Control Theory

*Dennis Hens* (r0852264)
*Nick Hosewol* (r0849415)

# Assignment 2: Velocity control of the cart

REPORT

Academic year 2023-2024

# Contents

## List of Figures

# 1 Design of a velocity controller

## 1.1 Controller choice

The main requirement for the velocity controller is that the steady-state error is zero on a constant velocity reference signal. This is achieved for a system with at least type 1. Ideally, the total system reaches an acceptable error (like one smaller than the general noise on the motor output) in an acceptable amount of time, on the order of 1 $s$ or less. As this is more a qualitative requirement (as there are no hard numerical requirements on the performance), this is not taken into account as a design parameter and is only checked after the design.
Because of the simplicity of the system, a Proportional-Integral (PI-) controller is chosen. The reason for this is that the PI-controller is very easy to implement due to its limited design parameters, while still yielding zero steady-state error when a step input is applied, due to the fact that this controller has an infinite gain at DC, and thus raising the type of the system with 1. The use of a PI-controller also ensures no amplification of high frequencies (in contrast to a lead compensator, for example), which could lead to (unnecessary and unwanted) noise amplicification.

## 1.2 Implementation of the controller

The controller is designed based on the frequency response, with particular emphasis on the phase margin (PM) as the most important parameter. The PM is a measure of the stability and the robustness of the closed loop system, as it quantifies how far the open loop system stays away from the value of $-1$, which would induce an infinite gain at the cross-over frequency (the resonance frequency), which induces instability at this frequency. Mathematically, this is more rigorously described by the Nyquist Stability criterion, which states that an encirclement of the point $(-1, 0)$ implies a pole in the right hand side of the real plane (as long as there are no zeroes there, as is the case for a general PI-controller, and the DC-motor model), thus leading to instability. A selection of 55° for the phase margin is made to achieve the necessary robustness, which means that even if there are some modelling errors, the system does not reach instability. Furthermore, for a second order system (or higher) the damping ratio heuristically increases for a larger phase margin (as $\xi \approx \frac{PM}{100}$), such that the system does not overshoot as much, leading to better transient behaviour. This also implies a lower settling time (as the system is underdamped). As a PI-controller inherently introduces phase lag, we have to account for this by designing for an additional phase lag of 15°. This means that the PI-controller does actually not inherently increase the phase characteristics, like a lead compensator would. The goal of the PI-controller is thus to shift the cross-over frequency to a lower value, such that the desired phase margin is reached.
To find the necessary cross-over frequency $\omega_c$, we search for the frequency at which the phase, $\phi$, of the uncompensated open loop system reaches the value given by equation 1. This is the desired phase $(-180° + PM)$at this frequency, plus the added phase lag of the PI-controller.

$$\begin{aligned} \phi &= -180° + PM + 15° = -180° + 55° + 15° \\ \phi &= -110° \end{aligned} \tag{1}$$

