Institute for Computer Science XVII
Robotics

Bachelor's thesis

Different Virtual Pose Instruction Plane control strategies for uniform linear motion on Telescopic Linear Driven Rotation Robots

Martin Cosmas Hesse

March 2023

First reviewer: Prof. Dr. Andreas Nüchter
Advisor: M.Sc. Jasper Zevering

Zusammenfassung

Es besteht ein großer Bedarf an der Erforschung des Mondes und anderer Planeten, auf denen Roboter mit widrigen Bedingungen konfrontiert sind, z.B. mit unwegsamem und unebenem Gelände, was eine große Belastung für das Fortbewegungssystem darstellt. Ein kugelförmiger TLDR-Roboter, der eine VPIP zur Steuerung verwendet, ist ideal geeignet, um diese rauen Bedingungen zu überstehen. In dieser Arbeit werden mehrere Steuerungen und Regelungen für die VPIP untersucht, um eine gleichmäßige lineare Bewegung zu erzeugen. Dazu werden eine Steuerung, mehrere PID-Regler und ein LQR implementiert. Alle diese Strategien werden simuliert. Die Steuerung zur Charakterisierung des Systems, die Regler zur Charakterisierung ihrer Reaktion auf eine Änderung der gewünschten Rotationsgeschwindigkeit des Roboters und ihrer Fähigkeit, diese auf verschiedenen Werten konstant zu halten. Erfolgreiche Tests mit einem PI-Regler validieren diesen Regelansatz und die Verwendung einer VPIP zur Steuerung eines TLDR-Roboters.

Abstract

There is a large need to explore and study the moon and other planets, where robots face adversary conditions, a.o. rough and uneven terrain, which puts a large strain on the locomotion system. A spherical TLDR robot using a VPIP for control is ideally suited to weather these harsh conditions. This thesis examines multiple control strategies for the VPIP to create uniform linear motion. This is done by implementing an open-loop control strategy, as well as multiple PID controllers and a LQR. All of these are simulated. The open-loop controller to characterize the system, the closed-loop controllers to characterize their response to a change in the desired rotational speed of the robot and their ability to hold it at various constant values. Successful tests using a PI controller validate this control approach and using a VPIP to control a TLDR robot.

Contents

List of Figures

List of Tables

Acronyms

DAEDALUS	Descent And Exploration in Deep Autonomy of Lava Underground Structures.
ESA	European Space Agency.
LiDAR	Light Detection and Ranging.
LQR	Linear-Quadratic Regulator.
OSIP	Open Space Innovation Platform.
PID	Proportional-Integral-Derivative.
ROS	Robot Operating System.
TLDR	Telescopic Linear Driven Rotation.
VPIM	Virtual Pose Instruction Map.
VPIP	Virtual Pose Instruction Plane.

Chapter 1

Introduction

Lunar lava caves, also called tubes, are structures found underneath the lunar surface. They typically have skylights, an opening to the surface, which grants access to the underground tube system. In comparison to the lunar surface relatively little is known about these underground tubes, which makes them a great place to further study lunar geology[7]. Similar lava tubes are also found on earth, where they form by fluid basalt flows, the same is believed to happen on the moon. The lunar lava tubes in the Marius Hills region share very similar geomorphic aspects to the ones on earth, excluding size, which is explained by the smaller lunar gravity[8]. This makes them a great place to start studying these structures and comparing them to earth. Besides the potential scientific benefit of studying these structures, they can also be used to create an advanced underground lunar base. Though further research of the tubes is necessary to use them for a permanent lunar outpost[3]. In 2019 the European Space Agency (ESA) became aware of this potential for exploration of lunar caves. Therefore they put out a Call for Ideas as part of there Open Space Innovation Platform (OSIP) program [6] to detect, map and explore lunar caves [7]. One of the ideas, that were further studied, as part of the SysNova initiative [5], was the Descent And Exploration in Deep Autonomy of Lava Underground Structures (DAEDALUS) project [4, 10]. The DAEDALUS project put forward a mission concept to examine the entrance and initial parts of an underground lunar lava tube. The mission uses a spherical robot with various sensors, such as cameras and Light Detection and Ranging (LiDAR) scanners. A crane lowers the robot down, through a skylight into the cave system. The spherical shape of the robot is ideal for the decent, during which it can freely spin. In doing so, the robot is able to scan its surroundings with its LiDAR scanners, to create a 3D map of the entrance. Once on the ground of the entrance the robot is untethered and able move around in the cave. The spherical shape is also ideal for this task, as it protects the sensors within a shell, and can easily traverse the rough and uneven terrain of lunar caves. For locomotion within the cave the robot uses a pendulum. By shifting the mass of the pendulum forward, the center of gravity shifts forward and therefore the sphere begins to roll forward. Using the same logic the sphere can move in all directions, including side to side. Even though this pendulum driven locomotion approach works well on relatively easy terrain, there are many obstacles, that can't be traversed. Especially those with a certain height, for which to pass the robot has to lift itself up. Based on the DAEDALUS mission concept a Telescopic Linear Driven Rotation (TLDR) robot was

developed. This robot uses a novel locomotion approach, but keeps the spherical shape [12]. Instead of a pendulum the TLDR robot uses telescoping rods, to push itself forward and away from obstacles. In doing so it is better suited for the uneven terrain, as it is able cross more obstacles, by e.g. pushing itself upwards. This is described in detail in Chapter 2.1. Due to the new idea of multiple rods pushing the robot, it is not obvious how far to extend the individual rods. There are different strategies on how solve this problem. One possibility to solve it, is the Virtual Pose Instruction Plane (VPIP), which calculates the desired length of each rod based on a geometric representation. This is further discussed in Chapter 2.2. This thesis will design and examine multiple control strategies for the VPIP to create a linear forward motion of a spherical TLDR robot.

Chapter 2

Background

In this chapter we take a closer look at the TLDR robot, how to get it moving, get it balanced, and how to combine these using the VPIP.

2.1 Telescopic Linear Driven Rotation Robot

The TLDR robot consists of a inner part including all relevant systems, including the rods, and an outer spherical shell, that is attached to the inner part and has holes at the relevant spots to let the rods extend to the outside of the robot shell. Figure 2.1 shows a rendering of a TLDR robot, that is used in the DAEDALUS mission. Besides the shell being the support point for the robot on the ground and protecting the inside from damage, it has no practical function. On either side of the center there are discs, offset to the sides, on which the rods are mounted. In the center, between the two discs, there should be a mechanical connection, otherwise the outer shell would have to hold the discs in place. The exact type and form of the center connection is irrelevant and can be designed to fit the needs of any payload.

2.1.1 Movement

We will first only look at a simple movement, where the rotation of the robot is enough to move it forward. Special cases, like those, where the entire robot has to push itself upwards, to pass an obstacle, will first be ignored. We can also treat the rods on opposite sides exactly the same, as we are currently only interested in forward / backward motion and not in curves. Also we are only interested in movement due to rotation and not the robot falling over, jumping or pushing itself away from somewhere. This leaves us with two options to initiate the rotation of the robot. The first is pushing the rods into the ground and thus creating torque. If the robot is supposed to move in one direction (from now on forward), the rods on the back have to extend, to push into the ground. This is illustrated in Figure 2.2. Obviously the angle, of the rods, that extend, between the bottom of the sphere an the rod (from now on ζ) has to be greater than 0 rad. If it were 0 rad it only pushes the robot upward and doesn't create any rotational speed. Similarly ζ has to be less than 0.5π rad, otherwise the extending rod never touches the ground, or if ζ is greater than 1.5π rad, the rod pushes against the desired rotation. But as the rods have a maximum extended length, there is also a maximum $\zeta < 0.5\pi$ rad, until which the rods extend.

Figure 2.1: Visualization of a TLDR robot. [12]

This is why in the last state in Figure 2.2 the rod, that was the first to extend, is retracted, even though its ζ is smaller than 0.5π rad. For any other values of ζ (including between 0.5π rad and 2π rad) the rods contract.
The second option to initiate a rotation of the robot, is by leverage, shown in Figure 2.3. For this we will extend the rods if ζ is greater than π rad. Therefore their mass moves forward and outward, so they create a torque due to their increased leverage. This leads to an increase in forward rotational speed. As soon as ζ is greater than 1.5π rad the rods contract, to prevent hitting the ground and therefore inadvertently reducing the rotational speed. This is the reason, why the rod, that was extended in the first state in Figure 2.3 isn't anymore in the second state. This covers most of the movement needs, although there some scenarios, for which this simple kind of movement isn't enough. These are either obstacles to tall to just roll up, or slopes and

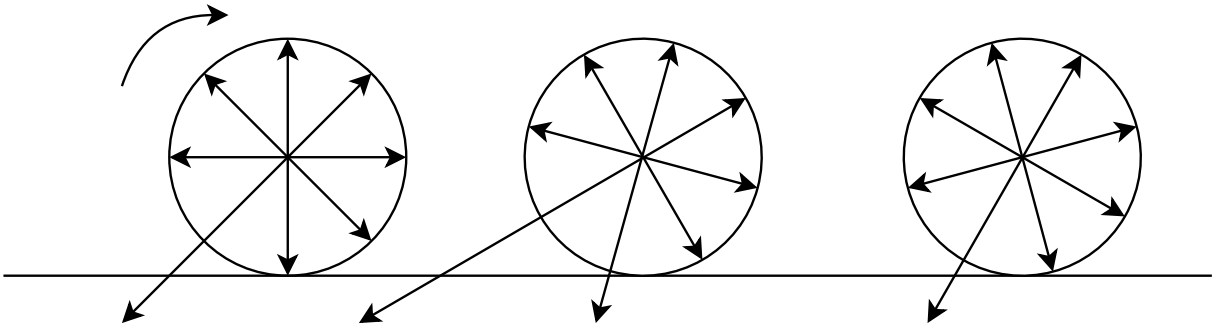


Figure 2.2: A sketch of a TLDR robot extending rods at the back pushing into the ground, thus initiating a rotation and movement to the front.

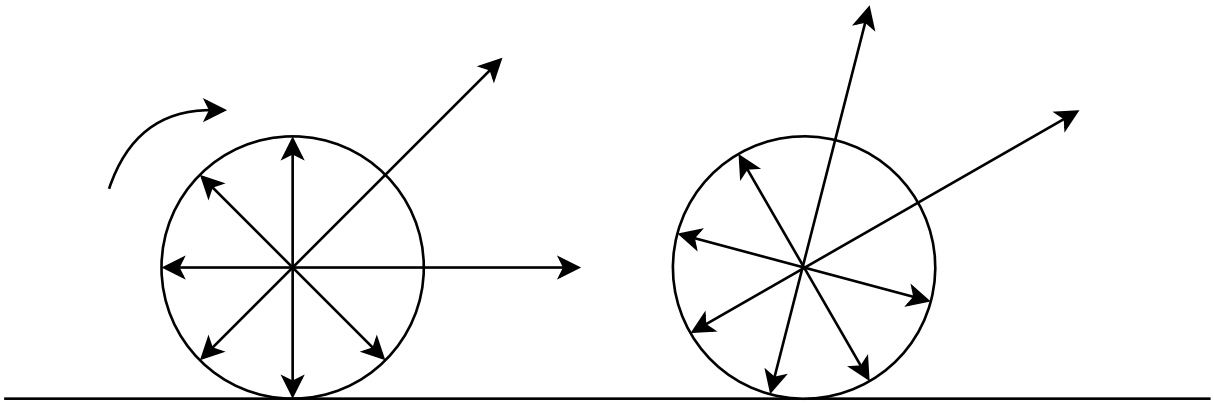


Figure 2.3: A sketch of the TLDR robot extending rods in the front, thus initiating rotation and movement to the front.

ridges so steep they need active braking. In case of downward slopes active braking is needed. This is achieved by initiating a backward rotation. As long as this is actively controlled, this will manage the increase of rotational speed due to gravity and not let the robot roll back uphill. More difficult to control are ridges, both up and down. In the case of an upward ridge i.e. a larger rock or plate, the robot has to lift itself upward. For this the robot rolls against the ridge (Figure 2.4, Phase 1) and a rod, or multiple, with ζ greater than 0 rad, but small enough to still easily reach the ground, extend into the ground. This pushes the robot up the ridge, leaning against it the entire time, another reason a spherical robot is an ideal choice (Figure 2.4, Phase 2). For any ζ smaller than 0 rad, the corresponding rod pushes the robot up, but away from the ridge, which leads to it falling over. Therefore only rods with ζ greater than 0 rad are used. Similarly we can lower the robot, by contracting one rod, or multiple, while the robot leans against the ridge. Before rolling over the edge, any rod, which has a ζ smaller than 0 rad, when the robot is completely over the edge, but still large enough so it reaches the ground, extends to support the robot (Figure 2.5, Phase 1). Then the robot rolls over the edge, but not so far, that it falls over the supporting rods. Then the supporting rods get contracted, to lower the robot down (Figure 2.5, Phase 2).

2.1.2 Balancing

Other than in the case of a pendulum driven robot, as planed for the DAEDALUS mission (Chapter 1), the TLDR robot can not move side to side. To create a curved motion, the robot therefore has to lean into the curve. The same is also true for balancing the robot just in reverse. Therefore these problems can be viewed as the same. To balance the robot evenly, the rods opposite of each other, have to extend to different lengths. This only effects the rods, that are currently pushing into the ground. The ones on the lower side have to extend further, and the ones on the higher side have to extend less. This won't affect the forward / backward movement of the robot, but will just balance it. To drive in a curve, instead of perfectly balancing the robot, it has to tilt in the desired direction. Therefore the length of opposite rods have to adjust accordingly.

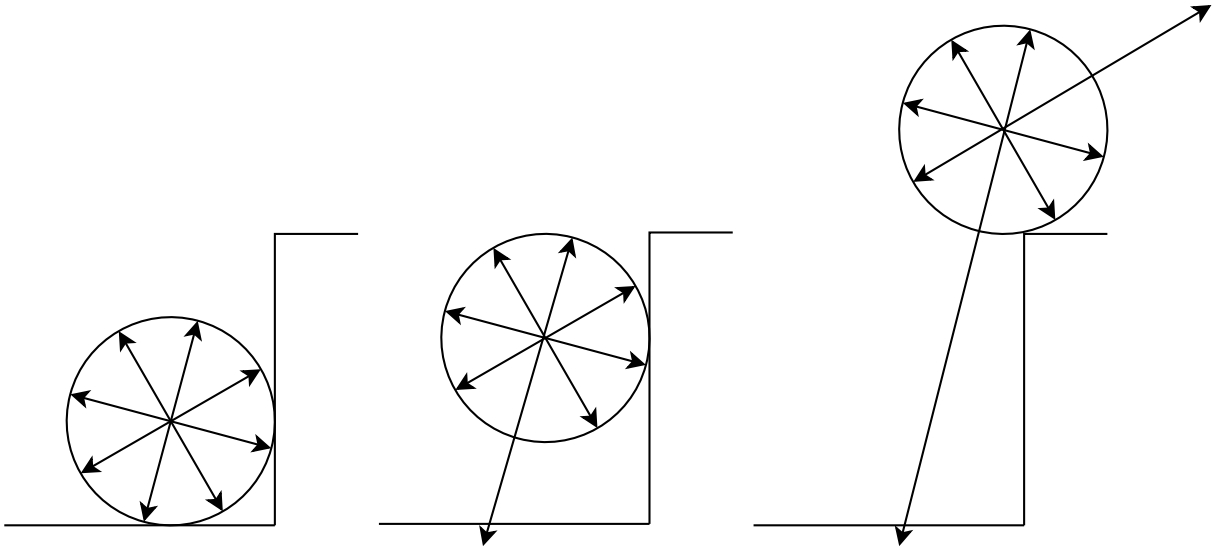


Figure 2.4: A TLDR robot pushing itself up a ridge, and using both pushing and leverage to keep moving at the top of the ridge.

Even though we now have a general strategy for how to balance and move the robot, we still don't know how far exactly the individual rods have to extend. Hard coding necessary extensions for different ζ for all the different movement possibilities and balancing cases is not only very complicated, but also leaves no opportunities to properly control the robot. One option, to combine movement and balancing and make the system controllable, is the VPIP.

