TTK4115 – Linear System Theory

Helicopter lab assignment

Department of Engineering Cybernetics
NTNU

Version 5.0 August 2019, SVR

Previous versions:
4.5, August 2015, KG
4.4, August 2013, MH
4.3, August 2012, MH
4.2, August 2011, DB
4.1, August 2010, DB
4.0, August 2009, JR
3.8, August 2008, AAE
3.7, August 2007, JM
3.6, August 2006, JB
3.5, August 2005, JH
3.4, August 2005, JS
3.3, August 2003, JS
3.2, September 2002, ESI
3.1, September 2001, ESI
3.0, June 2001, TAJ

1. Introduction

A. Preliminaries

The purpose of this assignment is to:

- Derive a model for a dynamic system (a helicopter) that can be used for control purposes.
- Implement PD controllers on a physical system in a real-time environment.
- Utilize and tune a linear optimal control algorithm (LQR).
- Gain experience with inertial measurement units (IMU).
- Implement a Kalman filter and utilize this algorithm for output feedback.

A.1. Evaluation and grading

The completion of this lab assignment is required to pass the course. A report and an oral presentation will account for 50% of the final grade.

A stapled copy of the report is to be handed in at the oral presentation. Moreover, a pdf file of the report, as well as all Matlab/Simulink files needed for the demonstration, is to be be uploaded in accordance with the specifications found on Blackboard.

Oral presentation

- The oral evaluation consists of practical demonstrations in the lab, accompanied by questioning. The questions will be related to theory from the course, as it pertains to your control system implementation.
- At the oral presentation you will show results from the different tasks of the project. Make sure you save each task (where the code/simulink diagram changed) as a separate file so that you can show it easily at the presentation.
- The oral presentation will be graded based on the performance of the helicopter and your theoretical understanding.

Report

The report is to be structured in the same way as the assignment and written in English. Avoid lengthy appendixes and keep the document brief and to the point.

The contents of the report should include:

- Answers to all questions posed in this document, including derivations, calculations, and experimental/analytical results.
- Thorough discussions on relevant observations. In particular, how theory relates to practice.

- Documented Matlab code and Simulink diagrams where applicable.
- Plots of measured/computed time-series to document experimental/analytical results.
- Numerical values for gains and physical constants, with appropriate units.

You do not need to include a separate chapter to introduce the helicopter system.

Consider using LaTeXwhen authoring the report. See the ITK Labreport guide/template on Github for some useful hints[1].

A.2. Practical information

- Ten helicopters are found in "Elektrobygget", rooms B117 (Helicopters 1–3), B121 (Helicopters 4–7) and B125 (Helicopters 8–10). You need your key card to access the room[2]. You are assigned to a specific helicopter, and should use that throughout the lab to avoid conflicts with other groups and to minimize time spent on adapting your code to a new helicopter, as they vary slightly.
- Each group will have 6 whole days (8 hours each) reserved in the lab. This means that you will have 48 hours exclusively reserved in the lab for your group. A student assistant will be present for 3 hours each day.
- The lab computers are not connected to the internet, which implies there are no e-mail or other internet-based ways to store your data. The easiest way to store your data is to bring a memory stick to the lab. Note that you need the put the Matlab files that are posted on blackboard on your memory stick before you come to the lab. It is also possible to use the "sambastud" server; see Section D.4 for more details.

A.3. General advice

- Each task is assigned to a day in the lab. It is wise to stay ahead of schedule in case you need more than a full day for one of the later tasks. There is little room for using the helicopter outside of the scheduled time-slots.
- Start the writing process early! It is advisable to write the report as you proceed. There will not be any extra time at the end of the lab to finalize the report.

B. The helicopter

The helicopter consists of a base on which a long arm is attached; see Fig. 1.1. The helicopter head, with two motors and propellers, is attached to one side of the arm. On the other side, a counterweight is placed. The helicopter has three rotational joints. The helicopter arm can rotate about the vertical axis. This will be referred to as the travel of the helicopter. The helicopter head can move up and down with respect to the base. This rotation of the helicopter arm will be referred to as the elevation of the helicopter. Moreover, the helicopter head can tilt with respect to the arm, which will referred to as the pitch of the helicopter. All three joint angles are measured using encoders. The helicopter is actuated by the two motors on the helicopter head, which are connected to the propellers.

[1] These files are also posted on Blackboard

[2] If you are denied access, talk to Lill Hege Pedersen who works in the Department's reception (room D144 in "Elektrobygget"), or contact the teaching assistant.

Fig. 1.1.: Helicopter.

B.1. Hardware

The helicopter lab stations use the Quanser Q8-USB data acquisition (DAQ) cards. These cards are used for both inputs and outputs. The joint angles for pitch, elevation and travel are measured using special counter circuits to read the encoders on the helicopter. The control signals (motor voltages) are set using a digital-to-analog converter (DAC). This signal is amplified by a linear amplifier inside the power supply and fed to the motors.

Joystick

A joystick is used for controlling the helicopter. The joystick displacement is measured by potentiometers in the joystick, and are transmitted to the PC via USB. In the *Joystick* Simulink block this signal is interpreted, and translated into a value in the range from -1 to 1. Within the *Joystick*-block the buttons on the joystick can be read. However, this functionality will not be used in this lab.

Power supply

The power supplied to the motors comes from a power supply located under each lab station's desk. It should be labeled "Quanser VoltPAQ-X2".

C. Safety

- Before you start, read the safety instructions attached to your work station.
- Keep your fingers (and limbs) away from the propellers while they are running, and while the power supply is turned on! The DAQ card will output transient signals now and again, which cannot be controlled, e.g. when the computer is switched on or off.
- With respect to fire safety: remember to turn off the power supply when you leave the lab.
- The system is often in an unstable state and behaves badly. It is also expensive and a bit fragile, so try to avoid crash landings! Be prepared to either turn off the power supply, and/or to stop the execution of the controller application. One person in the group should always be prepared to switch off the system when it is running, while another one should

be standing prepared to catch the counterweight of the helicopter to avoid crash landing. **Do not grab the end with spinning blades! The protective grids are very thin to be light enough for flight, it might break when you grab it which can result in serious injuries!**

- It should not be necessary to alter the helicopter setup in any way. If you experience hardware problems, notify the student assistants, teaching assistant or technical staff. **Do not try to fix it yourself.**

D. User guide

D.1. Starting up

Before you get started, make sure that the power supply of the helicopter is switched off: the power switch on top of your desk should be on "Stopp". To turn on the computer, press the power button on the computer below the helicopter platform. Once Windows has started up, click on the "Other User" button. Log in with your student user name and password. Once you are logged in, double click on the Matlab icon on the desktop or click on the Matlab icon in the task bar to start up Matlab.

D.2. Matlab, Simulink, QuaRC

The controllers for the helicopter are to be made with Matlab and Simulink. When Matlab has loaded, you should see a window similar to the one shown in Fig. 1.2. From this window Simulink can be run by clicking the Simulink button in the toolbar, or entering `simulink` in the Matlab Command Window and pressing enter.