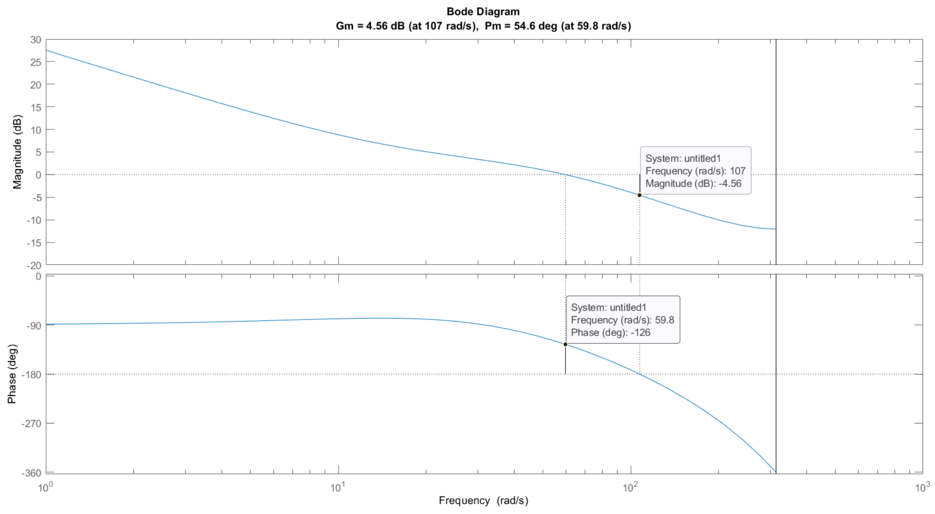
This frequency is now calculated using the phase plot of the transfer function that describes the DC-motors, as found in assignment 1. This is done easily using $MATLAB$s interpolation command $interp1$. The crossover frequency, which has the necessary phase, $\phi$, is $\omega_c = 59.25 rad/s$ or $f_c = 9.43 Hz$ for motor A. For motor B, which has a slightly different transfer function, the desired crossover frequency at $\omega_c = 60.90 rad/s$, which corresponds to a frequency $f_c = 9.69\ Hz$. Next, the integration time, $T_i$, one of the parameters of the PI-controller, is chosen such that the guess for the phase lag at the crossover frequency is exactly 15°, as anticipated. This can be easily derived using the transfer function, $D(j\omega)$, of a PI-controller. As the transfer function of the motors is not relevant, the integration times of the PI-controllers are identical for both motors.

$$\begin{aligned} D(j\omega) &= K\frac{j\omega T_i + 1}{j\omega T_i} \\ \iff \angle(D(j\omega)) &= arctan(\frac{-1}{\omega T_i}) \\ \implies T_i\omega_c &= tan(90° - 15°) \\ \iff T_i &= \frac{1}{\omega_c \cdot tan(15° \cdot \frac{\pi}{180°})} = 0.063\ s \end{aligned} \tag{2}$$

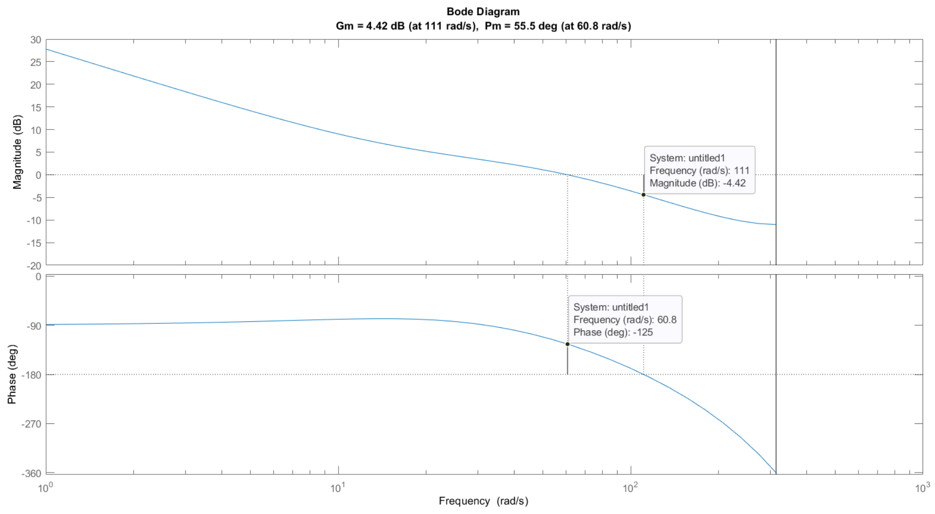
Thus, the integration time $T_i$ is determined uniquely by the desired phase at the desired cross-over frequency. Next, for the desired cross-over frequency to be the actual cross-over frequency of the total open loop system, the gain has to equal exactly 1 at this frequency. Equation 3 shows this constraint in discrete time. Here, $G(e^{j\omega_c})$ is the discrete time transfer function for the DC-motors.

$$|D(e^{j\omega_c}) \cdot G(e^{j\omega_c})| = 1 \tag{3}$$

Now as the PI-controller is designed in continuous time, while the transfer function of the motors is given in discrete time. The process of finding the right value for $K$ is executed iteratively. First, a value for $K$ is proposed. Then, the continuous time PI-controller is discretised with the *MATLAB* command *c2d*, using the Tustin method. The bode plot of $D(e^{j\omega}) \cdot G(e^{j\omega})$ is plotted in *MATLAB* using the command *bode*. Now, the value for $K$ is altered, until the cross-over frequency *MATLAB* calculates, equals the desired cross-over frequency. The correct results are found for $K = 0.74$ for motor A, and $K = 0.745$ for motor B. For these values, the open loop transfer function, $D(e^{j\omega}) \cdot G(e^{j\omega})$, is shown in figure 1 to indicate these values.