2.2 Virtual Pose Instruction Plane

Now we will take a look at the VPIP, which combines the movement and balancing of the robot. Here we will only look at a flat ground, as the VPIP doesn't provide a solution for the special movement cases, but only the ones, where pushing into the ground or extending rods for leverage is enough to initiate rotation.
Let the y-axis be the direction in which the robot is moving forwards, the z-axis perpendicular to the ground going through support point of the robot, and the x-axis completes a right handed coordinate system. The virtual plane, that describes the VPIP, is fixed at the support point of the robot on the ground. Now we describe the plane only using two angles: θ_{VPIP}, the pitch measured around the x-axis, and ϕ_{VPIP}, the roll measured around the y-axis. This is illustrated in Figure 2.6. Each of the rods now extends to the exact point, where they touch the VPIP, as shown in Figure 2.7. In Figure 2.6 we have a positive θ_{VPIP} and ϕ_{VPIP}. Therefore the rods at the back right side of the robot extend through the ground and the rods at the front left side retract to be above the ground. Thus the robot starts rolling to the front left. The exact length of the rods on the front right and back left depend on the the exact angles, that are chosen for θ_{VPIP} and ϕ_{VPIP}, and no general statement about these rods is possible. Using the VPIP

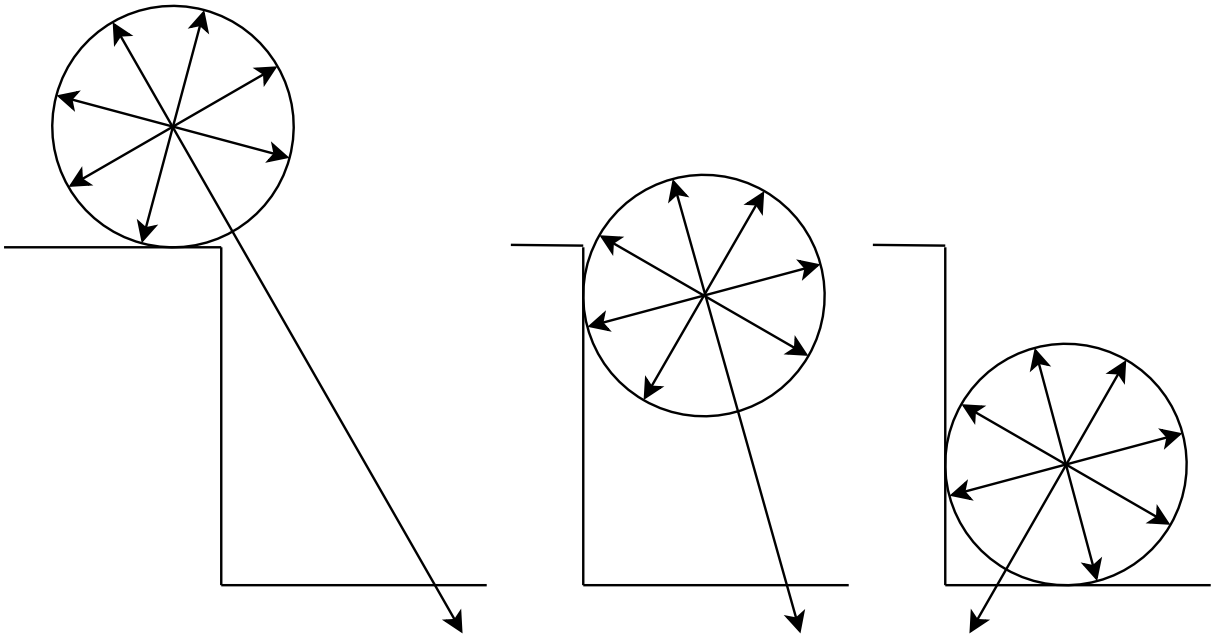


Figure 2.5: A TLDR robot lowering itself down a ridge.

we have created a controllable system. We can control the rotational speed of the robot just by changing θ_{VPIP}. If we desire a greater rotational speed, θ_{VPIP} is increased, so that rods in front of the robot extend, without touching the ground, in doing so creating torque by leverage without braking. The rods in the back extend further, thus creating torque by pushing into the ground. Therefore we are able to control the rotational speed of the robot, by changing θ_{VPIP}. Similarly we can control the balancing, by changing ϕ_{VPIP}. As soon as the robot tilts to the right, the VPIP also tilts to the right, thus the rods on the right extend further forcing the robot back upright. In an ideal case a θ_{VPIP} of 0^circ has no influence on the rotational speed of the robot at all. All the rods extend to the exact position where they touch the ground, but not any further. Thus no torque is created, only stabilizing the robot. Any existing rotational speed is conserved.

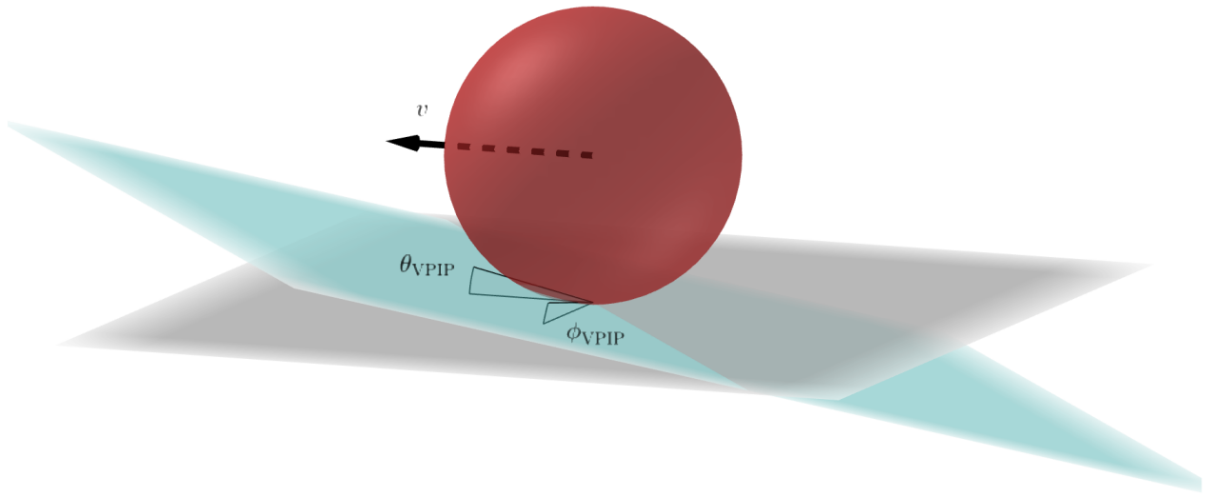


Figure 2.6: The VPIP (blue) is tilted by θ_{VPIP} and ϕ_{VPIP}, away from the ground (grey), thus moving the robot (red) into the direction of v. [12]

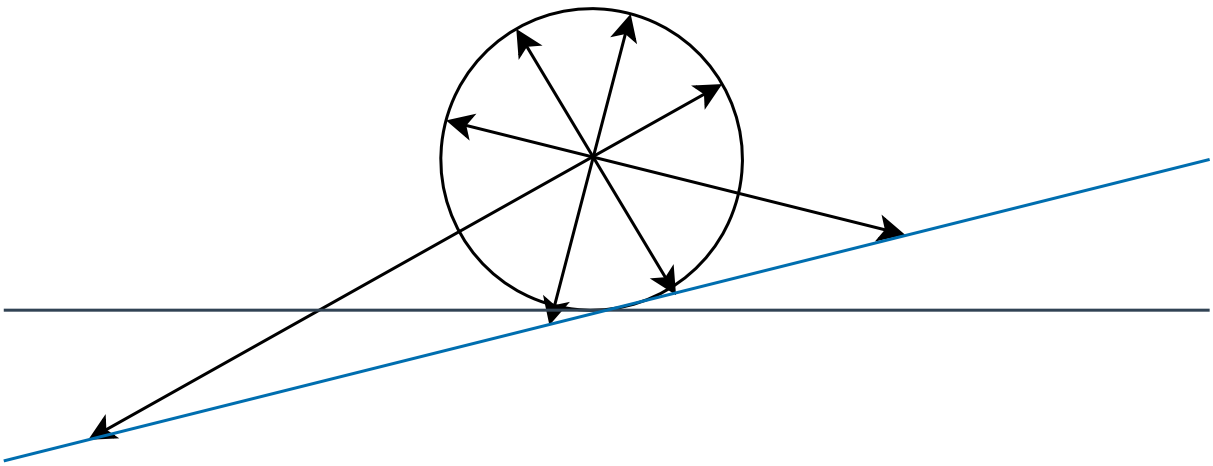


Figure 2.7: The rods of the robot extend to the point, where they touch the VPIP (blue).

Chapter 3

Theoretical Background

3.1 Virtual Pose Instruction Plane

Here we take a closer look at the mathematics behind the VPIP, which leads to a formula, with which we calculate the necessary length for each of the rods. Setting the support point of the robot to $(0, 0, 0)$ we derive

$$\Pi_{VPIP} = \{(x, y, z) \mid \tan(\phi_{VPIP}) \cdot x + \tan(\theta_{VPIP}) \cdot y + z = 0\}, \tag{3.1}$$

which describes the the VPIP only using θ_{VPIP} and ϕ_{VPIP}. To calculate the length for each of the rods, we need a mathematical representation for each rod, illustrated in Figure 3.1. We define the point $\mathbf{p}_m$ at the center of the side disc, and $\mathbf{p}_t$ as the tip of the fully contracted rod, which we assume to be at the edge of the side disc. With ϕ_r the roll angle of the sphere, d_m^s the distance between the center of the sphere and the side disc, r_m the radius of the sphere and r_s the radius of the side disc, we derive $\mathbf{p}_m$ to be

$$\mathbf{p}_m = \begin{bmatrix} \cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - \sin(\phi_r) \cdot d_m^s \end{bmatrix}. \tag{3.2}$$

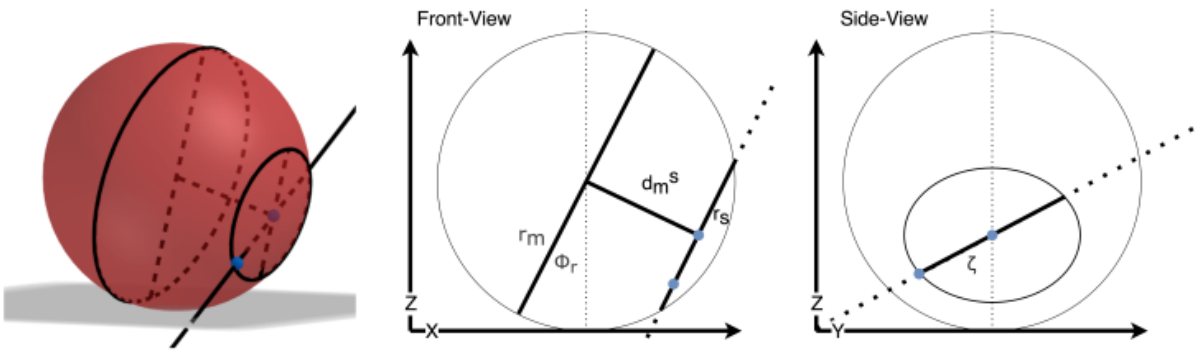


Figure 3.1: Sketch of a rod for calculating l. The blue points represent $\mathbf{p}_m$ and $\mathbf{p}_t$. [12]

Let $\mathbf{d}_{mt}$ be the vector from $\mathbf{p_m}$ to $\mathbf{p}_t$, then we derive $\mathbf{p}_t$ to be

$$\mathbf{p}_t = \mathbf{p}_m + \mathbf{d}_{mt} = \mathbf{p}_m + \begin{bmatrix} -\cos(\zeta) \cdot r_s \cdot \sin(\phi_r) \\ -\sin(\zeta) \cdot r_s \\ -\cos(\zeta) \cdot r_s \cdot \cos(\phi_r) \end{bmatrix}. \tag{3.3}$$

As we have a sphere d_m^s is constant.

$$d_m^s = \sin\left(\arccos\left(\frac{r_s}{r_m}\right)\right) \cdot r_m \tag{3.4}$$

Until now we have only looked at the right side disc in direction of the y-axis, but with the addition of a side factor s_f, which is $+1$ for the right side and -1 for the left, we derive a general $\mathbf{p}_m$ as

$$\mathbf{p}_m = \begin{bmatrix} s_f \cdot \cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - s_f \cdot \sin(\phi_r) \cdot d_m^s \end{bmatrix}. \tag{3.5}$$

From this we calculate the line $L_r(\zeta, \phi_r, s_f)$, that represents a specific rod, which goes through $\mathbf{p}_m$ in direction of $\mathbf{d}_{mt}$.

$$L_r(\zeta, \phi_r, s_f) = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \mid \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{p}_m + \lambda \cdot \mathbf{d}_{mt} \mid \lambda \in \mathbb{R} \right\} \tag{3.6}$$

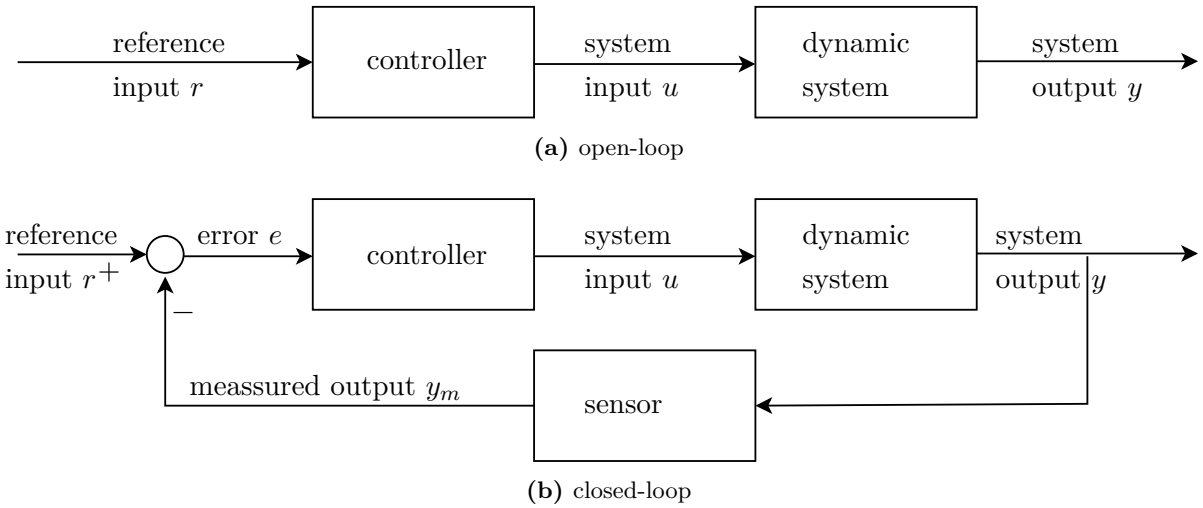
λ represents how far the individual rods extend, 0 representing to $\mathbf{p}_m$, which is not physically possible, 1 to $\mathbf{p}_t$, which is the minimum. By normalizing $\mathbf{d}_{mt}$, where $||\mathbf{d}_{mt}||$ is equal to r_s, as it is the vector from $\mathbf{p}_m$, the inside of the side disc, to $\mathbf{p}_t$, on the edge of the disc, and replacing λ with $r_s + l$, we derive $L_r(\zeta, \phi_r, s_f)$ to be

$$L_r(\zeta, \phi_r, s_f) = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \mid \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{p}_m + (r_s + l) \cdot \frac{\mathbf{d}_{mt}}{||\mathbf{d}_{mt}||} \mid l \in \mathbb{R} \right\}$$

$$= \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \mid \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_f \cdot \cos(\phi_r) \cdot d_m^s \\ 0 \\ r_m - s_f \cdot \sin(\phi_r) \cdot d_m^s \end{bmatrix} + (r_s + l) \cdot \begin{bmatrix} -\cos(\zeta) \cdot \sin(\phi_r) \\ -\sin(\zeta) \\ -\cos(\zeta) \cdot \cos(\phi_r) \end{bmatrix} \mid l \in \mathbb{R} \right\}, \tag{3.7}$$

with l the length of each individual rod. As the rods extend to the intersection of $L_r(\zeta, \phi_r, s_f)$ and Π_{VPIP}, we set the components of Equation 3.7 into Equation 3.1, to calculate l.

$$\begin{aligned} &\Pi_{VPIP} = L_r(\zeta, \phi_r, s_f) \\ \Leftrightarrow\ &\tan(\phi_{VPIP}) \cdot L_r(\zeta, \phi_r, s_f)_x + \tan(\theta_{VPIP}) \cdot L_r(\zeta, \phi_r, s_f)_y + L_r(\zeta, \phi_r, s_f)_z = 0 \\ \Leftrightarrow\ &l = \frac{\tan(\phi_{VPIP}) \cdot s_f \cdot \cos(\phi_r) \cdot d_m^s + r_m - s_f \cdot \sin(\phi_r) \cdot d_m^s}{\tan(\phi_{VPIP}) \cdot \cos(\zeta) \cdot \sin(\phi_r) + \tan(\theta_{VPIP}) \cdot \sin(\zeta) + \cos(\zeta) \cdot \cos(\phi_r)} - r_s \end{aligned} \tag{3.8}$$



(a) open-loop

(b) closed-loop

Figure 3.2: Schematic overview of an open- vs. closed-loop control system.

This calculation is done for all of the rods, giving us the length l for each one of them. This still leaves the possibility for a negative l, in which case the rod fully retracts, and a l larger than the maximum length of a rod l_{max}, which just means a full extension of the rod.

3.2 Open-loop vs. Closed-loop controller

In control theory we differentiate between open- and closed-loop control. The difference is whether the control system uses feedback (closed-loop) or not (open-loop). An open-loop controller, as depicted in Figure 3.2a, uses a reference input as the input for a controller, that passes a system input on to the dynamic system. This then results in a system output. Crucially the controller works completely independent from the system output, so there is no possibility to react to potential disturbances or unexpected system behaviour. It is not even possible to detect any of these disturbances. Also there is no way to know the initial state of the system, so this control system can only be used, in very simple cases, where we understand the dynamic system precisely, and the initial state is irrelevant. The only sensible controllers are a timer, which just lets the system run for a certain amount of time, or a look up table, which associates a system input with a specific desired output.
In the closed-loop case (Figure 3.2b) we use a sensor, that measures the system output. This then gets compared to the reference input, so that the controller no longer uses the reference input directly as an input, but an error, describing how far off the desired result we are. Therefore we detect and incorporate the output of the dynamic system into the controller. The output of the controller is passed to the dynamic system, same as in the open-loop case. For the closed-loop control system there are many different controllers, that all share the same goal of minimizing the control error. Following are descriptions of a few.

