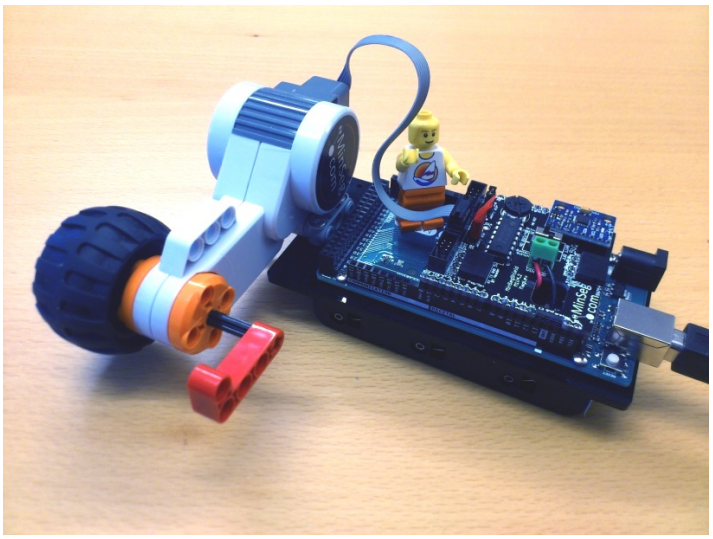
DC-motor PID control

This version: November 1, 2017



REGLERTEKNIK

AUTOMATIC CONTROL

LINKÖPING

Name:	
P-number:	
Date:	
Passed:	

Chapter 1

Introduction

The purpose of this lab is to give an introduction to PID control. We experiment with PID controllers to control the angle and angular velocity of a small arm attached to a DC-motor (see cover). This simple experiment mimics typical applications in practice, such as the robotic hand in the figure below.

Figure 1.1: Robotic hand using 15 DC-motors to position 5 fingers independently. High-precision control of joint angles is required for high-precision picking of objects, gentle grabbing, forming gestures etc.

1.1 Hardware set-up

The lab is based on three main hardware components.

To begin with, we have a standard desktop computer. This computer is used to automatically develop and deploy code using MATLAB and SIMULINK models.

To supply power to the DC-motor and perform measurements of motor angles, we use a board with an Arduino micro-controller which runs the auto-generated code. It also communicates with the desktop computer and thus allows us to look at the measurements.

The motor we experiment with is a simple DC-motor with a wheel and an arm attached. The motor is normally part of a LEGO Mindstorms kit.

The Arduino board together with the motor and attachments is called the MinSeg.

1.2 Troubleshooting

The wheels turn slowly and/or erratically Make sure the tires do not rub against the motor. You can pull the wheels apart as they slide on the wheel axis.

Complaints about COM port or connection when downloading to board Disconnect USB-cable and connect it again. Make sure cable is firmly attached on both ends. If it still does not work after several tries, save your model and restart MATLAB.

Complaints about OUT OF MEMORY when compiling code Save your model and restart MATLAB.

Chapter 2

Preparation

The questions below, and all questions throughout the document marked as **Preparation** must be completed by all students before attending the lab. Note that there are additional preparation exercises in Chapter 3.

Solutions to all questions should be available upon request from the lab assistant, and the preparation exercises in Chapter 3 are preferably written in this printed documented.

When the lab starts, it is assumed you have done all preparations, and have a clear idea of the tasks that will be performed during the lab.

Note, although there are 15 question below, many of them are variants of exactly the same question, repeated for different setups.

Finally, note that essentially all computations performed below relate to practical experiments that we will conduct during the lab, i.e., they all have physical meaning and it is important to see these connections during the lab.

Preparation 1 *Read Section 3.1-3.6 in the course book by Ljung & Glad.*

Preparation 2 (P-controlled angle) *Suppose we use a P-controller to control the angle* $y(t) = \theta(t)$ *to follow a reference angle* $r(t)$*, i.e.,* $u(t) = K_P(r(t) - y(t))$ *or equivalently* $U(s) = K_P(R(s) - Y(s))$ *where* $K_P > 0$*. The transfer function from requested voltage* $u(t)$ *to angle* $y(t)$ *is given by* $G(s) = \frac{1.35}{s(0.1s+1)}$*. Show that the closed-loop system from reference to angle is given by*

$$Y(s) = \frac{1.35K_P}{0.1s^2 + s + 1.35K_P} R(s) \tag{2.1}$$

Preparation 3 (P-controlled angle) *Suppose the motor is at the angle* $y(t) = 0$ *and we change the reference from* 0 *to* $r(t) = \pi/2$*. Assuming we are using the gain* $K_P = 3$*, which input voltage* $u(t)$ *will initially be requested by the controller when the step is performed?*

Preparation 4 (P-controlled angle) *Write the closed-loop transfer function in the standard form* $G(s) = \frac{\omega_0^2}{s^2+2\zeta\omega_0 s+\omega_0^2}$ *(i.e., identify the parameters* ζ *and* ω_0 *as functions of* K_P*) and explain based on this how speed and oscillatory behavior of the closed-loop system is related to* K_P*. Suitable theory can be found on page 37 in the course book.*

Preparation 5 (P-controlled angle) *For simple systems, overshoot and oscillations can be predicted to occur when poles become complex, and the larger the complex part is compared to the real part, the larger the overshoot and oscillations are. Show that the closed-loop poles are complex when* $K_P > \frac{25}{13.5}$.

