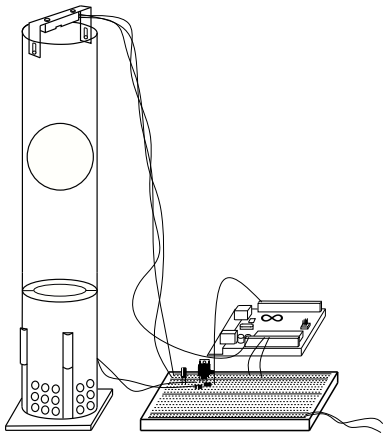
Control Theory Hands-on Practical

Hovering a table tennis ball at a fixed height

Dr. Tijs W. Alleman[1,2,*], Daniel Illana[1], Dr. Gauthier Vanhaelewyn[1], Dr. Elena Torfs[1,3]

July 1, 2024

[*]Correspondence: `tijs.alleman@ugent.be`
[1]BIOMATH, Department of Data Analysis and Mathematical Modelling, Ghent University
[2]Infectious Disease Dynamics Group, Department of International Health, Johns Hopkins University Bloomberg School of Public Health
[3]modelEAU, Department of Civil Engineering and Water Engineering, Université Laval
Available on GitHub: `https://github.com/twallema/hovering-table-tennis-ball`



Aim In this practical, we turn control theory into practice by hovering a table tennis ball at a fixed height in a transparent tube. We use an Arduino Uno to build a closed loop feedback controller. We measure the ball's height using an infrared distance sensor at the top of the tube, which we the use to control a fan at the bottom of the tube. First, we study the setup's open-loop behavior and gather response data to calibrate a mathematical model. We then use Matlab's Simulink environment to perform an in-silico tuning of several closed-loop feedback controllers, such as the Proportional-Integral-Derivative (PID) controller and Linear-Quadratic Regulator (LQR) state feedback controller. Finally, the performance of the tuned controllers is examined and compared using the experimental setup. This setup serves as an intuitive and visual test bed for the closed-loop feedback controllers introduced in the theoretical course. Our aim is to introduce students to the use of electronics, the unfortunate reality of noisy sensors and dead time, along with their implications on closed-loop stability, the application of control theory to non-linear systems, and to the use of mathematical modeling to support the in-silico design of a suitable controller.

Contents

1 Introduction

1.1 Electronics

To build the practical setup, we first need to familiarise ourselves with the bread and butter of electronics: the resistor, capacitor, diode and transistor. In this short intro we use the hydraulic circuit analogy, to explain some basic concepts. In this analogy, an electrical circuit is thought to be a hydraulic circuit consisting of pipes, valves, pumps, etc. Charge is represented by the water amount, voltage is represented by the water pressure, and current is represented by the water flow.

Resistor As the name implies, resistors add resistance to the circuit and reduce the flow of electrical current. In the hydraulical circuit analogy a resistance is represented by a throttling valve, or alternatively, as rocks in a river, hindering the flow of water. Resistors are typically represented as a pointy squiggle with a value next to it (Fig. 1). In electronic devices, resistors are used to avoid damage resulting from high currents. When building an electronic circuit, it is paramount to check the amount of current flowing through each analog component by using Ohm's law,

$$I = \frac{V}{R}, \tag{1}$$

where V is voltage in Volt (V), I is the electric current in Amps (A) and R is the resistance in Ohms (Ω). Most electronic components, f.i. an LED, have little internal resistance. When applying even the smallest voltage directly to an LED, the LED will light up briefly and then shutdown forever. Generally speaking, overloading electronic circuits by mistake is common and not very dangerous. **In any case, when noticing the characteristic smell of a shortcircuit (a combination of ozone and burnt plastic) during this practical, please pull out the wall plug immediately!** During this practical we will use a 2.2 $k\Omega$ resistor to limit the current of a voltage sent from the Arduino Uno to a transistor.

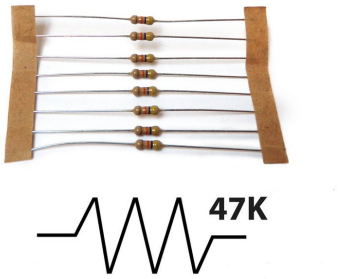


Figure 1: (upper) real life resistor. (lower) schematic representation of a transistor.

Capacitor A capacitor is a component that stores electricity and then discharges it into the circuit when there is a drop in current. You can think of a capacitor as a water storage tank that releases water when there is a drought to ensure a steady stream. Therefore, capacitors are typically used to filter a noisy current. Further, capacitors are preferred over batteries in applications that require a rapid discharge, such as in a defibrillator. The values that you will typically

encounter in most capacitors are measured in picofarad (pF), nanofarad (nF), and microfarad (μF). The most commonly encountered types of capacitors are ceramic disc capacitors that look like tiny M&Ms with two wires sticking out of them and electrolytic capacitors that look more like small cylindrical tubes with two wires coming out the bottom (Fig. 2). The main difference is that ceramic disc capacitors are non-polarized, meaning that electricity can pass through them no matter how they are inserted in the circuit. Opposed, electrolytic capacitors are polarized, meaning one leg needs to be connected to the ground side of the circuit and the other leg must be connected to power. For this practical we will be using a 22 μF electrolytic capacitor as an analog filter for the power supply of the IR proximity sensor. **Because the provided capacitor is electrolytic, please make sure you respect its polarity.**

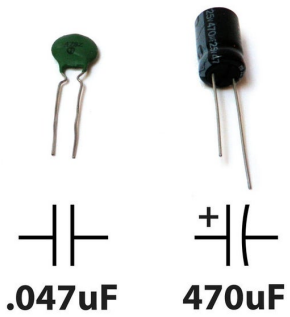


Figure 2: (left) real life ceramic capacitor and schematic representation (right) real life electrolytic capacitor and schematic representation.

Diode A diode only allows electrical current to pass through them in one direction, its hydraulic analogy is a non-return valve. Diodes are useful to prevent electricity from flowing in the wrong direction. When the voltage applied to an electric DC motor is abruptly switched off, the fan will keep spinning for a short while. As such, the motor will act as a generator of current until its blades come to a standstill. Charge will build up on what was the negative terminal of the motor and this charge will want 'a way out'. Adding a diode in parallel with the motor provides a path for dissipation of stored energy when the voltage is cut. Similar to the electrolytic capacitor, the diode is polarised and the ring found on one end of the diode indicates the side of the diode which connects to ground. **When building the setup, please mind that the ring on the diode points in the right direction (Fig. 12).**

Transistor A transistor can act as a switch or gate for electronic signals, opening and closing many times per second. It typically has three pins: the base, emitter and collector pins (Fig. 4). A transistor uses a small electrical current at its base pin to control the current passing between its collector and emitter pins. The amount of current allowed to pass between the collector and emitter pins is proportional to the voltage being applied at the base pin. A transistor is therefore a good way of controlling a high voltage/high current circuit (fan motor, 12 VDC) using a low voltage/low current signal (Arduino, 5 VDC). An (inperfect) hydraulic analogy for a transistor is an adjustable valve. Passing current through a transistor costs energy and therefore observing a

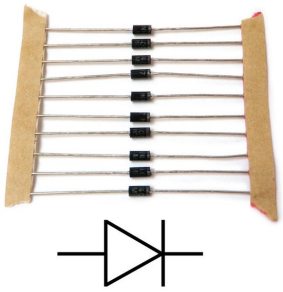


Figure 3: (upper) real life diode (lower) schematic representation of a diode.

voltage drop of 1 V over a transistor is common. This energy is dissipated in heat so **transistors heat up to well over 100 °C under load. Never touch your transistor while operating the setup!**

Figure 4: BD537 transistor used to control the fan's motor.

1.2 The Arduino Uno Rev 3, its components and signals

Using only resistors, capacitors, diodes and transistors, it is possible to build simple electronic devices that do a specific job, such as an egg timer or a security buzzer that goes off when you keep the fridge open too long. The behavior of such circuits is solely based on the chemical properties of the components, which need to be computed beforehand to make sure that the right amount of current is flowing through the right components. This makes these simple devices very good at their specific job, but they're limited in what they can do. The microcontroller, a tiny computer present on the Arduino Uno can be used can be used in conjunction with the aforementioned analog circuitry to allow for more advanced behavior. The amount of current flowing through the analog components is now variable and controlled, rather than constant and computed beforehand.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. The Arduino hardware consists of a small circuit board containing a microcontroller chip and several connections (Fig. 8), allowing us to read inputs such as light on a sensor, a finger on a button, or even a Twitter message - and turn them into an output - activating a motor, turning on an LED, publishing something online. This is accomplished by telling your board what to do when certain

inputs are encountered by sending a set of instructions to the microcontroller on the board. To do so you can use the Arduino Software (IDE), as well as other programming languages such as Python and Matlab. In this practical, we'll program our Arduino Uno using the Matlab's Simulink environment. Let's examine the Arduino's components closer.

Microcontroller You can think of the microcontroller chip itself as the *brains* of the board. The chip used in the Arduino Uno is the ATmega328p, made by Atmel. It's the large, black component in the center of the board. It's actually not alone but rather sits in a socket. If you were to remove the microcontroller, it would look like the one shown in Figure 5.

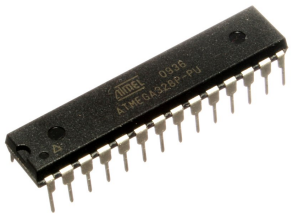


Figure 5: The brain of the Arduino Uno R3: the Atmel ATmega328p microcontroller.

Header sockets The microcontroller socket connects all 28 legs of the ATmega328 microcontroller chip to other sockets, referred to as header sockets, which have been arranged around the board and labeled for ease of use. They are the black sockets that go around the edge of the Arduino board. These are divided up into three main groups: digital pins, analog input pins, and power pins. All these pins transfer a voltage, which can either be sent as output or received as an input. These pins are important as they allow additional circuitry to be connected to the board quickly and easily.

Power pins You use the power pins to distribute power to inputs and outputs wherever it's needed (Fig. 8). GND marks the ground pins, which are essential to complete circuits. There is also a third ground by pin 13. All these pins are linked and share the same *common* ground. You can use the 5V and 3.3V pins to supply power to components or circuits. However, the Arduino can only supply about 40 mA of current without getting damaged. This is enough to power most sensors but not enough to power a larger motor. We'll thus have to supply the power for the blower fan externally.

Digital pins Signals are defined as time-varying *quantities* conveying information, typically a voltage. So when we talk about signals, just think of them as a voltage that's changing over time. You use the *digital pins*, which run across the top of the board (Fig. 8), to **send and receive** digital signals. Digital signals are made up of a sequence of *bits*. A bit represents only one of two states: off or on, 0 or 1, or in electrical terms, 0V or 5V, but no values in between. The amount of information conveyed by a digital signal is characterised by its number of bits. On the Arduino Uno, a digital signal has 10 bits, giving rise to a total of 1024 possible outcomes, corresponding to a

number between 0 and 1023. A temporal graph of a digital signal looks like a series of square waves (Fig. 6). In practice, ten bits means the Arduino Uno and your computer are communicating by sending out ten equally long pulses of 0V or 5V within a set time interval. F.i. the computer will send the Arduino Uno a sequence of 10 bits (zeros or ones) in 1 millisecond, implying each bit is 0.100 milliseconds in length. Then there may be a small pause, say 1 millisecond, after which the Arduino Uno gets 1 millisecond to send the computer ten zeros or ones. These timings typically obey a certain protocol, used globally, such as HDMI and USB serial.

Analog pins These pins are used to **receive** analog signals (Fig. 8). The fundamental difference between an analog signal and a digital signal is that an analog signal is continuous and can assume all values between 0V or 5V (Fig. 6). However, when receiving a voltage between 0 V and 5 V at pin A0-A5, the voltage is automatically converted to a digital value by the Arduino's on-board analog-to-digital converter before it is sent to your computer. **Therefore, both analog and digital inputs are represented by a number between 0 and 1023 on your computer, and therefore the maximum resolution of any signal on an Arduino Uno is 1/1024**.

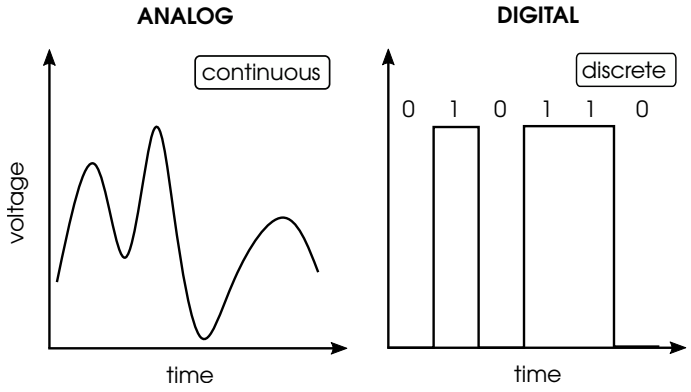


Figure 6: The fundamental difference between an analog and a digital signal. A continuous analog signal (left). A discrete 6-bit digital signal (right).

Pulse Width Modulation The very shrewd ones among you may have noticed that there seem to be **no pins for analog outputs**. In fact, there are, but they're hidden among the digital pins marked as PWM using the "~" symbol. PWM stands for *Pulse Width Modulation*, which is a technique you can use to mimic a continuous analog output using a discrete digital output. A digital output is either on or off but this can be done extremely quickly. If the output is on half the time and off half the time, it is described as having a 50 percent *duty cycle*. Because it is blinking faster than the human eye can perceive, an LED pulsed using a 50 percent duty cycle looks as though it is at half brightness (Fig. 7). With a DC motor as an output, a 50 percent duty cycle has the effect of moving the motor at half speed. So in this case PWM allows you to control the speed of a motor by pulsing it at an extremely fast rate. PWM works best for devices with sufficient inertia such as electric motors and pumps.

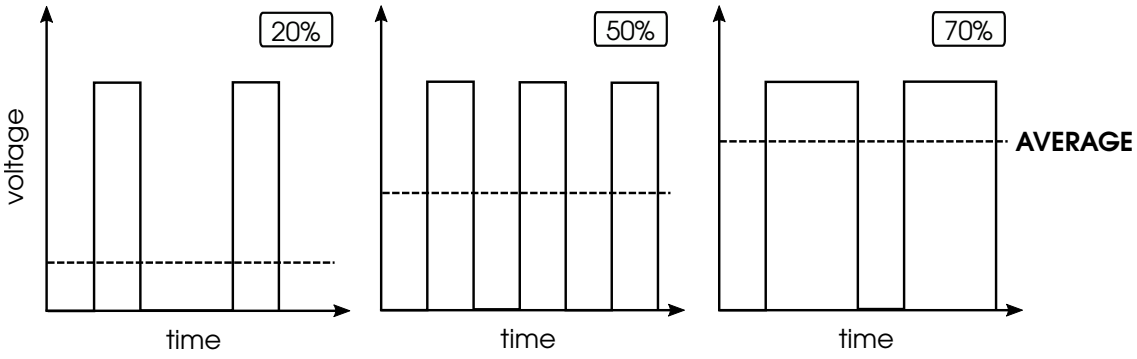


Figure 7: From left to right: A 20 %, 70 % and 90 % PWM duty cycle.

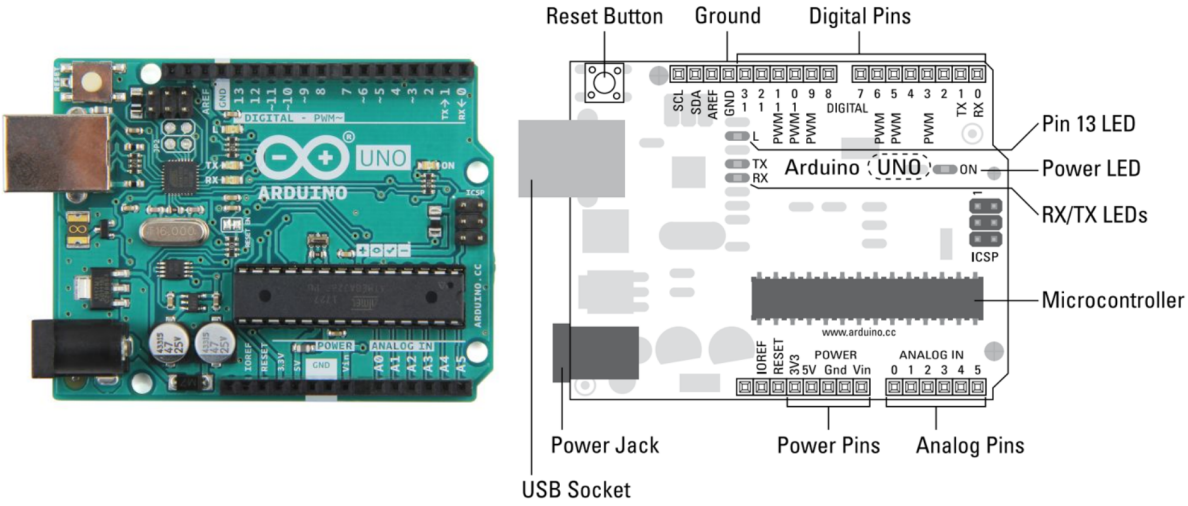


Figure 8: An actual Arduino Uno R3 (left) and a schematic highlighting the most important parts (right).

2 Practical setup

2.1 Materials

- Arduino Uno R3
- Male-male jumper wires
- Solderless breadboard
- Sharp 2Y0A21 IR proximity sensor
- 2.2 kΩ resistor
- BD139 or BD537 transistor
- Diode
- 22 μF electrolytic capacitor

- Sunon 40x40x10 mm 5 VDC blower fan
- Fan stand, obtained from `https://www.instructables.com/Float-a-Ping-Pong-Ball-at-Will`
- 12 VDC power supply
- Transparent polycarbonate tube, 65 cm long
- USB-A to USB-B cable
- Table tennis ball

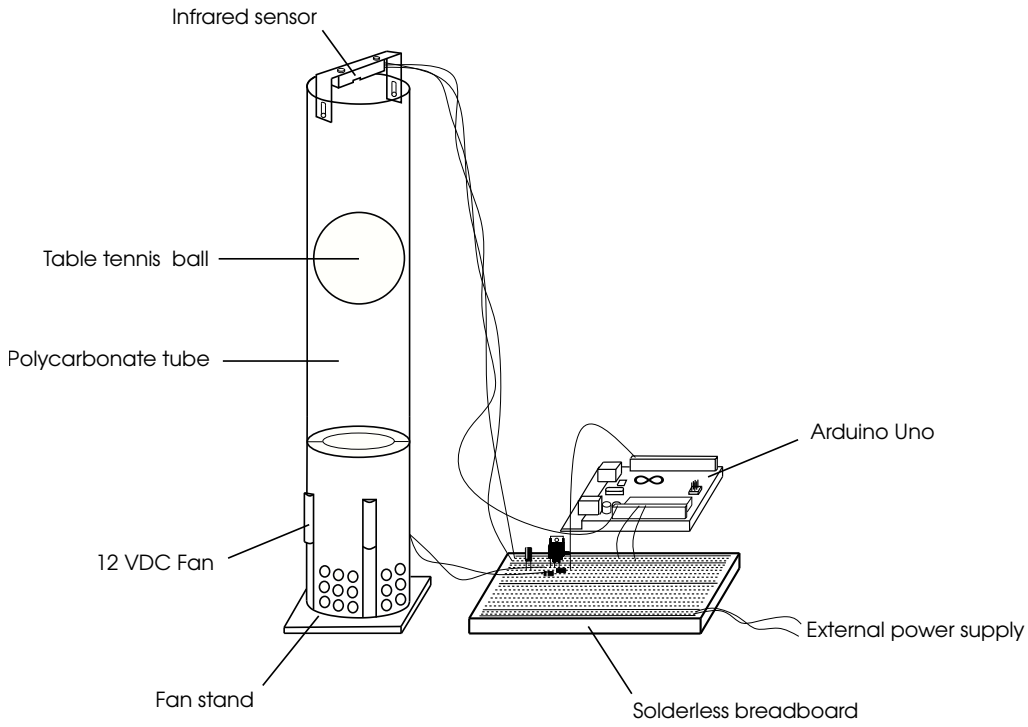


Figure 9: Schematic of the practical setup.