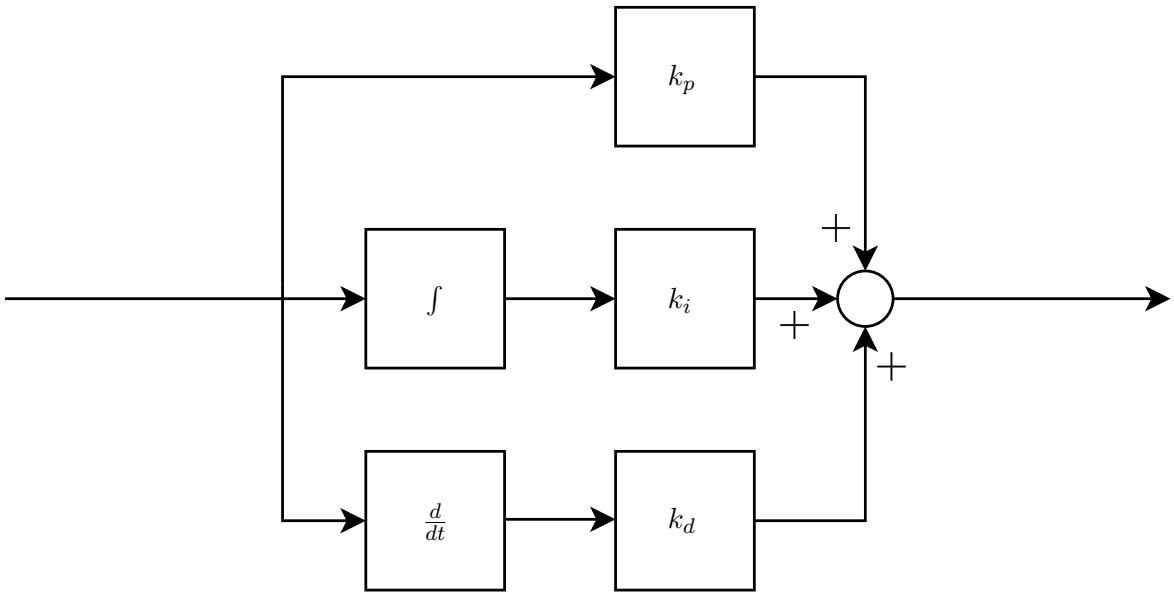


Figure 3.3: Schematic overview of a PID controller

3.3 PID controller

One of the most common controllers is a Proportional-Integral-Derivative (PID) controller. As with all controllers for closed-loop control systems, it takes a control error as an input, and calculates a system input from the error. The PID controller does this by adding three different parts together, as shown in Figure 3.3. The first is the proportional (P) part, defined by the k_p gain. It simplify multiplies the error by k_p. The second is the integral (I) part, defined by the k_i gain. For this we integrate the error over time and then multiply it by k_i. As an integration is not realizable in an actual implementation, we take the sum over all the error values and multiply that by the time in between two successive control cycles. This therefore represents the integration, after which we multiply the error sum by k_i. The third is the derivative (D) part, defined by the k_d gain. Similar to the I part, we first process the error and then multiply it by k_d. In this case we take the derivative of the error, before multiplying by k_d. As with the integration also a differentiation is not possible in a real implementation of the controller. Instead we take the difference of the latest and second to last error value and divide that by the time in between two successive control cycles. This then forms the differentiation, after which we multiply it by k_d. This gives us the final equation for a PID controller.

$$u(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau)\, d\tau + k_d \cdot \frac{d}{dt} e(t) \tag{3.9}$$

The three different parts of the PID controller make it possible to react in different ways to a changing control error. While the P part is responsible to get rid of the significant part of the error, the I part is able to neutralize error, that remains over a longer period of time, effectively looking into the past. The D part works against large changes in the error, preventing a large overshoot, once the desired value is reached, effectively looking into the future. The necessity

for and sizing of each of the parts depends on the dynamic system. By setting the different gains we tune the PID controller to the different systems, and to our preferred system response. If we leave out one or two of the gains (set them to 0), it is no longer considered a PID, but P, PI, PD or I controller. Depending on the system this may be preferable to create the desired system response.
One prominent problem with the integral part is integral windup. If we change the reference input of the system by a large amount at once, the resulting error is fairly large. During the time it takes the system to respond the integral winds up to a to high value, that leads to a large overshoot. In this case the I part no longer responds to a persisting steady state error, but to the large system response itself. This is not the desired effect of the I part, therefore there are multiple ways to reduce this behaviour. A simple solution is to limit the magnitude of the integral to a certain maximum value. If chosen correctly, this means, that the integral ignores most of the major change in the system response following a rapid major change in the reference input, and instead only focuses on a persisting steady state error. In a real implementation, in which the integral is replaced by a sum, we achieve this by capping the magnitude of the sum to certain value. [1]

3.4 LQR

The Linear-Quadratic Regulator (LQR) is a optimal state controller. To use a LQR we therefore must first define a linear state space system model. This is a method to describe a linear system, using a state $\mathbf{x}$ and a formula to calculate $\dot{\mathbf{x}}$.

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u}, \tag{3.10}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ the input matrix, and $\mathbf{u}$ the input signal. The aim of the LQR is to lead $\mathbf{x}$ towards 0. Now we set up a cost function

$$J = \int_0^\infty \mathbf{x}(t)^\top \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^\top \mathbf{R}\mathbf{u}(t)\, dt, \tag{3.11}$$

that describes how costly a specific system response is. In this function $Q \succeq 0$ describes the cost associated with $\mathbf{x}$ not being 0, and $\mathbf{R} \succ 0$ describes the cost associated with a large $\mathbf{u}$. In particular the diagonal entries $q_{i,i}$ describe the cost associates with x_i not being 0, and $r_{i,i}$ describe the cost associates with u_i. Therefore x_i converges towards 0 faster, if $q_{i,i}$ is increased, and an increase in $r_{i,i}$ reduces the maximum u_i used in the response. There is no general way to describe what effect the values $q_{i,j}$ and $r_{i,j}$ with $i \neq j$ have, so they are typically set to 0. This also ensures $\mathbf{Q} \succeq 0$ and $\mathbf{R} \succ 0$, as long as $q_{i,i} \geq 0$ and $r_{i,i} > 0$. Minimizing the cost function J you can show, that the following is true for $\mathbf{u}$: [11]

$$\mathbf{u} = -\mathbf{K}\mathbf{x} \text{ , with} \tag{3.12}$$

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^\top \mathbf{P}, \tag{3.13}$$

where $\mathbf{P}$ is the only positive-definite solution to the algebraic Riccati equation

$$\mathbf{P}\mathbf{A} + \mathbf{A}^\top \mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top \mathbf{P} + \mathbf{Q} = 0. \tag{3.14}$$

As long as we know the system described by Equation 5.3, and have set **Q** and **R**, we calculate **K** using Equation 3.14 and 3.13 once at the beginning. Then the controller calculates the system input using Equation 3.12 once per control cycle.
To find the solution for **K** we must first calculate **P** by solving Equation 3.14. As this Equation is quadratic in **P**, the solution is non trivial. But if and only if the underlying system described by **A** and **B** is stabilizable, there is exactly one positive-definite solution for **P**, which we then use to calculate **K**. For the system to be stabilizable, it has to be either controllable or the real part of all Eigenvalues of **A** have to be negative.

Chapter 4

Controlling of the VPIP

This thesis focuses on the controlling of the rotational speed ω of the robot, using θ_{VPIP}. So we don't consider ϕ_{VPIP}, which is always set to 0°, or the orientation of the robot. Further research has to be done in controlling the orientation using ϕ_{VPIP}.
In this chapter we setup multiple theories on control strategies for the pitch of the VPIP θ_{VPIP}. We run multiple experiments in Chapter 5, that are designed to verify or contradict the theories, described in the following.

(1) Any control system, that is used, works in the same way, both in the forward and backward direction. This also means braking is simply a backward acceleration, so no special considerations have to be given to braking or backward movement, as any control strategy can work with negative θ_{VPIP} and error values, to achieve these types of movement.

(2) As described in Section 2.2 a θ_{VPIP} of 0° theoretically has no influence on the movement of the robot. Similarly a θ_{VPIP} of 90° leads to no rods extending at all, which also has no influence on the movement of the robot. Therefore there has to be maximum useful angle for θ_{VPIP} somewhere between 0° and 90°, where the impact on the rotation of the robot is the largest. Increasing θ_{VPIP} further than that leads to a smaller impact.

(3) Analogous to **(2)** there is also a minimum angle for θ_{VPIP}, before it has any impact on the robot at all. Even though theoretically a θ_{VPIP} of 0° has no impact on the movement, the robot actually slows down, so a continuous $\theta_{VPIP} > 0°$ is necessary to achieve a constant positive rotational velocity, which means there is no impact on the robot. An even larger angle is necessary to accelerate.

(4) We theorize, that the system response for a specific θ_{VPIP} is characterized by a certain final value for ω, but more significantly also by a certain acceleration $\dot{\omega}$. This means even though an open-loop controller is able to achieve a specific desired rotational speed ω_{des}, a closed-loop controller with feedback, does so much faster, using more aggressive θ_{VPIP}.

(5) We assume, that the robot has very high friction, so the natural damping of the system is high. Therefore, if a PID controller is used, no D gain is necessary. So a PI controller achieves similarly good results.

(6) Based on the high natural damping of the system theorized in **(5)** and the necessary $\theta_{VPIP} > 0$ in **(3)**, we assume, that when using a PID controller the I gain is necessary, to get rid of an otherwise persisting steady-state error.

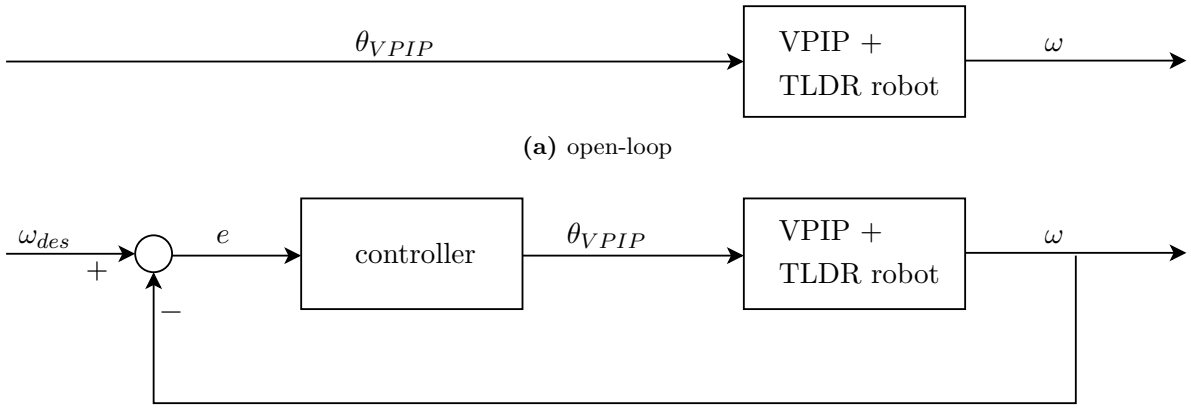
Chapter 5

Experiments

To validate the different theories presented in Chapter 4 we run multiple experiments, using both open-loop and closed-loop control systems with varying controllers. As the TLDR robot is a theoretical concept, that needs further research, there is no actual functioning TLDR robot, on which the experiments are run. Instead a simulation is used. The simulation used was first created and setup in [9]. It uses a combination of the Gazebo simulator and Robot Operating System (ROS) to simulate a TLDR robot, with a r_m of 23.2 cm, r_s of 7.0 cm and l_{max} of 37.8 cm. The simulation implements multiple locomotion approaches, including the VPIP, and supports different terrains. In all of the following experiments we only use the VPIP based locomotion on a flat ground. Also all test runs start the exact same way: the robot starts with 0 rad/s rotational velocity, from where the different control strategies take over.
As seen in Figure 5.1a the open-loop control system is very simple, using θ_{VPIP} as a reference input, feeding it directly into the dynamic system, in this case a TLDR robot using the VPIP locomotion approach. This allows us to characterize both the acceleration and final achieved rotational velocity for different VPIP angles. Some of these results (Section 6.1) are used to set up controllers for the closed-loop control approach.
Figure 5.1b shows the used closed-loop control system, for which we use varying controllers. We set a desired rotational speed ω_{des} as a reference input, from which the actual rotational speed ω is subtracted, to give us the control error e. This is the input for the controller, that is being tested, which gives us a θ_{VPIP} as an output. Same as in the open-loop case, this is passed to the simulation, from which we track the output ω, to validate the controllers.

5.1 Open-loop controller

The first test case we run is the open-loop control system. In total we run nine test runs with varying θ_{VPIP} angles. All of the tests runs start at standstill and are simulated until a final ω is reached. We run these tests to give us a first general understanding, of the effects of the VPIP on the rotational speed of the robot. We therefore test a wide spread of positive angles: $10°, 15°, 20°, 30°, 40°, 50°$ and $60°$. To characterize the backwards movement we also choose some negative angles, that are then compared to the result of the corresponding positive angle: $-30°$ and $-15°$.



(a) open-loop

(b) closed-loop

Figure 5.1: Schematic overview of the open- and closed-loop control systems used for our experiments.

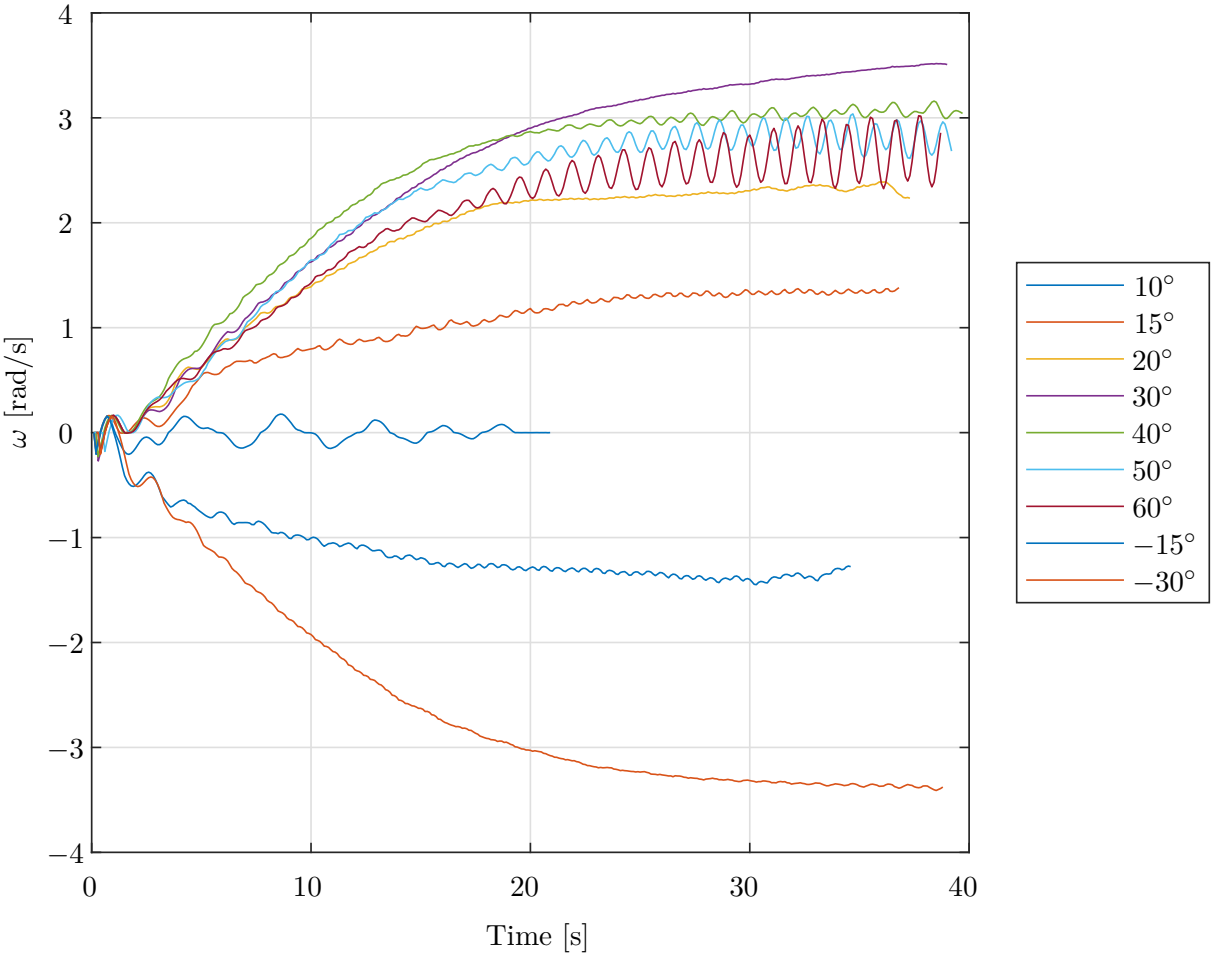


Figure 5.2: The resulting rotational speed ω of the TLDR robot for varying input angles θ_{VPIP}.

