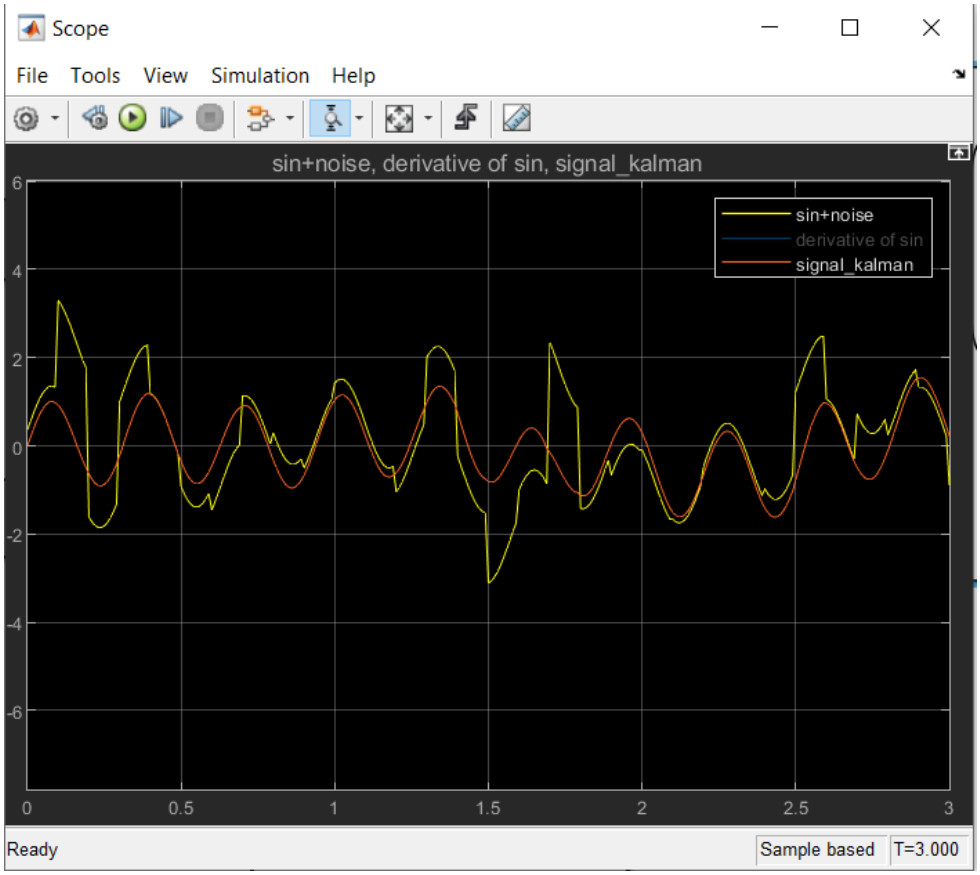


Figure 14: Output of the Kalman filter (orange) by applying sine and noise (yellow)

As Figure 14 shows, the Kalman filter smoothes the noisy sine wave and provides a better signal condition. This represents in the hardware realisation a reliable signal coming from the gyroscope and is therefore crucial for the further controlling. In chapter 8 the filtering is tested with the real sensor values.

7.3 Controller parameters

7.3.1 Ziegler Nichols method

At first the PID parameters should be calculated by Ziegler-Nichols method. In order to get the system swinging a small offset of 0.01 is added to the marked integrator of Figure 3. Afterwards a proportional factor is added to the closed loop in Figure 15 and increased in order to get a stationary oscillation.

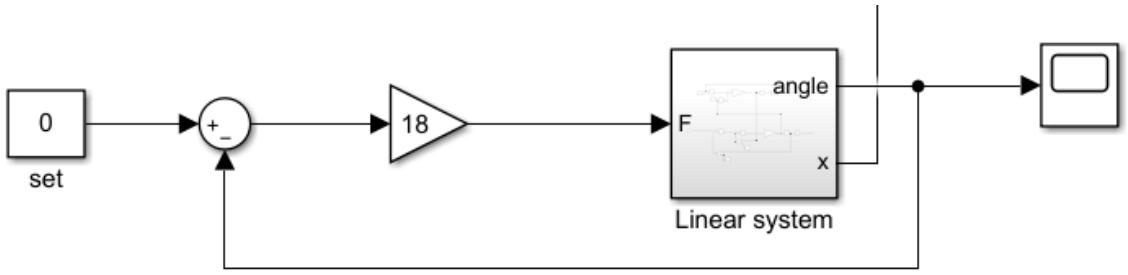


Figure 15: Setup for evaluating the controller parameters via Ziegler Nichols

The proportional factor is than increased to 18, so that a virtually stationary oscillation occurs, and the critical oscillation period can be measured in Figure 16.

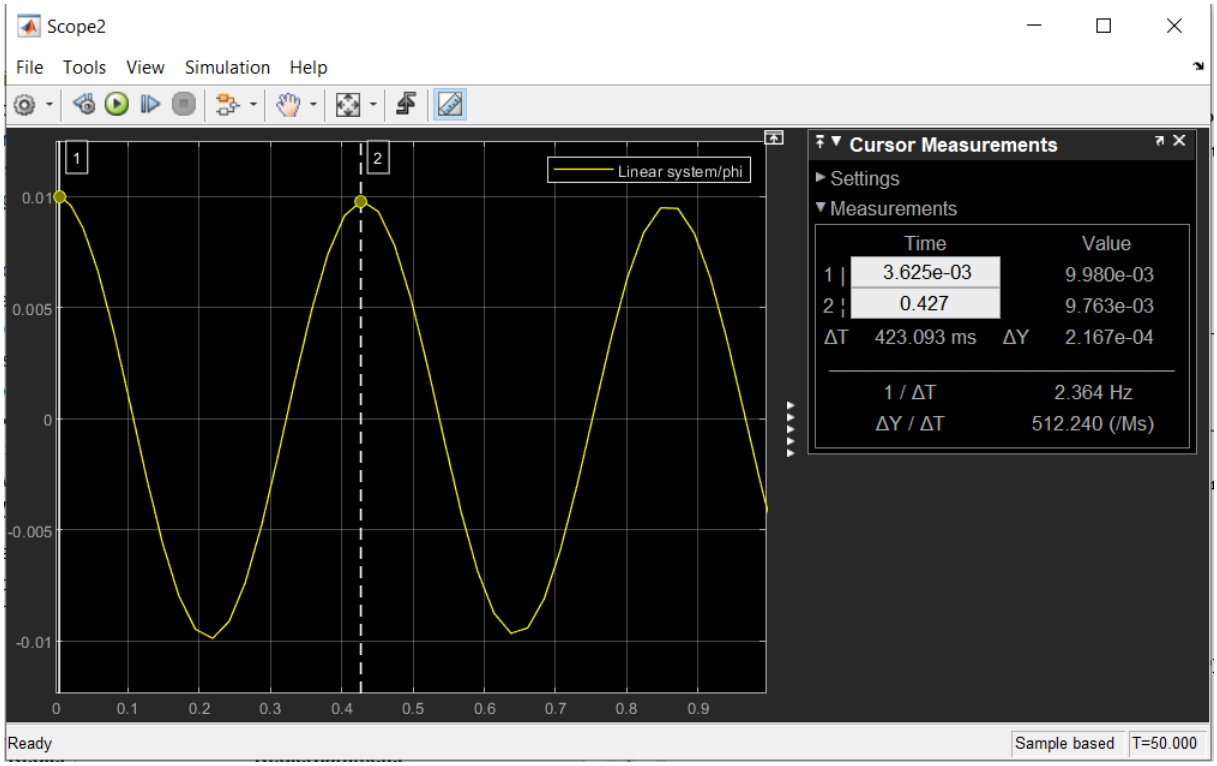


Figure 16: Measurement of the oscillation period $T_{krit} = 0.423s$

The calculation of the parameters is than done via the belonging formulas

$$T_n = T_{krit} \cdot 0.5$$

reference

$$T_v = T_{krit} \cdot 0.125$$

$$K_R = K_{R,krit} \cdot 0.6$$

$$K_P = K_R$$

$$K_I = \frac{K_R}{T_n}$$

$$K_D = K_R \cdot T_V$$

With $T_{krit} = 0.423s$ and $K_{R,krit} = 18$ the parameters result in

$$K_P = 10.8 \tag{7}$$

$$K_D = 0.57105$$

$$K_I = 51.06$$

Then these parameters are applied to both linear and non-linear models with the following results:

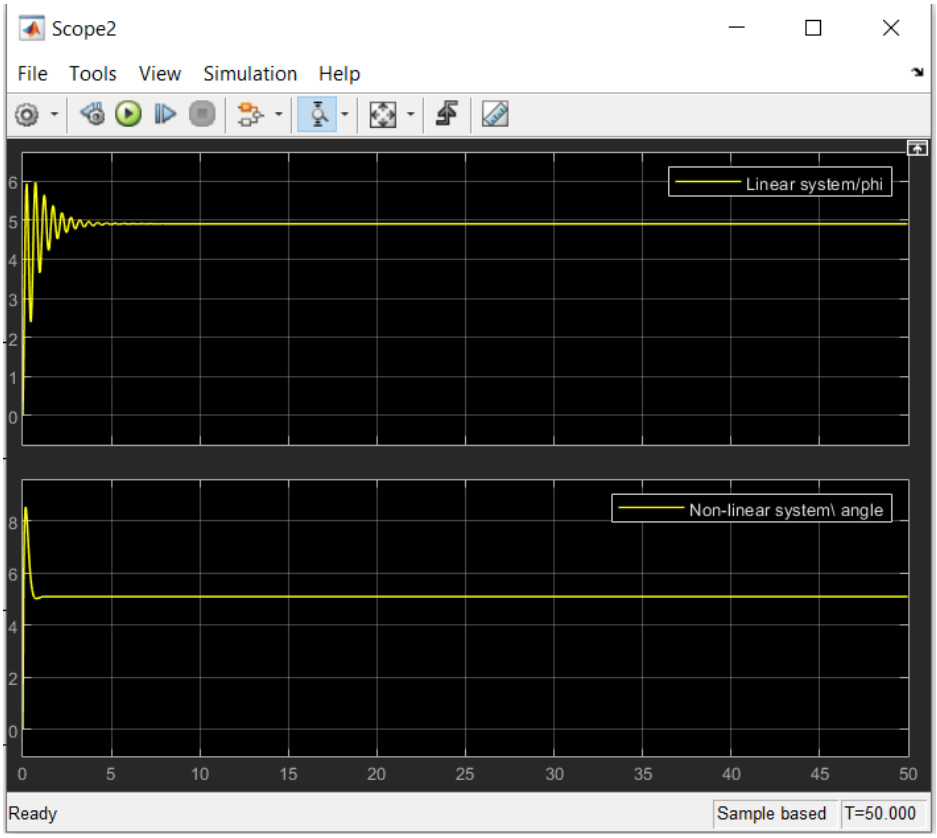


Figure 17: Controlling output of both models with the parameters of **(7)** and set point 5°

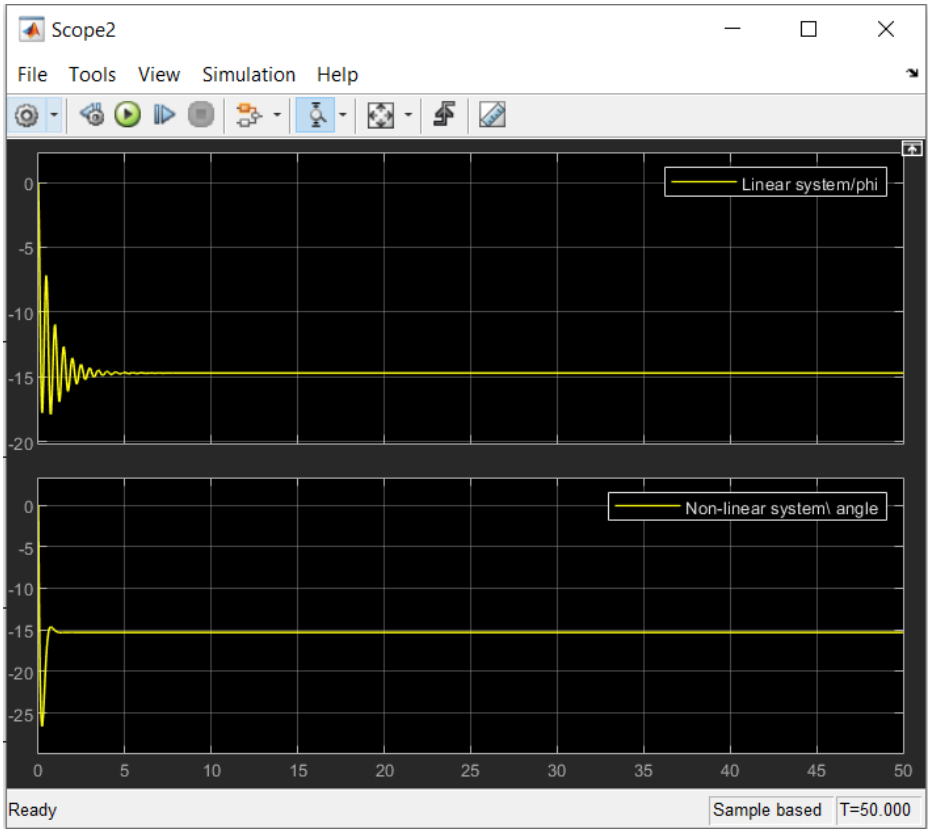


Figure 18: Controlling output of both models with the parameters of **(7)** and set point -15°

COnsider that the linear EQ is just valid for small angle range!!!

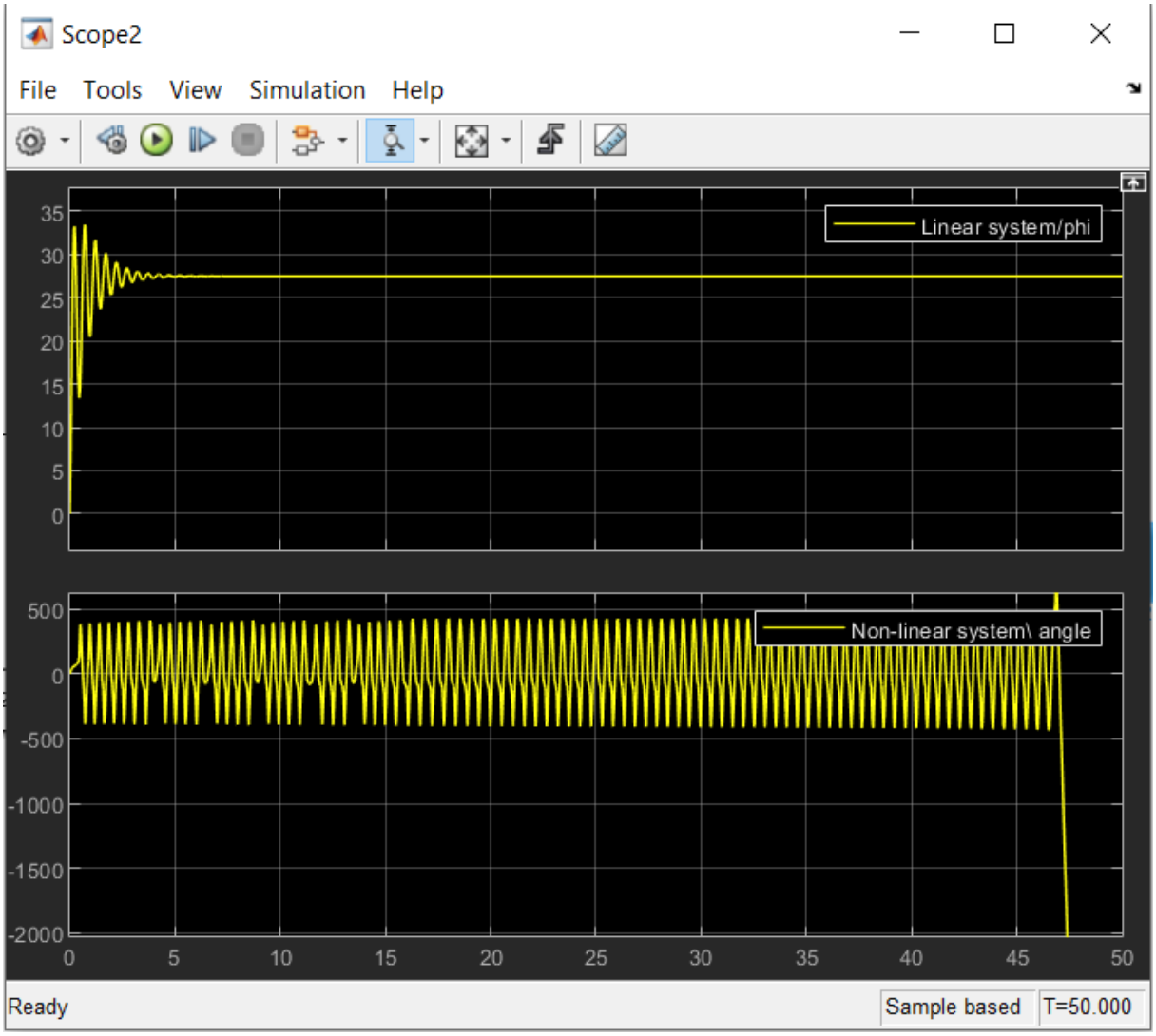


Figure 19: Controlling output of both models with the parameters of **(7)** and set point 28°

As can be seen in Figure 17 and Figure 18 the controlling of both models with the parameters of the Ziegler Nichols method is in principle possible, but for the non-linear model they have to be all negative. In addition, they only work for a limited range of about $\pm$ 25° when being applied to the non-linear model. Another outcome from the simulation is that the models have a rather large overshoot and the linear model is also highly underdamped at its response. In order to obtain a more suitable set of PID-parameters in the following the Auto-Tuner PID tool is used.