Preparation 6 (P-controlled angle) *Assume a disturbance* $v(t)$ *is acting on the input to the system, i.e., the actual input to the system is given by* $u(t) + v(t)$ *where* $v(t)$ *is unknown, see Figure 2.1.*

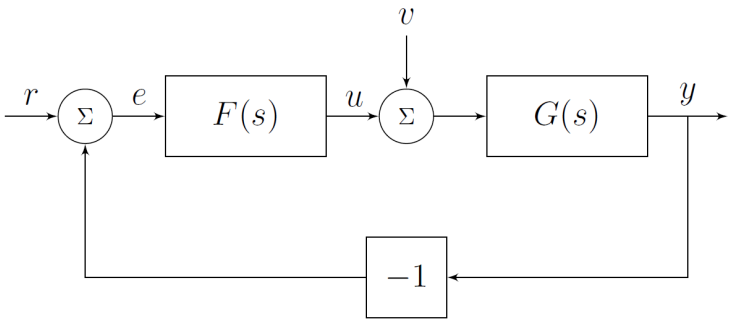


Figure 2.1: Disturbance acting on the input.

With $U(s) = F(s)(R(s) - Y(s))$*, show that control error* $e(t) = r(t) - y(t)$ *is given by*

$$E(s) = \frac{1}{1 + F(s)G(s)} R(s) - \frac{G(s)}{1 + F(s)G(s)} V(s) \tag{2.2}$$

Preparation 7 (P-controlled angle) *With* $\mathrm{G}(s) = \frac{1.35}{s(0.1s+1)}$ *and the P-controller* $\mathrm{F}(s) = \mathrm{K_P}$*, show that the error signal in the previous exercise evaluates to*

$$\mathrm{E}(s) = \frac{0.1s^2 + s}{0.1s^2 + s + 1.35\mathrm{K_P}}\mathrm{R}(s) - \frac{1.35}{0.1s^2 + s + 1.35\mathrm{K_P}}\mathrm{V}(s) \tag{2.3}$$

Preparation 8 (P-controlled angle) *Let the reference be a step with amplitude* A *(*$\mathrm{R}(s) = \frac{\mathrm{A}}{s}$*) and the input disturbance be a step with (unknown) amplitude* B *(*$\mathrm{V}(s) = \frac{\mathrm{B}}{s}$*) (i.e., we are modeling constant signals). Use the final value theorem to show that the steady-state value of the error,* $\lim_{t\to\infty} e(t)$*, converges to* $\frac{-\mathrm{B}}{\mathrm{K_P}}$ *when a P-controller* $\mathrm{F}(s) = \mathrm{K_P}$ *is used. Hence, as long as there is no input disturbance, we can track a constant reference angle without any steady-state error using a simple P-controller, but an input disturbance will cause a steady-state error. By increasing* $\mathrm{K_P}$ *this error can be decreased.*

Preparation 9 (PI-controlled angle) *Now use a PI-controller* $\mathrm{F}(s) = \mathrm{K_P} + \frac{\mathrm{K_I}}{s} = \frac{\mathrm{K_P}s + \mathrm{K_I}}{s}$*. Use the same setup as above with constant reference and disturbance, and show that the steady-state error converges to 0. In other words, integral action (*$\frac{1}{s}$*, pole in the origin) in the controller can help us to counteract a* constant *input disturbance.*

Preparation 10 (PD-controlled angle) *Now use a PD-controller to control the angle* $y(t)$*, i.e.,* $\mathrm{F}(s) = (\mathrm{K_P} + \mathrm{K_D}s)$*. Show that the closed-loop system from* $\mathrm{R}(s)$ *to* $\mathrm{Y}(s)$ *is given by (we skip the input disturbance now)*

$$\mathrm{Y}(s) = \frac{1.35(\mathrm{K_P} + \mathrm{K_D}s)}{0.1s^2 + (1.35\mathrm{K_D} + 1)s + 1.35\mathrm{K_P}}\mathrm{R}(s) \tag{2.4}$$

Preparation 11 (PD-controlled angle) *Show that all closed-loop poles are real if* $\mathrm{K_D} \geq \frac{\sqrt{54\mathrm{K_P}} - 10}{13.5}$*, i.e., the larger* $\mathrm{K_P}$ *we use, the larger* $\mathrm{K_D}$ *we need to keep all poles real.*