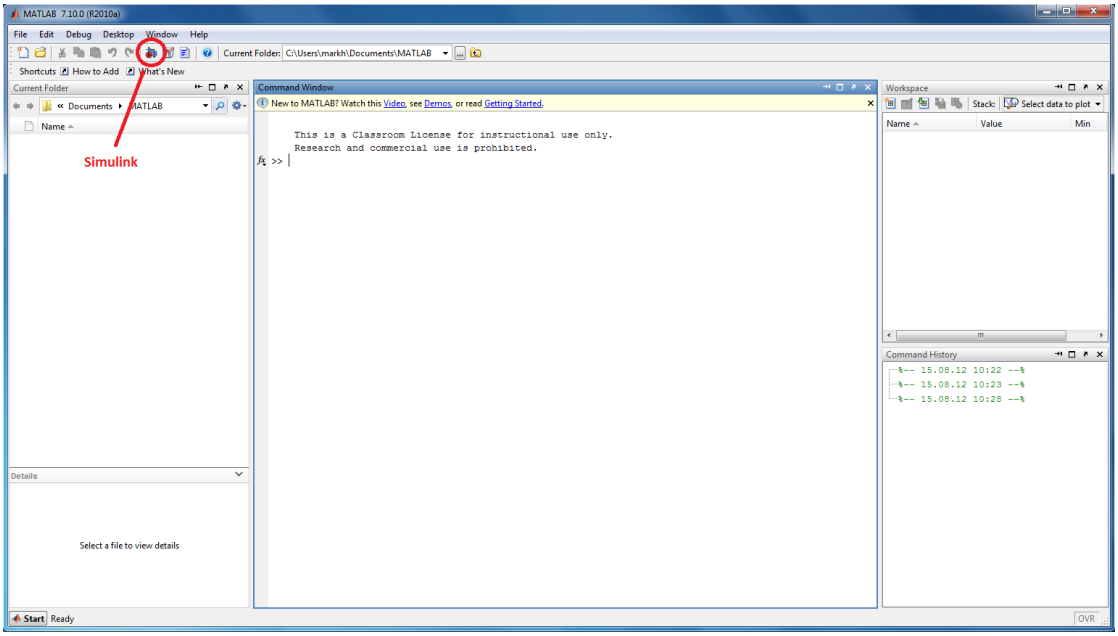

Fig. 1.2.: Matlab Desktop.

When Simulink has been loaded, a window similar to the one shown in Fig. 1.3 should appear. In this window you should be able to find all the Simulink blocks needed to create your controllers.

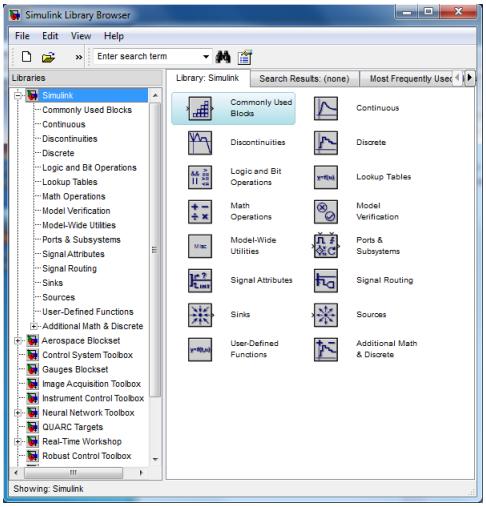

Fig. 1.3.: Simulink Library Browser.

A template Simulink model has been made, which you can use as a starting point for the assignment work. This file is described in Section D.3. A Simulink model file can be opened from Windows, Matlab, and Simulink. If you open such a file from Windows, a new instance of Matlab is run, so it is usually preferable to open such files in Matlab or Simulink directly.

When a block diagram implementing your controller has been made in Simulink, QuaRC is used to automatically generate source code for this model. QuaRC will also automatically compile the source code. When the source code has been built, Simulink is used to run the implementation in real time. With QuaRC installed, a new menu is available in Simulink. It is labelled `QuaRC`, and is shown in Fig. 1.4.

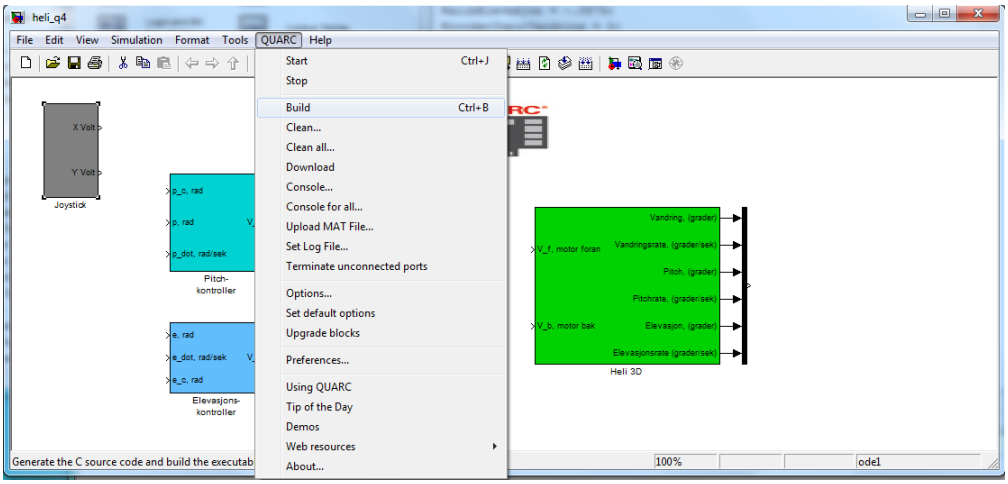

Fig. 1.4.: QuaRC menu in Simulink.

The menu item `QuaRC` → `Build` should be the only item in this menu needed throughout this assignment. The following happens when `Build` is selected:

- QuaRC generates ANSI C code for the Simulink block diagram.
- QuaRC compiles and links the source code.
- The executable code is loaded into the QuaRC Real-Time server.

After this ordeal, your Simulink block diagram implementation should be ready to run. You can now turn on the power supply. The power supply is the big black box underneath the table, labeled "Quanser VoltPAQ-X2". The power switch on your desk turns them on. Please turn them off when you are not using the helicopter or if your helicopter becomes unstable.

When the power supply is on, your controller application can be started by subsequently clicking the "Connect" and "Play" buttons in Simulink. This is shown in Fig. 1.5.