2.2 Electronic circuit

The electronic circuit diagram depicted in Fig. 11 is built on a solderless breadboard (Figs. 10, 12). A solderless breadboard consists of two halves split by the center divider and are electrically independent. Each half consists of roughly 50 terminal strips (5 hole columns) and two power rails. The power rails are horizontally connected over the full length of the breadboard, while each terminal strip is vertically connected. There is no vertical connection between the power rails and no horizontal connection between the terminal strips.

The electronic circuit diagram depicted in Fig. 11 can be broken down to two seperate parts: 1) A circuit for the IR proximity sensor, and 2) A transistor circuit to control the fan speed. The IR proximity sensor has 3 pins, a ground (GND) pin, a supply voltage (V_{CC}) pin and an output voltage (V_0) pin. We supply the sensor with 5 VDC power drawn from the Arduino Uno. As its output V_0, the sensor returns a voltage between 0 VDC and 5 VDC, either unfiltered to the Arduino's analog input pin (A0), or filtered via an RC filter, as illustrated in Fig. 12. An RC filter

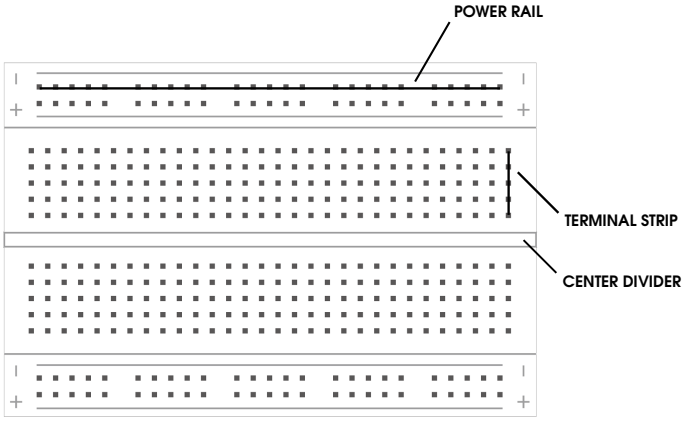


Figure 10: A solderless breadboard. The power rails are horizontally connected, while the terminal strips are vertically connected.

is an analog first-order low-pass filter, attenuating high-frequency noise on the sensor's output voltage. Using a resistor of $R = 22\ k\Omega$ and a capacitor of $F = 2.2\ \mu F$ yields an RC filter with a time constant of $\tau = R * C = 0.1\ s$,

$$H(s) = \frac{1}{RCs + 1} = \frac{1}{0.1s + 1}. \tag{2}$$

The transistor circuit used to drive the fan's motor combines two concepts from the introduction: a transistor controlled using Pulse Width Modulation (PWM) and a flyback diode. To power the motor, we need to send 12 VDC through it and then on to ground. To give the Arduino Uno control of the motor's power, you use a transistor, which functions as an electrically operated switch activated by means of PWM from your Arduino's digital pins. The emitter of the transistor is connected to common ground and the collector is connected to the motor. To control the flow between the emitter and collector, a low power PWM signal can supplied at the base pin (Fig. 4). You connect pin D9, which has PWM capabilities, to the base pin of the transistor and use a 2.2 $k\Omega$ resistor to limit the current. To limit the chances of a creating a reverse current after cutting the motor's power, we place a flyback diode parallel across the motor. The practical implementation of the motor's electronic circuit on the solderless breadboard is shown in Fig. 12. Note how the grounds from both power rails on the solderless breadboard and the Arduino Uno are shared, which is essential to making this setup work.

2.3 Coupling the Arduino to MATLAB

Make sure the `Arduino support package for Simulink` is installed locally on the computers that will be used (Simulink > `APPS` > `Get Add-Ons` > `Get Hardware Support Packages` > `Simulink Support Package for Arduino`). Connect the Arduino to the computer with the USB cable. Select `MODELLING` > `Model Settings` > `Hardware Implementation` > `Hardware board: Arduino Uno` to reveal the `HARDWARE` tab. To run a Simulink model on your Arduino board, select `HARDWARE` > `Run with IO`. To execute code continuously on your Arduino, set the simulation time to `inf`.

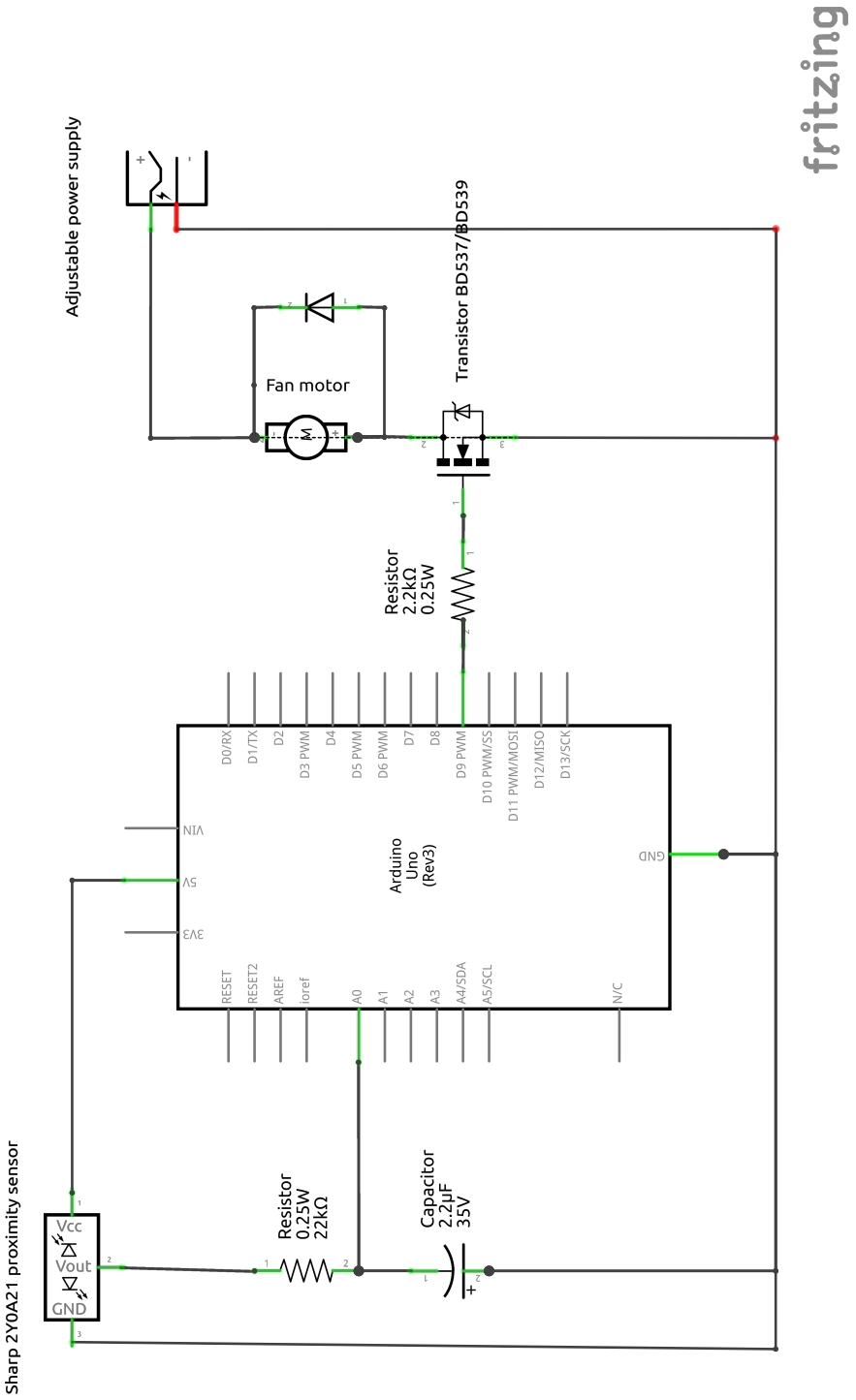


Figure 11: Electronic circuit diagram. Note how the IR sensor circuit and the transistor circuit are two seperate entities.

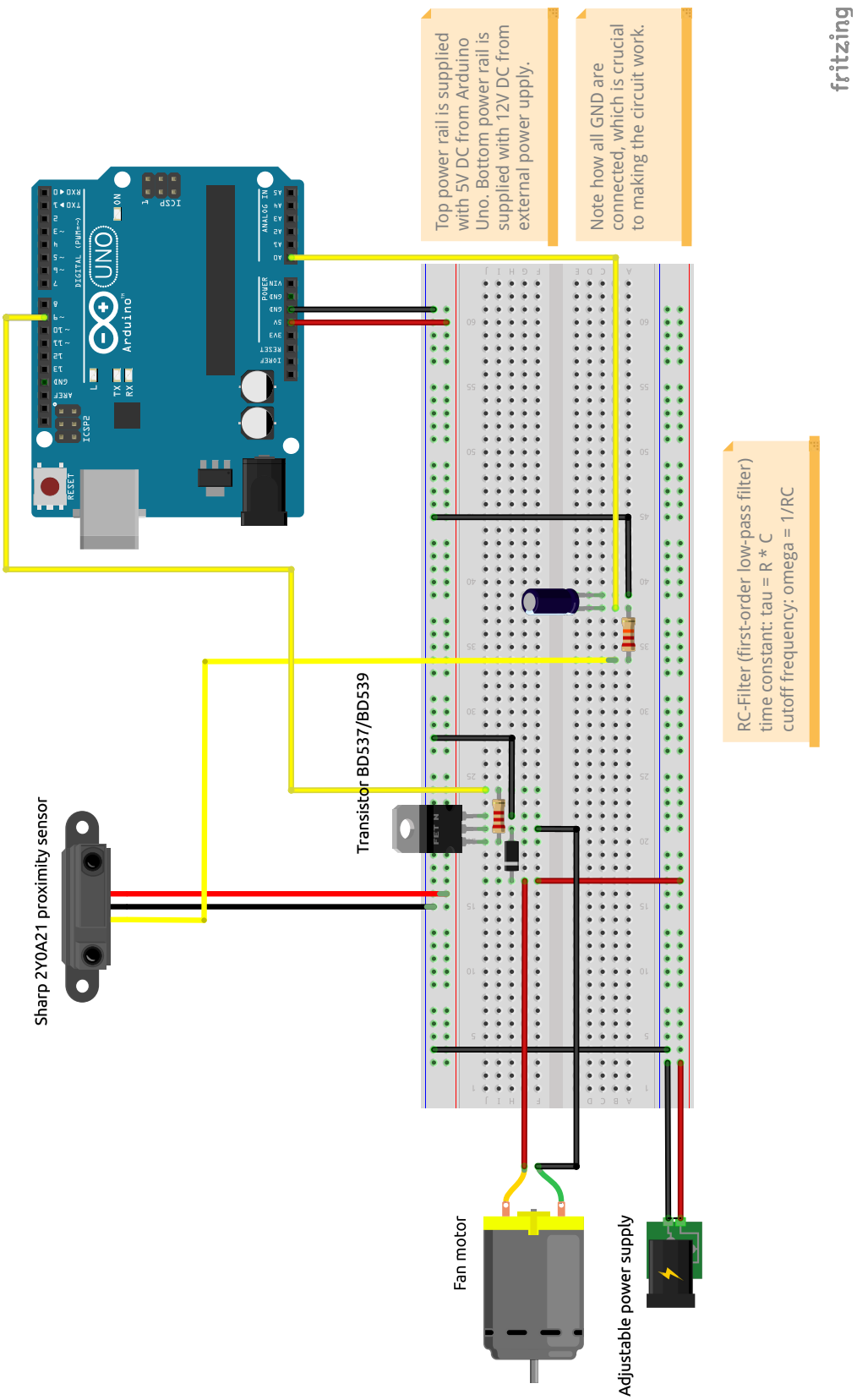


Figure 12: Electronic circuit built on the solderles breadboard.

2.4 Calibrating the IR proximity sensor

The relationship between the IR sensor's output voltage and its distance to the table tennis ball is needed. First, we collected data by hovering the ball at several heights (using a PID controller) and recording the sensor's output voltage (Fig. 13). We then expiremented with several mathematical relationships and found the following *logit* relationship adequately described the ball's height in function of the recorded voltage,

$$H(V) = H_{\max} + \frac{1}{\beta}\left[\ln\left(\frac{V_{\max} - V_{\min}}{V_{\max} - V} - 1\right) - \ln(\alpha)\right] \tag{3}$$

with $H_{\max} = 0.65\ m$, $\alpha = 2.19$, $\beta = 8.91$, $V_{\min} = 0.70\ V$, $V_{\max} = 4.51\ V$. At the bottom of the tube, a change in the ball's height results only in a very small change in the sensor's recorded voltage. Due to the noise on the recorded voltage, measuring height differences in the bottom 20 cm of the tube is difficult. As the ball gets close to the IR proximity sensor, above 58 cm, the sensor's output voltage drops back to 2 V.

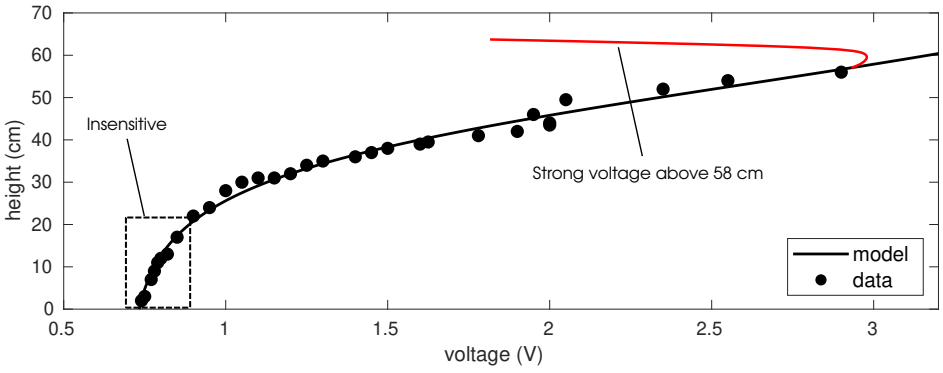


Figure 13: Relationship between the height of the table tennis ball and the IR sensor's output voltage.

2.5 Mathematical model

2.5.1 Model structure

The fan at the bottom of the tube blows air upwards around the table tennis ball, which hampers the flow of air thereby creating an upwards *drag force* proportional to the square of the relative velocity of the ball (Fig 14). Air molecules hitting the bottom of the ball are slowed down and subsequently deflected around the ball, resulting in the formation of a high pressure region. The incoming laminar airflow will adhere to the surface of the ball[1] up until the *separation point* where the formation of turbulent eddies begins. In the wake of the ball, a turbulent region of low pressure forms and it is the pressure difference between the bottom and top of the ball that generates the drag force. Opposed is the force of gravity, which pulls the ball down with a constant force irregardless of the ball's velocity. The total force exerted on the tennis ball along the z axis

[1]This is called the *coanda effect* which you may have know as the deflection of a vertical stream of water across the back of a spoon.

is therefore equal to,

$$F = \frac{1}{2}\rho A C_D v_{\text{rel}}^2 - mg, \tag{4}$$

where ρ_{air} is the density of air, equal to 1.225 $kg \cdot m^{-3}$. A is the surface area of the table tennis ball, equal to $4\pi r^2$ with a radius $r = 0.02$ m. C_D is the coefficient of drag, which is 0.47 (-) for Reynolds numbers below $Re < 10^5$ (S. Hoerner, 1965. Fluid-Dynamic Drag). m is the mass of the ping ball and is equal to 0.00283 kg. g is earth's gravitational constant equal to 9.81 $m \cdot s^{-2}$. v_{rel} is the relative velocity of air around the table tennis ball in $m \cdot s^{-1}$, which depends on the velocity of the airflow inside the tube (v_{air}), as well as on the velocity of the table tennis ball itself (v_{ball}). However, because the velocity of the airflow in the tube is not known exactly we make two assumptions. First, we assume it is related to the PWM signal used to control the fan's speed by means of a power law function. Second, we assume there is no pressure drop along the length of the tube, and hence, the air's speed is independent of the position in the tube (Excercise 3.1). Mathematically,

$$v_{\text{rel}} = v_{\text{air}} - v_{\text{ball}} = \alpha u(t)^{\beta} - v_{\text{ball}}, \tag{5}$$

where α $(m \cdot s^{-1})$ and β $(-)$ are parameters of the power law function and $u(t)$ is the value of the PWM signal, located between 0 and 255, and corresponding to a 0 % and a 100 % duty respectively. Hence, we can rewrite Eq. 4 as follows,

$$F = \frac{1}{2}\rho A C_D \left(\alpha u(t)^{\beta} - v_{\text{ball}}\right)^2 - mg. \tag{6}$$

Using Newton's second law of motion, we arrive at the following non-linear second order ODE describing the ball's movement inside the tube,

$$m\frac{\mathrm{d}^2 z}{\mathrm{d}t^2} = \frac{1}{2}\rho A C_D \left(\alpha u(t)^{\beta} - \frac{\mathrm{d}z}{\mathrm{d}t}\right)^2 - mg. \tag{7}$$

Next, as is common practice in control theory, we rewrite the mathematical model in a state-space representation. The system's states, denoted $\boldsymbol{x}$, are the position z and velocity v of the ball. Its only observed output y is the position of the ball z, its only input is $u(t)$. By substituting $v = dz/dt$ in Eq. 7, we arrive at the following non-linear model in state-space representation,

$$\begin{cases} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\big(\boldsymbol{x(t)},\ u(t)\big) = \begin{bmatrix} \dot{z}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \dfrac{\rho A C_D}{2m}(\alpha u(t)^{\beta} - v(t))^2 - g \end{bmatrix}, \\ y(t) &= g\big(\boldsymbol{x(t)},\ u(t)\big) = z(t). \end{cases} \tag{8}$$

2.5.2 Equilibrium and operating points

Equillibrium points A system is in equillibrium when the derivatives of all states are zero when no input is provided, mathematically, $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x(t)},\ 0) = 0$. Substitution in Eq. 8 yields the following conditions,

$$\begin{cases} v^* = 0\ m \cdot s^{-1}, \\ g = 0\ m \cdot s^{-2}, \end{cases} \tag{9}$$

and there is no condition on the ball's position z. Thus, the ball can only be at rest at any point in the tube with the fan switched off in the abscence of gravity. Therefore, on earth, where $g = 9.81\ m \cdot s^{-2}$, the system does not have an equillibrium point. In practice, when the fan is switched off, the ball will be in equillibrium at the bottom of the tube, where its weight is supported by the fan stand.