Preparation 12 (P-controlled angular velocity) *Now consider control of angular velocity,* $y(t) = \omega(t) = \dot{\theta}(t)$*. The transfer function from requested voltage* $u(t)$ *to angular velocity* $y(t)$ *is given by* $\mathrm{G}(s) = \frac{1.35}{(0.1s+1)}$ *(i.e., the only difference compared to the angle model is the removed integrator). Show that the closed-loop system from reference to angular velocity when a P-controller* $\mathrm{F}(s) = \mathrm{K_P}$ *is used is given by*

$$\mathrm{Y}(s) = \frac{1.35\mathrm{K_P}}{0.1s + 1 + 1.35\mathrm{K_P}}\mathrm{R}(s) \tag{2.5}$$

Preparation 13 (P-controlled angular velocity) *Show that the time-constant of the closed-loop system is* $\frac{1}{10+13.5\mathrm{K_P}}$*, i.e., increasing* $\mathrm{K_P}$ *decreases the time-constant and makes the closed-loop system faster.*

Preparation 14 (P-controlled angular velocity) *Show that the error signal with a reference and an input disturbance is given by*

$$\mathrm{E}(s) = \frac{0.1s+1}{0.1s+1+1.35\mathrm{K_P}}\mathrm{R}(s) - \frac{1.35}{0.1s+1+1.35\mathrm{K_P}}\mathrm{V}(s) \tag{2.6}$$

Preparation 15 (P-controlled angular velocity) *Let* $\mathrm{R}(s) = \frac{\mathrm{A}}{s}$ *and* $\mathrm{V}(s) = \frac{\mathrm{B}}{s}$*, and use the final value theorem to show that the steady-state value of the error converges to* $\frac{\mathrm{A}}{1+1.35\mathrm{K_P}} - \frac{1.35\mathrm{B}}{1+1.35\mathrm{K_P}}$ *when a P-controller* $\mathrm{F}(s) = \mathrm{K_P}$ *is used. In other words, constant non-zero angular velocity references cannot be followed without a steady-state error using a P-controller, even in the perfect case when there are no input disturbances.*

Preparation 16 (PI-controlled angular velocity) *Use a PI-controller* $\mathrm{F}(s) = \mathrm{K_P} + \frac{\mathrm{K_I}}{s}$*. Use the same setup as above with a reference signal and input disturbance, and show that the steady-state error converges to 0. In other words, integral action in the controller allows us to track a constant reference signal without steady-state error, and eliminates steady-state errors due to constant input disturbances.*

Preparation 17 (Summary of steady-state errors) *We considered a system with an integrator (pole in the origin) when we analyzed angle control, and a system without an integral in the angular velocity control case. We have also studied controllers without integrator (P, PD), and controllers with integrator (PI). In the table on the next page, mark the cases where theory predicts that we can track a constant reference perfectly, and eliminate a constant input disturbances. Since external signals (references, disturbances) act independently, you should consider the results independently, i.e. when you study reference tracking, simply assume input disturbance is 0, and when you consider input disturbances, you assume the reference is zero.*

Preparation 18 *Read the complete lab-pm. There are some theoretical questions in the pm which you are supposed to complete as preparation.*

Preparation 19 *Print this document. You must bring a physical copy to the lab.*

	F(s) (no integral)	F(s) (has integral)
G(s) (no integral)	☐ Reference tracked perfectly ☐ Input disturbance eliminated	☐ Reference tracked perfectly ☐ Input disturbance eliminated
G(s) (has integral)	☐ Reference tracked perfectly ☐ Input disturbance eliminated	☐ Reference tracked perfectly ☐ Input disturbance eliminated

Chapter 3

The lab

Items labeled **Preparation** are questions you are supposed to solve and fill out before attending the lab.

Items labeled **Task** are questions you solve when attending the lab and have access to the hardware.

In the lab, we will address two different scenarios ; control of angle, and control of angular velocity, and we will use various PID variants.

- P-control of angle
- PD-control of angle
- PID-control of angle
- P-control of angular velocity
- PI-control of angular velocity

The goal of the lab is to understand how the different parts of the PID controller influence closed-loop behavior and performance.

The design of the controllers will be completely model free (i.e., they are tuned by simply changing the gains without any computations). However, almost all phenomena seen (speed, steady-state errors, and oscillations) can be explained by the analysis you have done in the preparation exercises where a model of the DC-motor is used. Hence, it is important that you connect the dots between experimental results, and the predictions and analysis made in the preparation exercises.

3.1 DC-motor model

In the first lab, we derived and worked with a model describing the angular velocity $\omega(t) = \dot{\theta}(t)$ of the motor, when a voltage $u_A(t)$ was applied on the motor. Experiments typically found that $0.1\dot{\omega}(t) + \omega(t) = 1.9u_A(t)$ which corresponds to the transfer function $\frac{1.9}{0.1s+1}$ (look in your notes from the first lab!). In this lab, we will reuse this model, with some minor changes.

Figure 3.1: Our models now includes the motor driver (black chip) and describes the angle of the motor $y(t) = \theta(t)$ or angular velocity $y(t) = \omega(t) = \dot{\theta}(t)$, when sending voltage requests $u(t)$ to the motor driver.