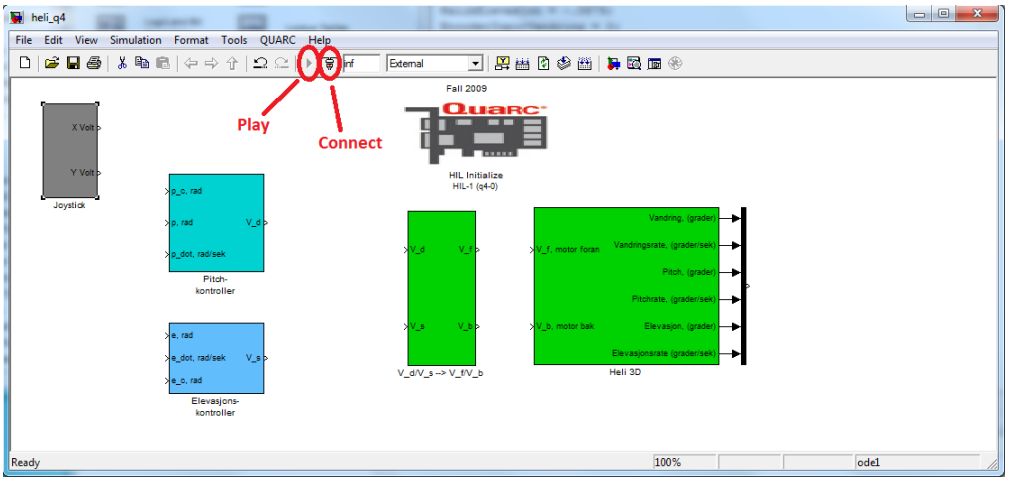

Fig. 1.5.: Start the Simulink program. Connect before playing.

When you make changes to the Simulink diagram, by e.g. adding new blocks, you will have to `Build` again before you can "Connect" and "Play". However changing the values but keeping the dimensions of constants can be done while the helicopter is running, and does not require a new `Build`. Errors can occur during building. It is usually good practice to wait until QuaRC finishes building, and for the model to be loaded into the QuaRC Server application before the controller application is started.

If you get strange error messages during building, there are a couple of tricks that usually help. Firstly, try to delete all compiled code by deleting the QuaRC files from the folder containing your model. Then, try to build everything again. Secondly, if this does not help your situation, restarting the computer might.

To save measured data from the system, you may use a "To File" or "To Workspace" block. The standard time-series format does not work. Instead use one of the others, such as "Array" to export your data; see Fig. 1.6. You can load the stored data into the Matlab Workspace by using the command `load xxxx`, where `xxxx.mat` is the name of the stored MAT file. Note that the first row of the array in which the data is stored contains the time sequence; the other rows contain the stored signals. Note that you can store multiple signals in one variable by "muxing" the signals with the "Mux" block in the Simulink Library Browser. Due to a limit

in the QuaRC driver, the following steps has to be made to store data sequences longer than 10 seconds: `Code` (in the Simulink menu) → `External mode control panel` → `Signals and Triggering`. Under `Trigger options` you should increase the duration to an appropriate value.

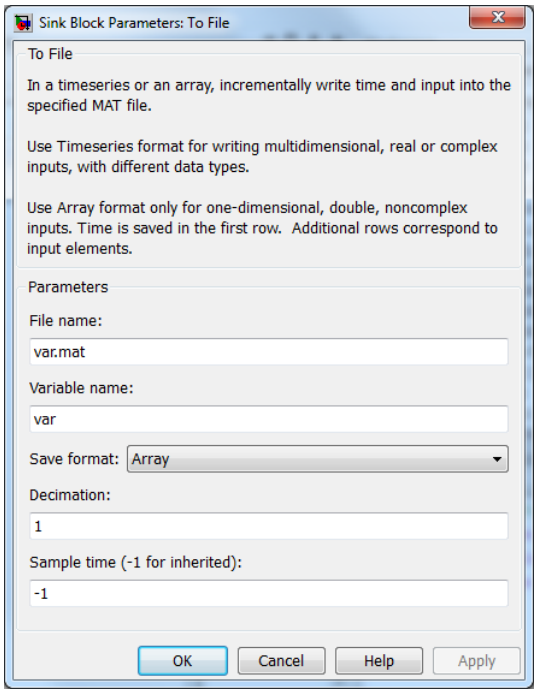

Fig. 1.6.: Sending data to a file

D.3. Starting point for the programming

A template Simulink model, `heli_q8.slx`, has been made which you can use as a starting point for the assignment. In addition to this file, initialization file containing the physical parameter values for the helicopter, `init_heli.m`, has been prepared. These files can be downloaded from "Blackboard"[3].

The initialization file must be run prior to the `QuaRC` → `Build` command to avoid any error messages during building. The reason for this is that in addition to the physical parameter values, calibration data for the encoders and other configuration specific data is also in this file.

It should **not** be necessary to rewire/reconnect any cables on the helicopter, the power supply, the joystick, the DAQ card's terminal board, etc. It is important that your end product uses the configuration given in the initialization file.

For a faster building procedure, copy the Matlab/Simulink files to your folder on the hard-drive of the computer:

`C:\Users\xxxx`

where `xxxx` is your username. Do not forget to copy the files to your memory stick after you are done! Note that no backups are made of the files on the lab's computers, and the computers are wiped without notice. Thus, it is your own responsibility to back up your files.

[3]The files have been tested on all the helicopters. However, due to some differences between the helicopters, some adjustments of the parameters might improve performance.

D.4. Data storage

Because there is **no internet in the lab**, it is not possible to store and transfer your data via e-mail or other internet-based methods. The easiest way to store your data is on a memory stick, so **do not forget to bring a memory stick to the lab**.

Alternatively, you can store your data on the "sambastud" server, which should be accessible from the lab's computers. Windows should automatically connect to this server when you log on, but in case it does not, you can try to following: right-click on "My Computer", select "Map Network Drive", enter

```
\\sambastud.stud.ntnu.no\username
```

as the path, and your username in the "Connect As" field. Be sure the "Reconnect at Logon" checkbox is unchecked. Click "OK" and fill in your password. To connect to the "sambastud" server outside the lab, you need to make sure that you are connected to the NTNU network (possibly via a VPN connection).

2. Tasks

The helicopter lab assignment is divided into six parts each corresponding to one day at the lab. The later tasks may be more difficult than the first ones - it is recommended to continue on next week's task if you finish early.

1. **Part 1 - Mathematical modeling:** A mathematical model of the system is derived to form the entry point for the rest of the assignment.
2. **Part 2 - Mono-variable control:** P and PD controllers for pitch and travel are implemented to control one of the system's outputs.
3. **Part 3 - Multi-variable control:** Optimal control is implemented and tuned.
4. **Part 4 - Inertial Measurement Unit:** An inertial measurement unit (IMU) is introduced and experiments are conducted to examine its characteristics.
5. **Part 5 - Modeling and Prediction:** A discrete-time stochastic model of the system is derived and examined.
6. **Part 6 - Filtering:** IMU measurements and predictions from the stochastic process model are combined with the Kalman filter algorithm and subsequently used for output feedback.