7.3.2 Linear model autotuned

In order to find a first reference for suitable PID-parameters the PID Tune block is used according to Figure 20.

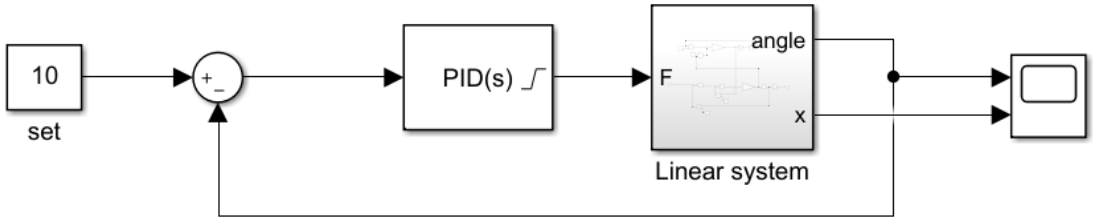


Figure 20: Additional PID Tune block for finding controlling parameters of the model

With a trade-off between robustness and speed of the controller we receive the parameter set showed in Figure 21.

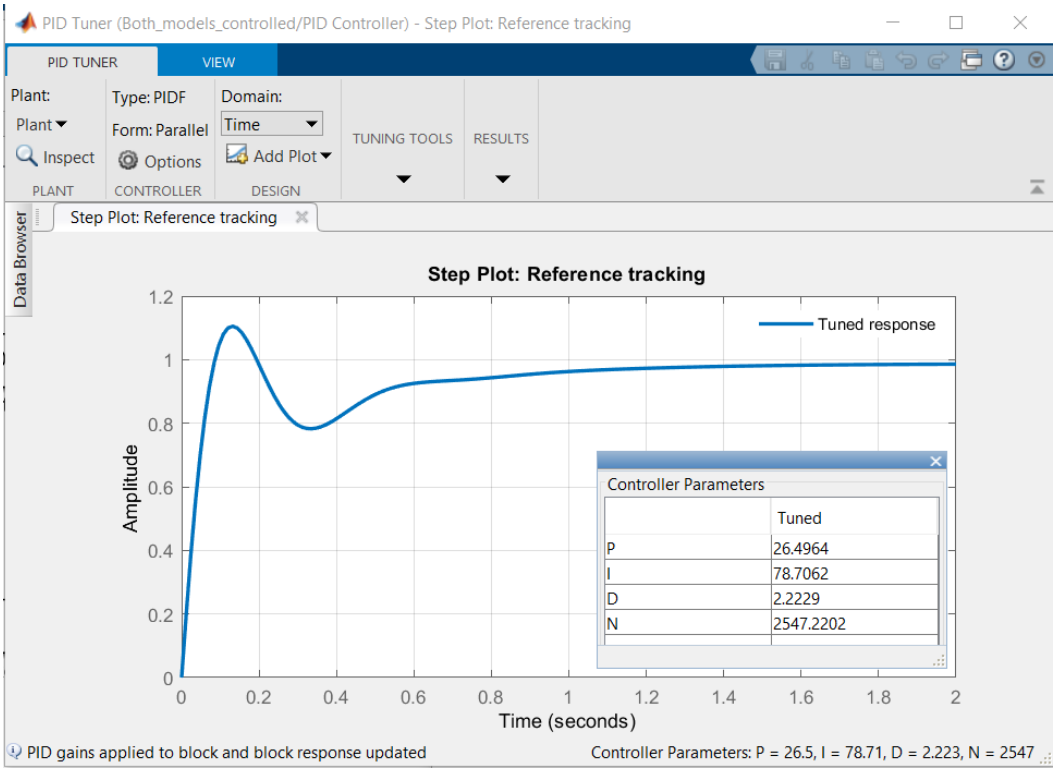


Figure 21: PID Tuner interface with calculated parameters for linear model

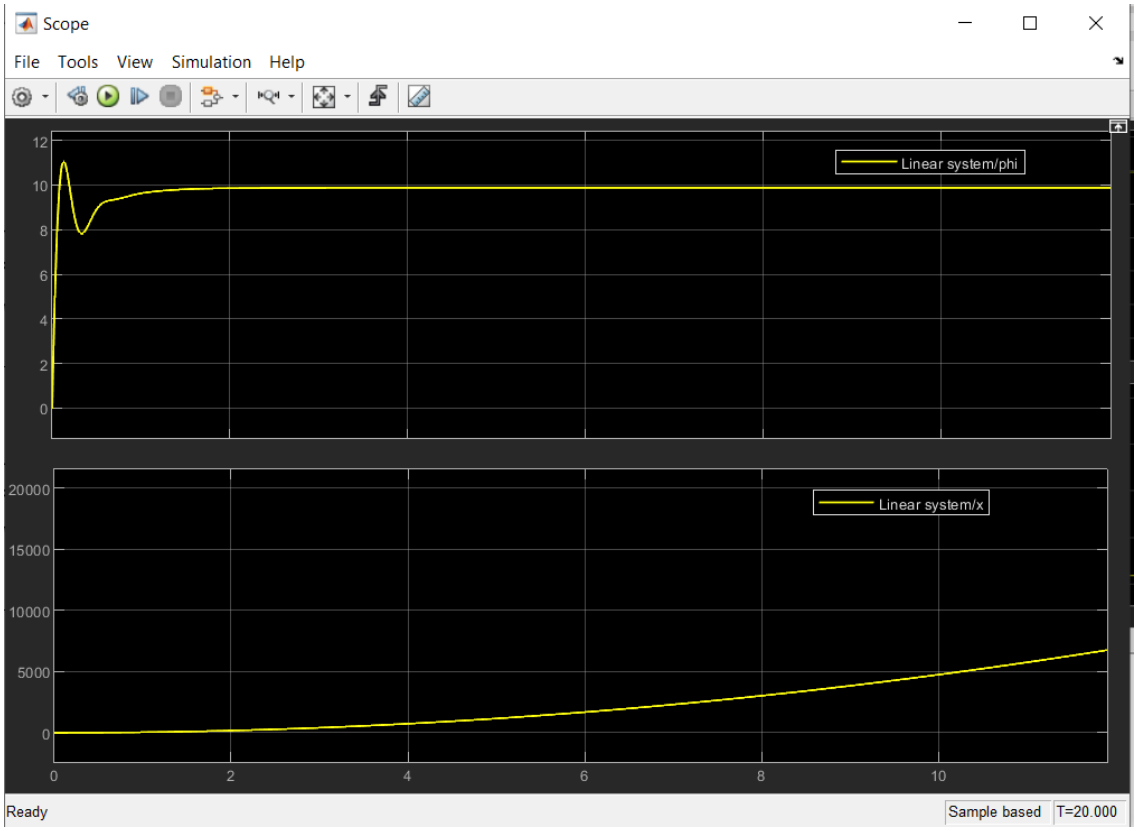


Figure 22: Controlled linear system with setpoint = 10°

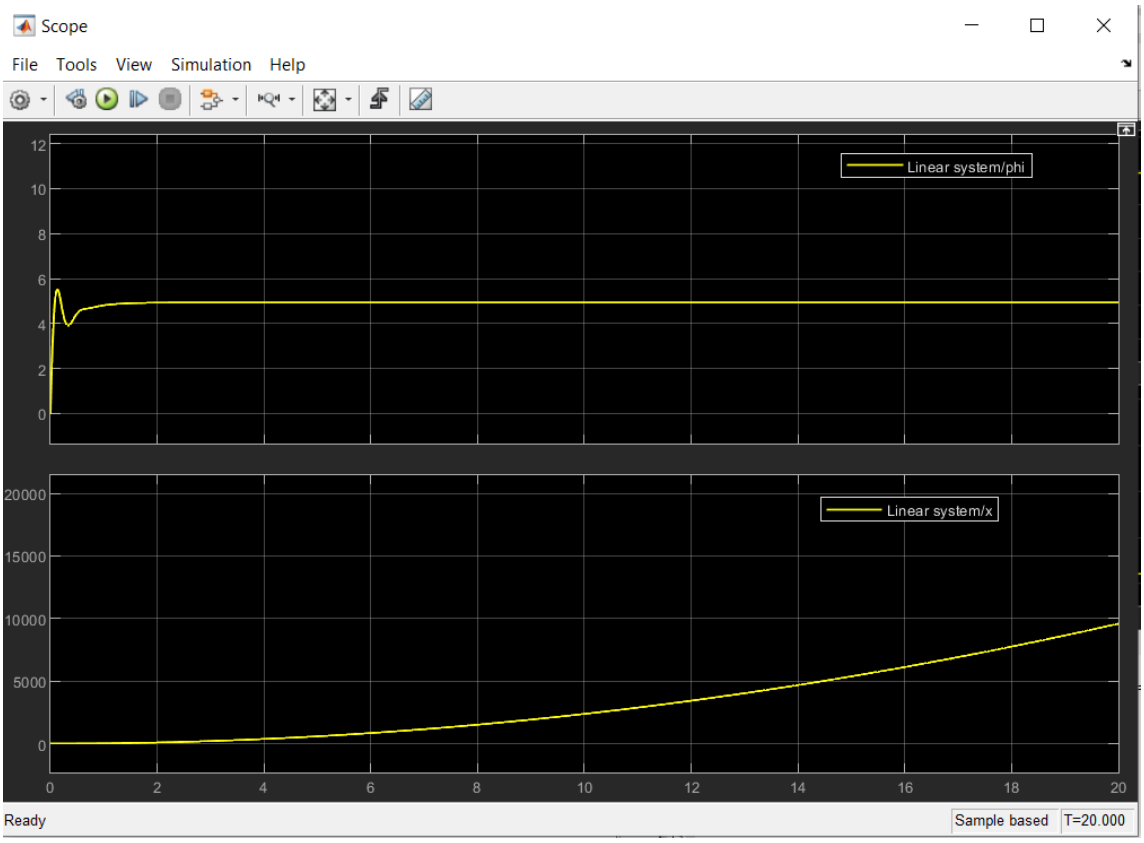


Figure 23: Controlled linear system with setpoint = 5°

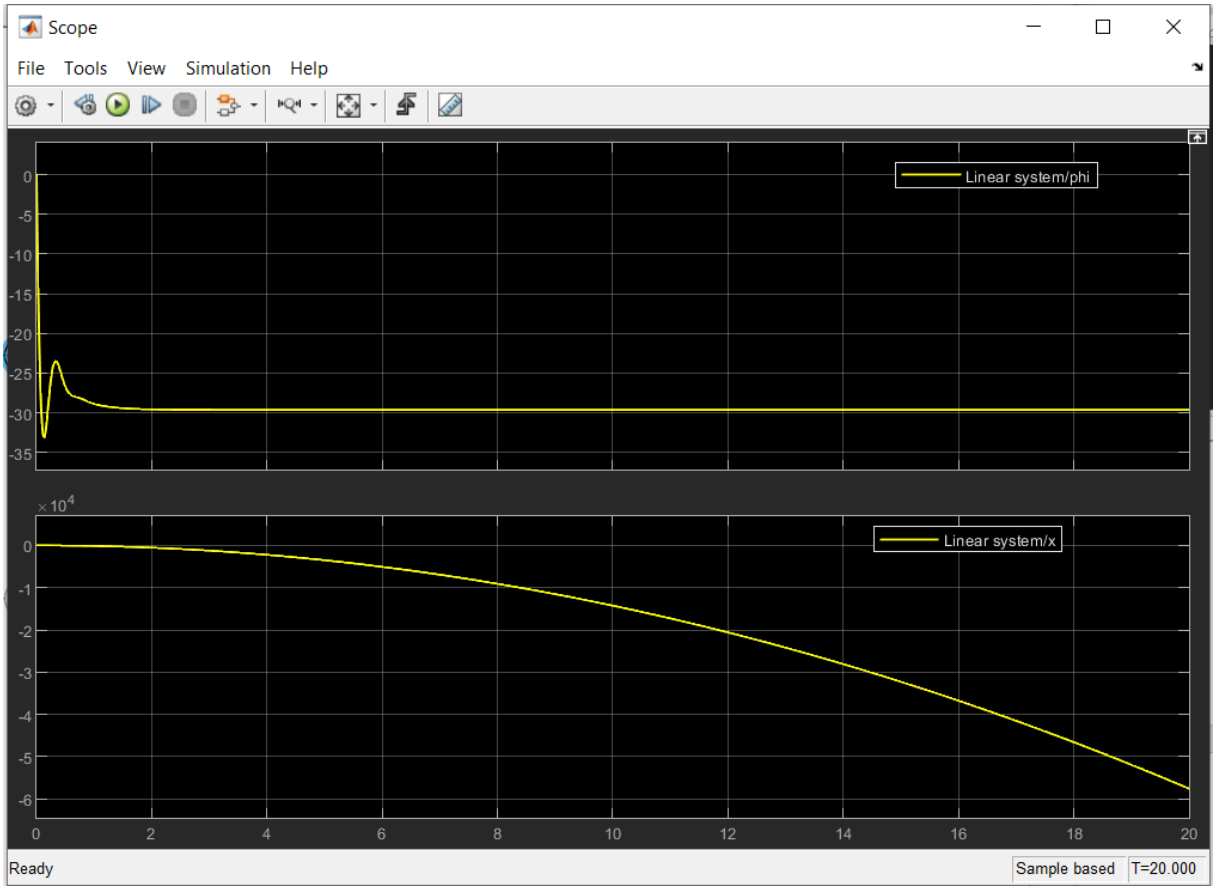


Figure 24: Controlled linear system with setpoint = -30°

The simulation results above show that the linear model is capable of controlling different angles quickly and with not to high overshoot. Of course the parameters are different to the ones found by Ziegler Nichols under 7.3.1 as the controller was set to be faster and more aggressive, but they are at least in the same proportional value range. As can be seen in the above simulations, the position of the cart is getting highly increased, because the Balanbot in reality would have to move continuously in the certain direction as it would otherwise fall to one side.

7.3.3 Non-linear model autotuned

The same process as under 7.3.3 is also done for the non-linear model where the PID-parameters result in Figure 25.