To begin with, our input $u(t)$ will not be the voltage applied to the motor ($u_A(t)$, which we measured using a multimeter), but the voltage sent to motor driver chip. The reason is that the voltage sent to the motor driver is the only signal we can manipulate directly and thus use for control. When 4.5V is sent to the motor driver chip by the Arduino micro-controller, the resulting voltage on the motor is 3.2V. Hence, a reasonable model for the motor driver (black chip) is a simple gain $u_A(t) = (3.2/4.5)u(t)$.

Our initial focus will be the angle of the motor ($y(t) = \theta(t)$). Going from a model of the angular velocity $\omega(t)$ to angle is done by integrating the velocity $y(t) = \int \omega(\tau)d\tau$

Preparation 20 *Show that a model for the angle $y(t) = \theta(t)$ given the input $u(t)$ is given by*

$$Y(s) = G(s)U(s) = \frac{1.35}{s(0.1s+1)}U(s) \tag{3.1}$$

Note that this is the model you have used in the preparation exercises, i.e., your theoretical predictions have been derived using a slight extension of the model you developed in the first lab.

3.2 Control of angle

The first part of the lab will be devoted to angle control. We will start with a simple P-controller, and then try to improve this by adding derivative action (PD) and integral action (PI, PID). Controlling the angle of a motor is a common task in applications. Think for instance of a robot arm, or creating a gesture on the robotic hand in Figure 1.1. A gesture corresponds to a particular angle in every joint motor, and we want to move between different gestures in a smooth fashion.

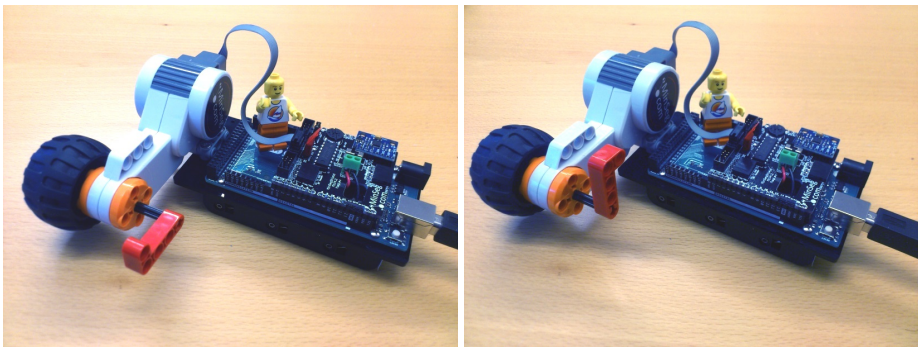


Figure 3.2: Miniature robot arm moved from initial angle $y(t) = 0°$ to final position $y(t) = 90°$. Performing a maneuver like this with specified speed and high precision is the task of the first part of the lab. Looks trivial, but it is surprisingly tricky.

Task 1 (Assembly) *Assemble the MinSeg as in Figure 3.2 and connect the USB cable, preferably in the monitor USB connection.*

A simple P-controller

Task 2 (Implement P-controller) *Open the model* ***minseg/pid/template1****. Implement the model in Figure 3.3. Note that text below blocks are arbitrary and can be changed.*

- ***Sum:*** *can be found in* ***Commonly used blocks****. Make sure you understand why there is a* $+$ *and a* $-$ *in the computation.*
- ***Slider gain:*** *This implements the controller gain* K_P*. It is found under* ***Math operations****.*
- ***Pulse generator:*** *This is the reference signal. It is found under* ***Sources****. Set the amplitude to* $\pi/2$*, period to 10 and pulse width* 50% *(i.e., create a reference signal* $r(t)$ *which switches between* 0 *and* $\pi/2$ *every 5 seconds). Note that the amplitude* $\pi/2$ *corresponds to the constant* A *in the preparation exercises.*
- ***Plot scope:*** *Found under* ***Sinks*** *(or copy the one in model).*

Remember to save the file after implementing the model!

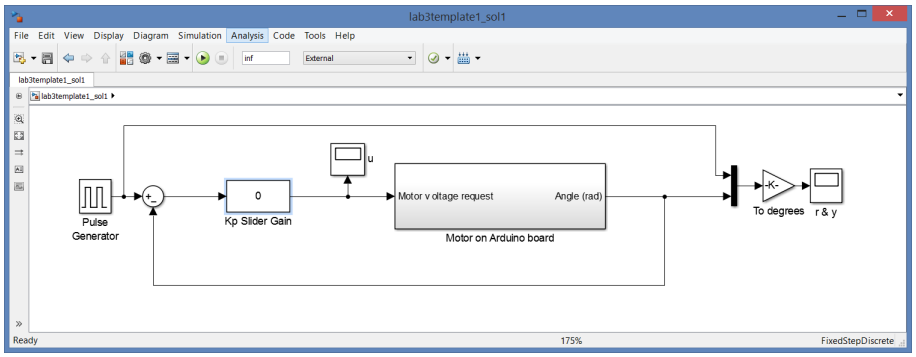


Figure 3.3: P-control of angle $\theta(t)$.