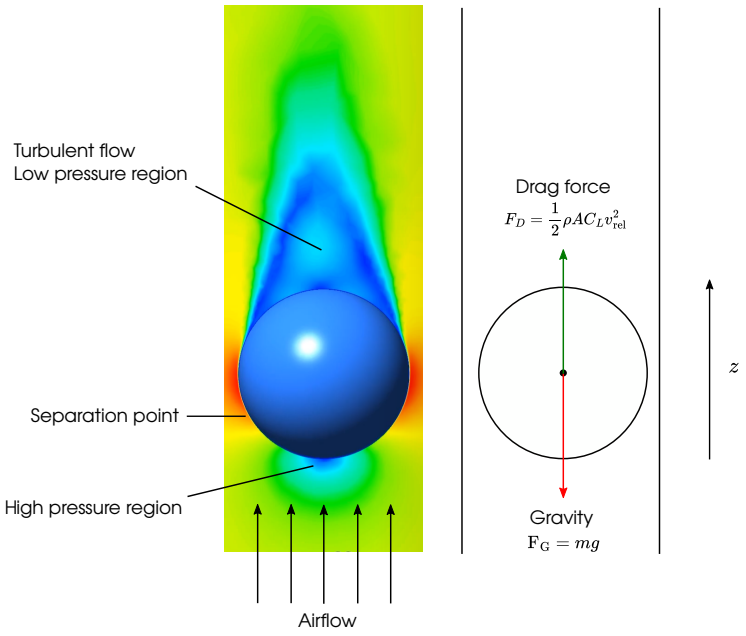


Figure 14: (left) Velocity contours of the flow around a smooth sphere placed in a laminar fluid flow. (right) Forces acting upon the sphere.

Operating points A system is in an operating point when the derivatives of all states are zero when input is provided, mathematically, $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x(t)},\ \bar{u}\right) = 0$. Substitution of this condition in Eq. 8 yields,

$$\begin{cases} v^* = 0\ m \cdot s^{-1}, \\ \bar{u} = \left(\dfrac{2mg}{\rho A C_D \alpha^2}\right)^{\dfrac{1}{2\beta}}. \end{cases} \tag{10}$$

From which we conclude: 1) Any position inside the tube can be an operating point as there is no condition on the ball's position z, 2) the ball hovers in an operating point because $v^* = 0\ m \cdot s$, and 3) the fan speed needed to hover the ball $\bar{u}$ only depends on system parameters. Therefore, there exists one unique fan velocity given by Eq. 10 that will make the ball hover at a fixed height in the tube, irregardless of the ball's position in the tube. When designing a controller, the aim is to stabilise a desired operating point $(z^*,\ 0,\ \bar{u})$.

2.5.3 Stability of the operating points

To examine the stability of the operating points of a non-linear system analytically, we first linearize the system and evaluate it at generic point $(\boldsymbol{x^*}, u^*) = (z^*, v^*, u^*)$. A linear state-space system has the following form,

$$\begin{cases} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t), \\ y(t) &= \boldsymbol{C}\boldsymbol{x}(t) + Du(t), \end{cases} \tag{11}$$

Linearising a non-linear system is done by means of a first-order Taylor approximation in the operating point,

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x^T}}\right|_{(z^*,v^*,u^*)} \quad \boldsymbol{B} = \left.\frac{\partial \boldsymbol{f}}{\partial u}\right|_{(z^*,v^*,u^*)} \quad \boldsymbol{C} = \left.\frac{\partial g}{\partial \boldsymbol{x^T}}\right|_{(z^*,v^*,u^*)} \quad \boldsymbol{D} = \left.\frac{\partial g}{\partial u}\right|_{(z^*,v^*,u^*)}. \tag{12}$$

Applied to Eq. 8,

$$\begin{cases} \dot{\boldsymbol{x}}(t) &= \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & \frac{\rho A C_D}{m}\left(v^* - \alpha\left(u^*\right)^\beta\right) \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{\alpha\beta\rho A C_D}{m}(u^*)^{\beta-1}\left(\alpha(u^*)^\beta - v^*\right) \end{bmatrix}}_{\boldsymbol{B}} u(t), \\ y(t) &= \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\boldsymbol{C}} \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} + \underbrace{0}_{D} u(t), \end{cases} \tag{13}$$

where $\boldsymbol{A}$ is the *system matrix*, $\boldsymbol{B}$ is the *input matrix*, $\boldsymbol{C}$ is the *output matrix* and D is the *feedthrough matrix*. The eigenvalues of the matrix $\boldsymbol{A}$ at the operating point $(z^*,\ 0,\ \bar{u})$ determine its stability and are found by solving its characteristic equation,

$$\det\left(\lambda I - A\right) = \begin{vmatrix} \lambda & -1 \\ 0 & \lambda + \frac{\rho A C_D}{m}\alpha\bar{u}^\beta \end{vmatrix} = 0 \tag{14}$$

$$\iff \lambda_1 = 0 \ \vee \ \lambda_2 = -\frac{\rho A C_D}{m}\alpha\bar{u}^\beta. \tag{15}$$

If all real parts of the eigenvalues are negative than the system is *asymptotically stable* and if at least one real part is positive and the others negative, the system is *unstable*. For $\alpha > 0$, our system has one negative eigenvalue and one zero eigenvalue, and therefore exerts *marginal or neutral stability*, which is between an asymptotically stable and an unstable system. When displaced by the fan, the ball will not return to a common steady state, nor will it go away without limit. To examine the system's stability,we can alternatively write down the linearised system's transfer function and examine its poles in the operating points,

$$H(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}, \tag{16}$$

applied to Eq. 13,

$$H(s) = \frac{\alpha\beta\rho A C_D/m}{s\left(s - \rho A C_D/m\left(v^* - \alpha(u^*)^\beta\right)\right)}, \tag{17}$$

whose poles in the operating point $(z^*,\ 0,\ \bar{u})$ are identical to those found by solving the characteristic equation (Eq. 15). In Fig. 15, the phase trajectories of the system are given for three fan velocities, below ($u^* = 131$), at ($\bar{u} = 151$), and above ($u^* = 171$) the unique fan speed needed to hover the ball (Eq. 10)[2]. If a point $(z,\ v)$ in the phase trajectory is chosen as the initial condition of the system, then the streamlines indicate the change in position and velocity of the table tennis ball over time. With an insufficient fan speed to hover the ball, any initial combination of position and velocity results in the ball landing at the bottom of the tube. Opposed, at a fan speed above the hovering speed, the ball always ends up at the top of the tube. Starting from any initial velocity, a fan speed of $u = 151$ guarantees the ball will eventually come to a standstill.

[2]For the calibrated values $\alpha = 0.19$ and $\beta = 0.63$, see Excercise 3.5.

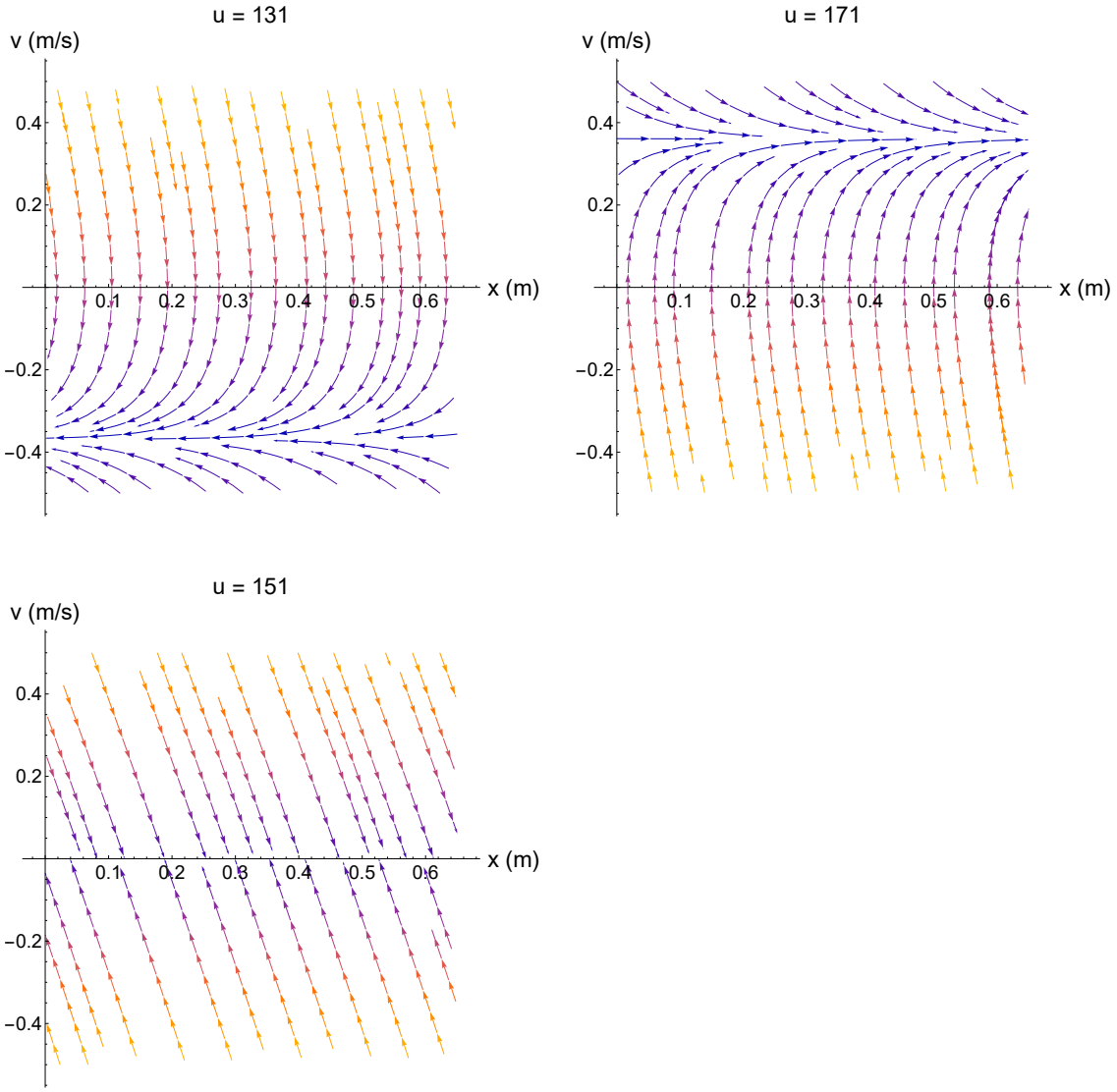


Figure 15: Phase trajectories for three different values of the fan speed $u(t)$.

2.5.4 Controllability and observability

Controllability The controllability matrix of the linearised system in an operating point $(z^*, 0, \bar{u})$ is computed as follows,

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} \end{bmatrix} = \bar{u}^{(2\beta-1)} \frac{\alpha^2 \beta \rho A C_D}{m} \begin{bmatrix} 0 & 1 \\ 1 & -\dfrac{\alpha \rho A C_D}{m} \bar{u}^{\beta} \end{bmatrix}, \tag{18}$$

and its rank is equal to the number of states (2). Therefore, the system is controllable, meaning an appropriate choice of fan speeds $u(t)$ can be used drive the ball from any initial state to any desired state within a finite timeframe.

Observability The observability matrix of the linearised system is computed as follows,

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{19}$$

and its rank is equal to the number of states (2). The system is therefore observable meaning the IR proximity sensor's measurement of the ball's position can be used in conjunction with a Luenberger state estimator to estimate the velocity of the ball. Hence, the system's known states can be used to estimate it's unknown states.

3 Hands-on practical

Overview of the practical

This hands-on practical consists of four parts. In the first part you will study the physical system by analysing its **open-loop response** or how it reacts without a feedback controller. This will allow you to assess how the system reacts to inputs by the fan, assess the stability of operating points (Section 2.5.3). Further, you will learn how to filter the IR proximity sensor's signals. Finally, you will perform several response experiments, which you will then use to **calibrate the mathematical model of the setup** (Eq. 8). In the second part of the practical, you will use the calibrated mathematical model to perform an in-silico tuning of a PID controller. In addition, you will tune a PID controller by means of the Cohen-Coon and Ziegler-Nichols **empirical tuning methods**. In the third part of this practicum, you will test the tuned PID controllers on the real system. You will study the **closed-loop response** of the system by studying the controller's P, I and D contributions more closely. You will further study the implications of dead time on the fan's response and measurement noise on the performance of the controller. The fourth and final part of the practical will introduce you to the design of the more advanced **Linear Quadractic Regulator** with **Luenberger estimator**. Its performance as compared to the PID controller will be assessed.

Part I: Open-loop control, time- and frequency response

3.1 Open-loop control of the fan's speed

Before implementing any form of feedback control, we investigate how to control the system manually (open-loop control). We'll use the Simulink model shown in Fig. 16 to control the fan speed. The model contains a `Knob` whose output is connected to a `Constant` block. The knob's output is limited to the interval 0-255 and represents the fan's motor speed u_{PWM}. Setting u_{PWM} to 0 results in $0/255 \cdot 12$ V $= 0$ V applied to the fan motor, while setting u_{PWM} to 255 results in $255/255 \cdot 12$ V $= 12$ V applied to the fan motor. A `display` block is used to more accurately show the value selected on the knob.

Exercise 3.1: Open-loop control of the fan's speed

1. Implement the model in Simulink (Fig. 16) or open **writePWM.slx**.
2. Start by turning the knob to $u_{\text{PWM}} \approx 180$. Run the Simulink model by pressing the `Run with IO` button in the `Hardware` tab. Pay attention to the system's response. What is the minimum value u_{PWM} needed to raise the ping pong ball from the stand?
3. What is the approximate u_{PWM} needed to hover the table tennis ball at 30 cm?
4. What is the approximate u_{PWM} needed to hover the table tennis ball at 50 cm? Are your findings in line with those obtained by analytically assessing the stability of the operating points (Section 2.5.3)?

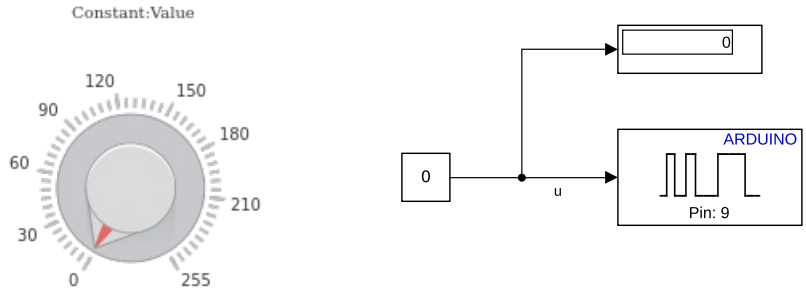


Figure 16: Simulink model used to control the fan speed.

3.2 Measuring and filtering the IR sensor's signal

Next, we will measure the step response of the system. For this purpose, you can use the Simulink model **readProximity.slx** (Fig. 17). An `analog read` block is used to read the sensor's output voltage at pin A0. However, the voltage was already transformed into a 10 bit digital signal by the Arduinos on-board analog-to-digital converter so a number between 0 and 1023 is returned. After changing the signal's datatype to `double`, we convert it back to a voltage between 0 V and 5 V. Next, the signal is smoothed by combining a `median filter` and a first-order system (low-pass filter). Next, Eq. 3 is used to convert the voltage into a distance from the IR sensor (Fig 13). However, because of a singularity in Eq. 3.17 at 0.70 V, the voltage is limited to a minimum of 0.701 V. Finally, distance from the sensor is converted to height in the tube and a `saturation`

block is used limit the height to a physcially plausible range.

Exercise 3.2: Measurement of the ping pong ball's height

1. Implement the model in Simulink (Fig. 17) or open **readProximity.slx**.
2. Manually control the height of the ping pong ball while observing the height using the `Scope`. How is the overall signal quality? Does the signal quality differ at different heights? What could you do to improve this signal?

Exercise 3.3: Filtering the IR sensor's signal

1. Add a first-order transfer function after the block converting the voltage to the range 0-5 VDC. Set the gain to $K = 1$ and time constant to $\tau = 0.2\ s$. Optionally connect the time constant to a `knob` for convenience. Visualise the filtered and unfiltered signal using a `scope`. Manually control the height of the ping pong ball while observing the height using the scope. How is the signal quality now?
2. Vary the first-order transfer function's time constant. How does the signal quality change? How does the timing of the filtered signal differ from the unfiltered signal? Find an appropriate time constant to use for the rest of the practical.
3. What is the name of such a filter? Can you explain how this filter works?

3.3 Collecting time response data

Having a mathematical model of the system is usefull to design a well-functioning controller. We previously derived a set of non-linear second-order ordinary differential equations representing the physical processes underlying the table tennis ball movement (Eq. 8). This model, based on a combination of physical laws and unknown parameters is a **grey box model**. In practice, systems

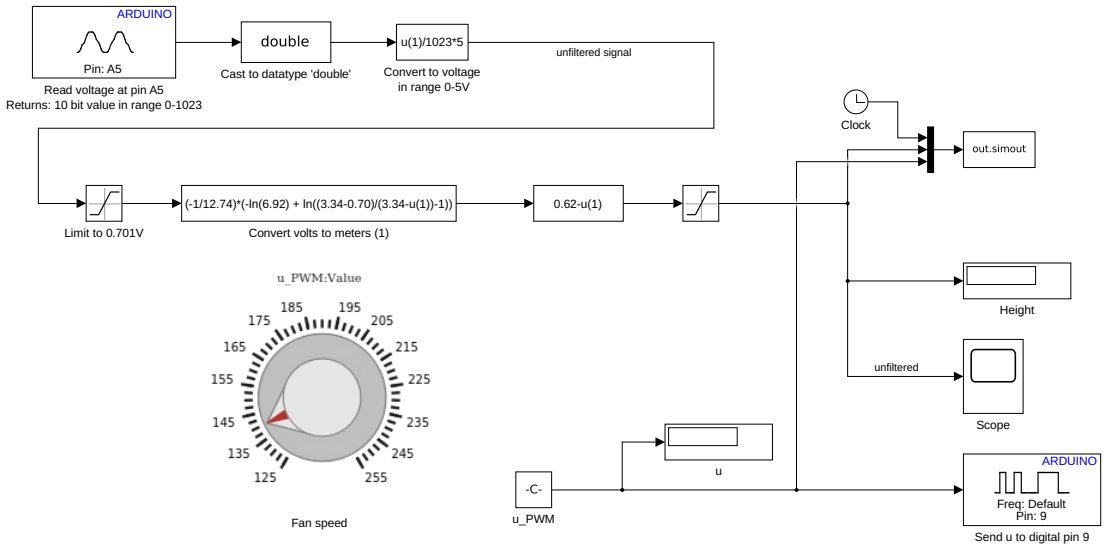


Figure 17: Simulink model used to read the ball's height in open-loop control.

are often too complex to come up with a grey or white box model. Many complex, non-linear system's dynamics are reasonably approximated by linear models under typical operating conditions. However, our system, with its neutral stability, is not well approximated by a first- or second order transfer function and hence, no such model is identified during this practical.