(a) Motor A

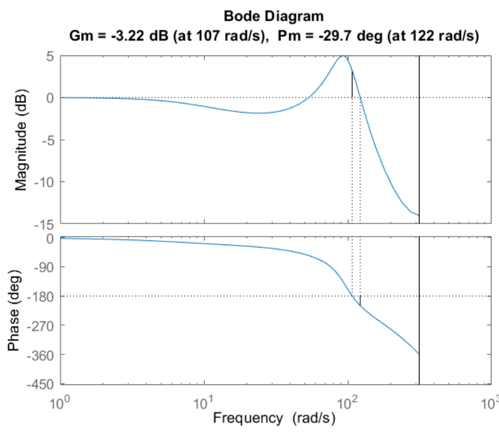


(b) Motor B

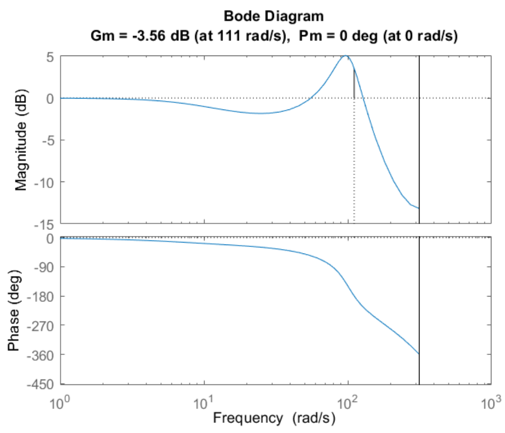
Figure 1: Bode diagram of the open loop frequency response

This concrete and applied analysis has shown the importance of the design parameters of the PI-controller. The integration time $T_i$ determines the frequency at which the PI-controller has a phase of $-45^\circ$ (as $\omega_{-45^\circ} = \frac{1}{T_i}$,

after all). As the phase lag at the operating frequency has to be as small as possible (remember, the PM needs to be large enough), $\frac{1}{T_i}$ should be much smaller than $\omega_c$. Increasing the $T_i$, therefore, decreases the phase lag of the PI-controller and directly increases the PM (and the other way around). Furthermore, decreasing $T_i$, increases the speed of integrating action, leading to a faster response. $T_i$ therefore poses a trade-off between speed of integrating action and phase lag of the PI-controller (which in turn influences the PM). K represents the proportional action of the PI-controller. The value for $K$ determines directly where the cross-over frequency lies, by shifting the whole bode plot upwards or downwards. An increase of $K$, shifts the open loop transfer function upwards and (for a monotonous decreasing transfer function, as is the case here,) shifts $\omega_c$ to a higher frequency. Therefore, the PM decreases. If $K$ is chosen too high, instability could be reached. $K$, once again, poses a trade-off between performance (of the proportional action, this time) and stability.
The final bode plot of the open loop system is shown in figure 1.
On this plot, both phase margins are indicated with the first vertical line on the phase plot. The specific values can be read in the box. The location where this line is located on the $x$-axis is the cross-over frequency of the system. The second vertical line indicates the gain margin of the system, which is a different measure for the stability of the system. As indicated, the phase margin matches the desired phase margin.
Figure 2 shows the closed loop transfer functions of both motors. This shape is completely in line with the expectations: First, we have the operating regime, where tracking is optimal. For higher frequencies, the system goes in resonance, and amplifies the reference signal. After the bandwidth frequency, $\omega_{BW}$, the signals are too fast too track and the signal is attenuated. As the system is discrete, the plot stops at $f = 50\,Hz$n the Nyquist frequency.