Task 3 (Experiment with proportional gain) *Download the controller to the Arduino with the green run button. What happens with the speed of the step-response when you increase* K_P*? (make sure you are looking at the correct plot, i.e., the plot of reference angle* $r(t)$ *and measured angle* $y(t)$*) You will have to increase the max value in the Slider gain as you investigate step-responses for increasingly larger* K_P *(try for instance* $1, 2, 4, 6, 8, 10$ *to really see the effects). Is the result consistent with* ***Preparation 4****?*

Task 4 (Overshoots and complex poles) *Find the value on* K_P *for which you clearly start seeing overshoots in the step response (value of* $y(t)$ *at some time larger than the value it finally converges to, or smaller when steps in the other direction are performed). How does this value compare to the theoretical prediction on complex poles (and thus possibly oscillatory response) for the closed-loop system in* ***Preparation 5****?*

Task 5 (Steady-state errors) *Let us study how the steady-state error changes when you alter* K_P*. How large are the steady-state errors for, e.g.,* $K_P = 1,5$ *and* 10*? Based on the observation coupled with results in* ***Preparation 8****, is it likely that there is no input disturbances?*

Task 6 (Tuning speed and steady-state behaviour) *Is it possible to tune the controller to achieve a very slow smooth movement while having a small steady-state error? Combine the theoretical predictions from* ***Preparation 4*** *and* ***Preparation8*** *and confirm in practice.*

Task 7 (Excessively large gain) *By increasing the gain, you see that the steady-state error becomes smaller, as theory predicts. However, what happens if you make* K_P *really large (say, 40)?*

Task 8 (Failure of theory) *From* ***Preparation 4*** *and page 37 in the course book we draw the conclusion that the closed-loop poles distance to the origin is proportional to* $\sqrt{K_P}$*. Hence, theory tells us the step-response should be significantly faster if we change* K_P *from 20 to 40. In practice this is not the case (check!, look for instance on the time it takes to go from 0 until it crosses the reference the first time). Can you figure out why it doesn't get any faster?* ***Hint:*** *Look at the plot of the requested control input* $u(t)$ *during the steps and note that the largest possible voltage available through an USB cable is 4.5V.*

Task 9 (Summary on proportional feedback) *Summarize the main effect* K_P *has on speed, steady-state error and overshoots & oscillations.*

Derivative action to reduce oscillations

When increasing K_P, overshoots and oscillations become problematic. To counteract this, we will add derivative action to the controller, i.e., let the control signal depend on how fast the error changes. If the error is deceasing rapidly ($\dot{e}(t)$ very negative), we should use less input voltage and perhaps even brake and apply voltage in the negative direction to avoid an overshoot. As we have seen in the preparation exercises, by using a sufficiently large K_D, complex poles can be avoided completely, thus reducing the likelihood of overshoots and oscillations.

Task 10 (Implement PD-controller) *Extend your model to include derivative feedback as in Figure 3.4 (the **Discrete Derivative** block is found under **Discrete**). As in the first lab, we do not have any direct measurement of the derivative of the angle $y(t)$, and we thus use an Euler approximation*[1] *to compute the derivative of the control error $e(t) = r(t) - y(t)$.*

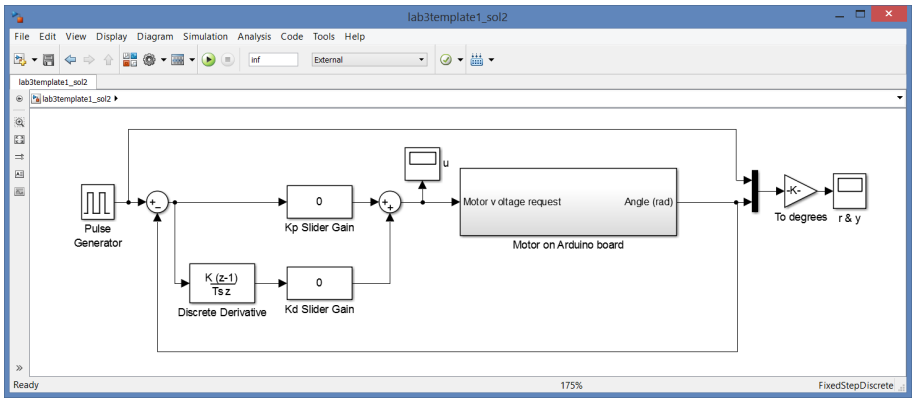


Figure 3.4: PD-controller for angle with numerical computation of the control error derivative.

Task 11 (Derivative action to reduce overshoots) *Start by using no derivative action, and use K_P such that it is fast but overshoots significantly (typically around 5). Now start adding derivative gain carefully. How large must K_D be to eliminate any overshoot? How does your tuning compare to the predictions on K_P vs K_D for obtaining real poles in **Preparation 11**?*

[1] With sample-time T_S, we have $\dot{e}(t) \approx \frac{e(t)-e(t-T_S)}{T_S}$. Our controller is setup to use $T_S = 0.04$ seconds.

Task 12 (Excessively large derivative gain) *What happens if you use a really large value on* K_D*? What could the reason be?* ***Hint:*** *we cannot measure the derivative* $\dot{\theta}(t)$ *but estimate it from noisy measurements of* $\theta(t)$.