1. Part I – Modeling, validation, and manual control

The helicopter is modeled as three point masses: two point masses represent the two motors that are connected to the propellers, and one point mass represents the counterweight. The model of the helicopter is depicted in Fig. 2.1. The cubes in the figure represent the point masses whilst the cylinders represent the helicopter joints. The rotations of the joints are defined in the following manner: p denotes the pitch angle of the helicopter head, e denotes its elevation angle, and λ denotes the travel angle of the helicopter. In Fig. 2.1, all joint angles are zero. This means that $p = 0$ if the helicopter head is horizontal, and that $e = 0$ if the arm between the elevation axis and the helicopter head is horizontal.

The propeller forces of the front and back propeller are given by F_f and F_b, respectively. It is assumed that there is a linear relation between the voltages V_f and V_b supplied to the motors and the forces generated by the propellers:

$$F_f = K_f V_f \tag{2.1a}$$

$$F_b = K_f V_b \tag{2.1b}$$

Here, K_f is a motor force constant. The two propellers are placed symmetrically about the pitch axis. The sum of the forces $F_f + F_b$ developed by the two propellers determines the net lift of the helicopter. The difference between the forces $F_b - F_f$ is proportional to the torque about the pitch axis.

The gravitational forces for the front and the back motor are denoted by $F_{g,f}$ and $F_{g,b}$, while the gravitational force of the counterweight is denoted by $F_{g,c}$. The amplitude of the gravitational

forces can be computed as mass times the gravitational constant g. Note that the gravitational forces always point in a vertical direction, whilst the direction of the propeller forces is dependent on the joint angles.

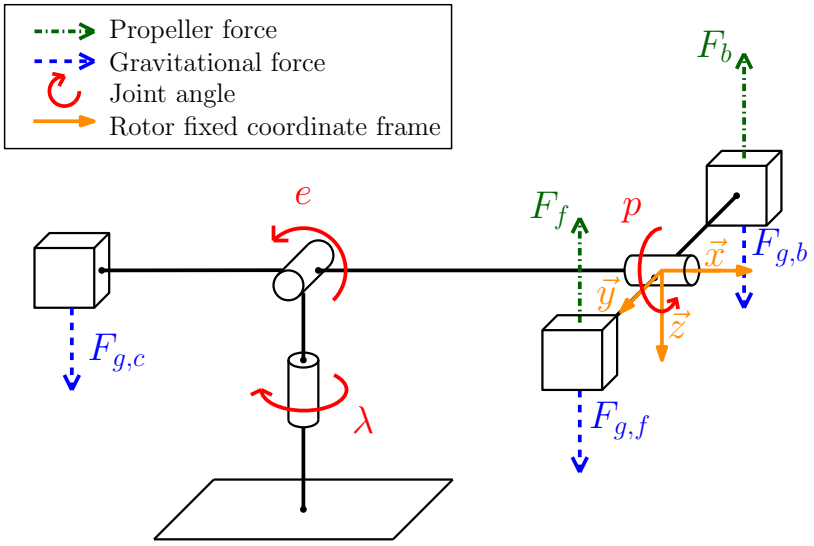

Fig. 2.1.: Helicopter model

The masses of the front and the back motors are assumed to be equal and given by m_p. The mass of the counterweight is given by m_c. The distance from the elevation axis to the head of the helicopter is given by l_h, whilst the distance from the elevation axis to the counterweight is given by l_c. Because the two propellers are placed symmetrically about the pitch axis, the distance from the pitch axis to both motors is the same and is given by l_p. The masses and distances are depicted in Fig. 2.2. Other forces, such as friction and centripetal forces, are neglected.

Note: The actual values for the masses and distances differ from helicopter to helicopter and are therefore not given here. These values can be found in an initialization file which can be downloaded from "Blackboard". Due to the mass disparities, some helicopters may be more docile than others. The motor force constant K_f also differs quite a bit from helicopter to helicopter. This constant is to be determined below.

1.1. Problem 1 - Equations of motion

Compute the linearized equations of motion (a set of coupled ODEs) for the pitch angle p, the elevation angle e, and the travel angle λ. The linearization point is the horizontal configuration shown in Fig. 2.1. Assuming small departures from the linearization point, one may used the small angle approximations:

$$\cos(x) \approx 1, \quad \sin(x) \approx x \tag{2.2}$$

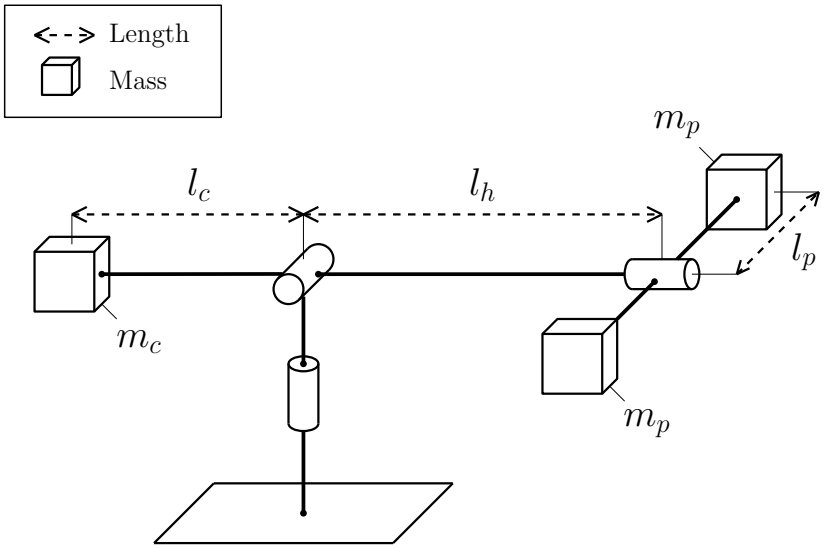

Fig. 2.2.: Masses and distances

Let the moments of inertia about the pitch, elevation and the travel axes be denoted by J_p, J_e, and J_λ, respectively. Show that the equations of motion can be stated in the form

$$J_p\ddot{p} = L_1 V_d \tag{2.3a}$$
$$J_e\ddot{e} = L_2(V_s - V_{s,0}) \tag{2.3b}$$
$$J_\lambda\ddot{\lambda} = L_3 V_s p \tag{2.3c}$$

Here, L_i for $i = 1, 2, 3$ are constants whilst $V_{s,0}$ is the motor-force needed to keep the helicopter horizontal. The sum of the motor voltages V_s and the difference of the motor voltages V_d are defined by:

$$V_s = V_f + V_b \tag{2.4a}$$
$$V_d = V_b - V_f \tag{2.4b}$$

The constant $V_{s,0}$ is the voltage needed to establish an equilibrium at the linearization point. Define $\tilde{V}_s = V_s - V_{s,0}$ and let $\tilde{V}_s$ serve as an input to be determined by the control system. Modify equations (2.3a) - (2.3c) to use $\tilde{V}_s$ and $V_{s,0}$ instead of V_s. Since a linearized system is desired, remove all higher order terms in the equations of motion. You are to demonstrate that the following equations result from the linearization procedure.

$$\ddot{p} = K_1 V_d \tag{2.5a}$$
$$\ddot{e} = K_2 \tilde{V}_s \tag{2.5b}$$
$$\ddot{\lambda} = K_3 p \tag{2.5c}$$

Here, the moments of inertia entering in the preceding equation are given by:

$$J_p = 2m_p l_p^2 \tag{2.6a}$$
$$J_e = m_c l_c^2 + 2m_p l_h^2 \tag{2.6b}$$
$$J_\lambda = m_c l_c^2 + 2m_p(l_h^2 + l_p^2) \tag{2.6c}$$

1.2. Problem 2 - Parameter identification

The encoder values are set to zero every time Simulink is connected to the helicopter. Assuming that the helicopter head rests on the table when Simulink is connected, the encoder outputs are not the same as the joint angles p, e and λ. Add constants to the encoder outputs such that the output of the encoder for the pitch is zero when the helicopter head is horizontal, and such that output for the elevation is zero when the arm between the elevation axis and the helicopter head is horizontal. Check that the encoders have the same positive direction as the model. If not, implement gains of (-1) to fix discrepancies.