(a) Motor A



(b) Motor B

Figure 2: Bode diagram of the closed loop frequency response

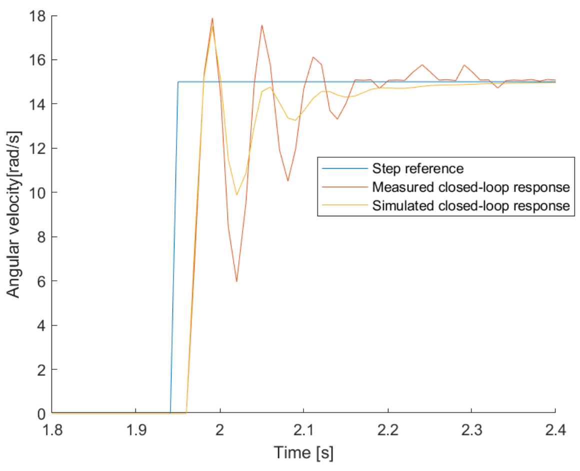
## 1.3 Analysis of the closed loop bandwidth of the system

The closed loop bandwidth of the system is defined as the point where the open loop transfer function reaches a gain of $-3\,dB$ [1], and depends on (or even, is approximated as) the cross-over frequency. To increase the cross-over frequency, one could decrease the phase margin, thus trading stability and robustness of the system for a higher bandwidth. The PM can, in theory at least, be decreased until $PM = 0°$, which would increase the bandwidth. This, however, would also lead to bigger overshoot and slower control action. A second way to increase the bandwidth a little bit, is decreasing the estimated phase lag of the PI-controller, as this directly increases the cross-over frequency. This would however slow down the integrating action (increase of $T_i$), which leads to a slower response time.
Theoretically, the PM can be decreased without big issues. In practice however, as discusses earlier, the PM is also an indicator for the damping of the system. In this case, lowering the PM leads to more oscillations and overshoot, which could be undesirable or detremental in certain situations. In practice, this poses a problem for the cart system if this creates a control signal above $12\,V$, as the *Arduino* cannot provide this voltage.
The software itself provides another theoretical limit: As the system is sampled at $100\,Hz$, the Nyquist frequency is only $50\,Hz$. If the system encounters a signal with higher frequency components, aliasing occurs, which means the system interprets this signal as one at a (corresponding) lower frequency. The bandwidth of the system can thus never be larger than $50\,Hz$.

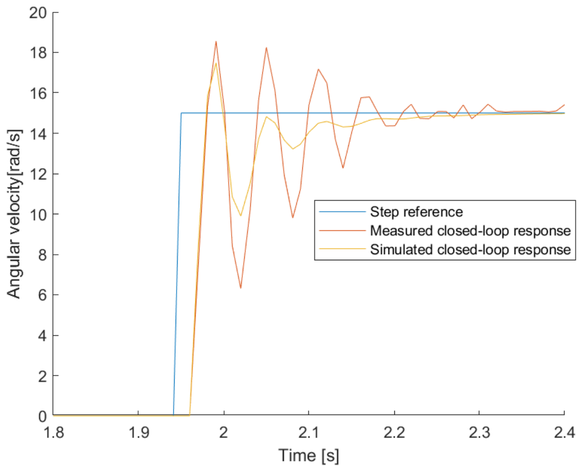
## 2 Experimental validation of the designed controller

### 2.1 Comparison between the measured and the simulated closed-loop response

In this section, the PI-controller is implemented to the cart and a step reference is imposed on the cart. Figure 3 shows the step input of 15 $\frac{rad}{s}$ on both motors. The simulated step response, as well as the measured step response are plotted on the same figure. The first noticeable thing is the expected delay between the system and the step reference. this is solely due to the structure of the model of the motors, but is thus validated here, as the simulation matches the measurements. At the first part of the transient response, an overshoot appears, almost perfectly as simulated, further showing the quality of the model. After the first peak, the oscillations of the measurements are greater than those in the simulations. At last, the real system and the simulations reach a zero steady state error, as is supposed to be the case. Due to disturbances, the steady state of the real system has little fluctuations, as is to be expected. All observations are valid for both motors. We can conclude that in this case, the model correlates extremely well with physical reality.