Task 13 *Try to make the system as fast as possible, with no movements remaining after 1 second and less than* 5° *overshoot (i.e., implement a controller you would be proud to sell!). Remember to look at the actual movements and not just the plots. Which tuning do you arrive at?*

Where does the steady-state error come from?

Based on the preparation exercises, we know that there should be no steady-state error, unless there is an input disturbance (or our models and assumptions are completely wrong!). There is no direct input disturbance acting on the system as far as we know, but there is an unmodeled effect that can be interpreted as a disturbance. The motor driver chip (see Figure 3.1) has an electrical dead-zone which causes it to output 0V for any

requested voltage $|u(t)| \leq 0.7$V (the exact size on the dead-zone varies between different individual chips), see Figure 3.5 for an illustration. By seeing the difference between the applied voltage with dead-zone (solid line) and our model $\frac{3.2}{4.5}u(t)$ (dashed line) as an input disturbance, we can try to compensate for it using integral action in the controller. According to the theoretical result in **Preparation 9**, this might eliminate the problem.

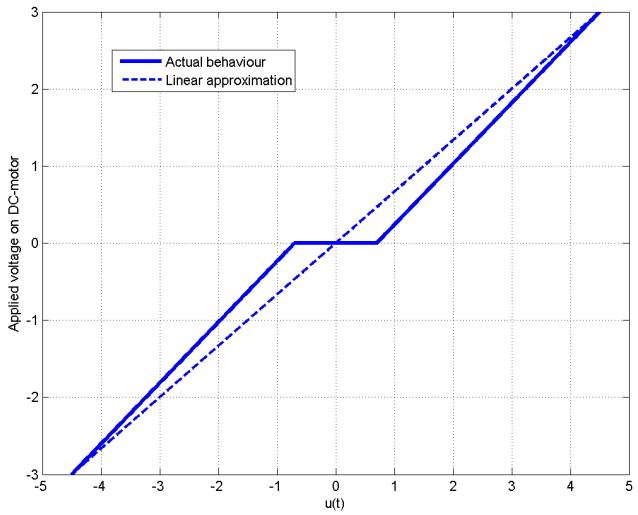


Figure 3.5: The motor driver chip has a dead-zone. When the requested voltage $|u(t)|$ is small (here $|u(t)| \leq 0.7$V), no voltage is applied on the motor. The difference between the actual voltage (solid line) and our linear approximation $\frac{3.2}{4.5}u(t)$ (dashed) can be thought of as an input disturbance $v(t)$. It is not constant as it depends on $u(t)$, but thinking of the effect of the dead-zone as an input disturbance is a good start.

Intuitively, what happens with a P-controller is that when the error $e(t)$ is small enough, the input $K_P e(t)$ will become so small that it is clipped in the dead-zone, effectively turning of the DC-motor, and it will slow down due to friction etc and stop at some point where typically $e(t) \neq 0$. By increasing K_P, the region where it is turned off will become smaller, but there will still be some region where the dead-zone causes it to be turned off.

Integral action to (try to) eliminate steady-state error

We will now try (and most likely fail!) to eliminate the steady-state error using integral action. In theory according to **Preparation 9**, if the input disturbance is constant, a PI controller should be able to eliminate any steady-state error.

Task 14 (Implement PID-controller) *Extend your* SIMULINK *model to include integral action as in Figure 3.6.*

- *The* ***Discrete-time integrator***[2] *block is found under* ***Discrete***
- *In the* ***Discrete-time integrator*** *block, set the sample time to* -1 *(this means it uses our sample-time* $\mathrm{T_S} = 0.04s$*).*
- *Change the period time in the pulse generator to 30 seconds (this will allow us to study the steady-state error for longer times before it switches reference value).*

Note that we have switched the location of the gain and the integral. With this order, the integral output will stay constant if $\mathrm{K_I} = 0$*. If we keep integrating while* $\mathrm{K_I}$ *is zero, and then change* $\mathrm{K_I}$ *to a non-zero value, the integral might have a huge value, leading to huge inputs.*

Task 15 (Integral action and oscillations) *Start with, e.g.,* $\mathrm{K_P} = 2, \mathrm{K_I} = 5$ *and* $\mathrm{K_D} = 0$*. Try reducing and increasing* $\mathrm{K_I}$*. What happens and what does this tell us about how* $\mathrm{K_I}$ *influence pole locations in the closed-loop system? Try adding derivative action* $\mathrm{K_D}$ *to see if oscillations introduced by the integral part can be reduced.*

[2] The integral is approximated using a rectangular approximation $\int_0^t e(\tau)d\tau \approx \mathrm{T_S}e(0) + \mathrm{T_S}e(\mathrm{T_S}) + \mathrm{T_S}e(2\mathrm{T}_s) + \ldots + \mathrm{T}_s e(t)$