Before one can implement a controller that is based on the equations of motion in (2.3a)-(2.3c), one needs to determine the motor force constant K_f. By measurement, obtain the value for the voltage $V_{s,0}$. When $V_s = V_{s,0}$ the helicopter will maintain the equilibrium value $e = 0$ (the arm between the elevation axis and the helicopter head is horizontal). Use this value to calculate the motor force constant K_f. Use the calculated value of K_f in the remaining problems.

In this task we need to fly the helicopter to get the correct voltage. Let the signal from the x-axis of the joystick connect directly to the voltage difference V_d and the signal from the y-axis of the joystick connect directly to the voltage sum V_s. You might need to scale the joystick signals.

1.3. Problem 3 - Verification and manual control

The first attempt to control the helicopter will be made using manual control. It is rather difficult to control the helicopter this way, so be prepared to turn off the helicopter and catch the counterweight. Be careful not to crash the helicopter!

We want to compare the physical behavior of the helicopter with the linearized model. Implement the linearized model in Simulink and compare the estimated outputs with the encoder outputs. The linearized output will only be valid around the linearization point, one person in the group should therefor lift the helicopter by holding under the bar between the rotors to the linearization point before the program is started. Make sure someone is ready to turn of the helicopter and take care that the fingers of the person holding the helicopter do not touch the grid around the rotors. Test out different scenarios where you would expect that the models are good, and others where you would expect that they are worse. Discuss your findings. Would it be vise to regulate the system based on an open loop estimate, such as the one you implemented here?

2. Part II – Mono-variable control

In this part of the assignment you will use the elevation controller already given in the Simulink diagram available from "Blackboard". Note that e_c is the reference value for e and should be set to zero. The output of the elevation controller is $\tilde{V}_s$. Add $V_{s,0}$ to the output of the controller to get V_s.

In this part of the assignment, we will design a controller for the pitch angle p that uses the x-axis of the joystick as a reference denoted as p_c.

2.1. Problem 1 - PD controller

A PD controller is implemented to control the pitch angle p. This controller is given as

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p} \tag{2.7}$$

with constants $K_{pp}, K_{pd} > 0$, where p_c is the desired reference for the pitch angle p. Substitute (2.7) in the equation of motion for the pitch angle in (2.5a). Apply the Laplace transform to the resulting differential equation to find the transfer function $G(s) = (p/p_c)(s)$.

2.2. Problem 2 - Pole placement

Use K_{pp} and K_{pd} to place the poles of the transfer function at desired locations. Test out experimentally how different locations of the poles of the system affect the closed loop behavior. Discuss the results, especially how this fits with the theory.

2.3. Problem 3 - Harmonic oscillator

An alternative method for choosing K_{pp} and K_{pd} is to realize that the closed-loop system can be regarded as an harmonic oscillator aka. a "'spring-mass-damper"' system. An harmonic oscillator can be written on the form:

$$G(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{2.8}$$

Examine how changes in the undamped angular frequency ω_0 and the damping ratio, ζ affect the poles and experimental response of the system. What is a good theoretical choice for ζ, and what does practice show?

3. Part III – Multi-variable control

In this part of the assignment, you will control the pitch angle p and the elevation rate $\dot{e}$ with an optimal multi-variable controller where the reference for the pitch angle and the elevation rate are provided by the output of the joystick.

3.1. Problem 1 - State-space formulation

Put the system of equations given by the relations for pitch and elevation in (2.5a)-(2.5b) in a state-space formulation of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.9}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices of suitable dimension. The state vector and the input vector are defined by

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{u} = \begin{bmatrix} \tilde{V}_s \\ V_\mathrm{d} \end{bmatrix} \tag{2.10}$$

Examine the controllability of the system and discuss the result of your analysis.

3.2. Problem 2 - Linear Quadratic Regulators

We aim to track the joystick reference $\boldsymbol{r} = [p_c,\ \dot{e}_c]^\intercal$ for the pitch angle p and elevation rate $\dot{e}$. The x-axis of the joystick is used to provide the reference p_c for the pitch angle whilst the y-axis is used for the reference $\dot{e}_c$ for the elevation rate.

A linear state-feedback controller with reference-feed-forward is desired.

$$\boldsymbol{u} = \boldsymbol{F}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x} \tag{2.11}$$

The matrix $\boldsymbol{K}$ is determined by the linear quadratic regulator (LQR) algorithm, which means that the control input $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$ optimizes the following cost function.

$$J = \int_0^\infty \left(\boldsymbol{x}^\intercal(t)\boldsymbol{Q}_\mathrm{LQR}\boldsymbol{x}(t) + \boldsymbol{u}^\intercal(t)\boldsymbol{R}_\mathrm{LQR}\boldsymbol{u}(t)\right) dt \tag{2.12}$$

For simplicity, let the weighting matrices $\boldsymbol{Q}_\mathrm{LQR}$ and $\boldsymbol{R}_\mathrm{LQR}$ be diagonal. The matrices $\boldsymbol{Q}_\mathrm{LQR}$ and $\boldsymbol{R}_\mathrm{LQR}$ should be chosen such that the response of the helicopter is fast and accurate. The gain matrix $\boldsymbol{K}$ can be computed by using the Matlab command `K=lqr(A,B,Q,R)`.

Choose the matrix $\boldsymbol{F}$ such that (in theory) $\lim_{t\to\infty} p(t) = p_c$ and $\lim_{t\to\infty} \dot{e}(t) = \dot{e}_c$ for fixed values of p_c and $\dot{e}_c$.

Test out the response for different values in $\boldsymbol{Q}_\mathrm{LQR}$ and $\boldsymbol{R}_\mathrm{LQR}$. Test what happens if you use a different $\boldsymbol{F}$. Discuss the results. Explain how the values in $\boldsymbol{Q}_\mathrm{LQR}$ and $\boldsymbol{R}_\mathrm{LQR}$ affect the experimental behavior of the helicopter.