θ_{VPIP}	initial acceleration [rad/s^2]	final ω [rad/s]
$-30°$	-0.19244	-3.4348
$-15°$	-0.10044	-1.3958
$10°$	-0.00020271	0.011113
$15°$	0.079745	1.3707
$20°$	0.13893	2.3794
$30°$	0.16257	3.5476
$40°$	0.18522	3.1273
$50°$	0.16448	2.8708
$60°$	0.14259	2.7045

Table 5.1: The initial acceleration (average over the first 10 s), and the final value (average over the last 5 s) of the open-loop responses, depending on their θ_{VPIP}.

θ_{VPIP} [°]	ω [rad/s]	θ_{robot} [°]	ϕ_{robot} [°]
40	± 0.1	± 2	± 15
50	± 0.2	± 3	± 20
60	± 0.3	± 5	± 25

Table 5.2: Magnitude of oscillations in ω, θ_{robot} and $\phi robot$ for varying high θ_{VPIP}. This is also shown in Figure 6.3.

Figure 5.2 shows the response of the TLDR robot for varying θ_{VPIP}. The first two seconds of the trajectory are the same for each of the different angles. Only after that point do the trajectories differ from one another. For an input angle θ_{VPIP} of 10° we see an oscillation around 0 rad/s. The final value for ω is 0 rad/s, so the robot moves neither significantly forward nor backwards, but only sways back and forward. Table 5.1 shows the initial acceleration and the final value for ω of the different open-loop responses. The initial acceleration is given by the average acceleration over the first 10 s of the test run, and the final value is the average value over the last 5 s. Both in the response to 15° and $-15°$ we notice a smaller and faster oscillation, compared to the oscillation of the response to 10°. For higher θ_{VPIP} (40°, 50° and 60°) we also observe an oscillation, as soon as the final value for ω is reached. The larger θ_{VPIP} is, the larger the oscillation gets. We show this in Table 5.2.

5.2 PID controller

The first closed-loop controller we test is the PID controller. We use a standard PID controller set up in Section 3.3, that runs at a frequency of 100 Hz, with varying k_p, k_i and k_d gains. The only small adjustment we make is to limit θ_{VPIP} to $\pm 50°$, as explained in Section 6.1. We start by only using a k_p gain, so a P controller, before adding in a k_i and k_d gain. All of the tests

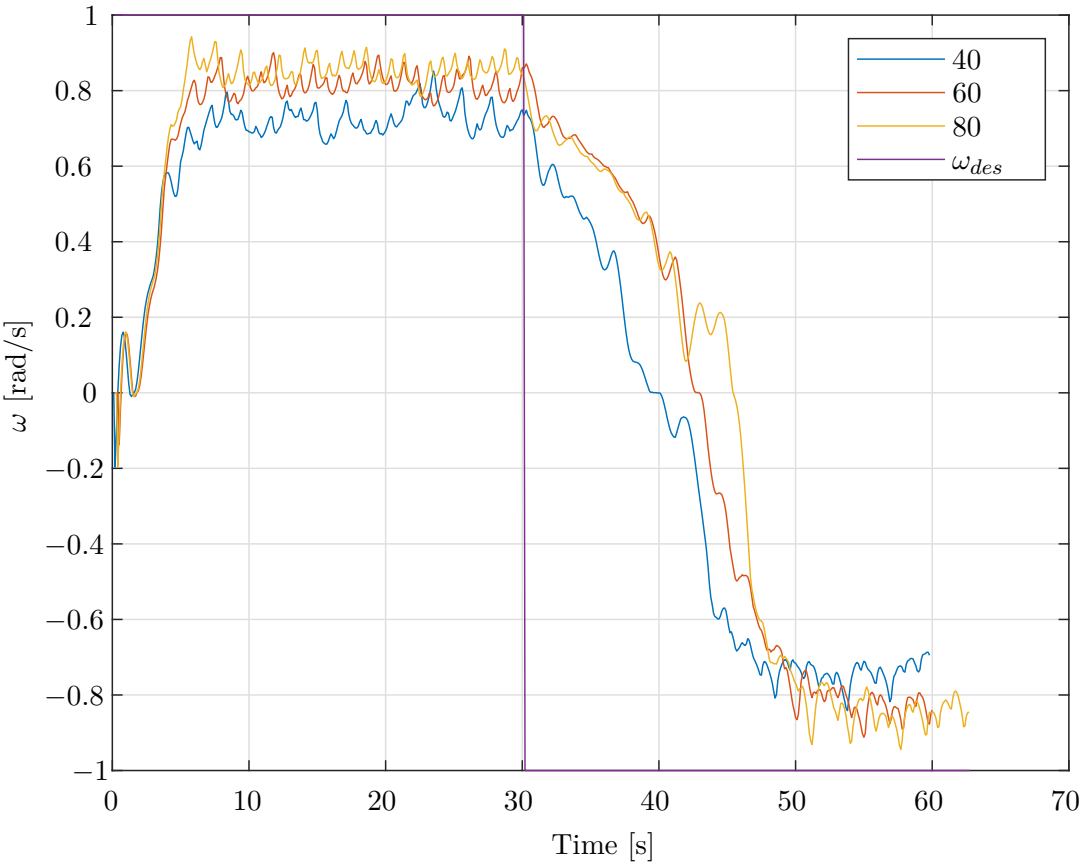


Figure 5.3: The response of ω to the test runs (a) using a P controller with varying k_p gains.

will use one of two test runs. The test runs start with the robot at standstill, from where it is supposed to accelerate to a rotational speed of 1 rad/s (ω_{des} is set to 1). After 30 s we either change ω_{des} to -1 (test run (a)) or to 0 (test run (b)). Test run (a) tests both the ability of the robot to brake, which we didn't test using the open-loop control structure, and the ability to move backwards with a specific target for ω and not just moving backwards in general. With test run (b) we also characterize the ability of the controller to control the robot at low rotational speeds.

5.2.1 P tuning

The first test runs use a simple P controller limiting θ_{VPIP} to $\pm 50°$. We run the test for a k_p of 40, 60 and 80.
The system response to test run (a) is depicted in Figure 5.3. We see the characteristic start pattern (first 1.5 s) described in Section 6.1, before the actual response starts. All of the response curves share a common course with varying parameters. Table 5.3 lists these parameters. We notice a small difference in the positive acceleration, which is the average acceleration over the first 5 s of the response. Then ω settles at a value below 1 rad/s leaving us with a steady-state

k_p	positive acceleration [rad/s^2]	braking [rad/s^2]	negative acceleration [rad/s^2]
40	0.12247	−0.075688	−0.11527
60	0.14257	−0.066200	−0.13580
80	0.15162	−0.054324	−0.15540

k_p	steady-state error at $\omega_{des} = 1$ [rad/s]	steady-state error at $\omega_{des} = -1$ [rad/s]
40	0.26390	−0.25275
60	0.17088	−0.15127
80	0.12229	−0.13693

Table 5.3: Acceleration, braking and steady-state error for the responses to the test run (a) using a P controller with varying k_p gains.

k_p	positive acceleration [rad/s^2]	braking [rad/s^2]
40	0.13061	−0.27897
60	0.14824	−0.094646
80	0.15324	−0.061238

Table 5.4: Acceleration and braking for the responses to the test run (b) using a P controller with varying k_p gains.

error, which is calculated by taking the average steady-state error from second 25 through 30. After 30 s ω_{des} is set to −1 rad/s, leading to the robot braking. We consider the braking phase to last until a ω of 0 rad/s is reached. The average deceleration over this period is also listed in Table 5.3. The next 5 s determine the negative acceleration, before ω settles at a final value above ω_{des}. We gather, that the magnitude of this negative steady-state error is similar to the corresponding positive one.
The results of test run (b), shown in Figure 5.4 are similar during the first 30 s. Both the initial acceleration and steady-state error are closely matched (cf. Table 5.3, 5.4 and Figure 5.3). The braking is much faster for the lower k_p gain of 40, but similar for the higher k_p gains. We also notice a large, slowly decaying oscillation. In the extreme case $k_p = 40$ up to ±0.3 rad/s and over 40 s of decay time.

5.2.2 I tuning

After the tuning of k_p is done we go on tuning k_i. For these test runs we set k_p to 40, as it doesn't have a significant impact on acceleration or braking, only on the remaining steady-state error, that in the PI controller is handled by the I part. In these test runs the integral is implemented using a sum, that starts at the beginning of the test run and sums up all errors. We test k_i gains of 0.5, 1, 2 and 5. For the I tuning we test only using the test run (a).
Figure 5.5 shows the response of the simulation to the PI controller. The response is very similar to the to the P controller (cf. Figure 5.3). Both positive and negative acceleration are similar,

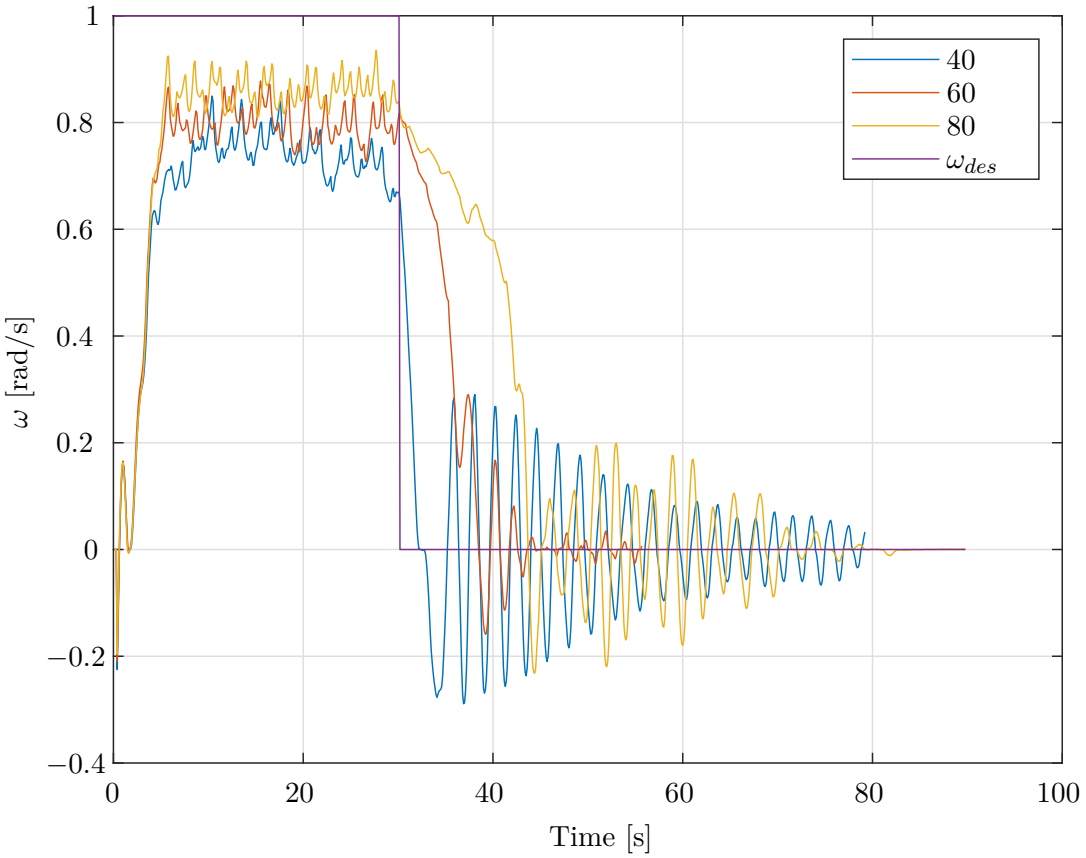


Figure 5.4: The response of ω to the test runs (b) using a P controller with varying k_p gains.

with braking again being considerably slower. The difference to the P controller lays in the steady-state error, shown in Table 5.5, and the large overshoot in the second phse of the tet run.

5.2.3 I tuning with Integral limit

As theoretically explained in Section 3.3 and seen in Figure 5.5 integral windup leads to a problem, if we have a large jump in ω_{des}. To mitigate this we implement a maximum value of 5 rad for the integral, in our implementation a sum. Whenever the sum is above or below 5 rad, it is set to ± 5 rad. Now we repeat the test runs (a) from Section 5.2.2, but only for k_i gains of 1, 2 and 5. We also test this controller using the same gains with test run (b).
Figure 5.6 shows the course of ω over the test run (a). Again the response is very similar to Figure 5.5, but this time the control responses also reach a steady-state error close to 0 rad/s for $\omega_{des} = -1$ rad/s, and without the large overshoot. This is listed in Table 5.6. The result for test run (b) is shown in Figure 5.7. The first 30 s show the same behaviour, after which we see similar behaviour to the P controller response (cf. Figure 5.4). The braking is faster than in test run (a), leaving us with some oscillations up to ± 0.2 rad/s in magnitude, decaying to under ± 0.1 rad/s within 20 s.

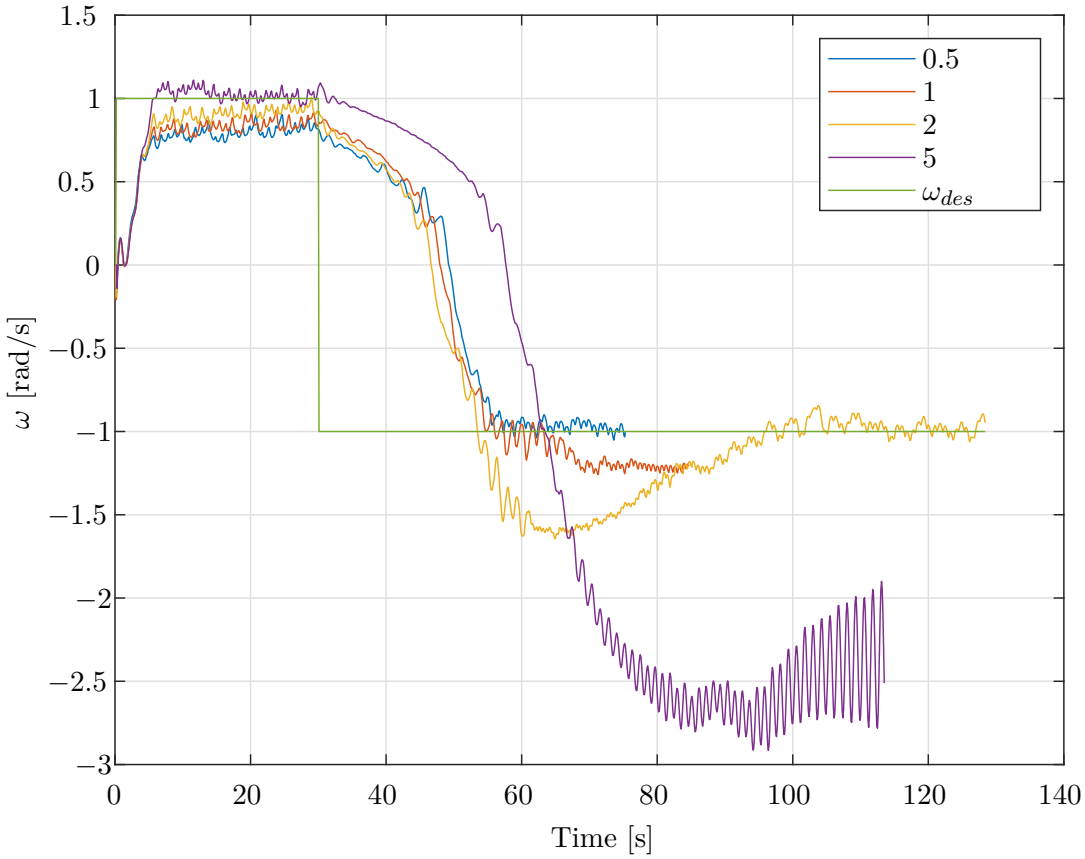


Figure 5.5: The response of ω to the test run (a) using a PI controller with a k_p gain of 40 and varying k_i gains.

k_i	steady-state error at $\omega_{des} = 1$ [rad/s]	steady-state error at $\omega_{des} = -1$ [rad/s]
0.5	0.16444	0.0070556
1	0.12935	0.24307
2	0.041075	0.0029226
5	−0.026303	1.4051

Table 5.5: The remaining steady-state error in ω for the responses to the test run (a) using a PI controller with a k_p of 40 and varying k_i gains.

k_i	steady-state error at $\omega_{des} = 1$ [rad/s]	steady-state error at $\omega_{des} = -1$ [rad/s]
1	0.12975	−0.12500
2	0.025124	−0.0042719
5	−0.047355	0.072218

Table 5.6: The remaining steady-state error in ω for the responses to the test run (a) using an integral limited PI controller with a k_p of 40 and varying k_i gains.