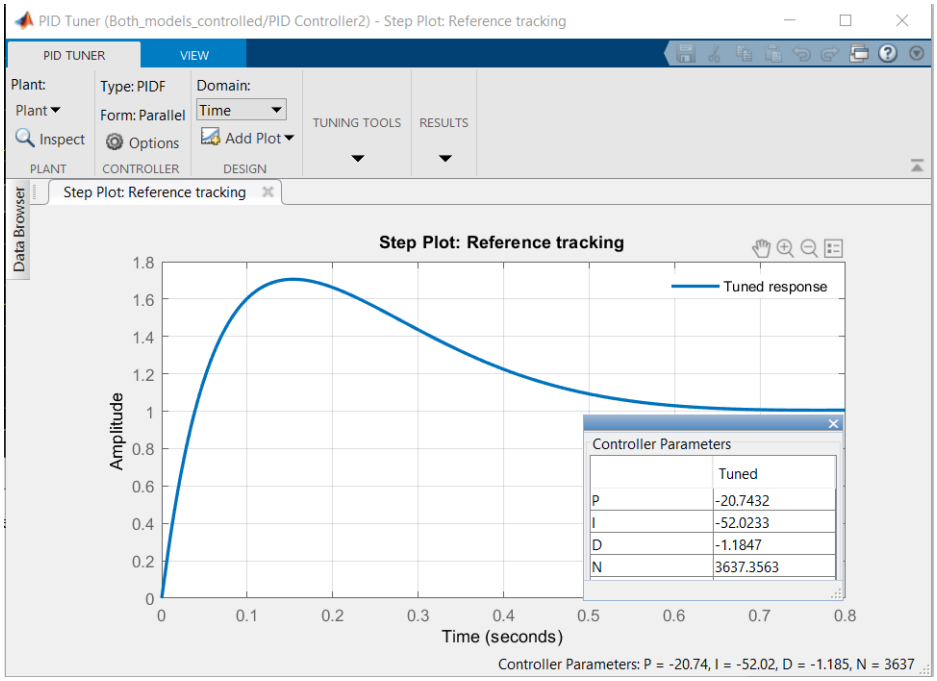


Figure 25: PID Tuner interface with calculated parameters for non-linear model

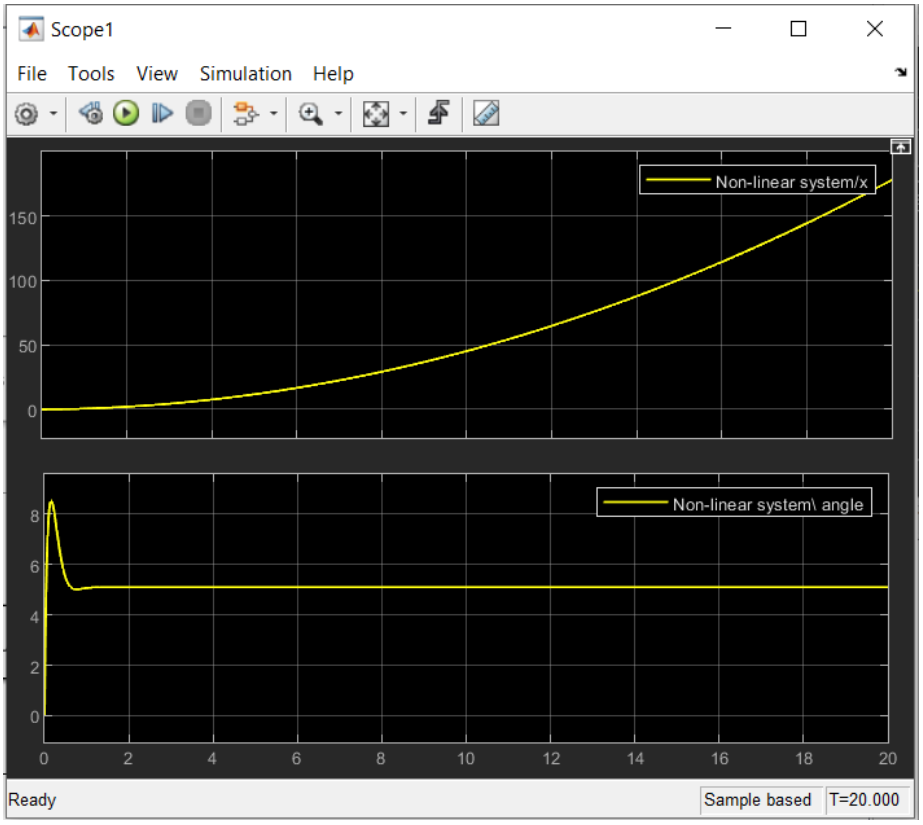


Figure 26: Controlled non-linear system with setpoint = 5°

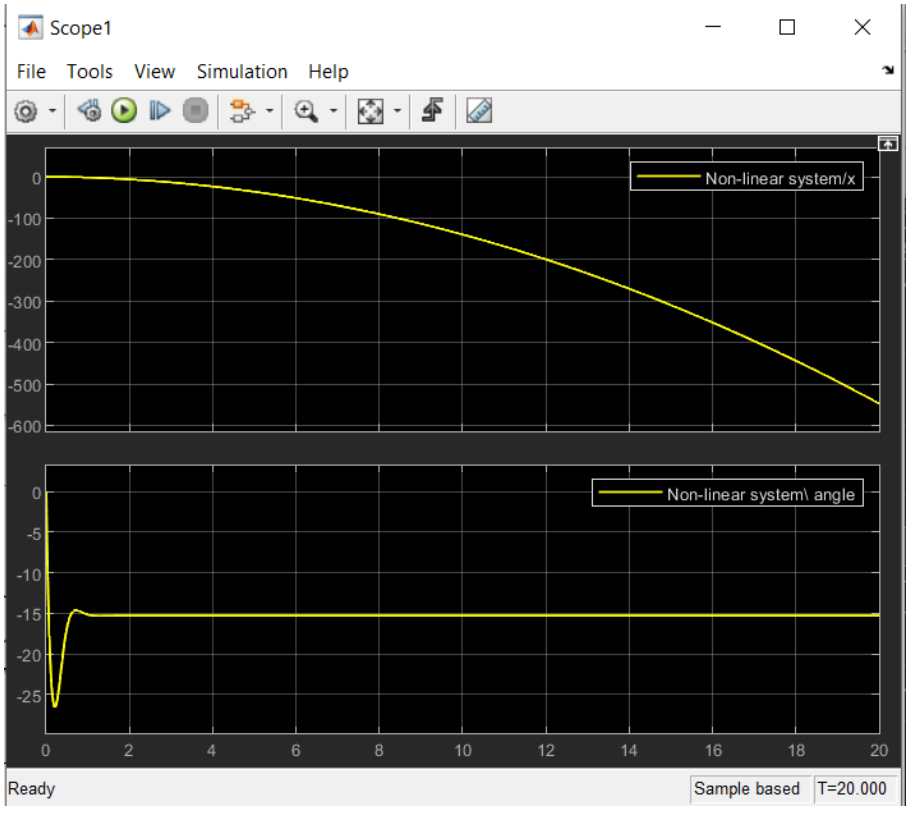


Figure 27: Controlled non-linear system with setpoint = -15°

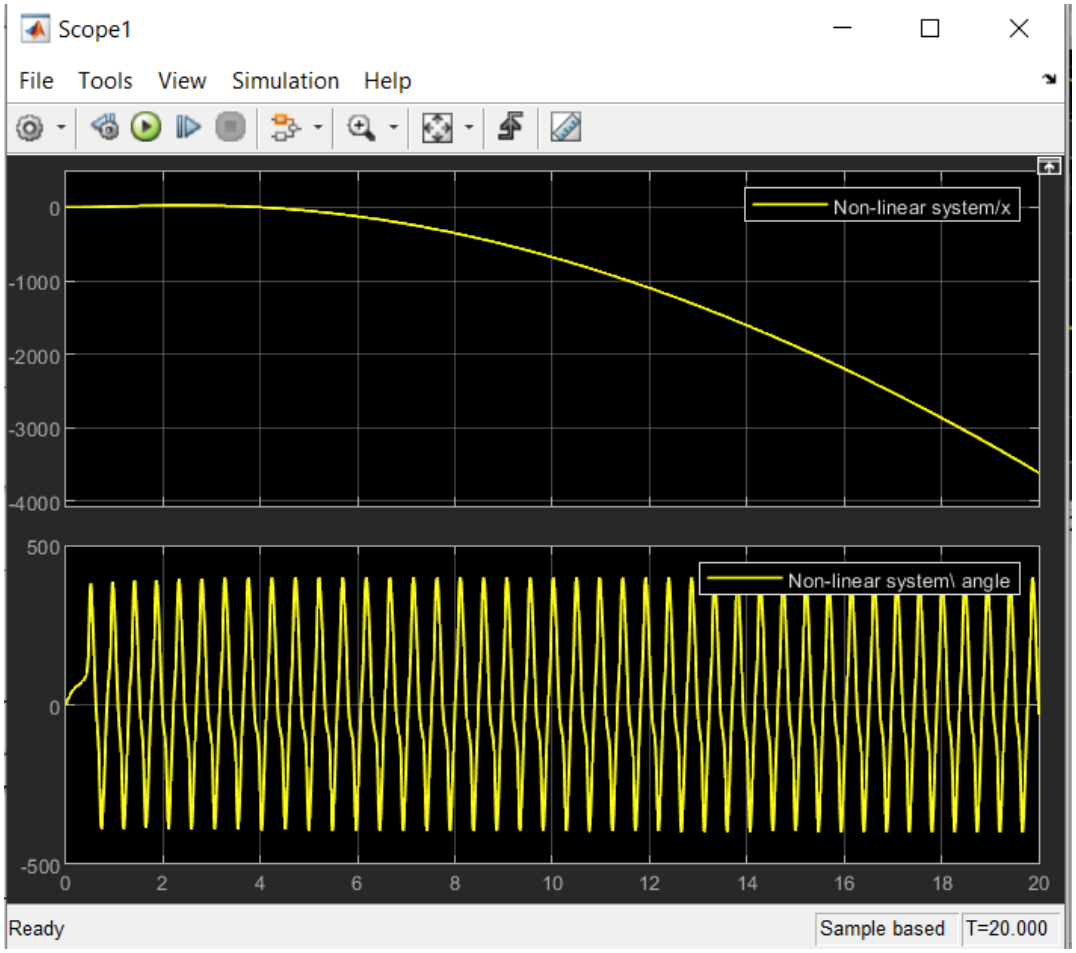


Figure 28: Controlled non-linear system with setpoint = 28°

As Figure 28 indicates the non-linear model is only capable of staying stable in a range of about $\pm 25°$. The PID parameters are again at least in the same scale and also negative as needed before. The advantage of the Auto-Tune tool is that the user is able to decide about the speed and the robustness of the controller as the parameters calculated by various methods are always only kind of a base for further improvement. In conclusion the PID parameter set of the linear model of Figure 21 will serve as a first reference point for the implementation on the real hardware as the next step.

8 Hardware implementation

Based on the experiences made from the simulation part and especially the controller parameter found the Balanbot should be operated in real hardware on the concept of Rapid Control Prototyping. The Balanbot consists of an Arduino Mega 2650 as controller and a MPU6050 motor/sensor shield. The last one needs to be configured via I2C so there is a need of setting registers first. In order to configure the Balanbot the framework in Figure 29.

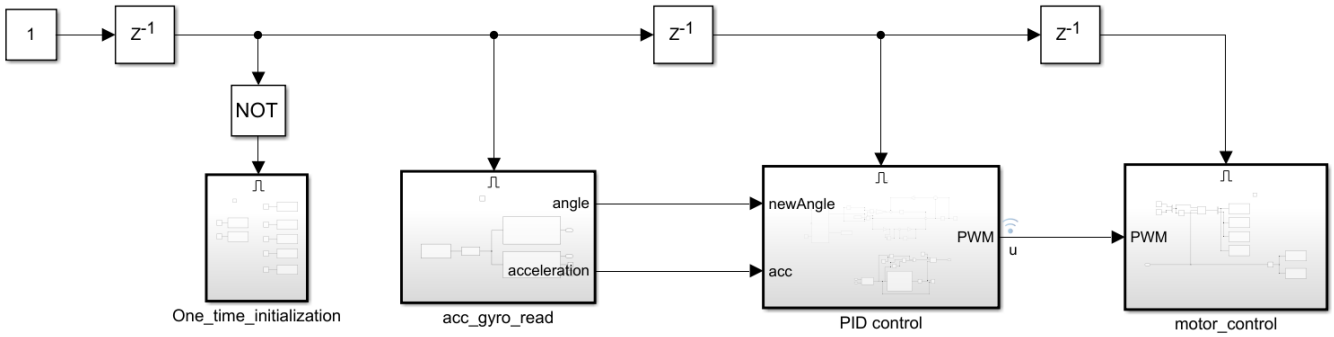


Figure 29: Simulink model framework for the hardware implementation of the Balanbot

As hardware target the Arduino Mega 2560 is chosen and a fixed-step solver with 10 ms rate is chosen according to Figure 30 in order to meet the requirement NF1 of 7.