3.3. Problem 3 - Integral action

Modify the controller from the previous problem to include an integral effect for the elevation rate and the pitch angle. Note that this results in two additional states γ and ζ, for which the differential equations are given by

$$\begin{aligned} \dot{\gamma} &= p - p_c \\ \dot{\zeta} &= \dot{e} - \dot{e}_c \end{aligned} \tag{2.13}$$

An augmented state vector shown below must now be used.

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \\ \gamma \\ \zeta \end{bmatrix} \tag{2.14}$$

Find the augmented $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{F}$ matrices. Implement the controller with integral action. Compare the behavior of the helicopter with and without integral effect. How does the introduction of integral effect change how we can choose $\boldsymbol{F}$? Test what happens if one sets $\boldsymbol{F}$ at a different value.

4. Part IV – Inertial Measurement Unit (IMU)

So far, encoders have been used to measure the states used for feedback. Encoders are, however, not always available in practical applications. An inertial measurement unit (IMU) called MPU-9250 produced by InvenSense will now be used to provide information about the system's states. This IMU is equipped with a gyroscope, an accelerometer, and a compass. We will use the gyroscope and the accelerometer for the subsequent tasks.

The IMU is mounted in the middle of the bar connecting the rotors. The IMU communicates with an Arduino Micro that sends the resulting data to the computer over a serial communication port. To communicate with the Arduino you need to download the "IMU.slx" file from Blackboard and copy the content into your project. The serial communication sets a random port number for the Arduino. This information must be passed on to Simulink. Follow the setup guide written in the Simulink diagram to set the correct port.

Measurements will not arrive in every Simulink time-step. The new-signal output will be equal to 1 when there are new measurements, and equal to 0 when there are no new measurements. The gyroscope and accelerometer outputs are vectors with 3 elements given in the order x, y, z as defined by Fig. 2.1. The IMU will measure acceleration and rotations in its local frame that rotates when the helicopter rotates, this frame is shown in orange in figure Fig. 2.1. This frame is defined such that the $\vec{y}$ axis always aligns with the bar between the rotors, and the $\vec{x}$ axis always aligns with the arm out to the rotors, even if the helicopter is rotated.

The gyroscope measures the rotational velocity in units rad/s, whilst the accelerometer measures the proper acceleration in units $\mathrm{m/s^2}$. In contrast to regular acceleration which measures the second derivative of linear position, proper acceleration evaluates to $0\mathrm{m/s^2}$ when the accelerometer is in free-fall. It therefore measures $9.81\mathrm{m/s^2}$ in the upwards direction when the accelerometer is standing still on earth's surface.

4.1. Problem 1 - IMU characteristics

Disconnect the voltage-signals into the helicopter block. Plot the gyroscope values against the encoder-rates and plot the accelerometer values by themselves. In this task we will look at the data from the IMU when we manually move the helicopter (and the rotors are not running!). Write down what you observe, with special emphasis on the following points:

- Hold the helicopter at the linearization point and rotate the helicopter about one axis at the time. How does the gyroscope output compare to the encoder-rates? You might need be rearrange the gyro-outputs so that the first element corresponds to pitch, the second to elevation, and the third to travel. This error is caused by the IMU not having the same coordinate system as our lab-setup. If alterations are needed, apply them to the accelerometer as well.
- Try to rotate the helicopter around one axis, and then rotate it around a second axis while maintaining the rotation about the first axis. Explain what you see.
- Look at the accelerometer output. What do you see?

4.2. Problem 2 - Gyroscope transform

You probably noticed that we could not use the gyro-measurements directly as they did not give correct results when the helicopter was rotated. To counteract this we will need to apply a rotation that uses the pitch and elevation angles to calculate the correct pitch-, elevation-, and

travel-rate. To save you some cumbersome trigonometry, a Simulink block that performs this rotation is supplied in the "IMU.slx" file. Use the encoder angles as input to this block.

4.3. Problem 3 - Observability

The transformed gyro-measurements are now used as pitch-, elevation-, and travel-rate measurements.

$$\boldsymbol{y}_{\text{gyro}} = \begin{bmatrix} \dot{p} \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} \tag{2.15}$$

Let a state and input vector be defined by:

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix}, \qquad \boldsymbol{u} = \begin{bmatrix} V_s \\ V_{\text{d}} \end{bmatrix} \tag{2.16}$$

Put equations (2.3a)-(2.3b) on state-space form with the preceding state, input and gyro output, viz.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.17a}$$

$$\boldsymbol{y}_{\text{gyro}} = \boldsymbol{C}_{\text{gyro}}\boldsymbol{x} \tag{2.17b}$$

Examine the observability of the system using the function `obsv(A,C)` in Matlab. If the full state is not observable, identify which subset of $\boldsymbol{x}$ that is. Discuss your findings.

4.4. Problem 4 - Accelerometer

In this task we will use the accelerometer output to measure the pitch and elevation. The IMU does not have a sensor that lets us measure these values directly. Instead we will utilize the fact that the accelerometer measures the force counteracting gravity when the helicopter is stationary. This force will always point straight upwards with a magnitude equal to the gravitational constant g.

1) Derive equations for the measured acceleration $[a_x, a_y, a_z]$ by the IMU given the elevation and pitch angle assuming that the helicopter is standing still. It is easier to first consider elevation alone assuming pitch equal to zero, and then consider the pitch afterwards.

2) Use the equations for the measured acceleration to derive the following equations for pitch and elevation.

$$p = \arctan\left(\frac{a_y}{a_z}\right) \tag{2.18a}$$

$$e = \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \tag{2.18b}$$

Discuss the validity of the assumption made to arrive at these equations.

4.5. Problem 5 - Observability part II

There are now two additional measurements. Repeat problem 3 and identify which states are observable with 5 measurements. Use the following output vector.

$$\boldsymbol{y}_{\text{IMU}} = \begin{bmatrix} \dot{p} \\ \dot{e} \\ \dot{\lambda} \\ p \\ e \end{bmatrix} \tag{2.19}$$

Discuss your findings.

4.6. Problem 6 - Noise

In this task, the noise affecting the measurement vector $\boldsymbol{y}_{\text{IMU}}$ is examined.

1) Produce an experimental time-series when the helicopter is laying still on the ground as well as a time-series when the helicopter is kept stationary at the linearization point while flying. You can use one of the controllers from the subsequent tasks to stabilize the system at the linearization point. Discuss the appearance of the noise in both cases. Would you classify it as white noise? Compare the signal covariances for the two cases. Explain the difference. Note down the experimental covariances in both cases.

NB! Remember that we do not get data each time-step! All NaN values should be removed before the variances are calculated.

2) As we measure all relevant states for our controllers, attempt to fly the helicopter using only the measurements as feedback. Be prepared to catch the counterweight and stop the helicopter.

Part V & VI: The Kalman Filter

We wish to do output feedback from our IMU instead of using the encoders for feedback. As you may have experienced in the previous task, it is not sufficient to simply feed the measurements back to the controller, as in the case of the encoders. Instead we will use an estimator to combine our measurements with our model of the system to get a better estimate of the state. In the following task we will develop the Kalman Filter which is described by the following equations.