You will perform experiments and use them to calibrate the unknown parameters α and β in Eq. 8. The first experiment is a **block-shaped response** experiment, which combines a step response experiment with a U-shaped response experiment (Fig. 18a). This experiment is preferred over the more simple step response experiment because it contains information on both the upward and downard trajectory of the table tennis ball, which differ from each other. Additionally, the U-shaped response contains information on the ball's **in-flight dead time** in response to changes in the fan speed. In the second **W-shaped response** experiment, a similar step-wise input is given, followed by a series of three step-wise changes in the fan speed, alternating below and above the fan speed needed to hover the ball $\bar{u}$(Eq. 10) (Fig. 18b). The block-shaped response experiments will be used as **calibration dataset**, while the W-shaped response experiment will be used as a **validation dataset** to assess the predictive accuracy of our calibrated model.

Exercise 3.4: Sytem response experiments

1. Open the script **writeBlockResponse.m**, which is preprogramed to perform a step response, followed by a U-shaped response (Fig. 18a). The intial step response is performed by stepping the fan speed to $u_{\max}$ for 7.5 s, followed by 7.5 s at the equillibrium fan speed $\bar{u}$ to stabilise the ball's spin for the second part of the experiment. Then, the U-shaped response is performed by stepping down the fan's speed to $u = \bar{u} - u_{\text{step}}$ for 2.5 s, then stabilising the ball at the equillibrium fan speed $\bar{u}$ for 5.0 s, followed by a 2.5 s step up to $u = \bar{u} + u_{\text{step}}$. By default, three initial step motor speeds $u_{\max} = 170,\ 185,\ 255$, and two U-sizes $u_{\text{step}} = 6, 10$ are permutated, and every combination is performed in duplicate to average out natural variations in the response. Thus, a total of 12 experiments are performed. You will provide the observed time needed before the ball was lifted from the stand (*lift-off dead time*), which is then automatically cut from the experiment. In the script, provide an accurate estimate of the fan's equillibrium speed `u_eq` (Excercise 1), and set the `output_name` to *blockResponse.csv*. The script uses the Simulink model *blockResponse.slx*. Limit the ball's height to 50 cm using a thin wooden stick.

2. Open the script **writeWResponse.m**, which uses the Simulink model *wResponse.slx* to log a W-shaped response experiment which we will use as a validation dataset (Fig. 18b). Five identical experiments are automatically performed to visualise the natural variation on the reponse. After all five experiments, you will provide one lift-off dead time which is then cut from the experiments. Use the automatically generated figure containing the fan speed and height response to **determine the in-flight dead time**.

$t_d =$

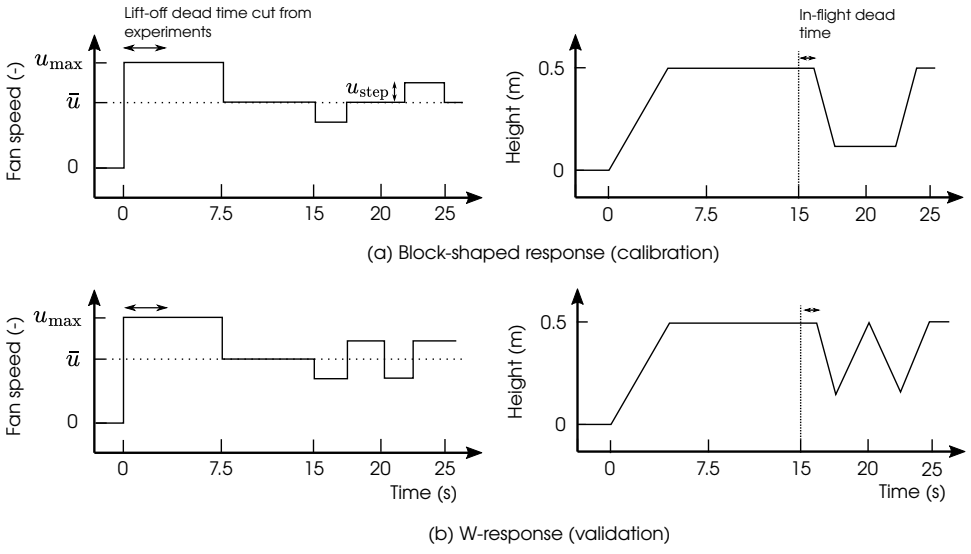


Figure 18: (a) Fan speed and ball height during a block-shaped response experiment. (b) Fan speed and ball height during a W-response experiment.

3.4 Understanding the system's open-loop frequency response

Finally, you will investigate the system's open-loop frequency response. You will do this by applying a sinusoidal input of varying frequency to the fan and observing the amplitude of the resulting sinusoidal changes in the ball's height, as well as the phase shift between the input and output signals. Using your observations, you construct an approximate **Bode diagram** allowing you to make further deductions on the system's dynamics.

Exercise 3.5: Frequency response of the system

1. Open the Simulink model **frequency.slx** and use the `knob` to set the equillibrium fan speed needed to hover the ball, $\bar{u}_{\text{PWM}}$ (Excercise 3.1). Set the frequency knob to 10 rad/s, which results in the fan speed sinusoidally varying between $\bar{u}_{\text{PWM}} \pm 30$, within $t = \pi/10 = 0.3\ s$ (= high-frequency input). Temporarily increase the value of $\bar{u}_{\text{PWM}}$ to overcome the additional energy needed to get the ball airborne. Next, hover the ball where the IR sensor's signal's quality is good, around 30 cm. Note that as the transistor heats up, its ability to pass a current increases. Hence, starting from a cold setup, you may need to initially use a value slightly higher than $\bar{u}_{\text{PWM}}$. What is the amplitude of the sinusoidal changes in the ball's height?

2. Decrease the frequency of the input signal to 1 rad/s in a step-wise manner. Prevent the ball from going fully down or up by making adjustments to $\bar{u}_{\text{PWM}}$. How does the amplitude of the ball's height change? Are the input and output signals in or out of phase?

3. Decrease the frequency of the input signal below 1 rad/s. Stop making temporary adjustments to $\bar{u}_{\text{PWM}}$ to prevent the ball from reaching either end of the tube. What is the amplitude of the ball's height? Not accounting for the dead time introduced by the initial energy needed to get the ball hovering, are the input and output signals in or out of phase?

4. Draw an approximate Bode diagram of the system. What kind of system is the Bode diagram characteristic of?

Part II: System identification and PID controller tuning

The second part of this practical can be done **independently from the lab set-up**. In this part of the practical, the unkown parameters in the mathematical model of the setup (Eq. 8) are identified using the response data gathered in Excercise 3.4. Then, the calibrated mathematical

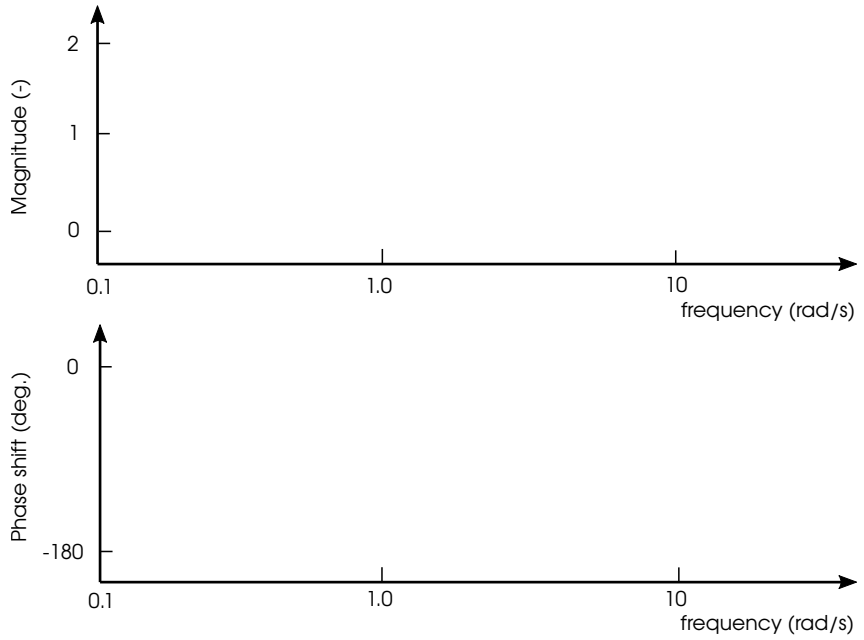


Figure 19: Approximate Bode diagram based on the setup's open-loop frequency response.

model is implemented in Simulink and used to design a suitable PID controller **in-silico**. Finally, two **emperical tuning techniques**, by Cohen-Coon and Ziegler-Nichols, are used to tune a PID controller.

3.5 System Identification

Exercise 3.6: Calibrating the mathematical model (Eq. 8)

1. Use **systemIdentification.m** to calibrate the unkown parameters α and β in Eq. 8 using the block-shaped response dataset gathered in Excercise 3.4.1. In the script, be sure to provide your estimate of the in-flight dead time (Excercise 3.4.2) and validate the paths to *blockResponse.csv* and *wResponse.csv*. The script will automatically minimize the sum-of-squared errors between the block-shaped response experiments and the mathematical model by finding optimal values for α and β. After the estimation of α and β is completed, a figure comparing the experiments and the mathematical model is provided (Fig. 20). Write down the optimal values of α and β below.

   $\alpha =$
   $\beta =$

2. Calculate the equillibrium fan speed $\bar{u}$ using Eq. 10. Use the following values for the constants: $\rho = 1.225\ kg \cdot m^{-3}$, $m = 0.00283\ kg$, $A = 4\pi r^2$ with $r = 0.02\ m$, $g = 9.81\ m \cdot s^{-2}$, and $C_D = 0.47$ (-). Is the computed value sufficiently close to the observed value (Excercise 3.1.3)?

$$\bar{u} = \left(\frac{2mg}{\rho A C_D \alpha^2}\right)^{\frac{1}{2\beta}} =$$

3. How is the goodness-of-fit of the mathematical model to the experimentally obtained data? What is the purpose of the validation experiment?

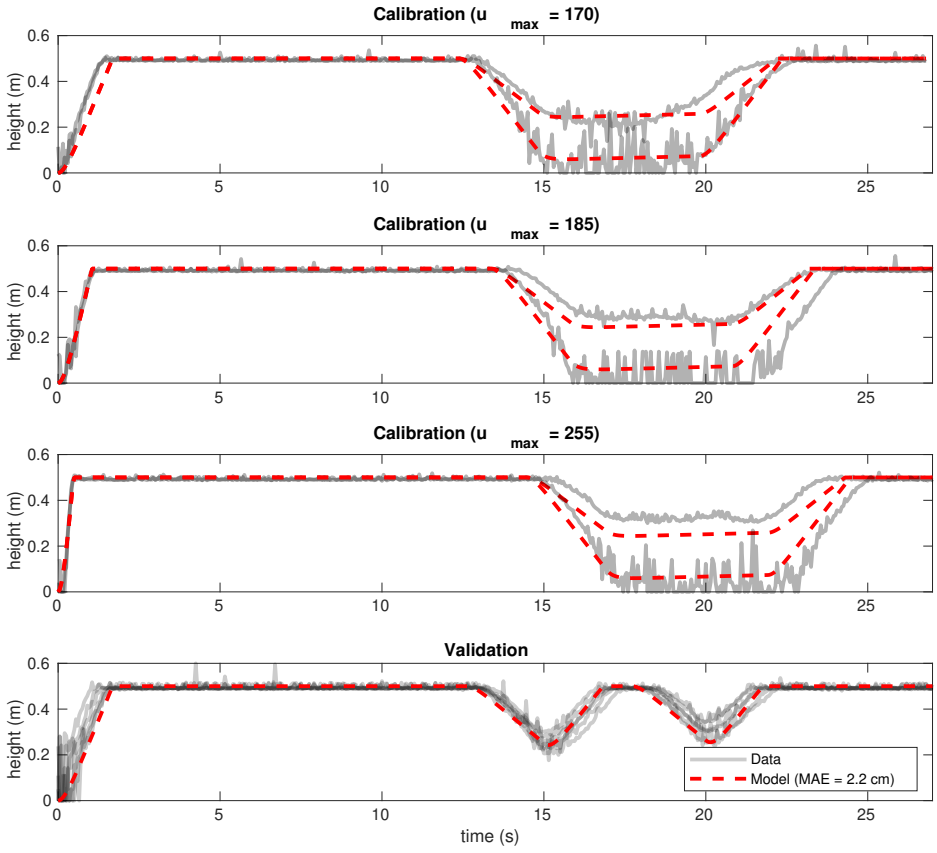


Figure 20: Goodness-of-fit of the mathematical model (Eq. 8) to the calibration and validation experiments (Fig. 18). The mean average error of the mathematical model in predicting the validation dataset is 2.2 cm.

3.6 Tuning a PID controller

In this section, you will learn how to tune a PID controller with the following transfer function,

$$C(s) = K_P + K_I \frac{1}{s} + K_D \underbrace{\left(\frac{s}{(1/K_F)\, s + 1} \right)}_{\text{filtered derivative}}, \tag{20}$$

where K_P is the proportional gain, K_I is the integral gain, K_D is the derivative gain, and $1/K_F$ is the derivative filter first-order time constant. The PID controller can be implemented either directly in Simulink using the `PID` block, or using the following basic Simulink blocks,

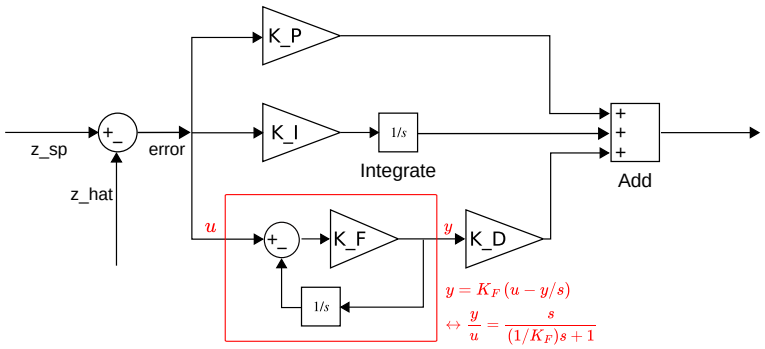


Figure 21: Implementation of a PID controller in Simulink. The derivation of the filtered derivative's transfer function is given in red.

Exercise 3.7: In-silico PID tuning

1. Implement the mathematical model of the setup (Eq. 8) with a PID controller in Simulink (Fig. 21). Saturate the ball's height between 0 cm and 60 cm. Do this by implementing the equivalent of the following Matlab code in Simulink using two consecutive `Switch` blocks,

```
if z >= upper_bound
  zdot = min(0, v); % pass only negative velocities at upper bound
elseif z <= lower_bound
  zdot = max(0, v); % pass only positive velocities at lower bound
else
  zdot = v;
end
```

   Placing a `Saturation` block after the integration of Equations 8 in Simulink is incorrect and not equivalent to the above Matlab code. Do not model the in-flight dead time or noise on the sensor's output initially. Make sure the initial value of the integral control action is set to zero. Simulate a stepwise increase in the setpoint from 0 cm to 40 cm, and use the following controller gains $K_P = 100$, $K_I = 10$, $K_D = 0$, $K_F = 100$. Observe the output of the integral control action. What value does the integral control action converge to?

2. Simulate a step-wise increase in the setpoint height from 20 cm to 40 cm. Set the initial output of the integral control action to the equillibrium fan speed $\bar{u}$. Tune the PID controller, what values for K_P, K_I, K_D and K_F yield satisfactory behavior?

   $K_P =$
   $K_I =$
   $K_D =$
   $K_F =$

3. Use a `Transport delay` block to model the in-flight dead time as a delay on the motor's output (be sure to set its initial output to the equillibrium fan speed $\bar{u}$). What is the impact of the dead-time on the stability of your previously tuned PID controller?

4. Use a `Spectral noise` (recommended) or `Random number` block to make the height measurement noisy. Subsequently filter the noisy signal using a low-pass filter (Excercise 3.3). Now, retune the PID controller. What values for K_P, K_I, K_D and K_F yield satisfactory behavior?

   $K_P =$
   $K_I =$
   $K_D =$

5. What is the impact of a noisy measurement on the derivative control action? What happens for high values of the proportional gain K_P in the abscense of integral and derivative control action, and in the abscense of a setpoint change?

6. Simulate a stepwise decrease in the fan's effectiveness after a step response experiment, thereby increasing the fan's equillibrium speed $\bar{u}$. Model this by increasing the fan's proportionality constant α (Eq. 8) by 5-10 % using a `Step` block. Reassess the performance of your tuned controller.

7. (Optional) The Simulink model **ODEModelPID_smithpredictor.slx** contains an implementation of the mathematical model of the setup (Eq. 8), with a PID controller and with a Smith predictor to compensate for the dead time on the fan's motor response. Asses the stability of your previously tuned PID controller when a Smith predictor is used.

Cohen-Coon tuning The empirical tuning method by Cohen and Coon is an **open-loop tuning technique** relying on many complex non-linear system's step response being adequately approximated by a sigmoidal first-order transfer function with dead time (at least under/near the relevant operating conditions).

$$H(s) = \frac{y(s)}{u(s)} = \frac{Ke^{-t_d s}}{\tau s + 1}, \tag{21}$$

where K is the system's gain, τ its time constant, and t_d its dead time. After applying a stepwise increase to the fan speed u_{step} PWM units above the equillibrium fan speed $\bar{u}$, the gain K can be computed as $K = \Delta y_{\text{step}}/u_{\text{step}}$, while the time constant τ and dead time t_d can be estimated graphically. The former is the time needed to reach 63.2 % of the step's output change Δy_{step}, while the latter is the elapsed time before a change in input has an effect on the system's output (Fig. 22).

For our setup, it is best to use the in-flight dead time gathered in Excercise 3.4.2 instead of the lift-off dead time, as the latter is not representative of the system's behavior once the ball is airborne. Our choice to subtract the fan's equillibrium speed $\bar{u}$ from the step size is based on the intuition that this will better mimic a first-order response (remember the ball does not go anywhere for fan speeds under $\bar{u}$). Cohen and Coon devised that a stable closed-loop response can be obtained by computing the K_P, K_I and K_D values using the tuning rules listed in Table 1.

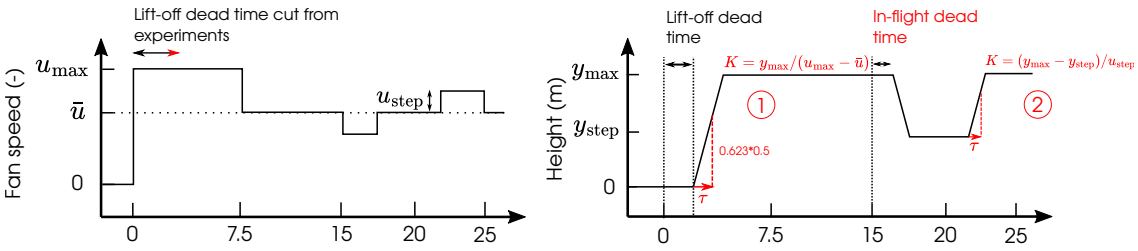


Figure 22: Using the block response experiments to estimate the gain, time constant and dead time of the approximate first-order response.