Figure 30: Solver configuration for the hardware implementation

For the following settings the Arduino support package for Simulink has to be installed.

Via the delay blocks the functions are executed in a consecutive order and the one time-initialization in Figure 31 is only done when launching the project for initializing the sensors and outputs.

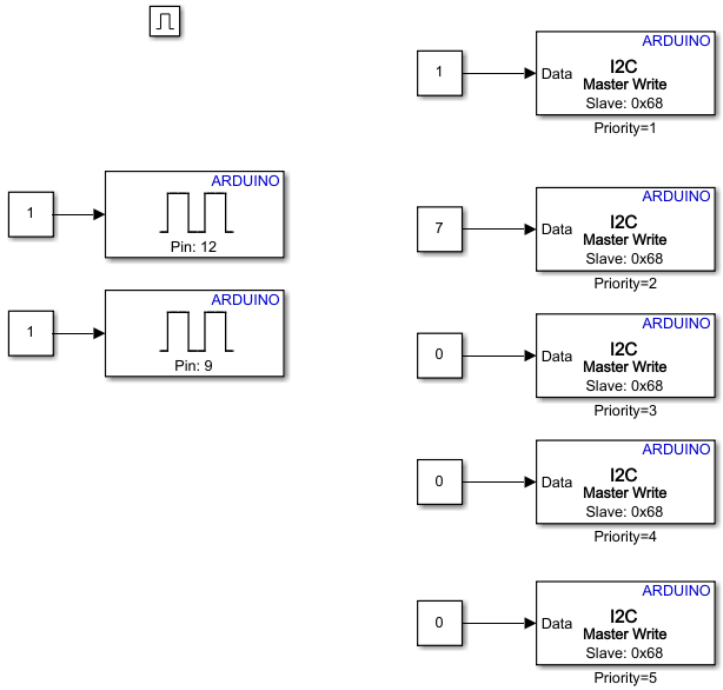


Figure 31: One time initialization of the model framework

The second subsystem acquires the data from the sensor and computes the angle and the angle acceleration via the given MATLAB function from the measured values.

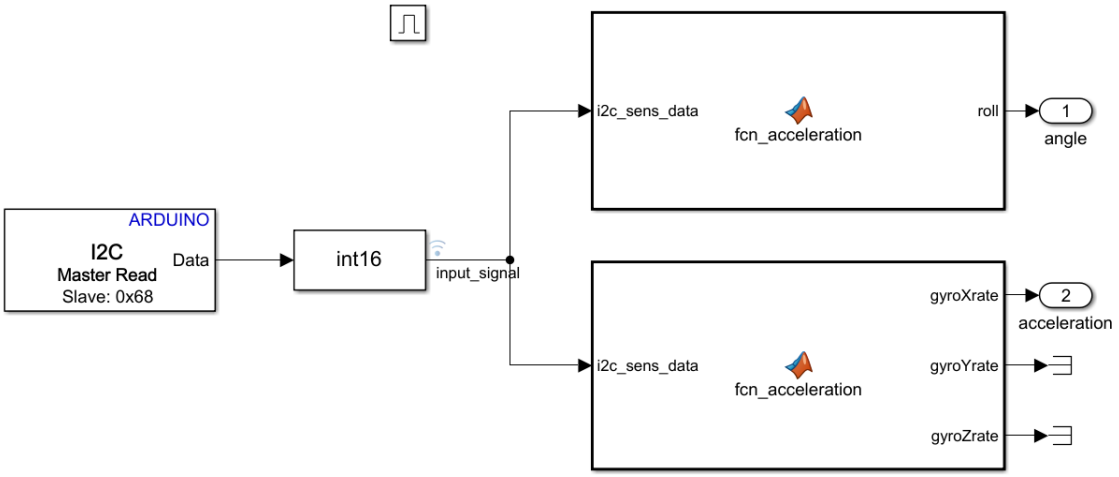


Figure 32: Gyro Sensor read model of the framework

In the next step the roll output of the sensor is compared to the signal coming out of the Kalman filter in order to check if the filtered signal is more reliable for further calculation.

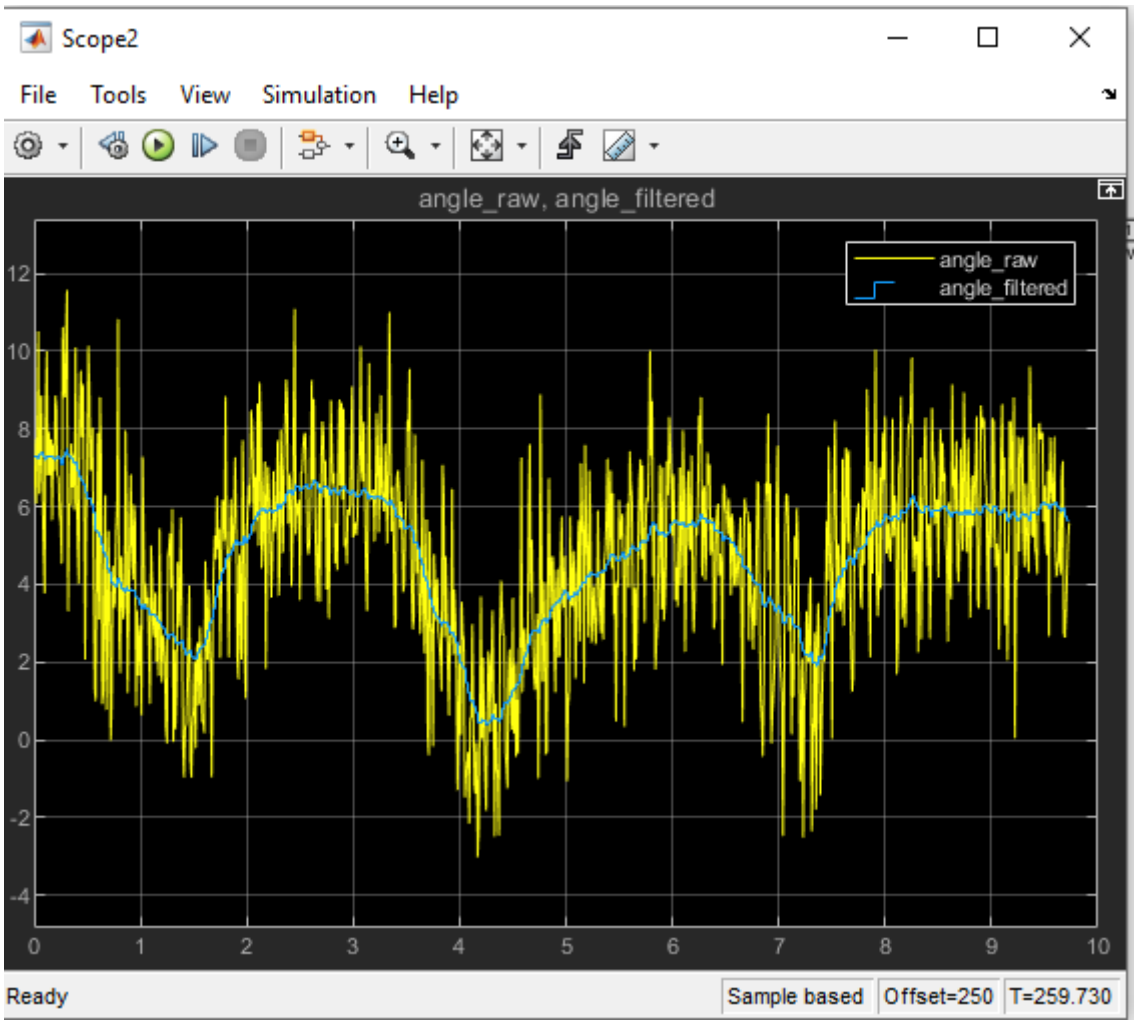


Figure 33: Output of the Kalman filter (blue) and angle values directly from the sensor (yellow)

As can be seen in Figure 33 the filtered angle signal provided by the Kalman filter is a lot smoother and low-noise and therefore more suitable for the controlling.

The third block of the framework from Figure 29 consists of the PID controller and the Kalman Filter. As can be seen in Figure 34 the Kalman Filter is fed by the incoming sensor values and provides the filtered angle values to the controller, which fulfils FR1 and NF2 of 7.