Correction with new data:

$$\boldsymbol{K}[k] = \bar{\boldsymbol{P}}[k]\boldsymbol{C}_\mathrm{d}^\top(\boldsymbol{C}_\mathrm{d}\bar{\boldsymbol{P}}[k]\boldsymbol{C}_\mathrm{d}^\top + \boldsymbol{R}_\mathrm{d})^{-1} \tag{2.20a}$$

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}}[k] + \boldsymbol{K}[k](\boldsymbol{y}[k] - \boldsymbol{C}_\mathrm{d}\bar{\boldsymbol{x}}[k]) \tag{2.20b}$$

$$\hat{\boldsymbol{P}}[k] = (\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{C}_\mathrm{d})\bar{\boldsymbol{P}}[k](\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{C}_\mathrm{d})^\top + \boldsymbol{K}\boldsymbol{R}_\mathrm{d}\boldsymbol{K}^\intercal \tag{2.20c}$$

Predicting ahead:

$$\bar{\boldsymbol{x}}[k+1] = \boldsymbol{A}_\mathrm{d}\hat{\boldsymbol{x}}[k] + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}[k] \tag{2.20d}$$

$$\bar{\boldsymbol{P}}[k+1] = \boldsymbol{A}_\mathrm{d}\hat{\boldsymbol{P}}[k]\boldsymbol{A}_\mathrm{d}^\intercal + \boldsymbol{Q}_\mathrm{d} \tag{2.20e}$$

These equations might at first glance seem intimidating, but worry not. In the following, we will gradually develop one equation at the time, which will hopefully make the Kalman filter seem quite natural.

5. Part V - Prediction step

In this part of the assignment the prediction step of the Kalman filter will be derived. Due to random noise and disturbances, as observed in the previous task, a stochastic model is warranted. Since the IMU gives data in discrete time, a discrete-time stochastic process model is necessary.

$$\boldsymbol{x}[k+1] = \boldsymbol{A}_\mathrm{d}\boldsymbol{x}[k] + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}[k] + \boldsymbol{w}_\mathrm{d}[k] \tag{2.21a}$$

$$\boldsymbol{y}[k] = \boldsymbol{C}_\mathrm{d}\boldsymbol{x}[k] + \boldsymbol{v}_\mathrm{d}[k] \tag{2.21b}$$

$$\boldsymbol{w}_\mathrm{d} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_\mathrm{d}), \qquad \boldsymbol{v}_\mathrm{d} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_\mathrm{d}) \tag{2.21c}$$

5.1. Task 1 - Discretization

Discretize the continuous-time deterministic system with the same time-step as Simulink to obtain $\boldsymbol{A}_\mathrm{d}$, $\boldsymbol{B}_\mathrm{d}$ and $\boldsymbol{C}_\mathrm{d}$. The provided Simulink diagram operates in fixed-step mode with a time-step of 0.002s. You may employ the `c2d` function in Matlab.

The covariance of the stochastic disturbance $\boldsymbol{Q}_\mathrm{d}$ serves as a tuning variable and can be left undetermined. The measurement covariance obtained in the previous task corresponds to the discrete-time covariance $\boldsymbol{R}_\mathrm{d}$. Use the values obtained with a flying helicopter.

5.2. Task 2 - State prediction

In this task we will implement the state prediction step. If we use the previous predicted state as the current state estimate we will get an open-loop estimation on the following form

$$\bar{\boldsymbol{x}}[k+1] = \boldsymbol{A}_{\mathrm{d}}\bar{\boldsymbol{x}}[k] + \boldsymbol{B}_{\mathrm{d}}\boldsymbol{u}[k] \tag{2.22}$$

Note that this estimator does not include the noise term, since this is unknown. Implement the open-loop estimator as a Matlab-function Simulink block. This block shall output the predicted next state ($\bar{\boldsymbol{x}}[k+1]$), taking the control-input ($\boldsymbol{u}[k]$), the predicted value of the current state ($\bar{\boldsymbol{x}}[k]$), and all relevant matrices as inputs. Use the predicted next state of the previous time-step as predicted value for the current time-step. This can be done in Simulink by connecting the predicted next state output through a unit-delay block into the predicted current state input. You can supply the initial condition for the state in the unit-delay block.

Compare the experimental values for the states (as obtained from encoders) with the predicted output. Discuss your observations.

Attempt to fly using only the open-loop estimator as feedback to your controller. As the model is only valid around the linearization point, gently lift the helicopter up to the linearization point before starting. As before be prepared to stop the helicopter and catch the counterweight, take care to avoid touching the grid around the rotors. Ensure that initial conditions are correct. Test out different behavior where you would expect a open-loop estimator to work, and cases where it would not work. Discuss your findings.

5.3. Task 3 - Prediction error covariance

The error in our open-loop prediction is defined by

$$\bar{\boldsymbol{\varepsilon}}[k] = \boldsymbol{x}[k] - \bar{\boldsymbol{x}}[k] \tag{2.23}$$

We now wish to examine how the covariance of this error develops over time. Derive an expression for the covariance of $\bar{\boldsymbol{\varepsilon}}[k+1]$, denoted as $\bar{\boldsymbol{P}}[k+1]$, given the error covariance of the previous time-step, $\bar{\boldsymbol{P}}[k]$. The definition of covariance and the equations 2.21a and 2.22 can be useful in finding this expression. The definition is as follows.

$$\bar{\boldsymbol{P}}[k+1] = \mathsf{E}[\hat{\boldsymbol{\varepsilon}}[k+1]\hat{\boldsymbol{\varepsilon}}^{\mathsf{T}}[k+1]] \tag{2.24}$$

Keep in mind that the covariance of uncorrelated terms is 0. You should end up with equation 2.20e with the difference that you are using the predicted covariance at the current-time step $\bar{\boldsymbol{P}}[k]$ instead of the estimated covariance $\hat{\boldsymbol{P}}[k]$.

Update your Matlab-function block to also accept the current error covariance matrix, $\bar{\boldsymbol{P}}[k]$, as input, and give the uncertainty in the predicted state, $\bar{\boldsymbol{P}}[k+1]$, as output. As in the previous task, feed the predicted uncertainty output to the current uncertainty input through a unit delay block. Choose a reasonable value for the uncertainty of the initial state, $\bar{\boldsymbol{P}}[0]$.

The matrix $\boldsymbol{Q}_{\mathrm{d}}$ describes the uncertainty in our model. Test out different diagonal $\boldsymbol{Q}_{\mathrm{d}}$ matrices. Discuss the results.

6. Part VI - Correction step

In this part of the assignment the correction step of the Kalman Filter will be implemented. This part will combine measurements and predictions to produce a better estimate of the current state.