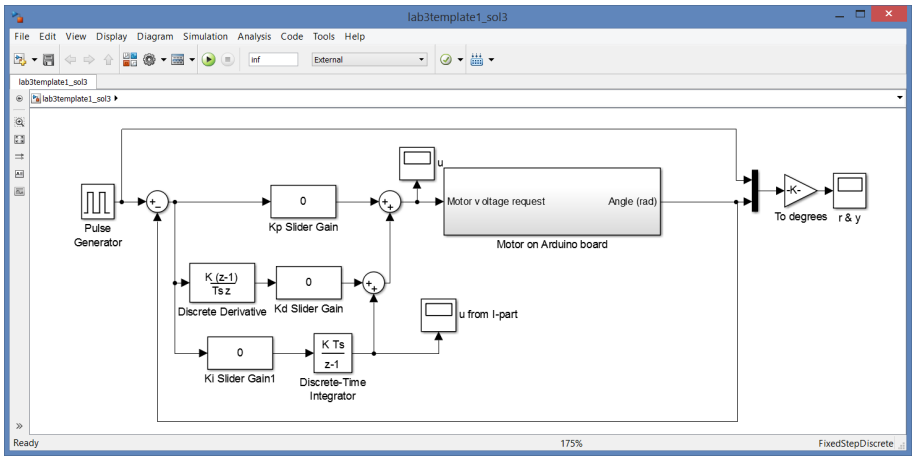


Figure 3.6: PID controller for angle control. Note that the order of gain and integral has been switched. By performing the computations in this order we ensure the integral stops integrating if we turn off the integral action with $K_I = 0$.

Task 16 (Steady-state error) *Can the steady-state error be eliminated using* K_I*? While running, look at the plot of the integral term added to the control input, and see how it keeps increasing and decreasing when the steady-state error is constant and non-zero. The integrator tries to find the value needed to compensate for the dead-zone, but cannot find any such suitable constant value (as it depends on the changing input).*

Although integral action probably fails to solve the problem completely here (due to the complicated dead-zone behavior, the input disturbance is not constant), it will be a very useful tool later when we control the angular velocity. **Always try integral action when you have steady-state errors**. If this does not work, you might have to use more tailor-made and advanced solutions, as we will do now.

Feedforward compensation of dead-zone

Since we know there is a dead-zone in the system, we can try to compensate for this directly, instead of trying to fix the problem using integral action (which didn't work out well). As roughly 0.7V is lost, we can simply add 0.7V to our control signal (or subtract 0.7V if the input is negative).

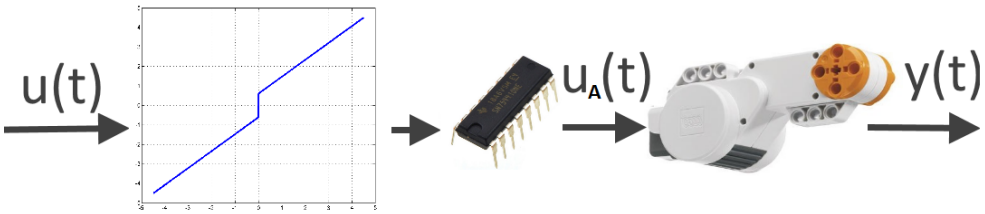


Figure 3.7: The computed input $u(t)$ is shifted to compensate for the known voltage loss in the motor driver chip dead-zone.

Task 17 (Dead-zone elimination) *Start the controller and turn of integral action by setting* K_I *to zero. Use* K_P *and* K_D *such that you have nice looking steps but with significant steady-state error. Double-click the block* ***Motor on Arduino board****, and open the block* ***Voltage request****. The value PWM offset is the dead-zone compensation value. A value of 255 corresponds to* $4.5\,V$ *compensation, hence the value 20 corresponds to* $0.35\,V$ *(*$(20/255)\cdot 4.5 = 0.35$*). Try to find a value (it can be changed while the controller is running) which leads to good behavior. A too small value will not eliminate the dead-zone completely leading to a remaining steady-state error, while a too large value will lead to a nervous behavior as the input to the DC-motor never is close to zero.*

At this point, you should have a PD-controller which essentially eliminates any steady-state error (as theory predicts when there is no input disturbance), while being fast with little overshoot and oscillations. We can thus smoothly control the exact position of the fingers in our robotic hand!

3.3 Control of angular velocity

Our task now is to have the motor rotate at a specified angular velocity. A typical application would be a cruise-controller in a car (to obtain a desired speed, the wheels must rotate with a particular angular velocity) or a robot arm which is programmed to move at a particular speed when performing a task such as welding or painting.

Task 18 (Implement PI-controller) *Open the model* ***template2*** *and implement a PI-controller of the angular velocity as illustrated in Figure 3.8. Note that the output from the Arduino board block is the angular velocity now (computed by Euler approximations from angle measurements)*

- ***Pulse generator:*** *Set the amplitude to* $250\pi/180$ *(i.e.,* $250^\circ/s$*), period to 20 and pulse width* 75% *(i.e., switch reference betwen* 0 *and* $250^\circ/s$*)*
- ***Discrete-time integrator:*** *Remember to set sample time to* -1 *(inherited sample time).*

Remember to save the file after implementing the model!