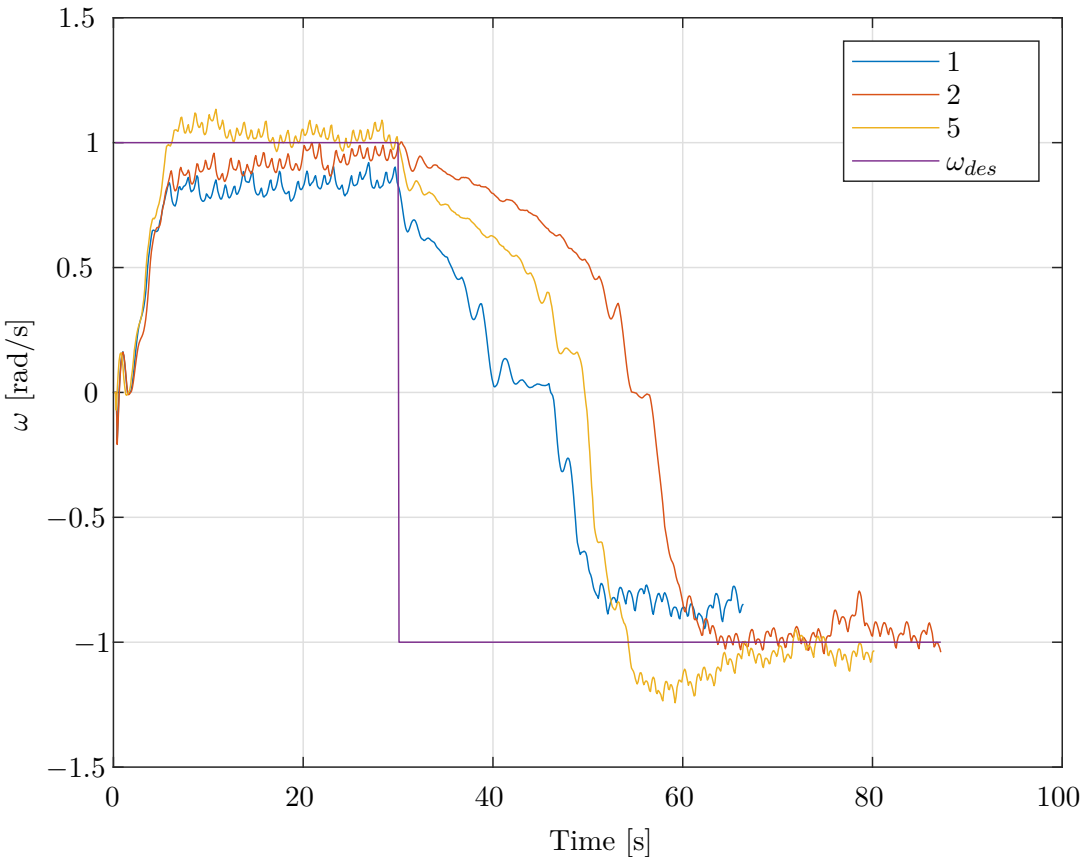


Figure 5.6: The response of ω to the test run (a) using a integral limited PI controller with a k_p gain of 40 and varying k_i gains.

5.2.4 D tuning

Now that we are done tuning the k_p and k_i gain, we go on testing a full PID controller by tuning the k_d gain. For this we run through test run (a) using $k_p = 40$, $k_i = 5$ and $k_d = 20$. As described in Section 3.3 the differentiation of the error can not be calculated. Therefore we take the difference between the last and second to last error and multiply it with the frequency of the controller.
The response both for ω and θ_{VPIP} is depicted in Figure 5.8. In the course of ω there is virtually no difference compared to the PI controller with $k_p = 40$ and $k_i = 5$ in Figure 5.6. The difference is in the large (up to $\pm 15°$) and very fast oscillations of θ_{VPIP}.

5.2.5 D tuning with filter

As seen in Figure 5.8 we have to change the implementation of the D part, as it otherwise leads to large oscillations in θ_{VPIP}. Instead of taking the difference between the last and second to last error value, we use average values. We save both the average error over the last second and over the second before that. We use the difference between these average values divided by 1 s,

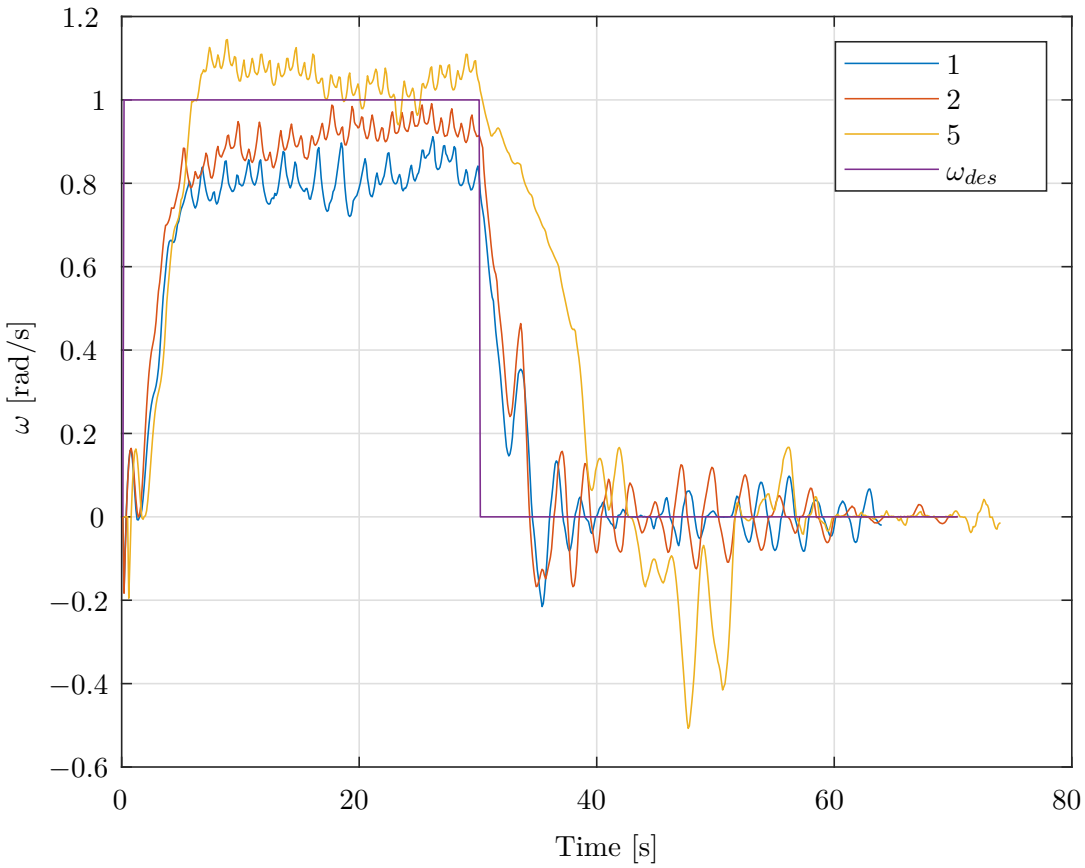


Figure 5.7: The response of ω to the test run (b) using an integral limited PI controller with a k_p gain of 40 and varying k_i gains.

as they lay 1 s apart from one another, to represent the differentiation. We run both test runs (a) and (b) for $k_p = 40$, $k_i = 5$ and varying k_d values of 20, 50 and 100.
Figure 5.9 shows the response of the system to test run (a). Comparing it to the PI controller in Figure 5.6 we see no significant difference. The acceleration is slightly lower and the oscillations are slightly larger for the highest k_d gain of 100. The result of test run (b) in Figure 5.10 convey a similar picture. The acceleration and braking are slightly decreased and higher k_d values also have higher oscillations.

5.3 LQR

To set up a LQR we must first set up a linear state space system. As we want to control ω, it must be a part of the state $\mathbf{x}$. To now set up $\mathbf{A}$ and $\mathbf{B}$, we need a formula for $\dot{\omega}$. Assuming we only want to control relatively low rotational speeds of up to ~ 1 rad/s, we use the initial acceleration from Table 5.1 for the positive θ_{VPIP} angles. As θ_{VPIP} is limited to 50° as with all controllers, we ignore the data point for $\theta_{VPIP} = 60°$. As the system has to be linear we now

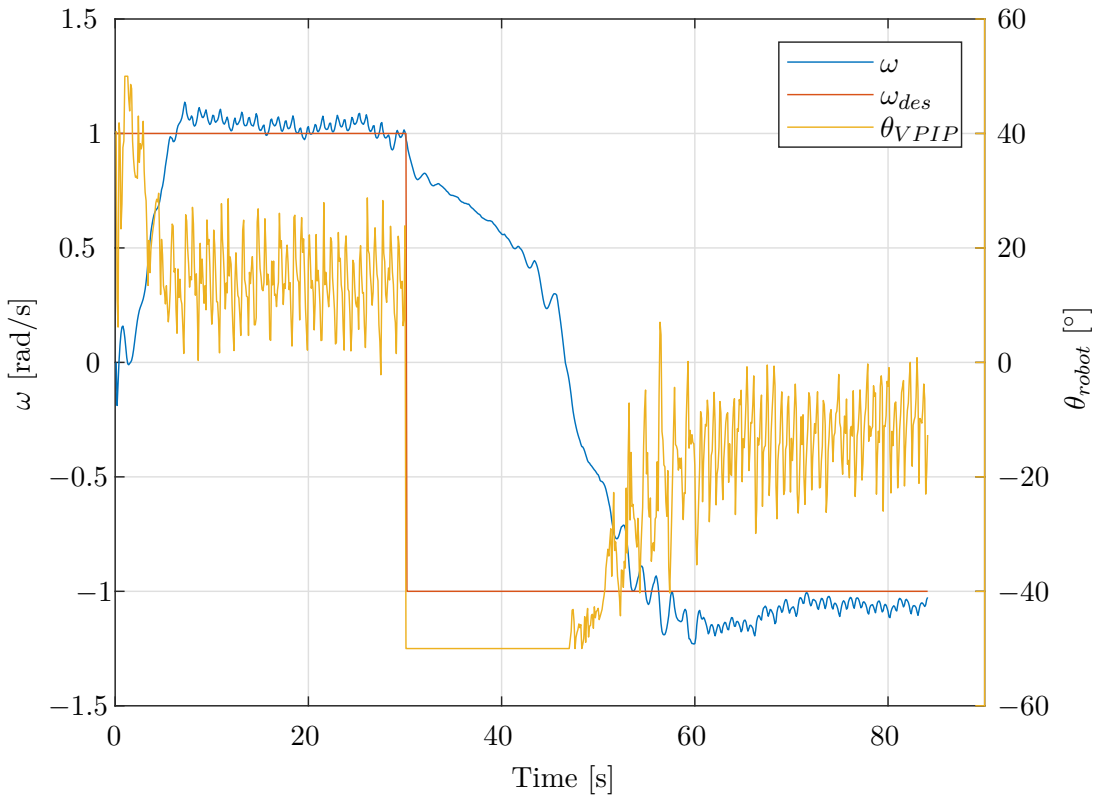


Figure 5.8: Response of ω and θ_{VPIP} to test run (a) with a PID controller using $k_p = 40$, $k_i = 5$ and $k_d = 20$.

approximate $\dot{\omega}$ with

$$\dot{\omega} \approx m \cdot \theta_{VPIP} + s, \tag{5.1}$$

with $m = 0.007$ rad/s^2/° and $s = -0.06$ rad/s^2. These values for m and s don't represent the best fit, but put more significance on the data point for $\theta_{VPIP} = 10°$, as it is crucial, that this point is adhered to (cf. Theory **(3)**). This gives us the linear state space system with

$$\mathbf{x} = \begin{pmatrix} \omega \\ s \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} m \\ 0 \end{pmatrix} \quad \text{and} \quad u = \theta_{VPIP}. \tag{5.2}$$

Unfortunately this system is not stabilizable, as it isn't controllable (s can't change) nor are the real parts of the Eigenvalues of $\mathbf{A}$ negative (both are 0). Therefore we transform the system to:

$$\mathbf{x}' = \begin{pmatrix} \omega \\ s \end{pmatrix}, \quad \mathbf{A}' = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{B}' = \begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{u}' = \begin{pmatrix} \theta_{VPIP} \\ s' \end{pmatrix}. \tag{5.3}$$

We now set up $\mathbf{Q}$ and $\mathbf{R}$ to be

$$\mathbf{Q} = \begin{pmatrix} q & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{R} = \begin{pmatrix} r & 0 \\ 0 & \infty \end{pmatrix}. \tag{5.4}$$

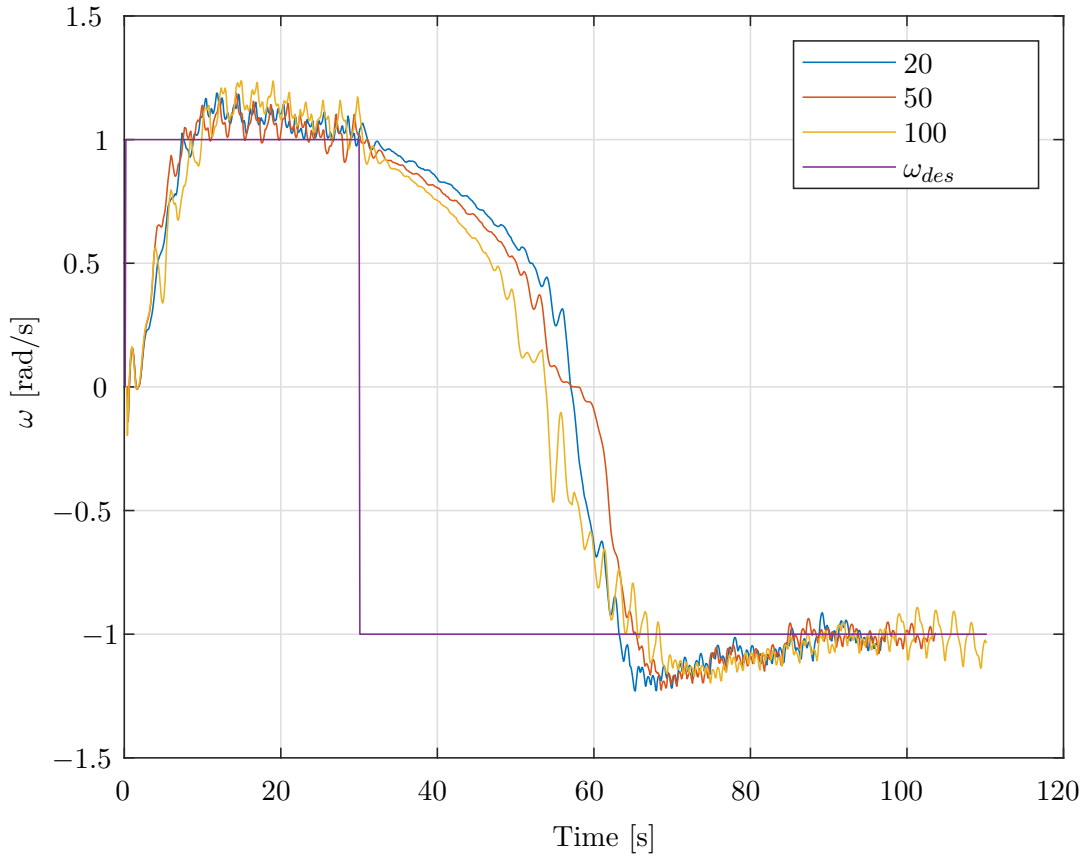


Figure 5.9: Response of ω to test run (a) using a PID controller with $k_p = 40$, $k_i = 5$ and varying k_d gains.

q_{22} and r_{22} are set to 0 and ∞ respectively as s must not change. We substitute Equations 5.3 and 5.4 into the Riccati Equation 3.14, and solve numerically for $\mathbf{P}'$.

$$\mathbf{P}' = \begin{pmatrix} \sqrt{\frac{q \cdot r}{m^2}} & \frac{r}{m} \\ \frac{r}{m} & \infty \end{pmatrix} \tag{5.5}$$

Using $\mathbf{P}'$ and Equation 3.13 we get numerically

$$\mathbf{K}' = \begin{pmatrix} \sqrt{\frac{q}{r}} & \frac{1}{m} \\ 0 & 0 \end{pmatrix}. \tag{5.6}$$

With $\mathbf{K} = \mathbf{K'}_1$ we get from Equation 3.12

$$\theta_{VPIP} = -\mathbf{K} \cdot \mathbf{x} = -\sqrt{\frac{q}{r}} \cdot \omega - \frac{s}{m}. \tag{5.7}$$

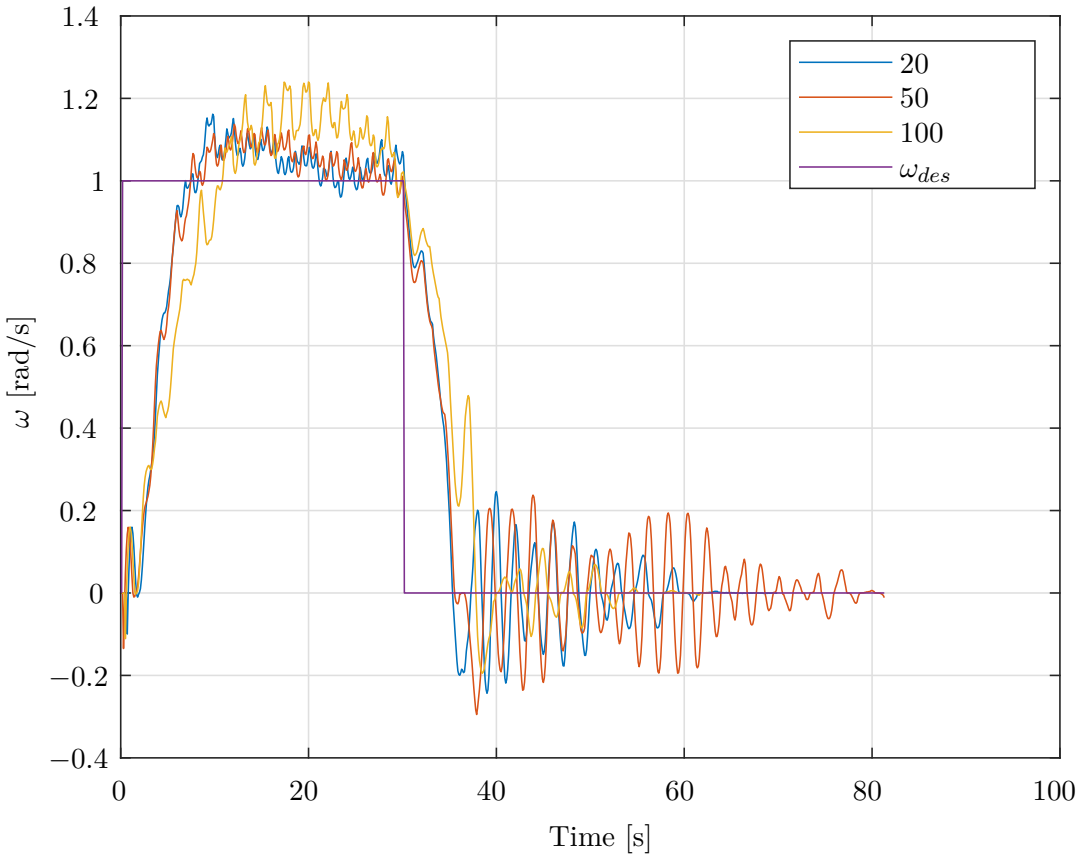


Figure 5.10: Response of ω to test run (b) using a PID controller with $k_p = 40$, $k_i = 5$ and varying k_d gains.