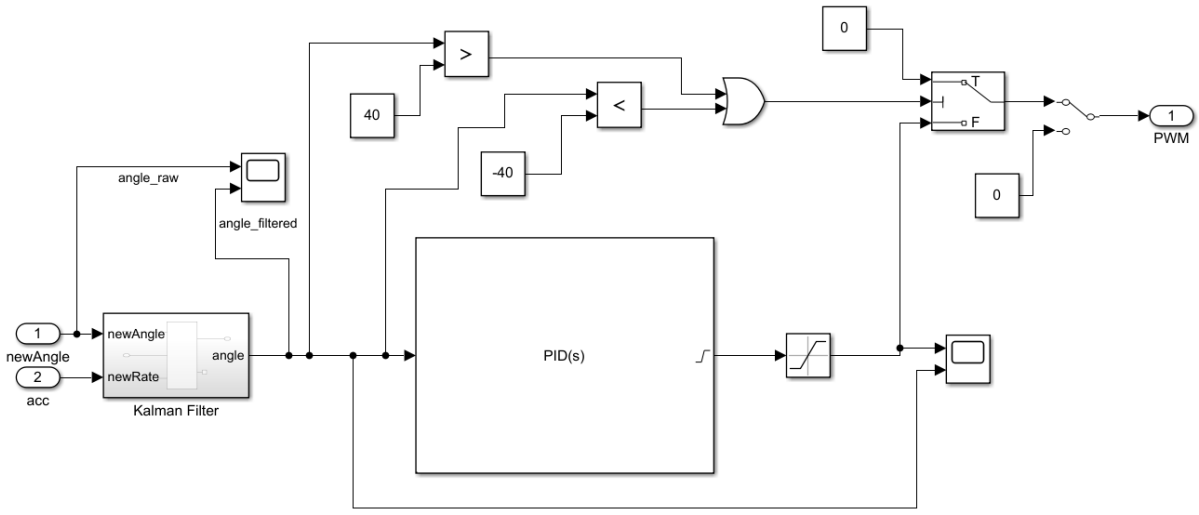


Figure 34: Configuration of the PID control subsystem

After the controller its output is saturated to -255 and 255 as the PWM value range is also limited. For safety reasons and also to fulfil FR2 of the requirements under 7, the PWM signal is zero for angles $> 40°$ or $< -40°$. In addition, there is a manual switch for turning of the motor in external mode for testing purposes. The PWM signal is than connected to the following motor control block in order to finally control the motor output pins.

In the last subsystem – the motor control – in Figure 35 the absolute value of the PWM signal (0-255) is applied to the two motor pins 5 and 6. According to the sign of the signal, the automatic switch changes the direction of rotation of the motors.

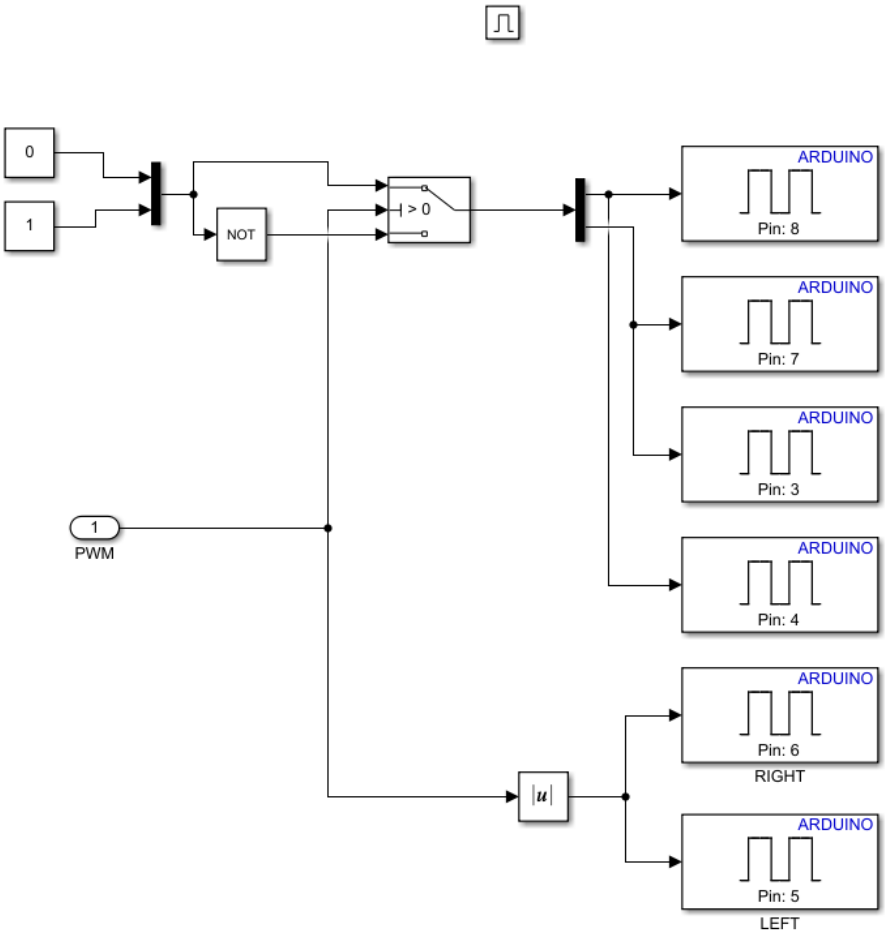


Figure 35: motor control subsystem of the framework

8.1 Testing

On the basis of the controlling of the real hardware the parameters for holding the system stable for a reasonable time (around 15 seconds) were derived by trying the linear model autotuned parameters from Figure 21. The final parameters used were then found empirically:

- P=15
- I = 80
- D = 0.5
- Filter Coefficient N = 800

9 Discretization

In the last step the system is discretization of the system is done with three different methods and a sample time of T=0.001 in order to analyse the discrete transfer functions according to its pole and zero allocation.

9.1 Tustin with T=0.001

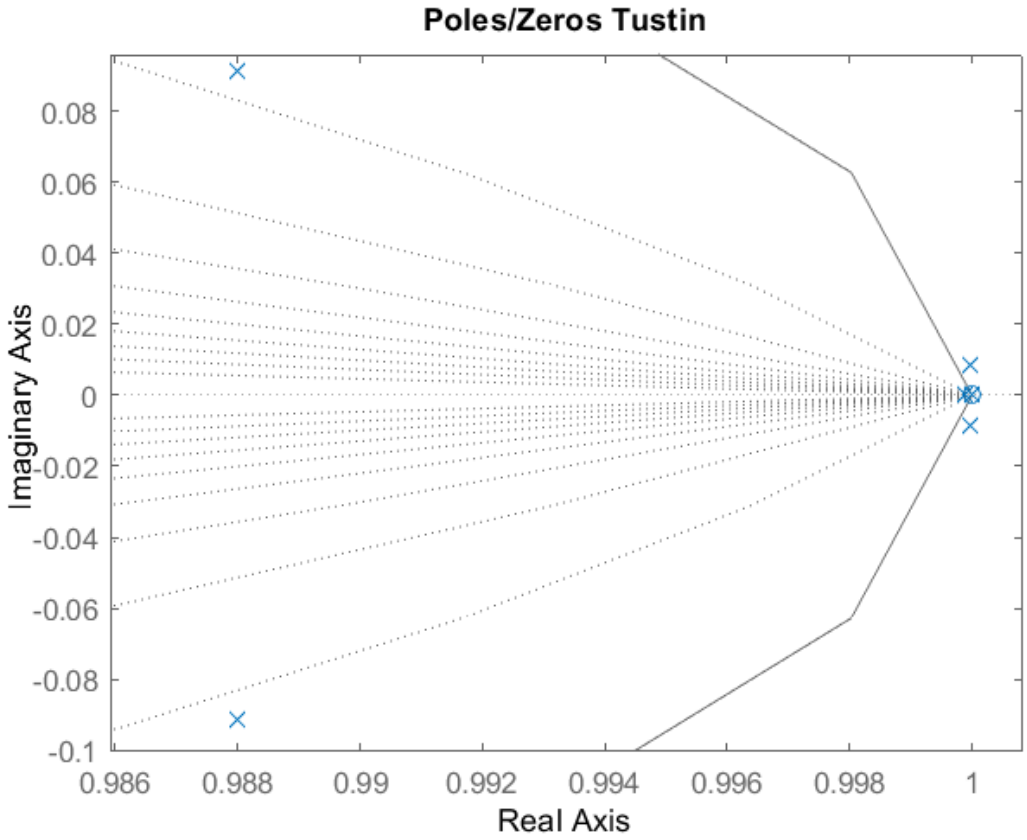


Figure 36: Pole-zero mapping of discretized transfer model with Tustin method with T=0.001

```
[poles_tustin,zeros_tustin]=pzmap(tustin)
poles_tustin = 8×1 complex
   1.0000 + 0.0084i
   1.0000 - 0.0084i
   1.0000 + 0.0000i
   0.9999 + 0.0000i
   0.9880 + 0.0912i
   0.9880 - 0.0912i
   1.0000 + 0.0000i
   0.9999 + 0.0000i
zeros_tustin = 1.0000
```


The poles and zeros of the linear system shall be purely on the x axis!
If everything is right