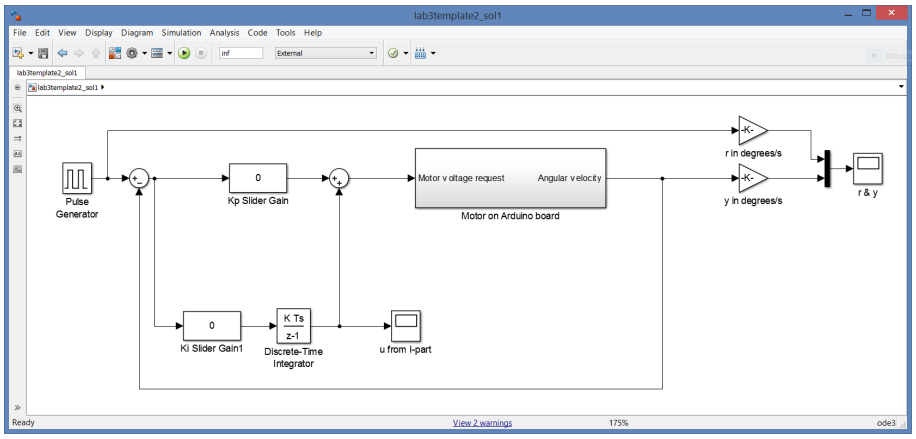


Figure 3.8: PI-controller for angular velocity

Task 19 (P-control of angular velocity) *Start with a simple P-controller (i.e., set* K_I *to 0) and experiment with* K_P *to see what happens. Note that the angular velocity is computed from the angle using numerical differentiation*

and thus suffers from amplification of measurement noise, as we have seen before. Hence, the signal we study now will have a significant level of noise which we cannot eliminate through control and we have to look at the average level of the steady-state error. Can you get rid of the steady-state error? Is the result consistent with ***Preparation 15****?*

Preparation 21 *The model for angular velocity is* $0.1\dot{\omega}(t) + \omega(t) = 1.35u(t)$. *If we want* $y(t) = \omega(t)$ *to be constant at* $4.36\ rad/s$ *(i.e.* $250°/s$*), which constant input voltage* $u(t)$ *is required?*

Preparation 22 *A P-controller computes* $u(t) = K_P(r(t) - y(t))$. *Based on simple logic, can a P-controller lead to a steady-state error* $e(t) = 0$ *, if you look at the result in the previous preparation exercise?* ***Hint:*** *Which input voltage do you have from a P-controller if you have zero steady-state error?*

Task 20 (Steady-state error with P-control) *Now add a dead-zone compensation using the same PWM-offset as you used in the angle controller to remove the input disturbance. Is the steady-state error eliminated for arbitrary choices of* K_P *as it was in the angle controller? Relate to your theoretical result in* ***Preparation 15, 21*** *and* ***22*** *on steady-state errors in angular velocity control using a P-controller.*

Task 21 (Steady-state error with PI-control) *Add integral action by increasing* K_I*. Can you eliminate the steady-state error? Relate to your theoretical result in* ***Preparation 16*** *on the steady-state error in angular velocity control using a PI-controller. Remove the PWM-offset to see that the I-part of the controller alone can fix the problem, taking care of both the steady-state error caused by a non-zero reference, and compensating for the input disturbance.*

Task 22 (What does the I-part learn?) *Look at the plot of the I-part of the input to see how the I-part learns the necessary value needed to keep the motor running at a specific angular velocity. Relate to* ***Preparation 21****.*

The reason we can compensate for the dead-zone using integral action here is because the effect from it is fairly constant when we are driving at a constant angular velocity, and the integral term simply has to learn to add another 0.7 volts to the input to reach the required level. When we control the position, the input should be zero in steady state, and for small control errors, the required compensation changes drastically from -0.7 volts to 0.7 volts when the sign switches. This is very hard for the integral to handle.

3.4 Summary and reflections

Summarize and reflect on what you have seen and learned in this lab.

Questions	Answers
1. Increasing K_P typically leads to	☐ A faster response ☐ A slower response
2. Increased K_P typically leads to	☐ Reduced steady-state error ☐ More oscillations and overshoots ☐ Less oscillations and overshoots ☐ Large control signals
3. Derivative action K_D is introduced to	☐ Reduce the steady-state error ☐ Speed up the system ☐ Amplify measurement noise ☐ Reduce oscillations and overshoots
4. A typical drawback of derivative action is	☐ Oscillations and instability ☐ Amplification of measurement noise
5. Integral action K_I is introduced to	☐ Reduce oscillations ☐ Speed up the system ☐ Reduce the steady-state error
6. The main drawback of integral action is that it might	☐ Introduce oscillations and instability ☐ Lead to smaller steady-state errors ☐ Amplify measurement noise ☐ Slow down the system
7. Whether we will have steady-state errors depends	☐ only on the open-loop system ☐ only on the controller ☐ on both system and controller
8. A model of the system is	☐ required to develop a PID controller ☐ not required to develop a PID controller

Most unclear to me is still: .....