Using the control law from Equation 5.7 ω goes towards 0 rad/s. Instead we want the control error e to go towards 0 rad/s. As $e = \omega_{des} - \omega$, we replace ω with $-e$ to get

$$\theta_{VPIP} = k \cdot e - \frac{s}{m}, \tag{5.8}$$

with the gain $k = \sqrt{q/r}$. As we set up the state space model only using the positive values of θ_{VPIP}, Equation 5.8 is only used for positive ω_{des}. For negative ω_{des} s has to be multiplied by -1 and for $\omega_{des} = 0$ rad/s s has to be 0 rad/s^2. This gives us

$$\theta_{VPIP} = \begin{cases} k \cdot e - \frac{s}{m} & \text{for } \omega_{des} > 0 \text{ rad/s} \\ k \cdot e & \text{for } \omega_{des} = 0 \text{ rad/s} \\ k \cdot e + \frac{s}{m} & \text{for } \omega_{des} < 0 \text{ rad/s} \end{cases} \tag{5.9}$$

We test this LQR with the control law described by Equation 5.9 both with test run (a) and (b) and use k gains of 20, 40 and 60, with the control cycle again running at 100 Hz.
Figures 5.11 and 5.12 show the response of ω to the two test runs. The initial 30 s show almost the same response, with higher k gains showing faster responses and lower steady-state errors.

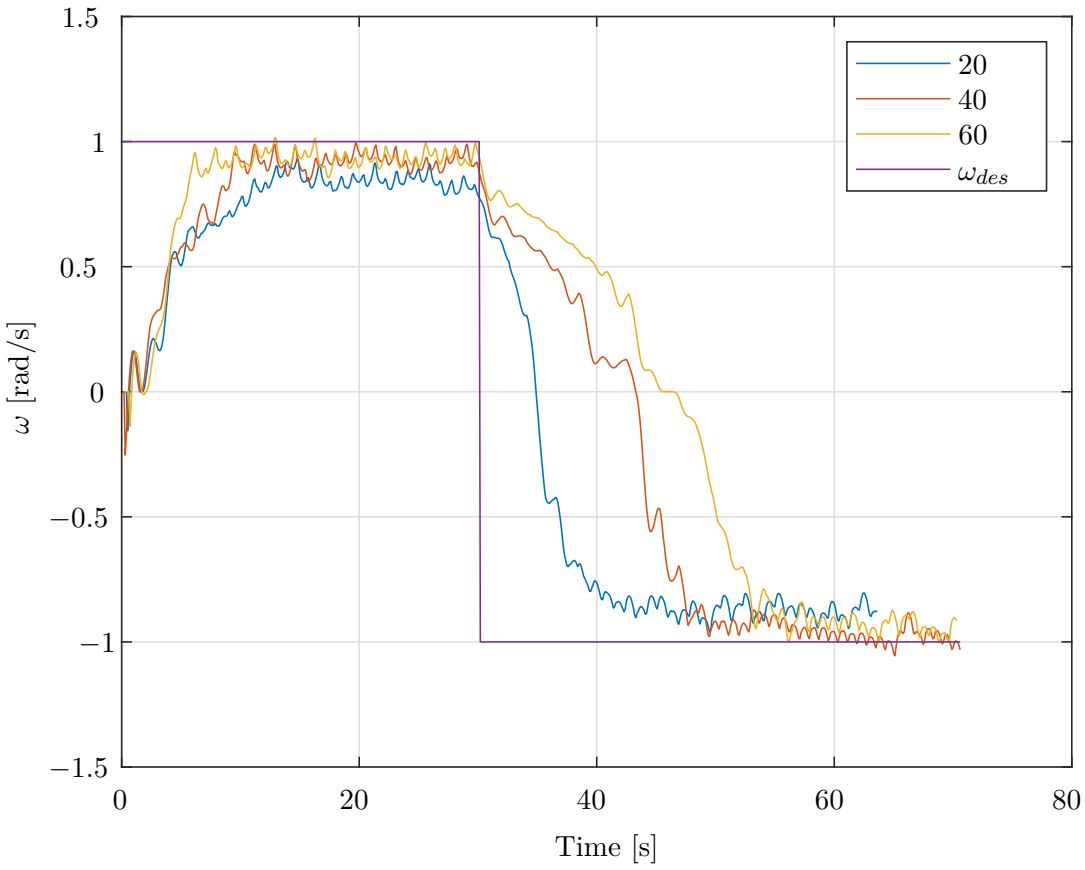


Figure 5.11: Response of ω to test run (a) using a LQR with varying k gains.

The magnitude of the steady-state error at $\omega_{des} = -1$rad/s in test run (a) is similar to the one at $\omega_{des} = 1$rad/s, as shown in Table 5.7. The braking and backwards acceleration are slower the higher the k gain is. For test run (b) all k gains show similar braking speeds and show a decreasing oscillation around 0 rad/s.

k	steady-state error at $\omega_{des} = 1$ [rad/s]	steady-state error at $\omega_{des} = -1$ [rad/s]
20	0.15342	−0.10279
40	0.063487	−0.010601
60	0.039298	−0.040483

Table 5.7: The remaining steady-state error in ω for the responses to the test run (a) using a LQR with varying k gains.

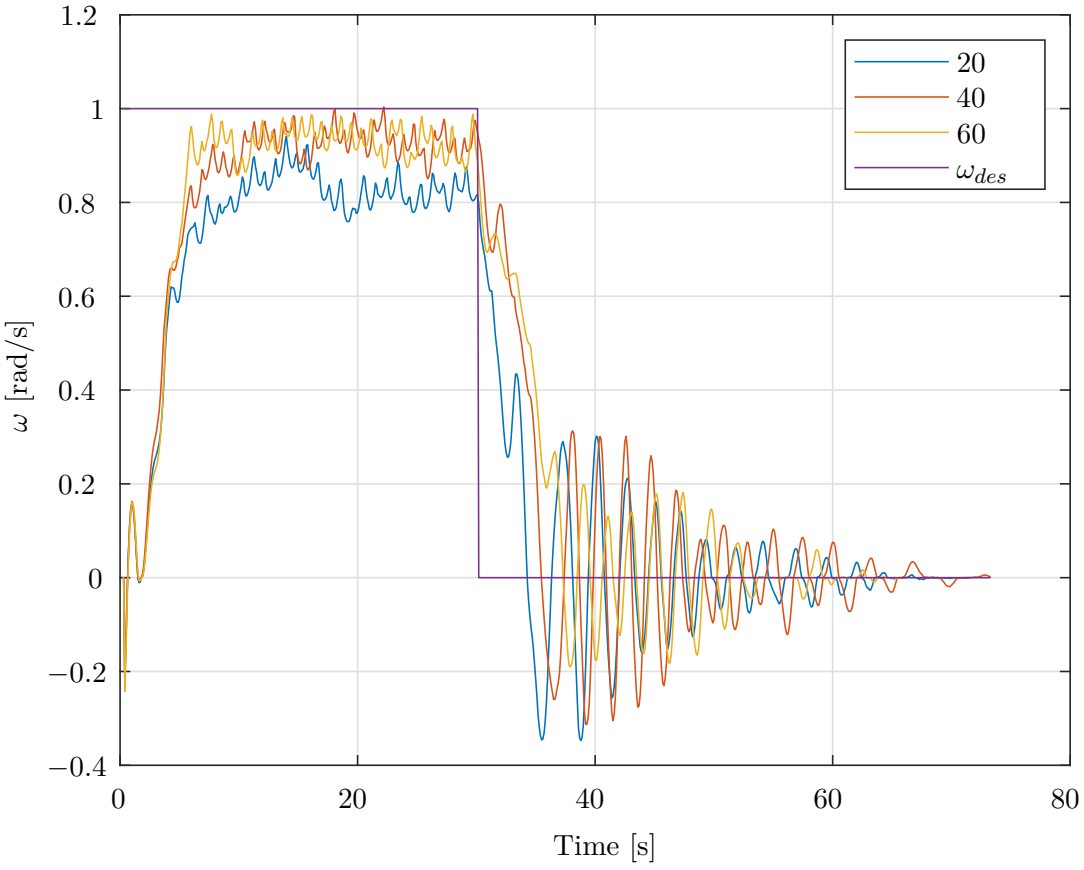


Figure 5.12: Response of ω to test run (b) using a LQR with varying k gains.

Chapter 6

Evaluation

In this chapter we take a closer look at the responses to the different experiments done in Chapter 5. We explain the results, that we observe, and then confirm or contradict the theories set up in Chapter 4, based on the results. We go through explaining the results of the experiments in the same order we did them in Chapter 5.

6.1 Open-loop controller

There is one observation we made with all experiments we did. The first 1.5 seconds of the simulation are always the same (cf. Figure 5.2). This is due to the way the simulation works, in particular the way the robot is created. When the robot is set into the world, all of the rods are completely retracted. Even though the necessary length of the rods get calculated immediately using the VPIP, it takes some time for them to extend and interact with the ground. Due the layout of the robot, in particular the starting angles ζ of the rods, that are not symmetrical about the xz-plane, the robot first rotates backward with up to -0.2 rad/s, than forwards with up to 0.15 rad/s before slowing down again. So at the point in time the rods interact with the ground and take over responsibility for the locomotion of the robot ω is roughly 0 rad/s, but $\dot{\omega}$ is negative. This leads to the fact, that initial movement in the negative direction is slightly easier, and therefore faster, than in the positive direction. This also explains the course of the graph in Figure 6.1. This graph plots the difference in magnitude of ω between a positive and a negative θ_{VPIP}. This is done for 15° and 30°. We notice, that in the beginning (first 15 s) the negative ω is larger than the positive. This is also seen in Table 5.1, the negative acceleration is larger in magnitude compared to the corresponding positive (0.079745 rad/s^2 vs. -0.10044 rad/s^2 and 0.16257 rad/s^2 vs. -0.19244 rad/s^2). The magnitude of the final values though are very similar again (1.3707 rad/s vs. -1.3958 rad/s and 3.5476 rad/s vs. -3.4348 rad/s). They are within 0.2 rad/s of each other, which we see in Figure 6.1. All of these observations support our Theory **(1)**, that says that backward acceleration works in the same way as forward. To verify the statement about braking we use the PID controller (Section 6.2).
For low θ_{VPIP} we observed an oscillation in ω. This oscillation is correlated with the pitch of the robot θ_{robot}, visualized in Figure 6.2 for 10°, 15° and $-15°$ θ_{VPIP}. We see, that the θ_{robot} is highly correlated with ω, so if the robot moves forwards, it is also tilted forwards and vice

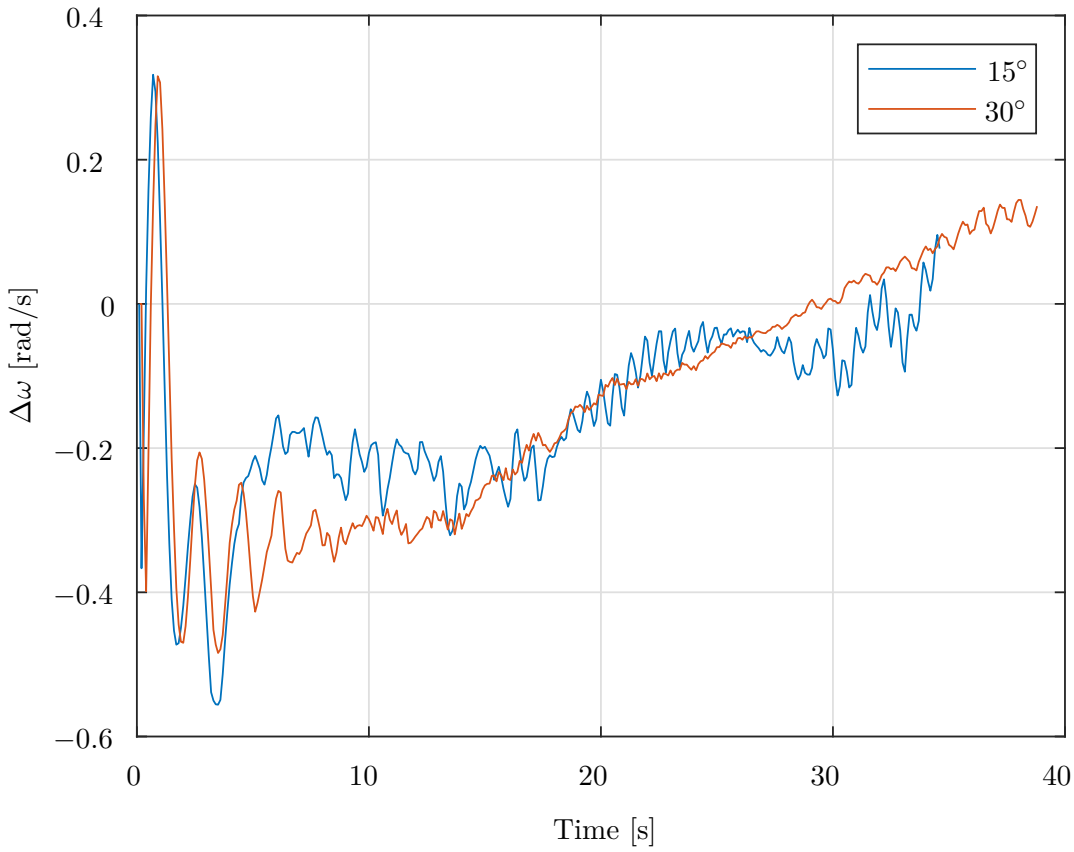
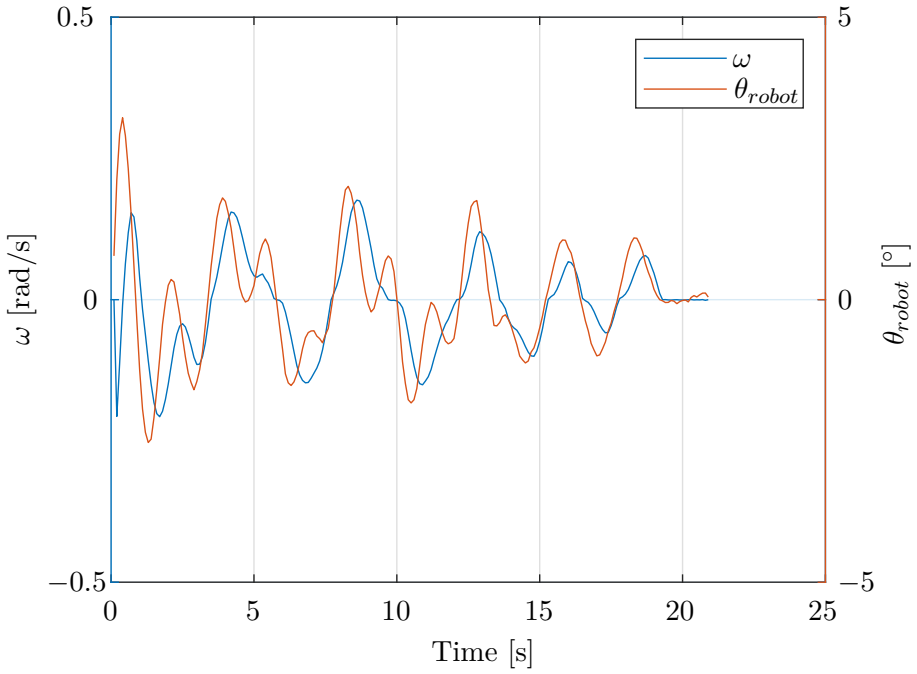
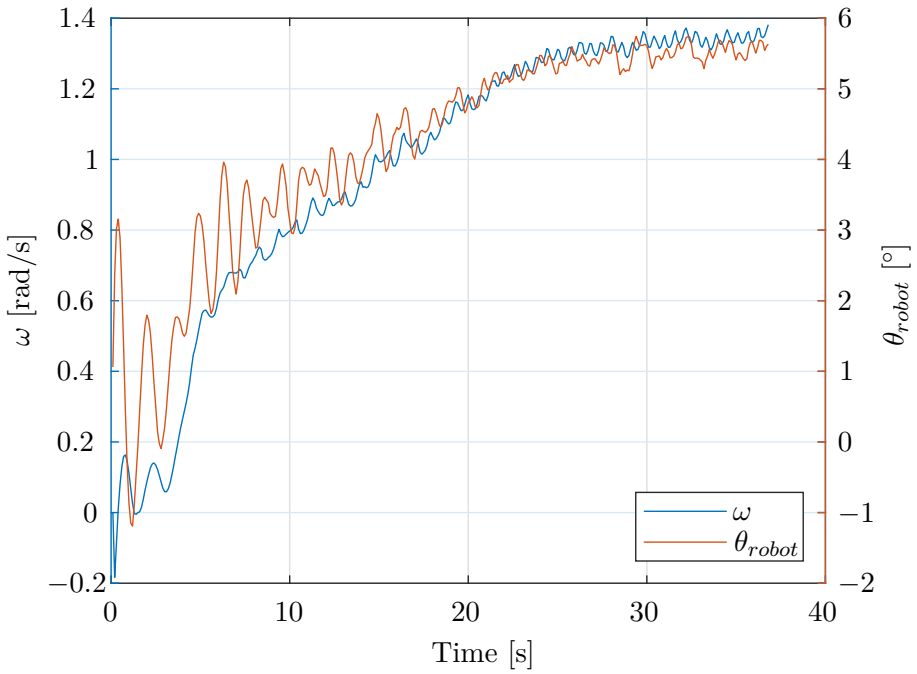


Figure 6.1: The difference in magnitude of rotational speed ω between the same positive and negative θ_{VPIP}.