9.2 ZOH with T=0.001

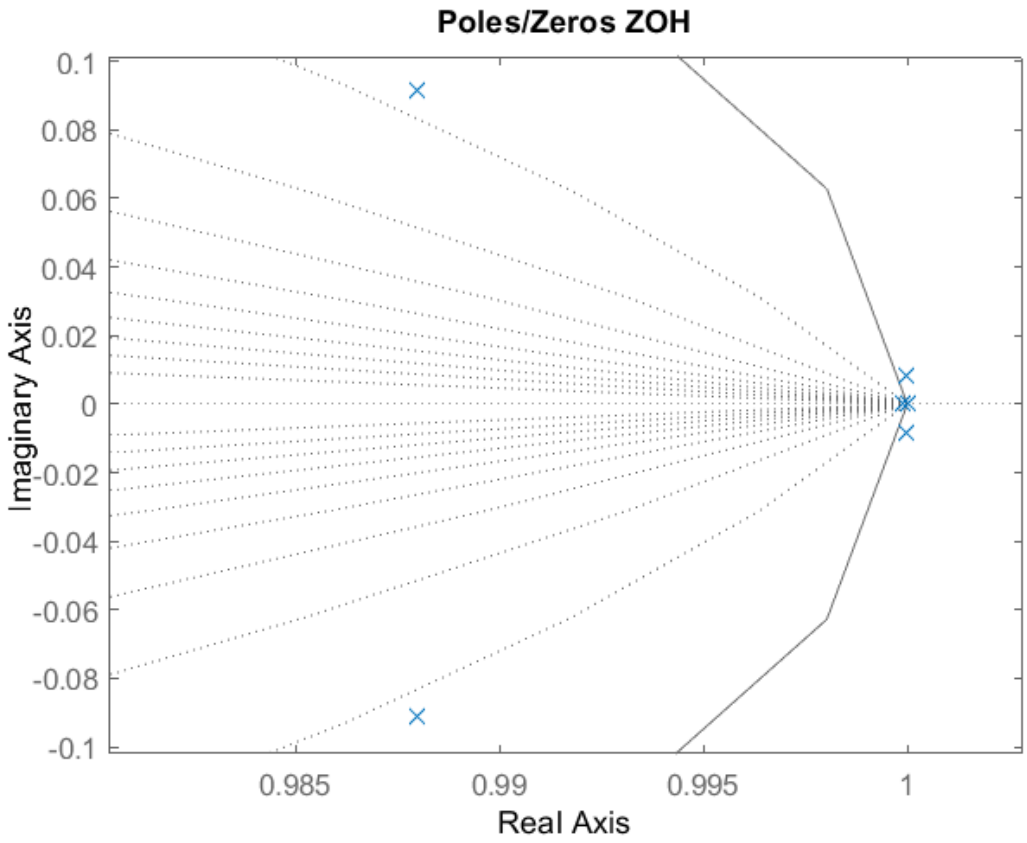


Figure 37: Pole-zero mapping of discretized transfer model with ZOH method with T=0.001

```
[poles_zoh,zeros_zoh]=pzmap(zoh)
poles_zoh = 7×1 complex

   1.0000 + 0.0084i
   1.0000 - 0.0084i
   1.0000 + 0.0000i
   0.9999 + 0.0000i
   0.9999 + 0.0000i
   0.9880 + 0.0913i
   0.9880 - 0.0913i

zeros_zoh                                                                =

  0×1 empty double column vector
```

9.3 FOH with T=0.001

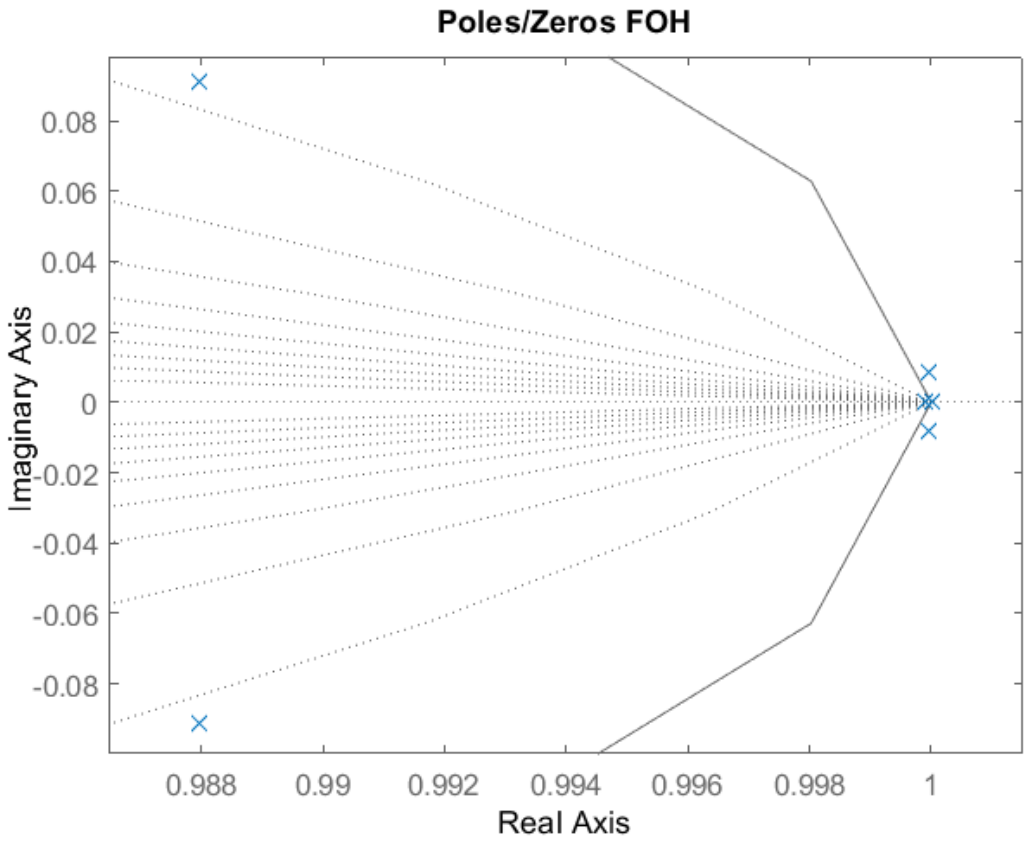


Figure 38: Pole-zero mapping of discretized transfer model with FOH method with T=0.001

```
[poles_foh,zeros_foh]=pzmap(foh)
poles_foh = 7×1 complex

   1.0000                                +                                0.0084i
   1.0000                                -                                0.0084i
   1.0000                                +                                0.0000i
   0.9999                                +                                0.0000i
   0.9999                                +                                0.0000i
   0.9880                                +                                0.0913i
   0.9880 - 0.0913i

zeros_foh                                                                       =

  0×1 empty double column vector
```

9.4 Tustin with T=0.01

```
[poles_tustin,zeros_tustin]=pzmap(tustin)
poles_tustin = 8×1 complex
   0.9964 + 0.0835i
   0.9964 - 0.0835i
   1.0000 + 0.0000i
   0.9989 + 0.0000i
   0.6086 + 0.7128i
   0.6086 - 0.7128i
   1.0000 + 0.0000i
   0.9989 + 0.0000i
zeros_tustin =

  0×1 empty double column vector
```

9.5 ZOH with T=0.01

```
[poles_zoh,zeros_zoh]=pzmap(zoh)
poles_zoh = 7×1 complex
   0.9964 + 0.0836i
   0.9964 - 0.0836i
   1.0000 + 0.0000i
   0.9989 + 0.0000i
   0.9989 + 0.0000i
   0.5593 + 0.7361i
   0.5593 - 0.7361i
zeros_zoh =

  0×1 empty double column vector
```

9.6 FOH with T=0.01

```
[poles_foh,zeros_foh]=pzmap(foh)
poles_foh = 7×1 complex
   0.9964 + 0.0836i
   0.9964 - 0.0836i
   1.0000 + 0.0000i
   0.9989 + 0.0000i
   0.9989 + 0.0000i
   0.5593 + 0.7361i
   0.5593 - 0.7361i
zeros_foh =

  0×1 empty double column vector
```

9.7 Conclusion

As the above three discretization results in 9.1, 9.2 and 9.3 show, with T=0.001 there are poles for every discretization method lying outside the unit circle, which in the z-domain means that the system is not stable. In contrast to that, by reducing the sample time to T=0.01 in 9.4, 9.5 and 9.6 the poles of the system functions are just within the unit circle and therefore stable. Hence, for discretization the sample time is crucial for defining the system´s stability. By means of building the Simulink model for the discretized system, the usage of discrete-time integrators is necessary in comparison to the continuous model.

10 Summary

For the controlling of the Balanbot the whole design process of a Model-Based Design approach was accomplished. Starting with deriving the system´s equations, the understanding for the behaviour of the system was deepened and could then be transferred to Simulink models. By taking into account some physical restraints it was possibly to linearize the system´s equations as we are in our case dealing with LTI-systems and linear hardware. The further step of deriving the transfer functions was important for analysing the system´s stability in MATLAB via investigating the allocation of poles and zeros. In the next step PID-parameters for the controlling were found by Nichols-Ziegler method as a first reference point. By using the PID Autotune Tool, the parameters came even closer to the ones finally used. For demonstration, the linear system equations were then discretized with different methods and showed that the sampling time is crucial for the stability of the system.

Sources

[1] A. Steinhuber, „ECE MBD: Laboratory session," Graz, 2019.

Finished on 20.02.2020

(Mario Waldherr)

(Iris Unterkircher)