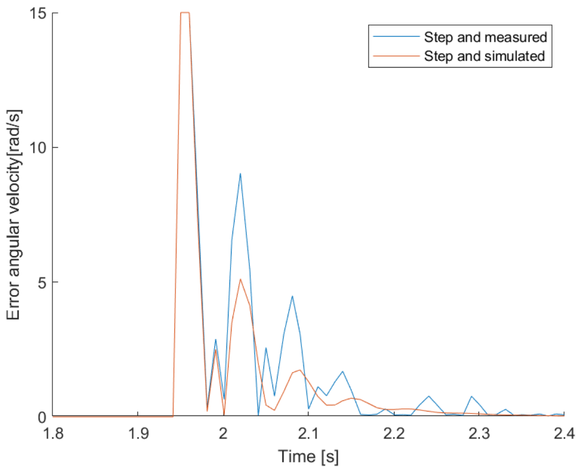
(a) Motor A



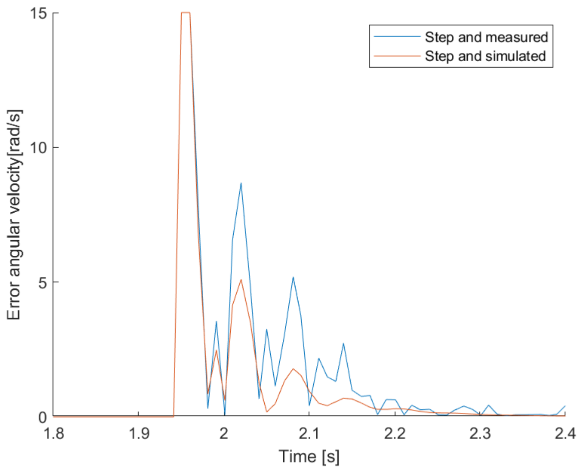
(b) Motor B

Figure 3: The step reference with its measured closed-loop response and its simulated closed-loop response

Figure 4 shows the absolute value of the tracking error of both the measurements and the simulations. As noticed before, the initial error matches the model closely. After, the oscillations are bigger in the real system than in the simulated system, but they die out at roughly the same time. Again, the behaviour of both motors is very similar.



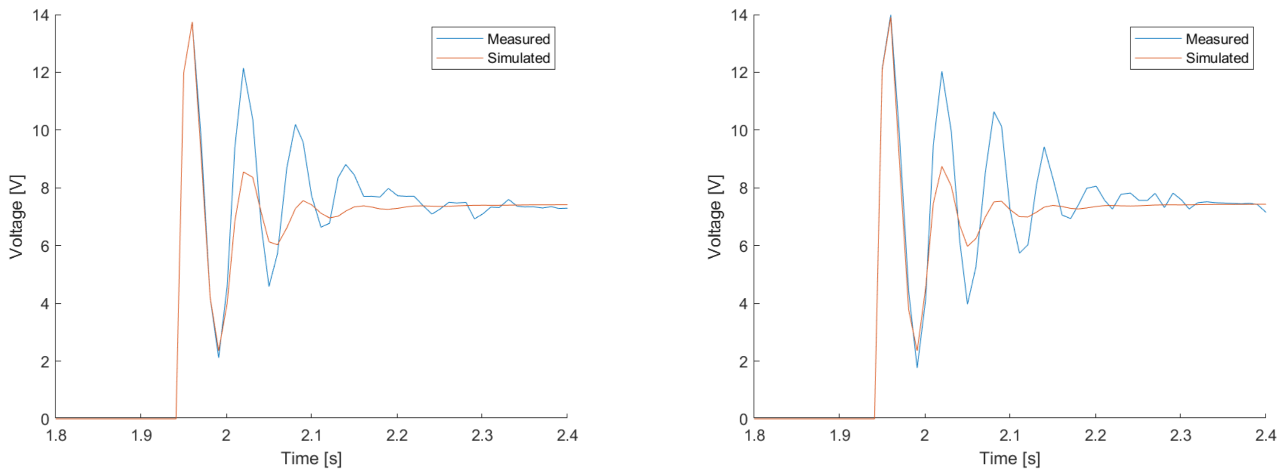
(a) Motor A



(b) Motor B

Figure 4: The measured tracking error of the step reference together with the simulated tracking error