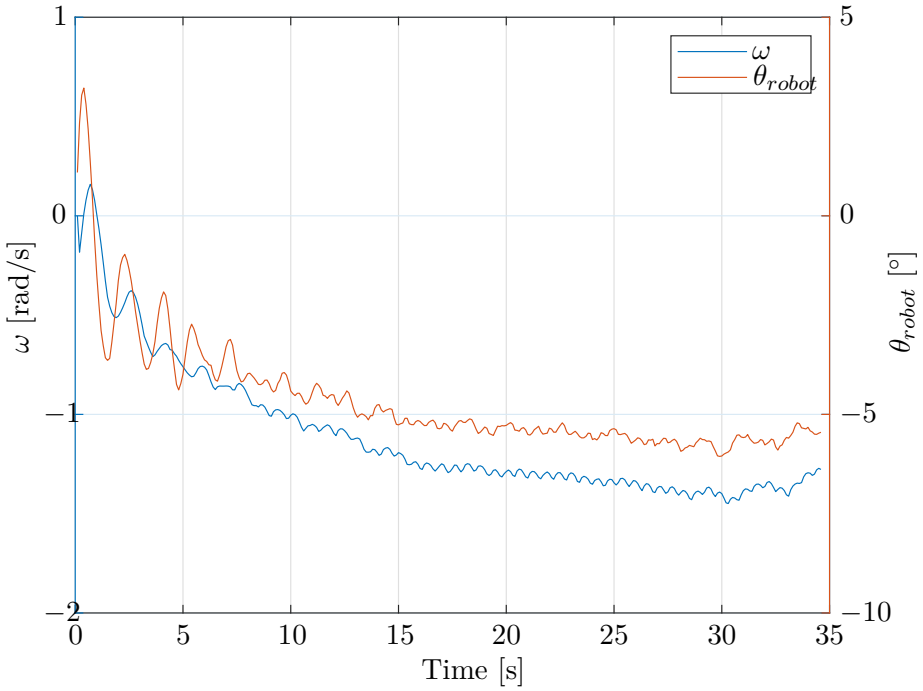
versa (cf. Figure 6.2b and 6.2c). We also notice, that the oscillations in θ_{robot} are shifted slightly forward compared to the ones in ω. We observe this the best in Figure 6.2a. This explains especially the oscillations for 10° of θ_{VPIP}, which are up to ±0.2 rad/s. As soon as the robot tilts forwards, it also starts to rotate forwards, which is then blocked by the rods, not moving out of the way, due to the low θ_{VPIP}. Then the robot tilts back, letting the robot rotate backwards. This cycle repeats over and over, resulting in the oscillations in ω seen both in Figure 5.2 and 6.2a. The smaller oscillations for 15° and −15° of θ_{VPIP} (under ±0.1 rad/s) have the same explanation, but the rotation of the robot is stabilized, by the movement itself, leading to lower and therefore faster oscillations.
Besides the oscillations at low θ_{VPIP} we observe similar oscillations at high θ_{VPIP}, that have the same effect on ω and θ_{robot}, but have a different cause. While the increase in rotational speed leads to a stabilization of θ_{VPIP} and therefore also ω itself, demonstrated by the fact, that there are virtually no oscillation for a θ_{VPIP} of 20° or 30° and much lower oscillation at 15° than at 10°, there are increasing oscillation for even higher values for θ_{VPIP}. The increase in oscillation is not limited to ω, but also the pitch θ_{robot} and roll ϕ_{robot} of the robot increases. This suggests a general unstableness of the system at such high θ_{VPIP}. We see this both in Figure 6.3 and



(a) $\theta_{VPIP} = 10°$

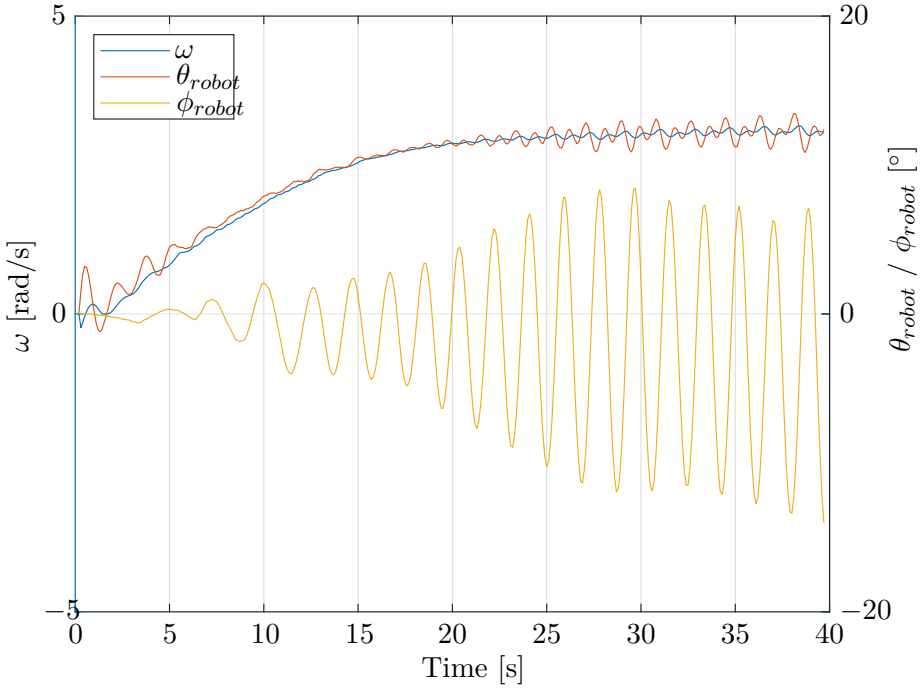


(b) $\theta_{VPIP} = 15°$

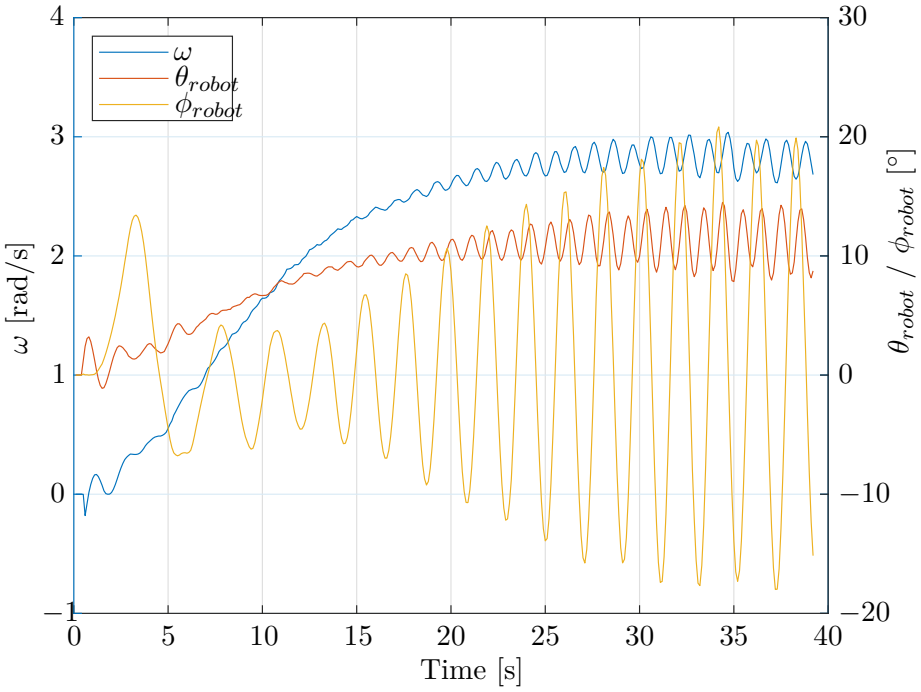


(c) $\theta_{VPIP} = -15°$

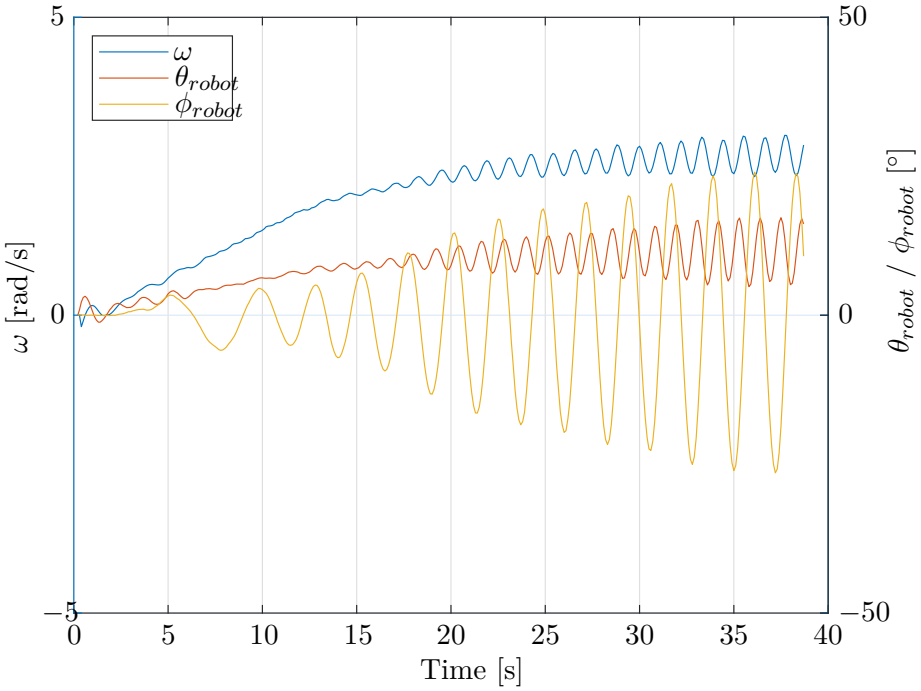
Figure 6.2: The oscillations of ω, once the robot reaches its final speed, at low θ_{VPIP} are a result of the oscillating pitch of the entire robot θ_{robot}.



(a) $\theta_{VPIP} = 40°$



(b) $\theta_{VPIP} = 50°$



(c) $\theta_{VPIP} = 60°$

Figure 6.3: The oscillations of ω, once the robot reaches its final speed, at high θ_{VPIP} are a result of the oscillating pitch θ_{robot} and roll ϕ_{robot} of the entire robot, due to general unstableness.

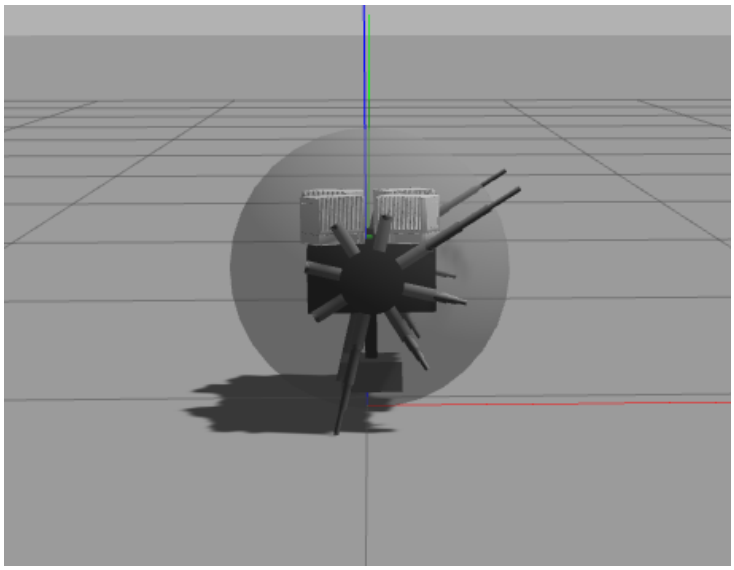


Figure 6.4: A TLDR robot in the simulation with a θ_{VPIP} of 60°. There is only one pair of rods pushing into the ground, thus creating little torque and offering little support.

Table 5.2. The oscillation of θ_{robot} are again highly correlated, to the ones we see for low θ_{VPIP}. The oscillation in ω are therefore again explained by the ones in θ_{robot}, but the high ω is no longer sufficient to stabilize them. This is were the oscillations in ϕ_{robot} come into play. Their frequency is exactly half that the oscillations in ω and θ_{robot}. Whenever the magnitude of ϕ_{robot} is large – so when the robot is tilted to either side – θ_{robot} is above its average value, although it again is shifted slightly backwards, due to some reaction time. During the time the robot tilts to the other side, it is also tilted backwards, relative to its average tilt, which isn't 0°, because the robot always tilts forwards, when moving forwards. So the oscillation in θ_{robot} and therefore in ω are caused by the oscillations in ϕ_{robot}. This suggests very low stability of the system at high θ_{VPIP}. In Figure 6.4 we see, that only one pair of rods actually touch the ground, due to the extreme angle of 60° for θ_{VPIP}. This one pair of rods only provide very little support to either side, thus allowing the robot to tilt to either side, when also tilting forwards. Only when the robot tilts over significantly (how much exactly depends on θ_{VPIP}, cf. Table 5.2), tilts it back and then to the other side, thus creating the oscillations in ϕ_{robot}.
From Table 5.1 we gather that a θ_{VPIP} of 40° has the largest initial acceleration, while 30° leads to the largest final rotational speed. This confirms our Theory **(2)**, that there is a maximum useful angle for θ_{VPIP}. Therefore the output of all closed-loop controllers are limited to a θ_{VPIP} of 50°. More precisely there are two different maximum useful angles, one $\sim$ 40° for maximum acceleration and a second one $\sim$ 30° for maximum ω. This effect is also explained by Figure 6.4, as there is only one pair of rods pushing into the ground, there is also less torque created, compared to more rods pushing into the ground but with less force. Why the maximum acceleration angle and the maximum speed angle are different is unclear, but there is also no reason for them to be the same, as acceleration and top speed are different concerns.
Also from Table 5.1 we gather, that a θ_{VPIP} of 10° has no effect on the long term speed of

the robot. We therefore need an angle $\theta_{VPIP} > 10°$ for θ_{VPIP} to get the robot moving, thus confirming part of Theory **(3)**. Whether a $\theta_{VPIP} > 0°$ is also necessary to keep moving is examined with an closed-loop controller (cf. Section 6.2).
Looking at Table 5.1 at a whole, we see that a certain θ_{VPIP} is both associated with a specific acceleration and a final ω, though the acceleration is more relevant for lower rotational speeds. Thus a closed-loop controller is able to make use of the different accelerations to reach and then hold a specific desired rotational speed ω_{des}. Whether it achieves this faster than an open-loop controller is discussed in Section 6.2, so Theory **(4)** is for now only partly confirmed.
Finally looking at Figure 5.2 we notice, that the robot is very inert. It takes multiple seconds before the robot reacts to any change in θ_{VPIP} and it takes over 20 s before the robot hits its final speed, no matter what θ_{VPIP} we choose.