Exercise 3.8: Cohen-Coon tuning

1. Use the block response experiments gathered during Excercise 3.4.1 to graphically estimate the gain K and time constant τ, both for the initial step (labeled 1 in Fig. 22), as well as

the upwards step during the U-shaped phase of the experiment (labeled 2 in Fig. 22). Tune the PID controller using the tuning rules of Cohen-Coon listed in Table 1.

2. Simulate the step response of the closed-loop system using the tuned P, PI and PID controllers. What controllers lead to a stable closed-loop response?

Table 1: Tuning rules of Cohen-Coon for a PID controller $C(s) = K_P + K_I/s + K_D s$.

	K_P	K_I	K_D
P	$\frac{1}{K}\frac{\tau}{t_d}\left(1+\frac{t_d}{3\tau}\right)$	-	-
PI	$\frac{1}{K}\frac{\tau}{t_d}\left(0.9+\frac{t_d}{12\tau}\right)$	$K_P * \left(\frac{1}{t_d}\ \frac{9+20(t_d/\tau)}{30+3(t_d/\tau)}\right)$	-
PID	$\frac{1}{K}\frac{\tau}{t_d}\left(4/3+\frac{t_d}{4\tau}\right)$	$K_P * \left(\frac{1}{t_d}\ \frac{13+8(t_d/\tau)}{32+6(t_d/\tau)}\right)$	$K_P * \left(t_d\ \frac{4}{11+2(t_d/\tau)}\right)$

Ziegler-Nichols tuning Opposed to the method of Cohen and Coon, the empirical tuning method of Ziegler and Nichols is a **closed-loop tuning technique**. First, the system is stabilised in a desired operating point. Then, the controller is reduced to its P-action, and a change in the setpoint is demanded. The proporational controller gain K_P is then increased until the system's closed loop response to the demanded setpoint change results in permanent oscillations. This gain, often referred to as the *ultimate gain* K_u, as well as the period of the continuous oscillations P_u are recorded and used to tune the system (Table 5).

Exercise 3.9: Empirical PID tuning methods: Ziegler-Nichols

1. Determine the ultimate control gain K_u and the period of the associated oscillations P_u using your simulink model of the setup (Excercise 3.7). Tune the controller using the tuning rules in Table 2.

2. Simulate a step-wise change in the setpoint from 20 cm to 40 cm, do the controllers

yield a satisfactory response?

Table 2: Tuning rules of Ziegler-Nichols for a PID controller $C(s) = K_P + K_I/s + K_D s$.

	K_P	K_I	K_D
P	$\frac{K_u}{2}$	-	-
PI	$\frac{K_u}{2.2}$	$\frac{1.2K_P}{P_u}$	-
PID	$\frac{K_u}{1.7}$	$\frac{2K_P}{P_u}$	$\frac{8K_P}{P_u}$

Part III: PID control of the experimental set-up

In this part of the practical, you will test your tuned PID controllers on the experimental setup and assess their performance. You will examine the frequency response of the closed-loop system and construct its Bode diagram, which you will then compare to the open-loop Bode diagram (Excercise 3.5). By using PID controllers in practice you will gain a deeper understanding of some common issues such as noisy measurements and dead time.

3.7 Testing PID control on the experimental setup

Exercise 3.10: PID control of the experimental set-up

1. Open the Simulink model **PID.slx**, which contains the implementation of a PID-controller. Set adequate parameters for the low pass filtering of the IR sensor's measurement (Excercise 3.3). Verify the initial condition of the integral controller's `Integrate` block is set to the fan's equillibrium speed $\bar{u}$ (Excercise 3.6.2). The height setpoint, controller gains (K_P, K_I, K_D, K_F) and the time constant of the height measurement's low pass filter τ_F can be tuned using the knobs while operating the experimental setup. Use the knobs to configure your optimal PID-controller. Change the setpoint from 0 cm to 40 cm and observe the response. Disturb the system by constricting the airflow with your hands. Does your controller work adequately?

Exercise 3.11: Contribution of the different control actions

1. Set the initial condition of the integral controller's `Integrate` block to zero for the duration of this excercise. Use the *Scope* to visualise the **individual contributions of the proportional, integral and derivative actions** of the controller. Start with the ball at the bottom of the tube. Set the proporational gain to $K_P = 600 - 800$, while disabling the integral and derivative control ($K_I = K_D = 0$). Raise the setpoint to 40 cm. Does the ball ever reach the setpoint? How is this phenomenon called?

2. Start with the ball at the bottom of the tube. Disable the proportional and derivative control by setting $K_P = K_D = 0$. Set the integral gain to $K_I = 50$. Raise the setpoint to 40 cm. What happens to the contribution of the I-action? Is the obtained system stable?

3. Starting with the ball stabilised at 40 cm, disable the integral and derivative control action by setting $K_I = K_D = 0$. Make stepwise changes in the setpoint and disturb the system. Is the system stable and its response satisfactory? Do you feel like any derivative action is needed at this point?

4. Starting with the ball stabilised at 40 cm, disable the integral and derivative control action by setting $K_I = K_D = 0$. Make the fan slightly less effective by holding your hand flat above the tube at some distance (circa 5 cm). Is the controller able to keep the ball at 40 cm? Restabilise the system at 40 cm and make the fan slightly less effective, but set the integral gain to $K_I = 10$. What happens to the output of the integral controller? Reduce the fan's efficacy at least 30 s and then remove your hand. What phenomenon do you observe?

Exercise 3.12: Effect of filtering on the derivative action

1. Start with the ball stabilised at 40 cm, set the derivative filter's gain to $K_F = 1$ and increase the derivative gain to K_D = 200-300. Take some stepwise changes in the setpoint. What do you observe?

2. Examine the effect of changing the derivative filter's gain K_F, the time constant of the first order derivative filter is equal to $\tau = 1/K_F$.

Exercise 3.13: Effect of dead time on stability

1. In **PID.slx**, add a `Transport delay` block of 1 s to the measurement of the ball's height to introduce additional dead time. Use stable controller gains and raise the setpoint from 0 cm to 40 cm. What is the effect of the added dead time on the stability of the controlled system?

3.8 Understanding the system's closed-loop frequency response

Finally, in analogy to Excercise 3.5, you will investigate the system's closed-loop frequency response. You will do this by applying a sinusoidal change of the height setpoint and observing the amplitude of the resulting sinusoidal changes in the ball's height, as well as the phase shift between the input and output signals. Using your observations, you will construct an approximate **Bode diagram**.

Exercise 3.14: Frequency response of the system

1. Open the Simulink model **frequencyPID.slx** and use the knobs to set your optimal controller gains. Make sure the initial condition of the integral controller's `Integrate` block is set to the fan's equillibrium speed $\bar{u}$. The setpoint varies sinusoidally around 35 cm with an amplitude of 10 cm and a variable frequency. The frequency can be changed using the knob, start a high frequency of 10 rad/s. What is the amplitude of the sinusoidal changes in the ball's height? Are the input and output signals in or out of phase?

2. Decrease the frequency of the input signal to 3 rad/s. How does the amplitude of the ball's height change? Are the input and output signals in or out of phase?

3. Decrease the frequency of the input signal to around 2 rad/s. What is the amplitude of the ball's height? Are the input and output signals in or out of phase?

4. Decrease the frequency of the input signal to around 1 rad/s. What is the amplitude of the ball's height? Are the input and output signals in or out of phase?

5. Decrease the frequency of the input signal to around 0.5 rad/s. What is the amplitude of the ball's height? Are the input and output signals in or out of phase?

6. Draw an approximate Bode diagram of the system. What kind of system is the Bode diagram characteristic of?

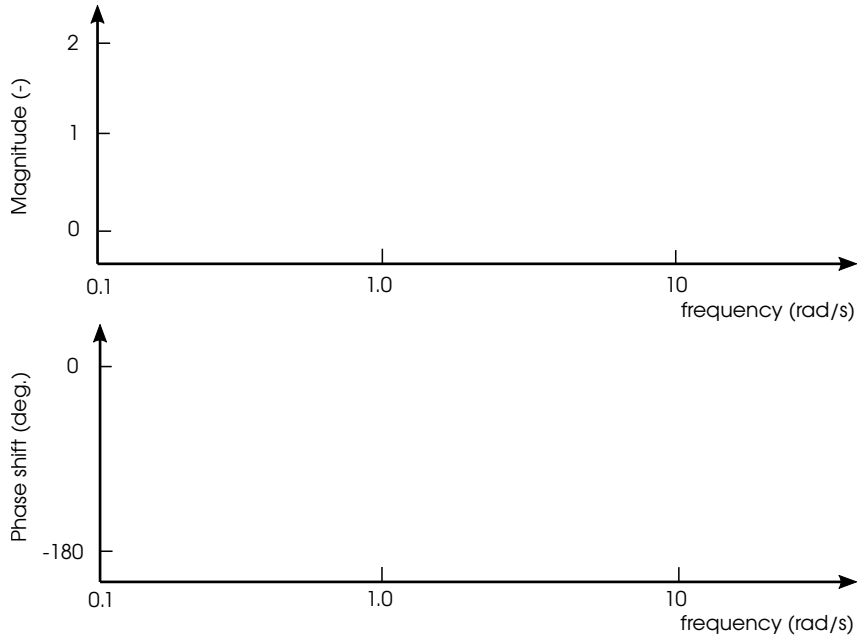


Figure 23: Approximate Bode diagram based on the setup's closed-loop frequency response.

Part IV: Advanced control strategies

3.9 Linear-Quadratic Regulator with Luenberger state observer

3.9.1 Control strategy and design

Linear-Quadratic Regulator In this section, we design an optimal state feedback controller to control the height of our table tennis ball. Mathematically, we attempt to drive our non-linear system to a desired operating point $\boldsymbol{x}_{\text{sp}} = (z, v)$ using an input $u(t)$. To accomplish this, the input $u(t)$ is chosen as the sum of two components (Fig. 24): The first, $\bar{u}$, is the input needed to keep the system in an operating point, and is the solution to the equation $\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{t}), \bar{u}) = 0$ (Eq. 10). The second, u^*, is the force needed to drive the system from any initial condition to the desired operating point, mathematically, $u^* = \boldsymbol{K}(\boldsymbol{x} - \boldsymbol{x}_{\text{sp}})$, which is the product of the feedback gain matrix $\boldsymbol{K}$ and the deviation from the setpoint.

The values of the feedback matrix K must then be chosen so that the real parts of the eigenvalues of the (linearised) system with state feedback $(\boldsymbol{A} + \boldsymbol{B} \cdot \boldsymbol{K})$ are negative, stabilizing the operating point. This is possible if the system is controllable, which we have previously verified (Eq. 18). For nonlinear systems the feedback matrix $\boldsymbol{K}$ is typically designed using the system matrix $\boldsymbol{A}$ and input matrix $\boldsymbol{B}$ from the system linearised in the operating point (Eq. 13). However, this strategy should always be used with care as nonlinear systems may have multiple operating points[3]and the size of the stable region around the desired operating point is unknown. Our system has an infinite number of operating points, as the ball can be stabilised at any height z inside the tube as long as the fan's hover speed $\bar{u}$ is maintained. However, because each operating point is neutrally stable, designing the gain matrices based on a linearised system is likely to work.

Instead of intuitively choosing the entries of the gain matrix $\boldsymbol{K}$ to obtain negative eigenvalues for the system with state feedback, it is possible to design an *optimal* gain matrix $\boldsymbol{K}$. Doing so entails defining the desired qualities of an *optimal* controller and translating them into an objective function. Typically, these qualities are: 1) Reaching the setpoint quickly, where a state-cost matrix $\boldsymbol{Q}$ governs the relative importance of reaching the setpoint of different states. For our setup, asigning more importance to reaching the height setpoint than the velocity setpoint ($v = 0\ m \cdot s^{-1}$) will result in oscillations. 2) Minimise the cost needed to reach the setpoint, where an input-cost matrix $\boldsymbol{R}$ governs the relative importance (read price) of using different inputs to reach the setpoint. Because our system has only one input, the input-cost matrix is not relevant to this practical. 3) To avoid offset. Minimising this objective function yields an optimal gain matrix $\boldsymbol{K}$, which is referred to as the Linear-Quadratic Regulator.

Exercise 3.15: Designing an appropriate state feedback gain matrix $\boldsymbol{K}$

1. What size should the state feedback gain matrix $\boldsymbol{K}$ be?

[3]Linear systems have only one operating point!

2. Use the system matrix $\boldsymbol{A}$ and input matrix $\boldsymbol{B}$ of the linearised system (Eq. 13) to design a suitable state feedback gain matrix $\boldsymbol{K}$. Do this symbolically in Wolfram Mathematica by evaluating Eq. 13 in an operating point $\boldsymbol{x^*} = (z^*,\ 0,\ \bar{u})$, and then computing conditions for the entries of $\boldsymbol{K}$ to make the eigenvalues of $\boldsymbol{A} + \boldsymbol{BK}$ negative.

3. Implement the mathematical model of the setup (Eq. 8; Excercise 3.7.1) with state feedback (Fig. 24) in Simulink. Implement dead time on the fan's motor response (Excercise 3.4.2). Assume the position and velocity of the ball are both known so no Luenberger state observer is needed. Simulate a step in the setpoint from 20 cm to 40 cm. Assess the response for different state feedback gain matrices. Use the builtin Matlab function `lqr(A, B, Q, R)` to design state feedback gain matrices. Experiment with varying the state-costs of the height setpoint Q_{11} and the velocity setpoint Q_{22} when designing LQR gains. Find a suitable gain matrix to try out on the experimental setup.

Luenberger state observer Only the position of the table tennis ball is monitered by the IR sensor while state feedback requires all system states, both the position and velocity of the table tennis ball, to be known. To resolve this problem, a Luenberger state observer can be designed because the system is observable (Eq. 19). Similar to the state feedback, the observer is designed using the linearised system. The state observer is a linear system consisting of a predictor and a corrector,

$$\underbrace{\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{Bu}}_{\text{predictor}} + \underbrace{\boldsymbol{L}\left(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{x}} - \boldsymbol{Du}\right)}_{\text{corrector}} \tag{22}$$

which models the evolution of the system's states $\hat{\boldsymbol{x}} = (\hat{z},\ \hat{v})$ by using the sensor's measurement of the ball's height y, the fan speed u, and a (linearised) process model as its inputs. The predictor, $\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{Bu}$, uses the system's (linearised) process model, inputs $u(t)$ and an initial condition to compute the dynamic trajectory of the system's states. However, the predicted state trajectory will diverge from the actual state trajectory as time progresses. Therefore a corrector term, $\boldsymbol{L}\left(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{x}} - \boldsymbol{Du}\right)$, equal to $\boldsymbol{L}$ times the difference between the sensor's measurement y and the predicted measurement using the model $\hat{y} = C\hat{x} - Du$, is added to the equation. Designing the matrix $\boldsymbol{L}$ so that the eigenvalues of the matrix $(\boldsymbol{A} - \boldsymbol{LC})$ are strongly negative drives the predictive error of the observer $\boldsymbol{\epsilon} = \boldsymbol{x} - \hat{\boldsymbol{x}}$ to zero. For a controllable and observable system, the **separation principle** states that matrices $\boldsymbol{K}$ and $\boldsymbol{L}$ can be designed independently.

Exercise 3.16: Designing an appropriate corrector gain matrix $\boldsymbol{L}$

1. What size should the corrector gain matrix $\boldsymbol{L}$ be?

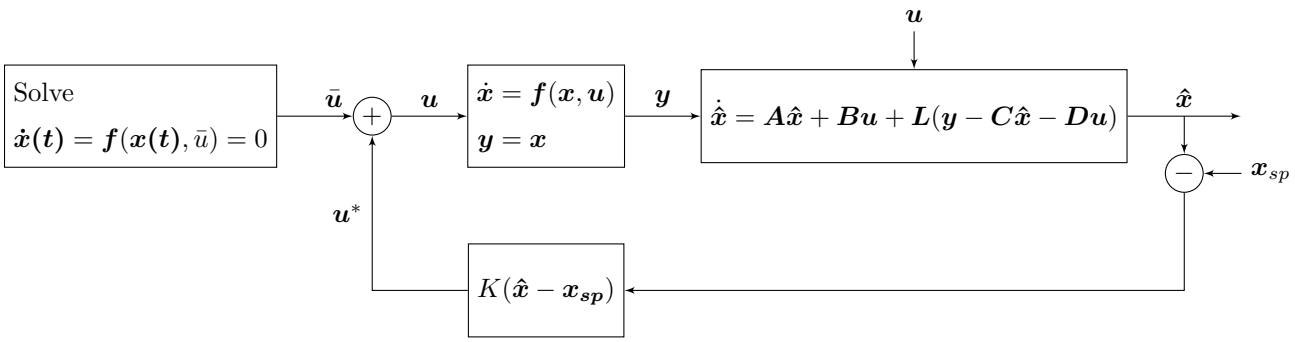


Figure 24: Schematic representation of a state feedback strategy with Luenberger state observer to bring a non-linear system to a setpoint $\boldsymbol{x}_{\mathrm{sp}}$.

2. Use the system matrix $\boldsymbol{A}$ and output matrix $\boldsymbol{C}$ of the linearised system (Eq. 13) to design a suitable state observer corrector gain matrix $\boldsymbol{L}$. Do this symbolically in Wolfram Mathematica by evaluating Eq. 13 in an operating point $\boldsymbol{x^*} = (z^*,\ 0,\ \bar{u})$, and then computing conditions for the entries of $\boldsymbol{L}$ to make the eigenvalues of $\boldsymbol{A} - \boldsymbol{LC}$ negative.

3. Adjust your Simulink implementation of the mathematical model of the setup with state feedback (Excercise 3.15.4) to include a Luenberger state observer. Based on the separation principle, the state feedback gain matrix $\boldsymbol{K}$ and the corector gain matrix $\boldsymbol{L}$ can be designed independently. Assume only the position of the ball is measured and add artifical measurement noise, which you subsequently filter (Excercise 3.7.4). Simulate a step in the setpoint from 20 cm to 40 cm and visualise the difference between the measured and estimated heights of the table tennis ball, as well as the estimated velocity, for different corrector gain matrices $\boldsymbol{L}$.

3.9.2 Testing LQR control on the experimental setup

Exercise 3.17: LQR control of the experimental setup

1. Open the Simulink model **LQR.slx**, which contains the implementation of an LQR controller. Be sure to change the solution algorithm to the fixed-step Backward Euler

`ode1be` method. The state feedback gain matrix's elements k_{11} and k_{12} can be tuned using the knobs. Raise the setpoint from 0 cm to 40 cm and observe the system's response. How does the response compare to that of a PID controller?