Figure 5 shows both the measured control signal and the simulated control signal. Note that the 'measured control signal' is not directly measured on the poles of the DC motor, but what is supposed to be applied, as this voltage measurement would introduce a delay that's not physically there. The first control signal follows from a big error, as the cart starts in rest. This induces a supposed control voltage of $14\,V$. This is, however, not a voltage the *Arduino* can deliver, so this is clipped to $12\,V$. This clipping could be one of the reasons why the real systems dips lower, deviating the simulations and the measurements. Note as well that the control signal follows the same trend as the angular velocity curves. It's interesting to note that, due to the steady state fluctuations, the control voltage also fluctuates.



(a) Motor A

(b) Motor B

Figure 5: The measured control signal (voltage) of the step reference together with the simulated control signal

## 2.2 Comparison between the measured and the simulated closed-loop response while a constant force disturbance is applied

Now, the PI-controller is tested while a constant force is applied on the cart. To apply this constant force, the cart is driving on a slope, such that gravity tends to pull the cart downwards. Figure 6 shows the block diagram of the cart system with the constant force. In this diagram, $r[k]$ is the reference signal, $e[k]$ is the tracking error, $D[z]$ is the PI-controller, $G(z)$ is the transfer function of the DC motors and $f[k]$ is the constant disturbance force.

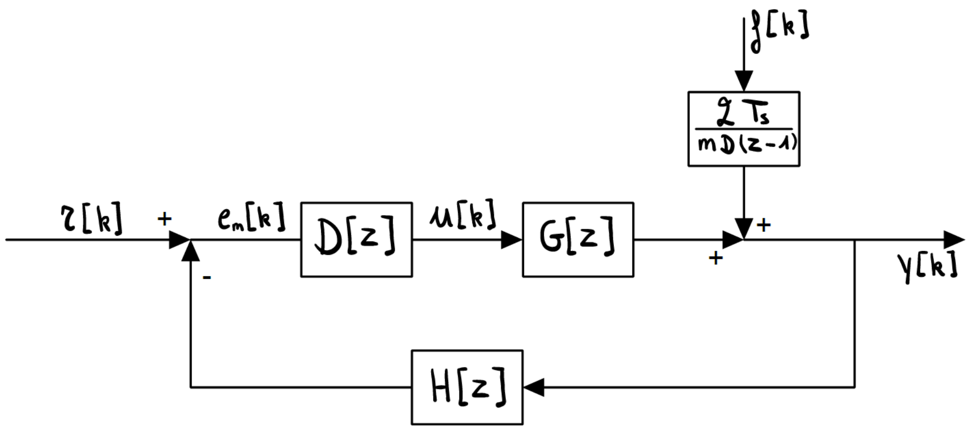
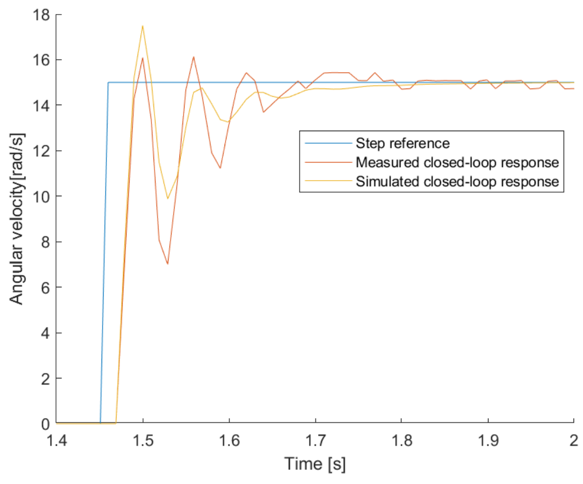