6.2 PID controller

6.2.1 P tuning

Both in Figure 5.3 and Table 5.3 we see, that both acceleration and braking in test run (a) see only little difference, when changing k_p. This is because for the initial acceleration and deceleration θ_{VPIP} is at $\pm 50°$ anyways, no matter what k_p is. Therefore the resulting $\dot{\omega}$ doesn't change significantly. But we do notice, that braking is considerably slower than accelerating. This is because, even though they are initiated the same way, by changing θ_{VPIP}, the actual process is somewhat different. During acceleration rods at the back side of the robot extend, thereby push into the ground, thus creating torque. But during braking the relevant rods are at the front of the robot retracting, essentially being squished between the robot and the ground, which also creates torque in the opposite direction. The extending of the rods and thereby pushing into the ground is more efficient than the compression. Therefore acceleration is faster than braking. This means, that our Theory **(1)** is partly correct, as backwards movement works in the same way as forward movement, but braking is slower than accelerating, but doesn't need any special control.
We also notice both in Figure 5.3 and Table 5.3, that there remains a significant steady-state error, which decreases when increasing k_p, but doesn't vanish. This verifies Theory **(6)**, that states we need an I part to get rid of the steady-state error. Because an error remains, there also remains a θ_{VPIP} of $\sim 10°$ in all of the test cases, thereby confirming Theory **(3)**, which states, that an angle $\theta_{VPIP} > 0°$ is not only necessary to start moving but also to keep moving. If $0°$ is enough to keep on moving, there wouldn't be a steady-state error remaining. The magnitude of the remaining steady-state error at $\omega_{des} = -1$ rad/s is the same as the corresponding one at $\omega_{des} = 1$ rad/s, therefore once again supporting Theory **(1)**.
The first 30 s of test run (b), shown in Figure 5.4, show the same response, as up that point test run (a) and (b) are identical. The braking in test run (b) is much faster for the lower k_p gain of 40, seen in Table 5.4. Because the error at the beginning of the braking process is much lower than in test run (a), θ_{VPIP} doesn't reach $-50°$. Though this is only true for $k_p = 40$. Apparently braking is more efficient, when using a more gradual approach, than an extreme value of $-50°$ for θ_{VPIP}. This fast braking leads to a overshoot of 0.3 rad/s, which in addition to the general unsteadiness of the system at low ω explored in Section 6.1, leads to the

oscillations. The P controller reduces the oscillations only slowly, as the impact of small changes on an an already small $\theta_{VPIP} < 10°$ is very limited (cf. Section 6.1). For the larger k_p gains the braking is slower, so the overshoot is smaller, thus the oscillations are smaller and decay faster.

6.2.2 I tuning

The similar acceleration and braking speeds compared to the P controller (Figure 5.5 vs. Figure 5.3), are explained by the fact, that θ_{VPIP} is at $\pm 50°$ no matter whether we use a P or a PI controller. The positive steady-state error at $\omega_{des} = 1$ rad/s is much smaller than with the P controller (Table 5.5 vs. Table 5.3). k_i gains of 0.5 and 1 still have a persisting steady-state error greater than 0.1 rad/s, but k_i gains of 2 and 5 get rid of the steady-state error entirely. This supports both our Theory **(6)**, that an I part is necessary to get rid of the steady-state error, and Theory **(3)**, as even with no error the integral leads θ_{VPIP} to be greater than 0°, to achieve constant forward motion. In the steady-state error at $\omega_{des} = -1$ rad/s and the Figure 5.5 we see the effect that integral windup has on ω. While the low k_i gain of 0.5 and the corresponding integral windup coincidentally match up perfectly, the other responses are worse. For a k_i gain of 1 there is an overshoot of 25%, that doesn't get cancelled in a timely manner, due to k_i still being small. At $k_i = 2$ the overshoot increases to 60%, but is cancelled after a further 35 s, leaving us with no steady-state error. Finally a k_i gain of 5, which showed a good result for the initial step response, leads to a overshoot of more than 150%. This is to large, leaving the robot in an uncontrollable state. So even though this implementation of the I part shows good results for the initial step response, we have to adapt it to make it a useful controller. We do this by implementing a limit to the integral (Section 5.2.3).

6.2.3 I tuning with Integral limit

The first 30 s of the test runs (both (a) and (b)) show the same characteristics as described in Section 6.2.2. The steady-state error at $\omega_{des} = 1$ rad/s for $k_i = 1$ is slightly above 0.1 rad/s, while it is essentially 0 rad/s for the higher k_i gains, gathered from Table 5.6. The second half of the response to test run (a) changes to the better, as seen in Figure 5.6. The limit on the integral lowers the effect that integral windup has on ω drastically. The magnitude of the remaining steady-state error at $\omega_{des} = -1$ rad/s matches the one at $\omega_{des} = 1$. Only the highest tested k_i gain of 5 still shows an overshoot – now just 20% –, while the other responses show no overshoot at all. Clearly the integral limit has a large improvement over the PI controller without one.
The impact of the integral limit on test run (b) is much lower. As explained in Section 6.2.1 the braking is slightly slower than in the response to the P controller with a k_p gain of 40 (Figure 5.4), due to the integral increasing the overall gain. Therefore the oscillations are also lower and decay faster but generally remain, due to the unstableness of the system explored in Section 6.1.

6.2.4 D tuning

The fact that the course of ω in Figure 5.8 has virtually no difference to the one using a PI controller (cf. Figure 5.6) shows, that this implementation of the D part has no effect on the robot. This because this implementation of the D part is flawed, which also explains the

large oscillations in θ_{VPIP}. As ω has small but fairly fast oscillations thorough all test runs, the difference between the last and second to last error, has a large effect on θ_{VPIP}. This explains the large oscillation and why there is no impact on the robot, which due to its high inertness automatically filters θ_{VPIP} and acts as if the fast oscillations aren't there. To implement a useful D part we must do the same, which we do in Section 5.2.5.

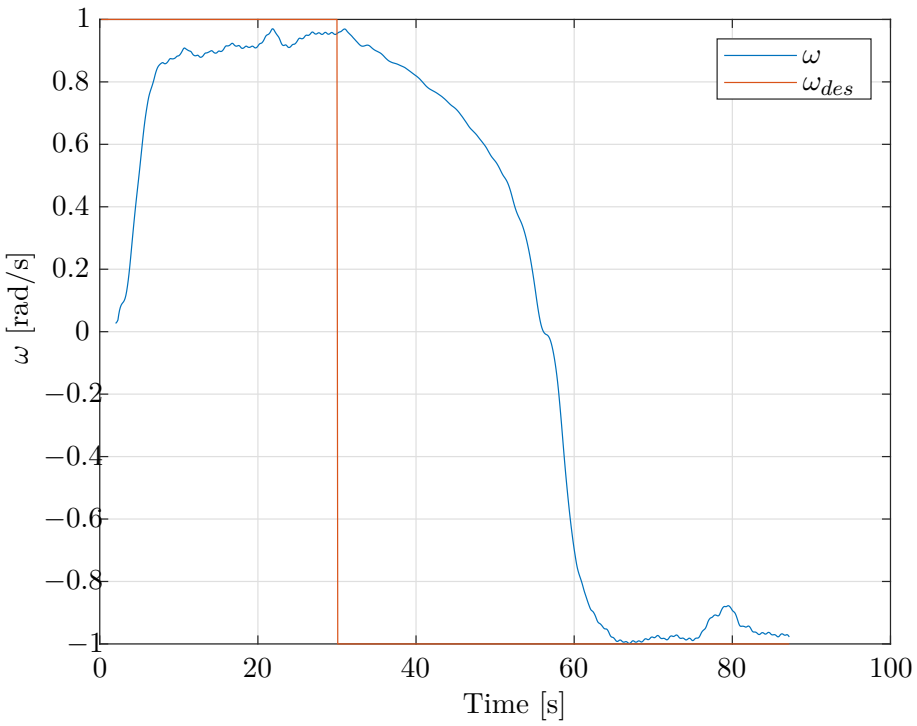
6.2.5 D tuning with filter

Both Figures 5.9 and 5.10 show a decreased acceleration and braking for higher k_d gains compared to the PI controller (Figures 5.6 and 5.7). This is expected, as the k_d gain works against a large change in the control error. Therefore the responses slow down. The increased oscillations for higher k_d especially 100 are caused by the D part. As explained in [1] a to large k_d gain, leads to an increase in oscillations as the system overcorrects due to the high k_d gain. This is what happens, increasing the unavoidable high frequency oscillations originally caused by the systems instability at low ω. So a high k_d gain increases the high frequency oscillations, whereas a low k_d gain, e.g. 20, has no effect on the robot. This confirms our Theory **(5)**, showing that no D part is necessary to control the robot. We also see this illustrated, if we ignore the high frequency oscillations by plotting the rolling average over the last two seconds of the response of ω to a PI controller in Figure 6.5. We notice, that no low frequency oscillations, against which the D part is effective, exist, supporting Theory **(5)**.

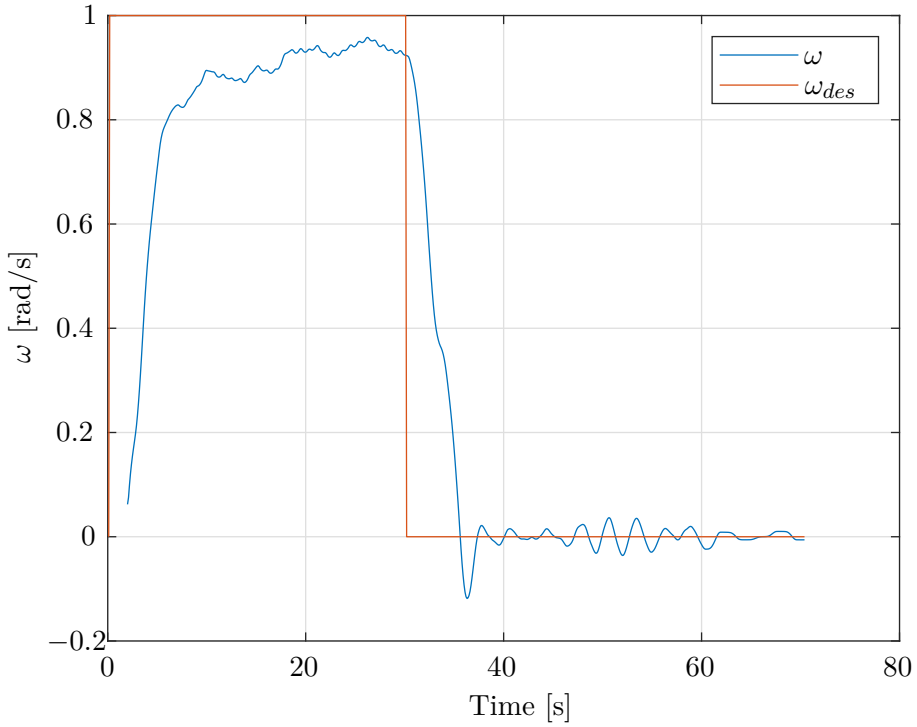
Tuning the PID controller, we saw the best results for a PI controller, with a k_p gain of 40 and a k_i gain between 2 and 5 depending on whether minimal overshoot or faster response times are desired. We also saw rise times to 1 rad/s of slightly above 5 s, in line with the accelerations of ideal θ_{VPIP} (40°), but not achievable using an open-loop controller with the aim of 1 rad/s, which leads to a $\theta_{VPIP} < 15°$. This supports Theory **(4)**, by showing that a closed-loop controller achieves a ω_{des} much faster. Finally we also demonstrated the ability to reach and stay at a given ω_{des}.

6.3 LQR

Looking at Equation 5.9 we notice, that due to the simplifications and assumptions we made the LQR is a P controller with a constant offset. Therefore we expect the response to be similar to the one of the P controller. Looking at Figures 5.11 and 5.12 vs. Figures 5.3 and 5.4 we see that this is in fact the case. Though using the LQR the steady-state error is much smaller than in the corresponding P controller case (Table 5.7 vs. Table 5.3). This is due to the constant offset added to θ_{VPIP} when using the LQR. Both the oscillations around 0 rad/s in test run (b) and the fact, that higher gains lead to slower braking, have the same reasons as discussed in Section 6.2.1. So even though the LQR shows an improvement over the P controller, it shows no improvement over the PI controller (cf. Figure 5.6 and 5.7). This is expected using the highly simplified state space model we used to set up the LQR. Further research has to be done creating a more complex and detailed model of the workings of the VPIP and the TLDR robot. With such a model a more complex LQR will be set up, giving it the possibility to create a more desirable system response.



(a) test run (a)



(b) test run (b)

Figure 6.5: The rolling average over the last two seconds of ω for the response to the test runs using a PI controller with $k_p = 40$ and $k_i = 2$.

Chapter 7

Conclusion and Future Work

This thesis set up, implemented and tested various control strategies for the VPIP. Starting with the open-loop control, we set the pitch of the VPIP θ_{VPIP} to various constant values, looking at the rotational speed of the robot. We saw, that the VPIP works as a locomotion approach for a TLDR robot and that changing θ_{VPIP} both changes the acceleration and final top speed of the robot. The highest rotational speed of 3.5 rad/s was reached with a θ_{VPIP} of 30°, while the greatest acceleration of 0.19 rad/s^2 was achieved using a θ_{VPIP} of 40°. For any higher angles we saw an decrease in effectiveness and in lateral stability, making any θ_{VPIP} of more than 50° undesirable. We also noticed, that there is a minimum angle of $\sim$ 10° to get the robot rolling and keep it moving with a constant velocity. We also confirmed, that the VPIP works in the same way backwards as it does forwards, although braking is slower than acceleration. For optimal braking we noticed, that θ_{VPIP} should change gradually and not jump to an extreme value at once. We also tested various PID controllers, with different k_p, k_i and k_d gains. As we expected from the fact, that an angle larger than 0° is necessary to keep moving, a P controller leaves us with a considerable steady-state error. We eliminated the steady-state error by introducing an I part to the controller, where we saw the effects of integral windup. To beat this we introduced a limit to the integral of 5 rad, which left us with the overall best system responses for a k_p gain of 40 and a k_i gain of somewhere between 2 and 5, depending on whether minimal overshoot or faster response times are desired. Introducing a D part to the system didn't have a positive effect, as the system is highly inert anyways, leaving us with no low frequency oscillations, that can be controlled with a D part. The oscillations, that we did see, are all due to the lower stability of the robot at lower rotational speeds. Finally we set up a LQR. To do this we simplified the system quite drastically, so the resulting LQR was simply a P controller with a constant offset. Even tough this produced better results than the P controller with significantly lower steady state errors, it didn't preform as good as the PI controller, which is more adaptive using an I part instead of the constant offset to θ_{VPIP}.
Of course there is still much future work to do. Further research has to be done into setting up a more precise mathematical representation of the entire system. This then allows the creation of a more detailed LQR, which is necessary to create an alternative to the PI controller. A Quadratic-Quadratic Regulator as described in [2] can also be used instead of an LQR, so that the system doesn't have to be linearized. Although the oscillations at $\omega = 0$ rad/s are limited,

they have to be reduced, by either further tuning the gains of the PID controller, or potentially using a better suited controller. There also has to be research into steering with the VPIP. This is done by changing the roll of the VPIP ϕ_{VPIP}, which same as with θ_{VPIP} also has to be controlled. When these two controllers are combined, we will have a control strategy, that allows us to control both the rotational speed of the robot as well as its orientation. We will then go on testing the TLDR robot in more complex situations, like on uneven terrain or at ridges, which requires further development of the VPIP. For this the Virtual Pose Instruction Map (VPIM) introduced in [12] is used. Instead of using a virtual plane, we use a virtual map, that represents the surrounding terrain. In doing so the calculation of the length of the rods also takes into account the terrain surrounding the robot, so the controllers for the VPIP have to be readjusted and verified to also work for the VPIM. Finally everything that is tested in a simulation also has to be tested using a real prototype of the TLDR robot. So even though this thesis showed promising results for the control of the rotational speed of the robot using a closed-loop controller, much is still to be done.

Bibliography

[1] KL Åström and T Hägglund. Pid controllers: Theory, design and tuning. *Instrument Society of America*, 1995.

[2] Jeff Borggaard and Lizette Zietsman. The quadratic-quadratic regulator problem: Approximating feedback controls for quadratic-in-state nonlinear systems. In *2020 American Control Conference (ACC)*, pages 818–823, 2020.

[3] Cassandra R. Coombs and B. Ray Hawke. A Search for Intact Lava Tubes on the Moon: Possible Lunar Base Habitats. In Wendell W. Mendell, John W. Alred, Larry S. Bell, Mark J. Cintala, Thomas M. Crabb, Robert H. Durrett, Ben R. Finney, H. Andrew Franklin, James R. French, and Joel S. Greenberg, editors, *Lunar Bases and Space Activities of the 21st Century*, page 219, September 1992.

[4] ESA. ESA plans mission to explore lunar caves. `https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/ESA_plans_mission_to_explore_lunar_caves`.

[5] ESA. SysNova. `https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/SysNova2`.

[6] ESA. The Open Space Innovation Platform (OSIP). `https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/The_Open_Space_Innovation_Platform_OSIP`.

[7] ESA. Seeking innovative ideas for exploring lunar caves. `https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/Seeking_innovative_ideas_for_exploring_lunar_caves`, 2019.

[8] Ronald Greeley. Lava Tubes and Channels in the Lunar Marius Hills. *Moon*, 3(3):289–314, December 1971.

[9] Kedus Mathewos. Simulation and Examination of Rod-Based Locomotion Options for Spherical Robots, 2022.

[10] Angelo Pio Rossi, Francesco Maurelli, Vikram Unnithan, Hendrik Dreger, Kedus Mathewos, Nayan Pradhan, Dan-Andrei Corbeanu, Riccardo Pozzobon, Matteo Massironi, Sabrina

Ferrari, Claudia Pernechele, Lorenzo Paoletti, Emanuele Simioni, Pajola Maurizio, Tommaso Santagata, Dorit Borrmann, Andreas Nüchter, Anton Bredenbeck, Jasper Zevering, Fabian Arzberger, and Camilo Andr´es Reyes Mantilla. DAEDALUS - Descent And Exploration in Deep Autonomy of Lava Underground Structures. Technical Report 21, Institut für Informatik, 2021.

[11] Russ Tedrake. *Underactuated Robotics*, chapter 8. 2023.

[12] Jasper Zevering. TLDR Robot Telescopic Linear Driven Rotation Robot - A Locomotion Approach for Spherical Robots, 2021.

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Würzburg, March 2023