4 Solutions

Linearisation, controllability and observability in Wolfram Mathematica

The following Wolfram Mathematica code was used to compute the derivatives of Eq. 8 symbolically, in order to obtain a linearised mathematical model of the setup (Eq.13, Section 2.5.3). Mathematica was further used to check the controllability and observability of the setup (Section 2.5.4).

```
(*Linearise system symbolically*)
Clear[v, u, rho, r, A, Cd, m, g, alpha, beta]
f = {{v}, {(0.5*rho*A*Cd/m)*(alpha*u^beta - v)^2 - g}} ;
FullSimplify[D[f, z]]
FullSimplify[D[f, v]]
FullSimplify[D[f, u]]

(*Define linearized model in generic point x=(z,v,u)*)
a = {{0, 1}, {0, (rho*A*Cd/m)*(v - alpha*u^beta)}};
b = {{0}, {(alpha*beta*rho*A*Cd/m)*u^(beta - 1)*(alpha*u^beta - v)}};
c = {{1, 0}};
d = {{0}};
model = StateSpaceModel[{a, b, c, d}]

(*Check controllability*)
v = 0;
FullSimplify[ControllabilityMatrix[model]]
MatrixRank[ControllabilityMatrix[model]]
ControllableModelQ[model]

(*Check observability*)
ObservabilityMatrix[model]
MatrixRank[ObservabilityMatrix[model]]
ObservableModelQ[model]
```

Excercise 3.1

1. NA.
2. Approximately 162 (may differ from setup to setup).
3. Approximately 152 (may differ from setup to setup).
4. Approximately 153 (may differ from setup to setup). Because the fan speed needed to hover the ball at 30 cm and 50 cm are almost identical, the assumption that there is no pressure drop along the length of the tube is valid. In other words, the air's speed in the tube is quasi-independent of the position in the tube and hence Eq. 5 is valid.

Excercise 3.2

1. NA.
2. The signal is noisy and its overall quality is poor, especially when the table tennis ball is in the bottom 20 cm of the tube. This is caused by the low sensitivity of the sensor in this region (Fig. 13). Between 58 cm - 63 cm, the sensor falsly thinks the ball's height drops to around 40 cm.

Excercise 3.3

1. An example of a Simulink model can be found in Fig. 25. The signal quality has improved.
2. The low pass filter introduces a delay in the signal (dead time). A lower time constant results in a noisier signal but less dead time. A higher time constant results in a smoother but more delayed signal. Finding an appropriate time constant is a balancing act between signal quality and dead time. Both a poor quality signal quality and dead time lower a feedback controller's ability to stabilise the system. We recommend a (fast) filter time constant $\tau_F = 0.05\text{-}0.10\ s$ when using a PID controller on the experimental setup.
3. A low-pass filter, which amplifies low-frequency signals and attenuates high-frequency signals. In our case, the sensor's noise is in the high-frequency spectrum.

Excercise 3.4

1. See Fig. 28.
2. $t_d \approx 0.5\ s$, see Fig. 27.

Excercise 3.5

1. The amplitude of the sinusoidal changes in the ball's height (output signal) is very small, almost non-existent. The ball hovers in the tube.
2. As the frequency of the input signal is lowered, the amplitude of the output increases. The input and output signals are almost fully out of phase (-180 degrees).
3. The amplitude of the output now covers the entire length of the tube. The ball starts at the bottom of the tube and reaches the top of the tube during each cycle. Ignoring the dead time induced by the additional energy needed to get the ball in the air, the input and output signals are now more or less in phase (0 degrees).
4. The approximate Bode diagram is characteristic of a first-order system (Fig. 26). The system's breakout frequency is located at approximately 1 rad/s. As previously discussed, a first-order system is a low-pass filter, as low-frequency inputs are amplified while high-frequency inputs are attenuated.

Excercise 3.6

1. $\alpha = 0.19,\ \beta = 0.63$. May differ between setups.
2. $\bar{u} = 151$, and thus in line with the value found during Excercise 3.1.

3. The mathematical model is able to adequately describe the time course of the validation experiment (Fig. 20), as gauged by a small mean average error of 2.2 cm. Performing *out-of-sample* testing on a *validation* dataset allows us to confidently assert the validity and predictive performance of our mathematical model. Other performance indicators could be the values of the parameters, which must make sense if the parameters have physical meaning, or the size of confidence intervals on the parameter estimates (obtained using the Fisher Information Matrix or Markov-Chain Monte-Carlo sampling).

Excercise 3.7

1. The integral control action converges to $\bar{u}$ (Fig. 29). The PID controller's integral action thus 'learns' what fan speed is needed to hover the ball. Therefore, it is best to initialise the integral control action as $\bar{u}$ (`Initial condition` in the I-controller's `Integrate` block). The Simulink model, including the in-flight dead time and noisy measurements, is shown in Figs. 36, 37, and 38.
2. $K_P = 1000;\ K_I = 0;\ K_D = 100;\ K_F = 1$ (Fig. 30).
3. More in-flight dead time destabilises the closed-loop response (Fig. 31).
4. $K_P = 50;\ K_I = 0;\ K_D = 0;\ K_F = 1$ (Fig. 32).
5. The derivative control action destabilises the closed-loop response. Large proportional gains can also destabilise the closed-loop system, even in the abscence of a setpoint change, this is due to the combination of the sensor's noise and the system's dead time (Fig. 33).
6. As the I-controller is essentially 'learning' how to hover the ball by finding $\bar{u}$, setting $K_I = 0$ makes the system incapable of adjusting $\bar{u}$. However, external disturbances, such as the electronic components "warming up", may cause changes of $\bar{u}$ over time. Therefore, the use of only a P-controller will make the system incapable of handling external disturbances (Fig. 34). An alternative controller's response ($K_P = 80;\ K_I = 10;\ K_D = 50;\ K_F = 1$) is shown in Fig. 35.
7. The use of a Smith predictor enhances the closed-loop stability of the system. Higher proportional gains K_P can now be applied before the system becomes unstable.

Excercise 3.8

1. The controller gains tuned using the empirical method by Cohen and Coon are shown in Tables 3 and 4.
2. The response to a step-wise increase in the setpoint height from 20 cm to 40 cm for the controllers listed in Table 3 are shown in Fig. 39 (with dead time but without noisy measurements). Only the P-controller leads to an adequate closed-loop response, the tuned integral gains K_I are too high to achieve a stable closed-loop response. Note how there is no offset when using a P-controller (Q: How is this related to the system's stability?). The Cohen-Coon tuning rules are based on the assumption that the system's response are adequately approximated by a linear first-order system. As our system is non-linear with neutral stability, the approximation likely does not hold. Always use empirical tuning rules based on linear approximations with care when designing a suitable controller.

Table 3: Controller gains tuned using the empirical method by Cohen and Coon. For an initial step size of $u_{\max} = 170$, $K = 0.5/(170 - 151) = 0.026$, $\tau = 0.8\ s$ and $t_d = 0.5\ s$.

	K_P	K_I	K_D
P	73	-	-
PI	58	78	-
PID	91	91	14

Table 4: Controller gains tuned using the empirical method by Cohen and Coon. For an upwards step during the U-shaped response part of the block response experiments with $u_{\text{step}} = 10$, $K = (0.5 - 0.1)/10 = 0.0025$, $\tau = 1.5\ s$ and $t_d = 0.5\ s$.

	K_P	K_I	K_D
P	67	-	-
PI	56	56	-
PID	85	92	15

Excercise 3.9

1. The ultimate control gain $K_u \approx 114$ and the period of the resulting permanent oscillation $P_u \approx 3.5\ s$ were determined in silico using the calibrated mathematical model (Fig. 40). It is possible to determine the ultimate control gain using the setup. The gains of the controllers tuned using the rules of Ziegler and Nichols are given in Table 5.
2. The resulting system's closed-loop response using the tuned controller gains is shown in Fig. 41. The P and PI controllers provide a stable closed-loop response while the PID controller does not. The P controller's response is preferred over the PI controller's response. Similar to the tuning method of Cohen and Coon, the integral gain K_I is too high. Based on the empirical method from Ziegler-Nichols, the use of a P-controller seems most adequate.

Table 5: Controller gains tuned using the empirical method by Ziegler and Nichols. For an ultimate control gain $K_u \approx 114$ and period of the resulting permanent oscillation of $P_u \approx 3.5\ s$.

	K_P	K_I	K_D
P	57	-	-
PI	52	17.8	-
PID	67	38.3	153

Excercise 3.10

1. The controller with gains $K_P = 80;\ K_I = 10;\ K_D = 50;\ K_F = 1$, and filter time constant $\tau_F = 0.05\ s$ works adequately for setpoint changes and disturbances (Excercise 3.7.6). Note how the integral controller's output converges to $\bar{u}$ (at least after some time, as the electronic components warm up).

Excercise 3.11

1. No, the ball will repeatedly go up and down the tube. The ball's average position will be located below the setpoint, and this is termed "offset".
2. The I ccontroller's output gradually increases to a value over the fan's equillibrium speed $\bar{u}$, raising the ball of the stand. As the ball crosses the desired setpoint, the I controller's starts gradually decreasing but too slow; the ball makes it way to the top of the tube. As the I controller's output gradually decreases back under the fan's equillibrium speed $\bar{u}$, the ball starts making its way down the tube again and the cycle is repeated. The obtained system is oscillatory and thus unstable, the I controller suffers from "windup".
3. With the proportional gain set to $K_P = 80$, the system's reaction to setpoint changes and disturbances is adequate and stable. Integral or derivative action does not seem needed.
4. Without the integral controller 'learning' the new (increased) equillibrium fan speed $\bar{u}$, the controller is unable to maintain a height of 40 cm. With the integral controller enabled, its output gradually increases and the ball maintains its height of 40 cm. Removing the disturbance suddenly results in "integral windup".

Excercise 3.12

1. The system becomes less stable because the derivative controller amplifies the high-frequent noise from the sensor. Intuitively, this makes sense: measurement noise is a small change in the absolute value of the measurement over a very small period of time. Noise thus has a large rate of change (derivative), resulting in a large D-controller action. What you observe is the D-controller reacting to the noise of the IR sensor. However great a D-controller may seem at stabilising a system, this is only true on paper. Ultimately, the use of derivative action can be beneficial, but it has to be used in the right situations and in the right amounts. Your starting assumption should be to see if the process will run satisfactorily on PI control alone. Adding derivative should be done cautiously and with appropriate filtering of the signal.
2. Lowering the derivative filter gain to $K_F = 0.5$ makes the system more stable because the sensor's noise is filtered more (derivative filter time constant is lowered to 2 s). Increasing the derivative filter gain to $K_F = 2$ makes the system less stable.

Excercise 3.13

1. Added dead time destabilises the closed-loop system.

Excercise 3.14

1. The magnitude of the amplitude is close to zero, the signals are out of phase.
2. The magnitude of the amplitude is close to one, the signals are out of phase.
3. The magnitude of the amplitude is higher than one, the signals are shifted by roughly 90 degrees.
4. The magnitude of the ampltiude is close to one, the signals are in phase.
5. The approximate Bode diagram is characteristic of a second-order system (Fig. 42).

Excercise 3.15

1. The system's states $\boldsymbol{x} = (z, v)$ are of size 2×1, and the size of $u^* = \boldsymbol{K}\left(\boldsymbol{x} - \boldsymbol{x}_{\text{sp}}\right)$ must be 1×1. Therefore, the feedback gain matrix $\boldsymbol{K}$ must be of size 1×2. The feedback gain matrix is typically of size $m \times n$, where m is the number of inputs and n is the number of system states.
2. The following Wolfram Mathematica code can be used to compute the values of k_{11} and k_{12} that result in the real parts of the eigenvalues of the matrix $\boldsymbol{A} + \boldsymbol{B} \cdot \boldsymbol{K}$ to be equal to -5 and -2, thereby stabilising the system. We obtain $k_{11} = 118$ and $k_{12} = 30$.

```
(*Define model parameters*)
rho = 1.225;
r = 0.02;
A = 4*Pi*r^2;
Cd = 0.47;
m = 0.00283;
g = 9.81;
alpha = 0.19;
beta = 0.63;

(*Define the steady state fan speed*)
ubar = (2*m*g/(rho*A*Cd*alpha^2))^(0.5/beta);

(*Define operating point*)
u = ubar;
v = 0;

(*Define linearized model in generic point x=(z,v,u)*)
a = {{0, 1}, {0, (rho*A*Cd/m)*(v - alpha*u^beta)}};
b = {{0}, {(alpha*beta*rho*A*Cd/m)*u^(beta - 1)*(alpha*u^beta - v)}};
c = {{1, 0}};
d = {{0}};
model = StateSpaceModel[{a, b, c, d}]

(*Compute conditions on entries in matrix K guaranteeing negative \
eigenvalues of A + B.K *)
Clear[k11, k12]
eigK = FullSimplify[Eigenvalues[a - b . {{k11, k12}}]];
Reduce[Re[eigK[[1]]] == -5 && Re[eigK[[2]]] == -2  , {k11, k12}, Reals]
```

If $k_{11} = 115$, we can use Mathematica to find what values of k_{12} are needed to stabilise the system. We find that if $k_{11} = 115$ then $k_{12} > 21$ to stabilise the system.

```
Reduce[Re[eigK[[1]]] < 0 && Re[eigK[[2]]] < 0 && k11 == 115, {k12}, Reals]
```

Additionaly, Wolfram Mathematica can find *optimal* values of the feedback gain matrix $\boldsymbol{K}$ by optimising an LQR problem. For a state cost matrix $\boldsymbol{Q} = [[14000, 0]; [0, 2000]]$, we find optimal gains $k_{11} = 118,$ and $k_{12} = 34$.

```
(*Design gains using LQR*)
q = {{14000, 0}, {0, 2000}};
r = {{1}};
K = LQRegulatorGains[model, {q, r}, "FeedbackGains"]
OLpoles = LQRegulatorGains[model, {q, r}, "OpenLoopPoles"]
CLpoles = LQRegulatorGains[model, {q, r}, "ClosedLoopPoles"]
```

3. The step response from 0 cm to 40 cm, in the presence of 0.5 s of delay on the fan's motor response, and in the abscense of measurement noise, is given in Figure 43. Using $k_{11} = 118,$ and $k_{12} = 34$, found by optimising an LQR problem with $\boldsymbol{Q} = [[14000, 0]; [0, 2000]]$. The Simulink model used to perform the simulation is shown in Fig. 44.

Excercise 3.16

1. The state observer corrector gain matrix $\boldsymbol{L}$ must be of size $n \times k$, where n is the number of system states, and k is the number of measured outputs. For our system $\boldsymbol{L}$ is thus of size 2×1.
2. The following Wolfram Mathematica code can be used to compute the values of l_{11} and l_{21} that result in the real parts of the eigenvalues of the matrix $\boldsymbol{A} - \boldsymbol{L} \cdot \boldsymbol{C}$ to be equal to -500 and -500, thereby stabilising the system quickly. We obtain $l_{11} = 995$ and $l_{21} = 245000$.

```
(*Compute conditions on entries in matrix L guaranteeing negative eigenvalues of A - C.L *)
Clear[l11, l21]
eigL = FullSimplify[Eigenvalues[a - {{l11}, {l21}} . c]];
Reduce[Re[eigL[[1]]] == -500 && Re[eigL[[2]]] == -500 ,  {l11, l21}, Reals]
```

3. The output of the Luenberger state observer during the closed-loop response for a setpoint change from 20 cm to 40 cm, with a dead time of $t_d = 0.5\ s$ on the fan's response and in the abscense of measurement noise, and using state feedback with $k_{11} = 118$ and $k_{12} = 34$, is shown in Fig. 45. For $l_{11} = 995$ and $l_{21} = 245000$, the estimated velocity differed from $v = 0\ m \cdot s^{-1}$ when the ball was at rest (prior to the setpoint change), resulting in an unsatisfactory observer design. The result shown here was obtained for $l_{11} = 100$ and $l_{21} = 250000$, and is satisfactory. The Simulink model used to perform the simulation is shown in Fig. 44. To implement the Luenberger state observer in Simulink, we rewrite the observer $\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x} - Du) = (A - LC)\hat{x} + (B - LD)u + Ly$ in state-space format,

$$\begin{cases} \dot{\hat{x}} &= \underbrace{(A - LC)}_{A} \begin{bmatrix} \hat{z} \\ \hat{v} \end{bmatrix} + \underbrace{\begin{bmatrix} (B - LD) & L \end{bmatrix}}_{B} \begin{bmatrix} u \\ y \end{bmatrix}, \\ y &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{C} \begin{bmatrix} \hat{z} \\ \hat{v} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}}_{D} \begin{bmatrix} u \\ y \end{bmatrix}. \end{cases} \tag{23}$$

Excercise 3.17

1. We found that a state feedback gain matrix $\boldsymbol{K} = [118,\ 35]$ and a corrector gain matrix $\boldsymbol{L} = [100;\ 250000]$ resulted in a satisfactory closed-loop response.

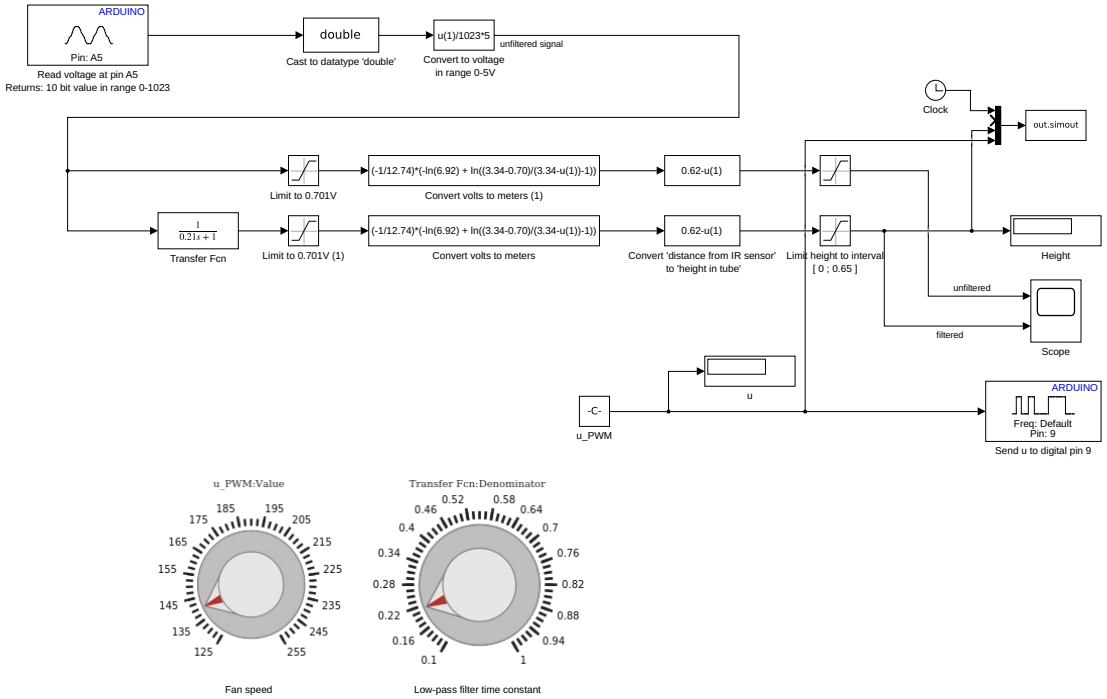


Figure 25: Simulink model used to read the ball's unfiltered and filtered height in open-loop control.