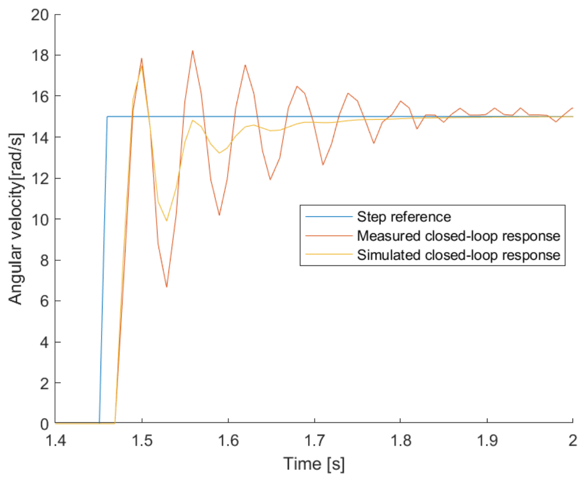
Figure 6: Block diagram of the PI controller with disturbance $f[k]$

To convert the force into its effect on the rotational velocity, Newton's second law is used, together with forward Euler discretisation: $f[k] = m\frac{v[k+1]-v[k]}{T_s} = m\frac{D}{2}\frac{(z-1)\omega[k]}{T_S}$. This result is simply added to the effect the

DC motor had on the wheels, as Newton's second law is linear.
Figures 7 and 8 show the same figures as earlier, but the measurements are now taken while driving on a slope.

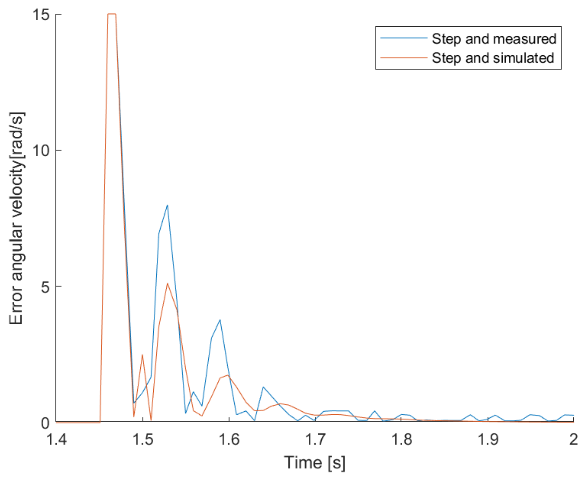


(a) Motor A

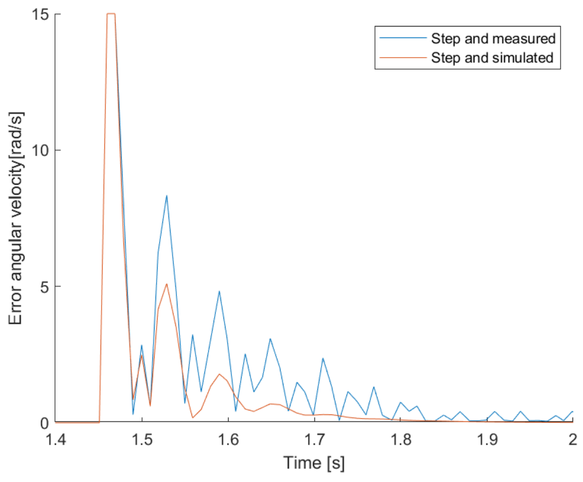


(b) Motor B

Figure 7: The step reference with its measured closed-loop response and its simulated closed-loop response