6.1. Task 1 - Correction

Denote a measurement-corrected estimate at time-step k by $\hat{\boldsymbol{x}}[k]$. Linear measurement correction is achieved by computing a weighted average of a measurement from the IMU and a predicted output signal derived from the predicted state. Let $\bar{\boldsymbol{y}}[k]$ denote the predicted measurement so that $\boldsymbol{y}[k] - \bar{\boldsymbol{y}}[k]$ represents the disagreement between prediction and measurement.

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}}[k] + \boldsymbol{K}[k](\boldsymbol{y}[k] - \bar{\boldsymbol{y}}[k]) \tag{2.25a}$$
$$\bar{\boldsymbol{y}}[k] = \boldsymbol{C}_{\mathrm{d}}\bar{\boldsymbol{x}}[k] \tag{2.25b}$$

Here, $\boldsymbol{K}[k]$ serves as a weighting matrix. Moving around the terms of equations 2.25a - 2.25b gives us the following equation where its easier to see that $\boldsymbol{K}[k]$ decides how much to weight the measurement in relation to the predicted state.

$$\hat{\boldsymbol{x}}[k] = (\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{C}_{\mathrm{d}})\bar{\boldsymbol{x}}[k] + \boldsymbol{K}[k]\boldsymbol{y}[k] \tag{2.26}$$

As we do not receive measurements at each time-step we can only do the weighted average when there is new measurements available. When this is not the case, let the estimate be the prediction. This way the estimate is always our best estimate of the true state.

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}}[k] \tag{2.27}$$

Update your Matlab-function block to also output the estimate as this will be used in feedback. When there is a new measurement, use the weighted average given in 2.25a. When there is no new measurement, let the estimate be the same as the prediction which is given in 2.27. Also change the prediction step to use the estimate instead of the last predicted output.

$$\bar{\boldsymbol{x}}[k+1] = \boldsymbol{A}_{\mathrm{d}}\hat{\boldsymbol{x}}[k] + \boldsymbol{B}_{\mathrm{d}}\boldsymbol{u}[k] \tag{2.28}$$

Nb: Take extra care to ensure that you are using estimates and predictions from the correct time-steps.

6.2. Task 2 - Correction covariance

As before we want to know the covariance of the error in our estimate. The error of the estimate is denoted as

$$\hat{\boldsymbol{\varepsilon}}[k] = \boldsymbol{x}[k] - \hat{\boldsymbol{x}}[k] \tag{2.29}$$

The covariance of the error can be written as

$$\hat{\boldsymbol{P}}[k] = \mathsf{E}[\hat{\boldsymbol{\varepsilon}}[k]\hat{\boldsymbol{\varepsilon}}^{\mathsf{T}}[k]] \tag{2.30}$$

Use equation 2.26 together with equations 2.21b to find an expression of $\hat{\boldsymbol{P}}[k]$ given $\bar{\boldsymbol{P}}[k]$. You should end up with equation 2.20c.

When there is no new measurement available let the estimated error covariance be equal the predicted error covariance, as the estimated state in this case will be equal the predicted state.

$$\hat{\boldsymbol{P}}[k] = \bar{\boldsymbol{P}}[k] \tag{2.31}$$

Update your Matlab-function block to also output the estimated error covariance, $\hat{\boldsymbol{P}}[k]$. Similarly to how we altered the state prediction step in 2.28, change the equation for the error covariance prediction step, $\bar{\boldsymbol{P}}[k+1]$, to use the estimated error covariance, $\hat{\boldsymbol{P}}[k]$, instead of the covariance of the previous prediction error, $\bar{\boldsymbol{P}}[k]$.

6.3. Task 2 - Weighting matrix K

There are multiple ways of choosing $\boldsymbol{K}[k]$, for example through pole placement. What separates the Kalman filter from other linear observers is that it minimizes the covariance estimate to find a time-dependent $\boldsymbol{K}[k]$. The diagonal of the estimate error covariance $\hat{\boldsymbol{P}}[k]$ contains the error variance of each state, and the off-diagonal elements contain the error covariance between states. We need to decide which elements of $\hat{\boldsymbol{P}}[k]$ we want to minimize and how to weight them in relation to each other. The Kalman filter only considers the variances (the diagonal elements) and weights them equally. We therefore want to minimize:

$$\mathrm{var}(\hat{\varepsilon}_p[k]) + \mathrm{var}(\hat{\varepsilon}_{\dot{p}}[k]) + \mathrm{var}(\hat{\varepsilon}_e[k]) + \mathrm{var}(\hat{\varepsilon}_{\dot{e}}[k]) + \mathrm{var}(\hat{\varepsilon}_\lambda[k]) + \mathrm{var}(\hat{\varepsilon}_{\dot{\lambda}}[k]) = \mathrm{tr}(\hat{\boldsymbol{P}}[k]) \tag{2.32}$$

The trace function returns the sum of the diagonal elements of a matrix. Find the time-varying $\boldsymbol{K}[k]$ that minimizes $\mathrm{tr}(\hat{\boldsymbol{P}}[k])$. The following properties of trace will be useful in the derivations.

$$\frac{\partial \mathrm{tr}(\boldsymbol{AB})}{\partial \boldsymbol{A}} = \boldsymbol{B}^{\mathsf{T}} \quad , \quad \frac{\partial \mathrm{tr}(\boldsymbol{ACA}^{\mathsf{T}})}{\partial \boldsymbol{A}} = 2\boldsymbol{AC} \tag{2.33a}$$

$$\mathrm{tr}(\boldsymbol{D}) = \mathrm{tr}(\boldsymbol{D}^{\mathsf{T}}) \quad , \quad \mathrm{tr}(\boldsymbol{A}+\boldsymbol{B}) = \mathrm{tr}(\boldsymbol{A}) + \mathrm{tr}(\boldsymbol{B}) \tag{2.33b}$$

You should end up with equation 2.20a.

Implement the equation for $\boldsymbol{K}[k]$ in your Matlab-function block.

6.4. Task 3 - Tuning

Use the encoders to provide feedback. How does altering $\boldsymbol{Q}_{\mathrm{d}}$ affect the estimated states? Tune the $\boldsymbol{Q}_{\mathrm{d}}$ matrix until you are satisfied with the estimated states. Discuss the results.

6.5. Task 4 - The Kalman Filter

You have now implemented the Kalman Filter.

Before we can fly using the Kalman Filter estimate, we need to take into account that the Arduino sometimes uses 6 seconds to start-up. Prevent the controller from flying the helicopter

and the estimator from predicting ahead, before the IMU has initialized. Place your controller in a subsystem, and your Kalman Filter in another, if this is not already the case. In both subsystems insert a "Trigger" block. The subsystem should now have a trigger input. Connect the New data line to the trigger input.

Replace the encoder measurements with your state estimates and fly the helicopter with state feedback provided by the Kalman Filter. Do some additional tuning if need be to get a good response. If you notice any interesting behavior make sure to discuss it in the report. Compare the behavior when flying with the measurements directly, when flying with an open-loop observer, when flying with the Kalman Filter, and when flying using the encoder.