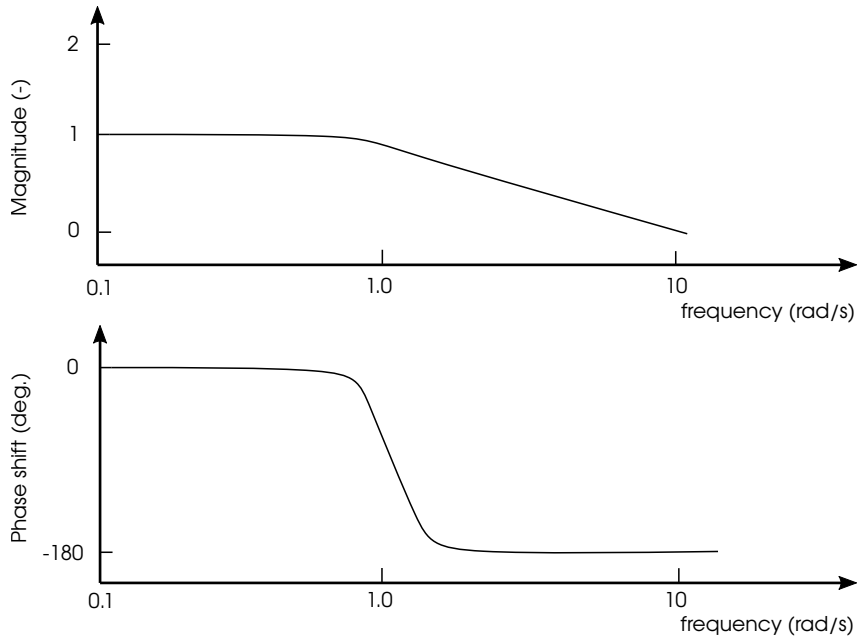


Figure 26: Approximate Bode diagram based on the setup's open-loop frequency response.

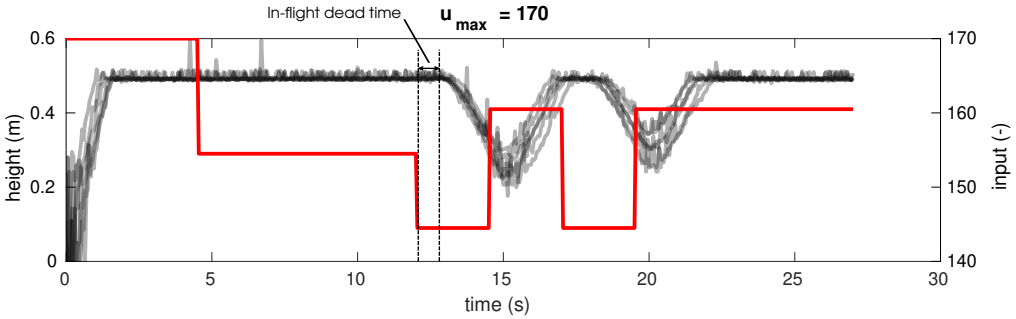


Figure 27: Outcome of the W-shaped response experiment performed using *writeWResponse.m*. The in-flight dead time is approximately $t_d \approx 0.5\ s$.

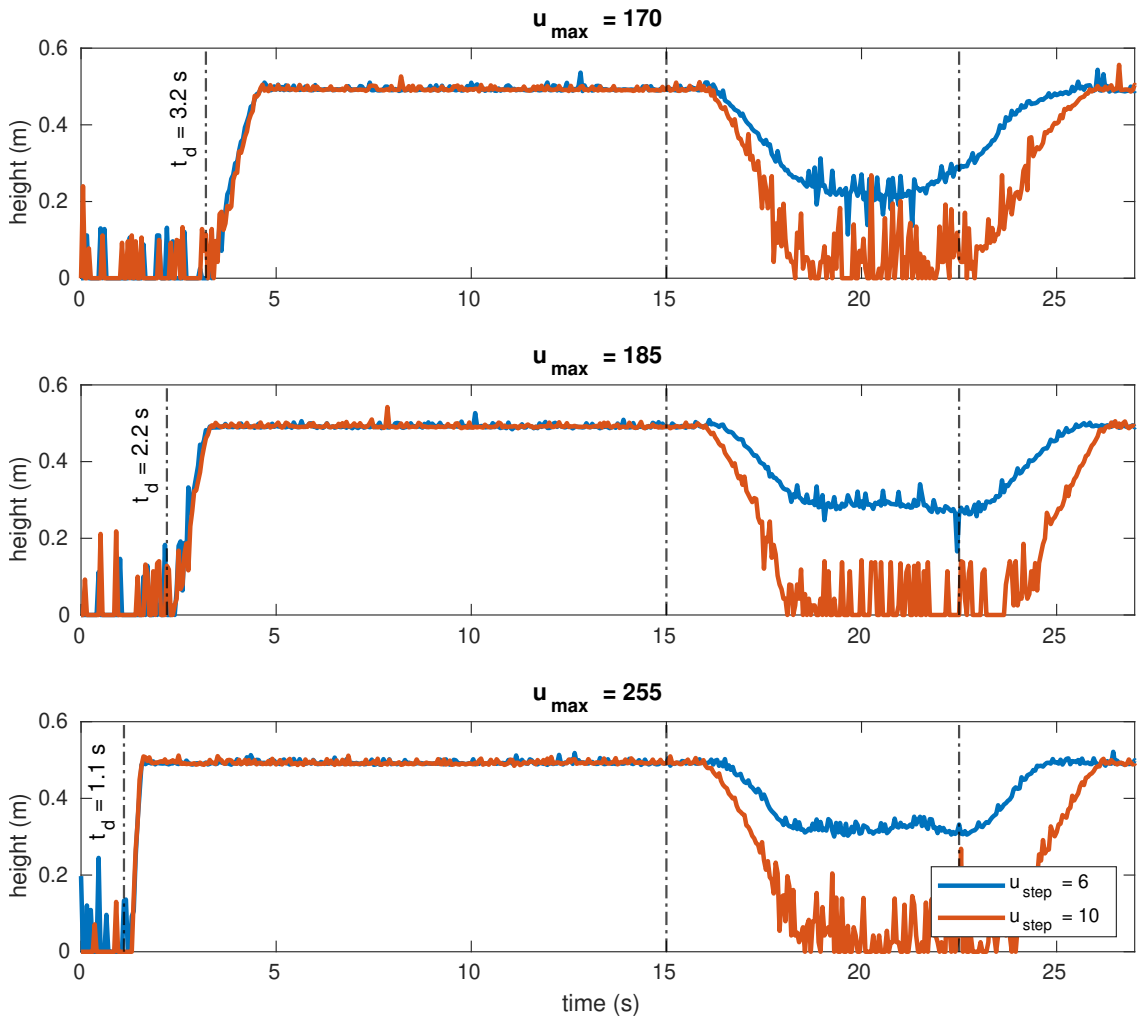


Figure 28: Outcome of the block-shaped response experiment performed using *write-BlockResponse.m*.

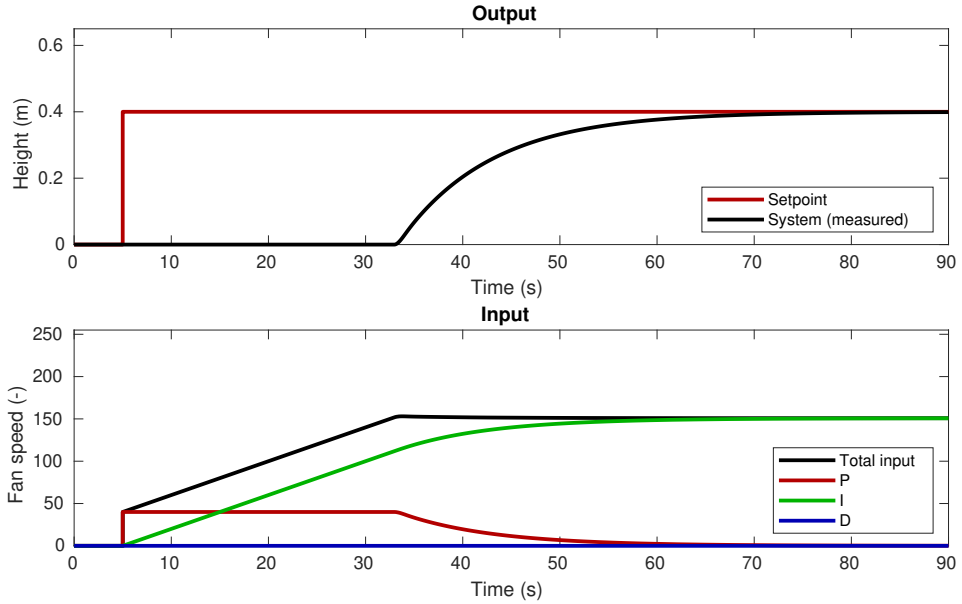


Figure 29: Closed-loop response for a setpoint change from 0 cm to 40 cm, in the abscence of added measurement noise or dead time. $K_P = 100$; $K_I = 10$; $K_D = 0$; $K_F = 1$. The output of the I controller converges to $\bar{u}$.

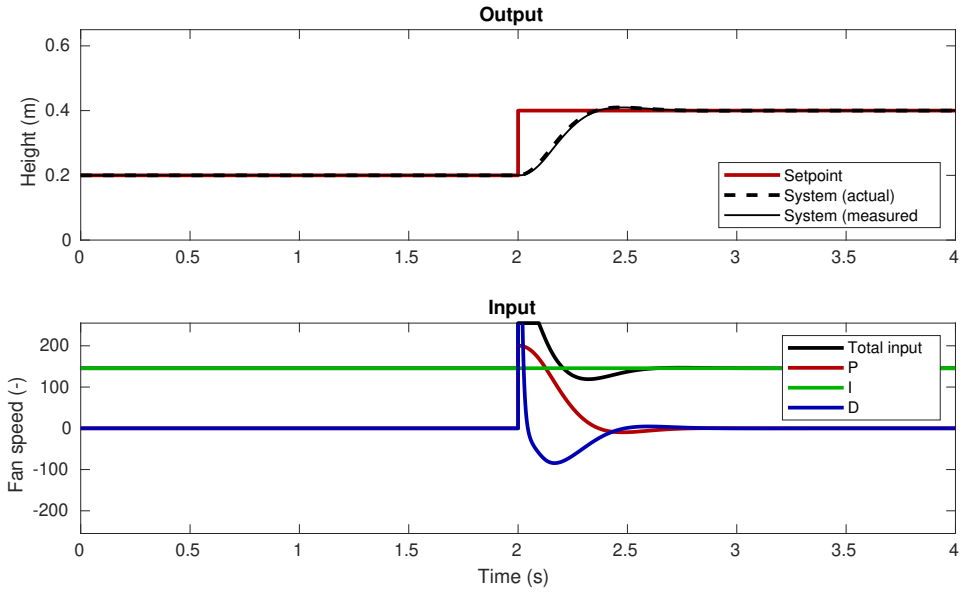


Figure 30: Closed-loop response for a setpoint change from 20 cm to 40 cm, in the abscence of added measurement noise or dead time. $K_P = 1000$; $K_I = 1$; $K_D = 100$; $K_F = 1$.

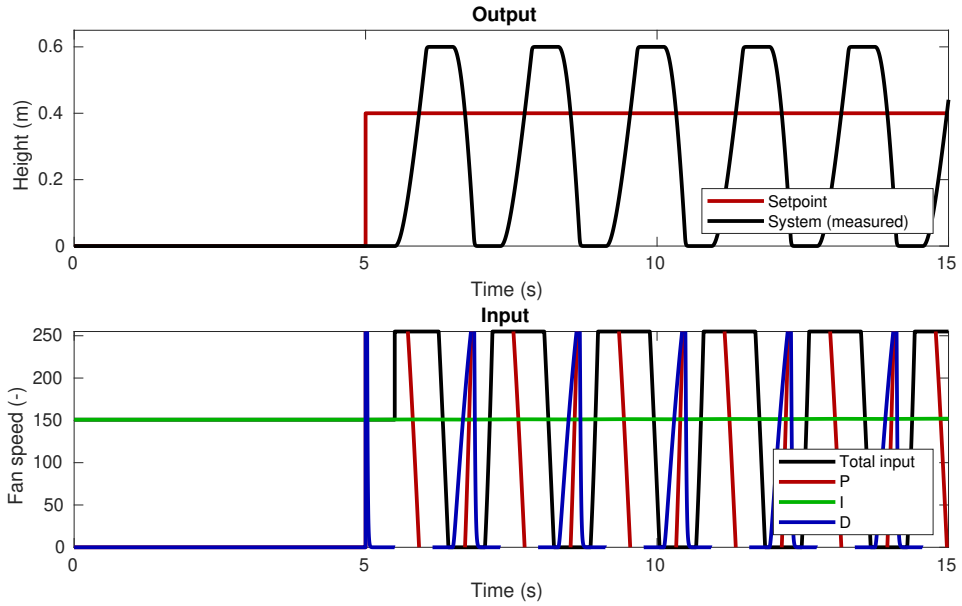


Figure 31: Closed-loop response for a setpoint change from 20 cm to 40 cm, in the abscence of added measurement noise but with a dead time of $t_d = 0.5\ s$ on the fan response. $K_P = 1000$; $K_I = 1$; $K_D = 100$; $K_F = 1$. The closed-loop response has become unstable.

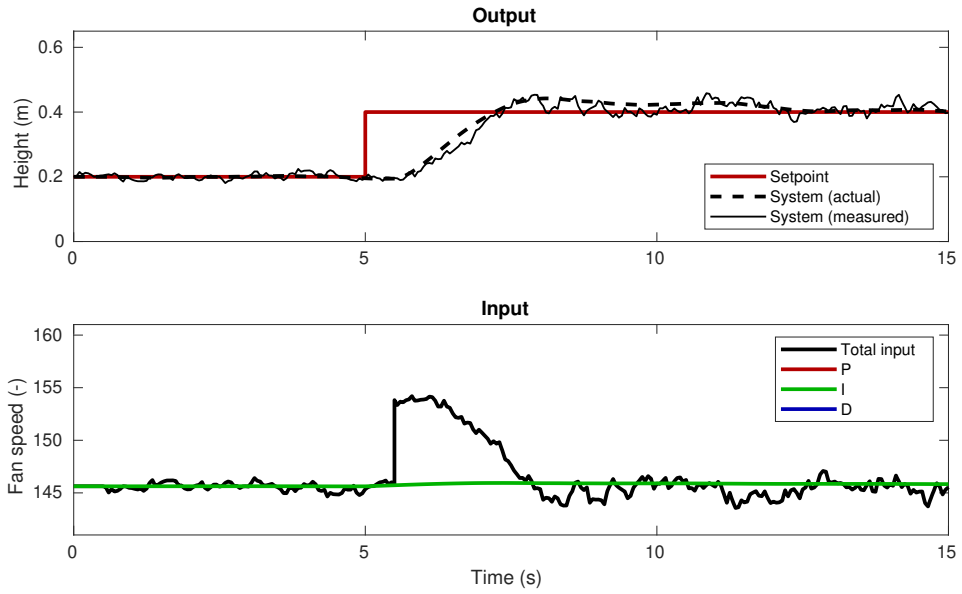


Figure 32: Closed-loop response for a setpoint change from 20 cm to 40 cm, with added measurement noise and subsequent low-pass filtering with time constant $\tau = 0.2\ s$, and with a dead time of $t_d = 0.5\ s$ on the fan response. $K_P = 50$; $K_I = 1$; $K_D = 0$; $K_F = 1$.

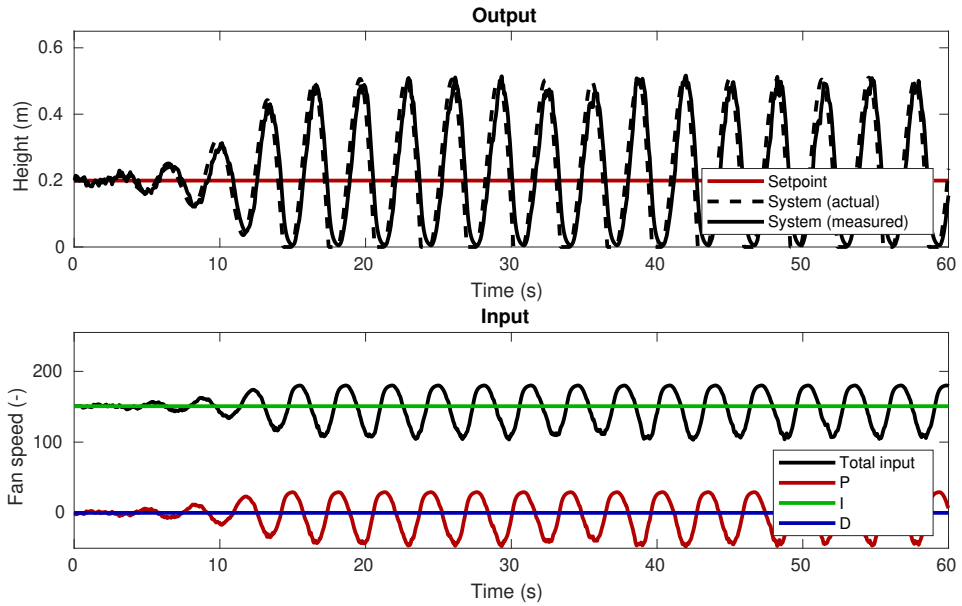


Figure 33: Closed-loop dynamics for high proportional gains in the abscence of a setpoint change. $K_P = 150;\ K_I = 0;\ K_D = 0$.

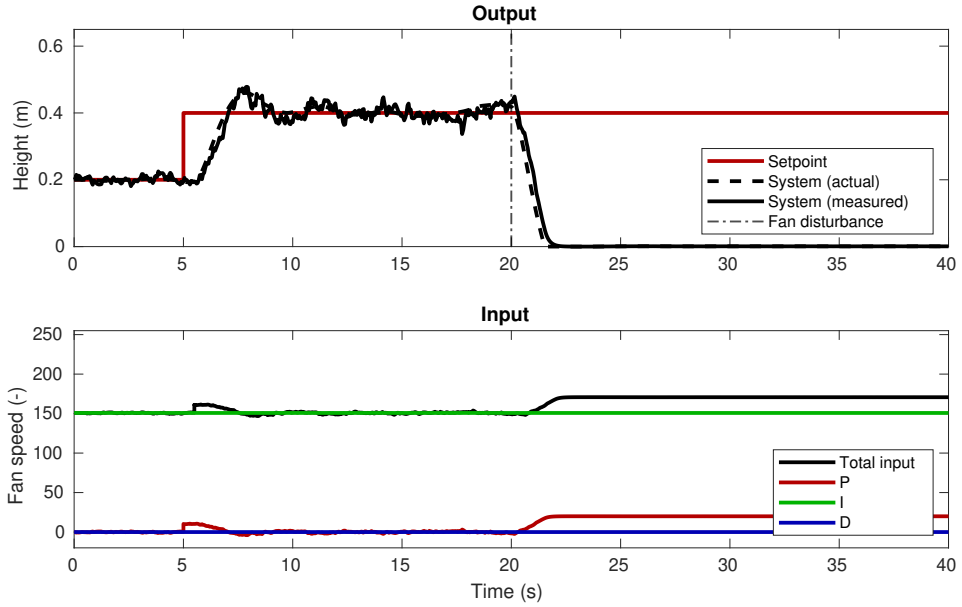


Figure 34: Closed-loop response for a setpoint change from 20 cm to 40 cm at time $t = 5\ s$, followed by a 7.5 % decrease in the fan's effectiveness (modeled as a decrease of the parameter α in Eq. 8) at time $t = 20\ s$. $K_P = 80;\ K_I = 0;\ K_D = 0$.