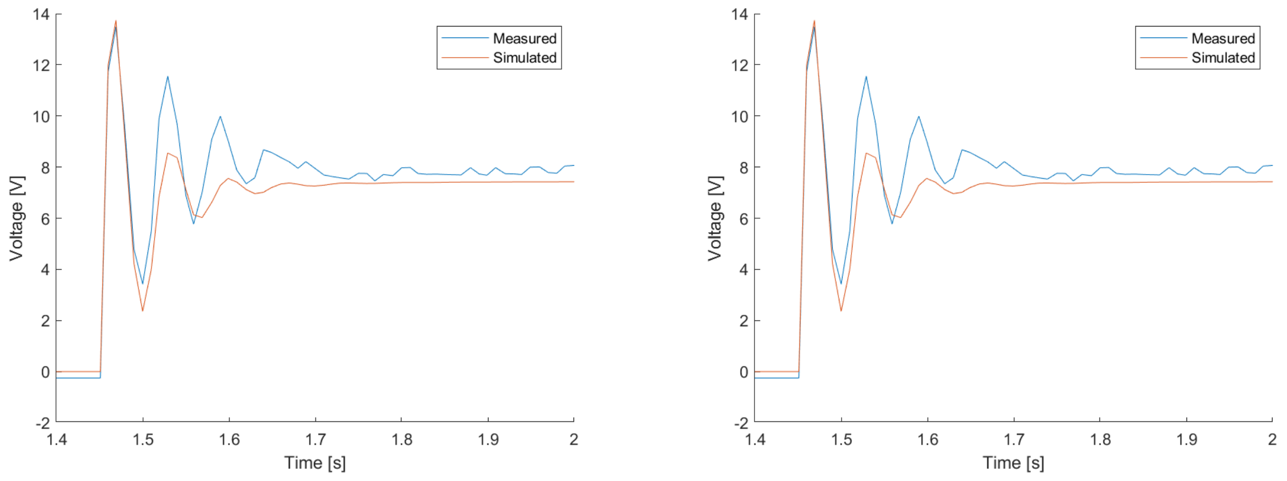
(a) Motor A



(b) Motor B

Figure 8: The measured tracking error of the step reference together with the simulated tracking error

In the figures, the same trends as earlier are noticeable. The simulated plots remain the exact same, as they do not incorporate the disturbance on the cart. In the measurements, a clear trend appears, where all peaks are systematically lower as when there was no disturbance. This is intuitive: The gravity pulling the cart down negatively influences the cart if actuated with the same control signals. It's very interesting to note that the measurements, due to the disturbance, now follow the simulation more clearly, while the simulation does not incorporate the disturbance. This probably has no physical reason, except for the fact that the disturbance acts as a 'braking term', which damps the oscillations, which were present in figure 3, a little bit, creating a resemblance with the simulated plot.



(a) Motor A (b) Motor B

Figure 9: The measured control signal (voltage) of the step reference together with the simulated control signal

Figure 9 shows the control voltage of the system with the force disturbance. Initially, the control signals look very similar, as the errors are similar too. Due to the fact that the velocity curve now resembles the measurements more, the control signals are very similar as well. The main important difference is that the steady state control signal is visibly higher than when there was no disturbance present. This makes sense intuitively: The error in speed in steady state, is generally larger, and biased towards being negative (as the force slows down the cart), this leads to an 'average' control signal which is larger.
Even though there is a constant disturbance, the controller still succeeds in decreasing the steady state error to zero (that is, lower than the general noise of the output).

Practically, it is very easy to imagine a 'disturbance' the controller cannot correct. Assume you're pulling on the cart, allowing no movement. This would evidently introduce a steady-state error, as the motors are simply not strong enough.
Theoretically, this is also possible. If the closed loop transfer function from the force to the output is constructed, this assumes a reference signal of zero. For the system to, theoretically, have zero steady state error, the output signal of the closed loop force velocity transfer function, $\frac{Y(z)}{F(z)}$, should be uniquely zero in steady state (thus following the reference of zero). If this response is simulated for a constant force, however, the steady state value is not zero. This is due to the fact that the block representing Newton's second law, also contains an integrator term, which 'counters' the integration action of the PI-controller. If the response is simulated without this block (physically, using a constant angular velocity disturbance instead of a force disturbance), the steady state response is exactly zero, as should be the case if zero steady state error is desired. Therefore, we concluded that, even a constant force, induces a steady state error. In the measurements, this error was simply not great enough to be distinguished in the noise.
In all cases, however, a ramp disturbance (force, or velocity) will always induce a steady state error.

## References

[1] Pipeleers and Swevers. *Control Theory - handouts*. Aug. 2020.