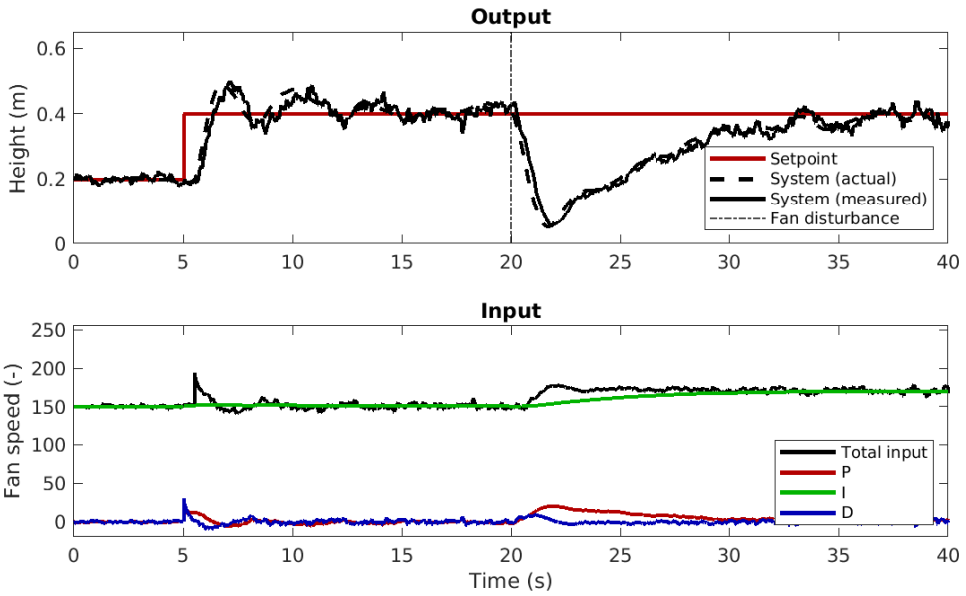


Figure 35: Closed-loop response for a setpoint change from 20 cm to 40 cm at time $t = 5\ s$, followed by a 7.5 % decrease in the fan's effectiveness (modeled as a decrease of the parameter α in Eq. 8) at time $t = 20\ s$. $K_P = 80$; $K_I = 10$; $K_D = 50$; $K_F = 1$.

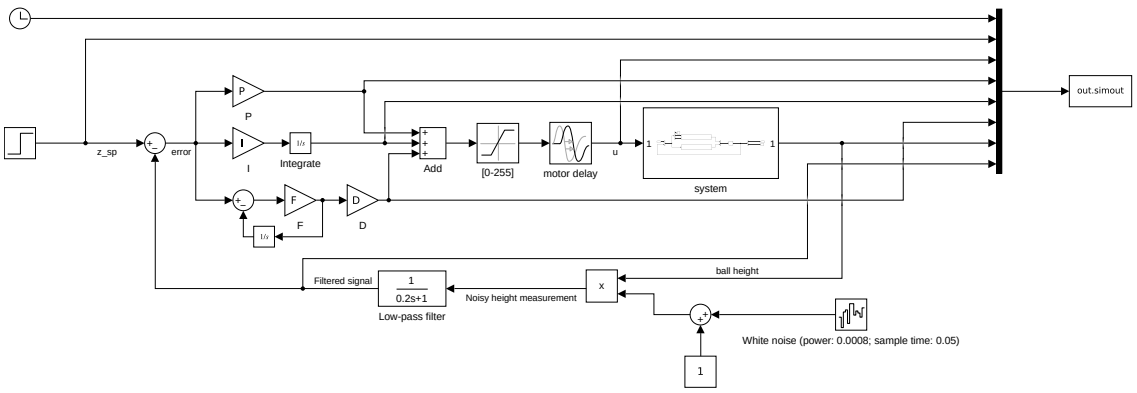


Figure 36: Simulink model *ODEModelPID.slx*. Contains the mathematical model of the setup (Eq. 8) in the subsystem labeled `system` (Fig. 37), in closed-loop with a PID controller. The in-flight dead time is added using a `Transport delay` on the fan's motor response. Noise on the IR proximity sensor's measurement is added using a `White noise` block, and is subsequently filtered using a low-pass filter.

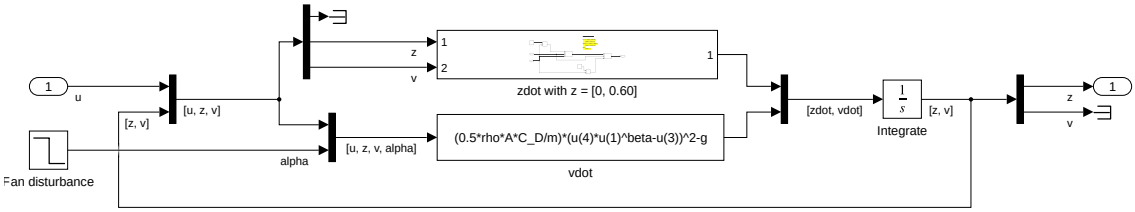


Figure 37: Simulink subsystem `system` in *ODEModelPID.slx* (Fig. 36), containing the implementation of the mathematical model (Eq. 8). Includes saturation of the ball's height between 0 cm and 60 cm.

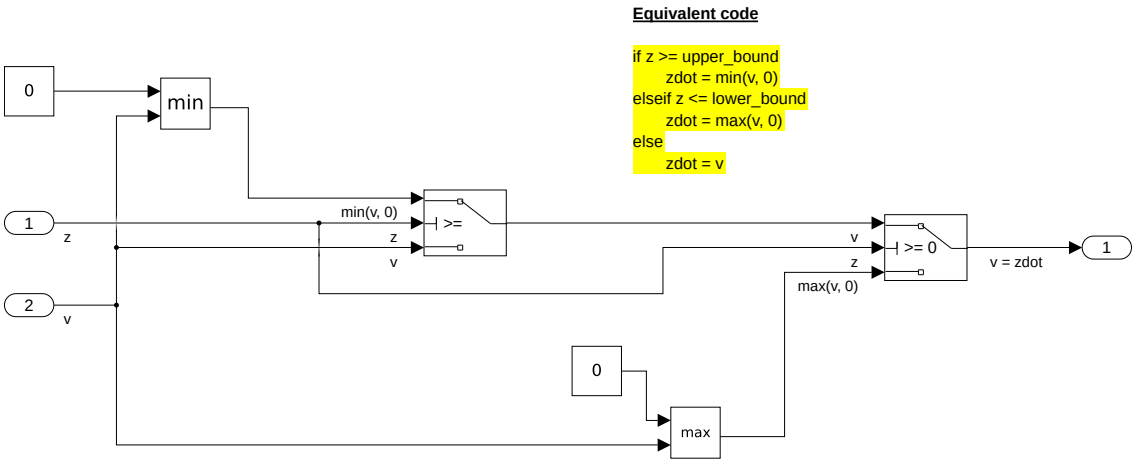


Figure 38: Simulink subsystem `zdot` in *ODEModelPID.slx* subsystem `system` (Fig. 37), containing the implementation of the derivative of the ball's height $dz/dt = v(t)$ (Eq. 8) with saturation of the height $z(t)$ between 0 cm and 60 cm.

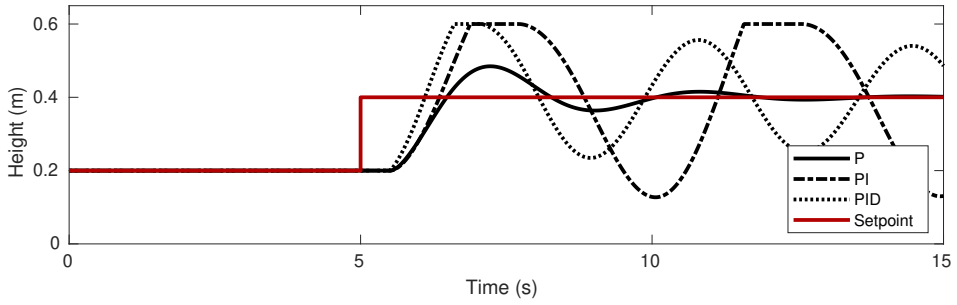


Figure 39: Closed-loop response for a setpoint change from 20 cm to 40 cm, using P, PI, and PID controllers tuned using Cohen and Coon's empirical method (Table 3).

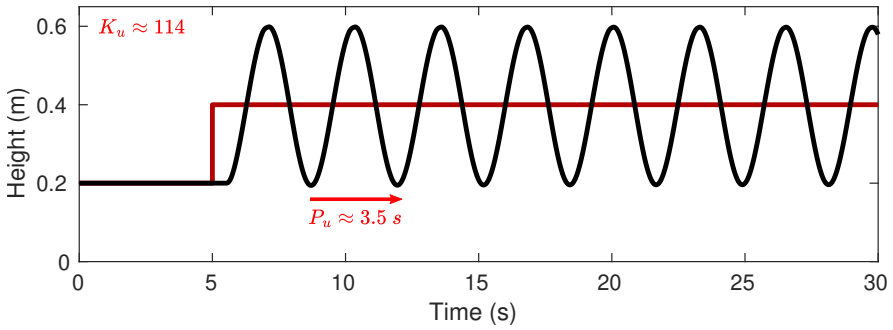


Figure 40: Closed-loop response for a setpoint change from 20 cm to 40 cm, with a P-controller set to the ultimate control gain of $K_u \approx 114$. The period of the resulting permanent oscillation is equal to $P_u \approx 3.5\ s$.

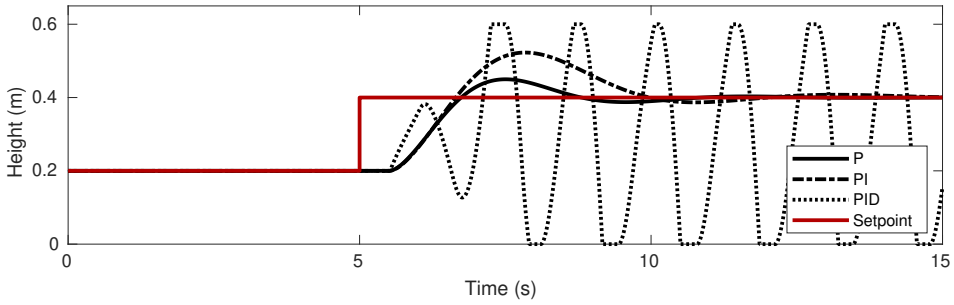


Figure 41: Closed-loop response for a setpoint change from 20 cm to 40 cm, using P, PI, and PID controllers tuned using Ziegler and Nichols empirical method (Table 5).

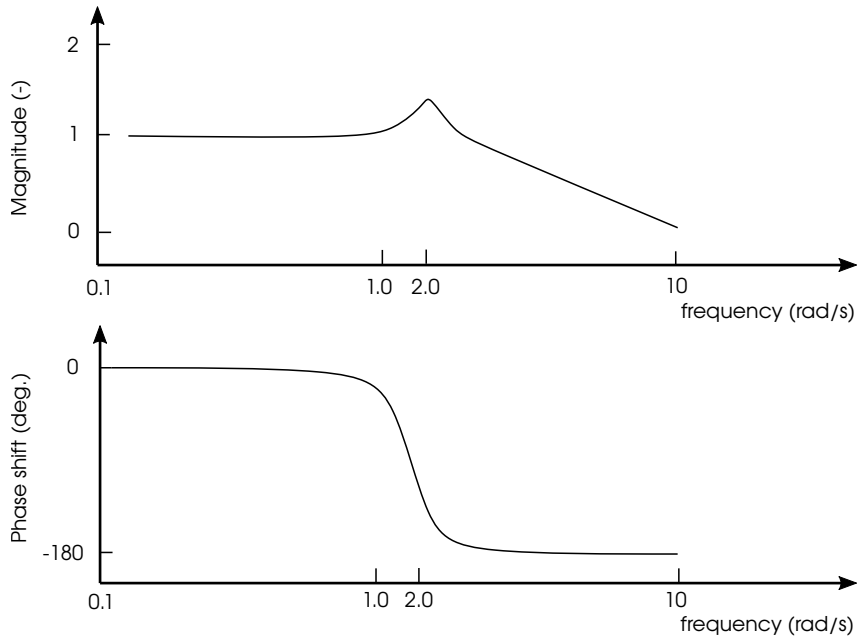


Figure 42: Approximate Bode diagram based on the setup's closed-loop frequency response.

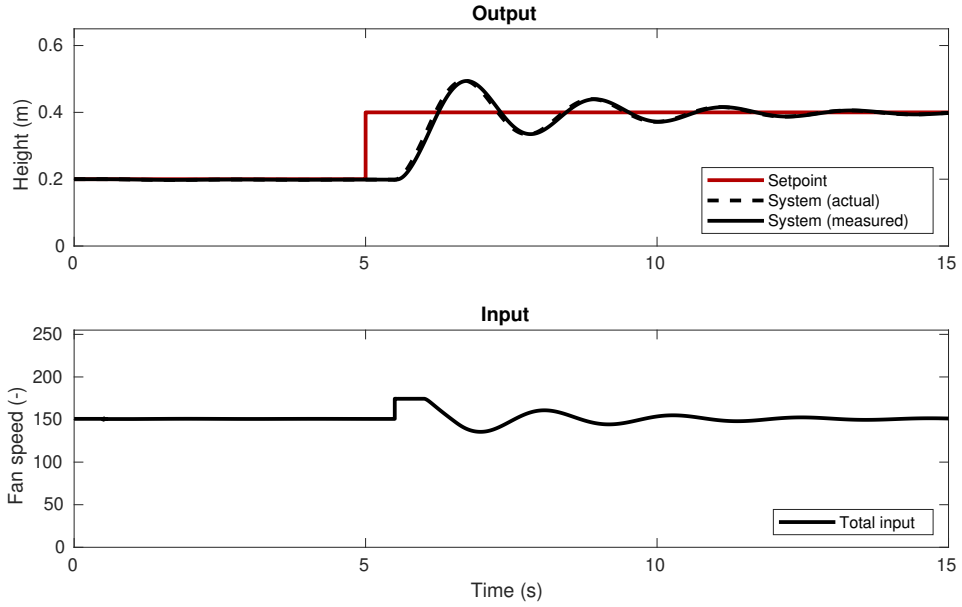


Figure 43: Closed-loop response for a setpoint change from 20 cm to 40 cm, with a dead time of $t_d = 0.5\ s$ on the fan's response and in the abscense of measurement noise. Using state feedback with $k_{11} = 118$ and $k_{12} = 34$.

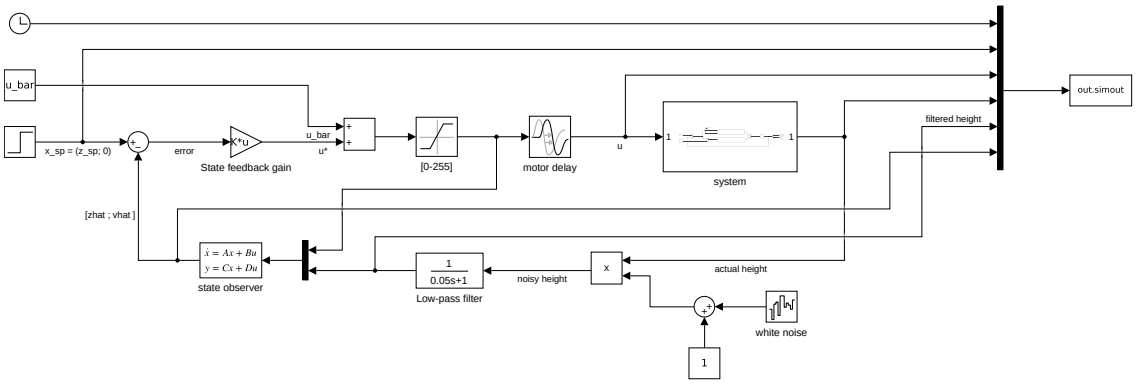


Figure 44: Simulink model *ODEModelLQR.slx*. Contains the mathematical model of the setup (Eq. 8) in the subsystem labeled `system` (Fig. 37), in closed-loop with state feedback. Includes a Luenberger state observer to estimate the ball's velocity, which is not measured. The in-flight dead time is added using a `Transport delay` on the fan's motor response. Noise on the IR proximity sensor's measurement is added using a `White noise` block, and is subsequently filtered using a low-pass filter.

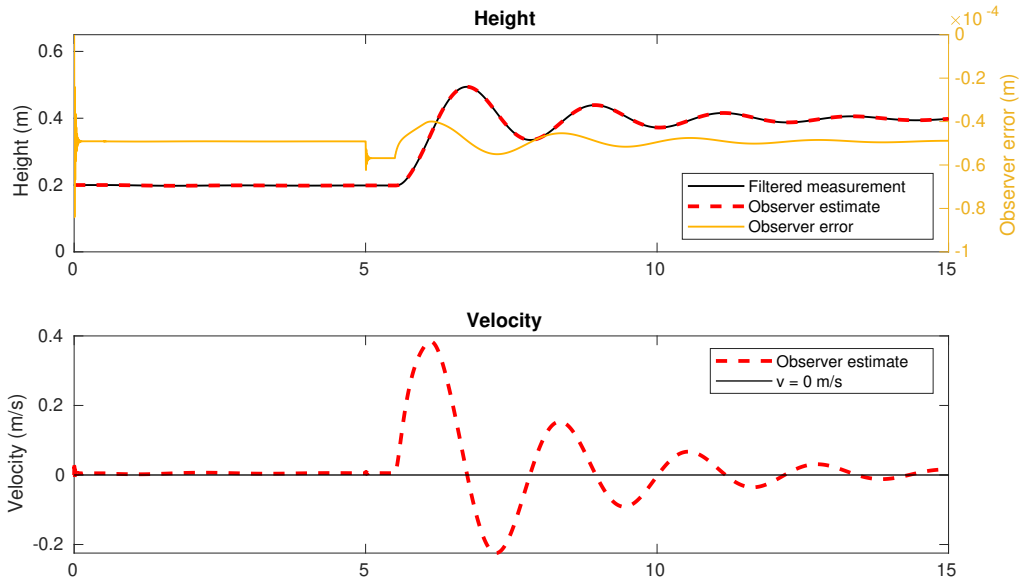


Figure 45: Output of the Luenberger state observer during the closed-loop response for a setpoint change from 20 cm to 40 cm, with a dead time of $t_d = 0.5\ s$ on the fan's response and in the abscense of measurement noise, and using state feedback with $k_{11} = 118$ and $k_{12} = 34$ (Fig. 43). For $l_{11} = 995$ and $l_{21} = 245000$, the estimated velocity differed from $v = 0\ m \cdot s^{-1}$ when the ball was at rest (prior to the setpoint change), resulting in an unsatisfactory observer design. The result shown here was obtained for $l_{11} = 100$ and $l_{21} = 250000$, and is satisfactory.