TTK4115 – Linear System Theory

Helicopter lab assignment

Department of Engineering Cybernetics
NTNU

Version 5.5 August 2024, AEO

Previous versions:
5.4 August 2023, DØH
5.3 August 2022, DØH
5.2 August 2021, DØH
5.1 August 2020, SVR
5.0, August 2019, SVR
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

The helicopter lab is meant to be an arena for you to implement and test the theoretical concepts covered in the lectures on a real physical system. You will be asked to complete a set of tasks, each focusing on separate topics in the course. As you complete these tasks, the helicopter will be gradually more advanced, with the implementation of controllers and estimators, and measurements from an inertial measurement unit. To fully grasp the content of the course we highly encourage you to experiment and be creative in your assigned lab time-slot. If you have an idea of how to improve your system, or if you see something interesting to investigate further, we encourage you to test it out! Remember to write about it in the report, and you can bring it up during the evaluation when we are discussing the relevant task.

Throughout the lab you will be asked to experiment according to your hypotheses and test-plans developed in the lab preparation assignment. The experiments should be done systematically and thorough, but not unnecessarily long. Remember that it is allowed to revise your test-plan when you gain experience from testing the system.

At the end of the lab we want you to understand the concepts of **controllers** and **estimators**, both in terms of them being non-optimal or optimal. Additionally be aware of limitations or possibilities when including **observations/measurements** from instruments like an inertial measurement unit.

1.1 Evaluation and grading

1.1.1 Oral evaluation

An oral evaluation will take place in November after the lab is completed. The evaluation will be an informal conversation where you can show us what you have done. We will also ask questions regarding your implementation, our observations, and theory relevant to the tasks. The oral evauliation is graded **PASS/FAIL**. The grade will be based on your understanding of theory, implementation and results of all the tasks in the lab.

At the oral evaluation we will ask you to show some of the results from each lab day. These results can be various interesting experiments you have made in each of the different lab days. Make sure you prepare separate files for the different experiments/configurations you want to show and discuss, such that we spend as little time as possible on technical troubles during the evaluation. Include a copy of all necessary Matlab files. speaking from experience, you will probably change your main Matlab script later which will corrupt all previous lab-days. At the presentation we will want to look at some scopes, so make sure you have scopes connected to all interesting signals (encoder measurements, IMU measurements, estimates). Familiarize yourself with the scope tool such that you can easily set up a new scope with other signals on our request during the evaluation. Also, make sure the helicopter can be flown using the joystick in the lab during the evaluation.

1.1.2 Report

A lab-report should be handed in at the oral evaluation. It should be printed in color on A4 paper. The report should be formatted in the same way as the lab assignment. Everything related to part 1 should be written together so that the examinator does not have to jump back and forth. Keep the report brief and to the point. You can assume that the lab setup and relevant theory is known. Include the names together with the **student number** of group participants (not candidate number). The report should be between 15 and 25 pages.

The report should be written in LaTeX. See the ITK Labreport guide/template on Github[1] for some useful hints.

The report should include:

- Your hypotheses and test-plans for the different tasks.
- The results and a discussion of all the experimentation that have been done, a discussion on any discrepancies between your hypothesis and observations.
- Answers to any questions that arise throughout the lab assignment or report specific questions from the lab preparation assignment.
- Relevant plots.
- **Consider including:**
  - Any derivations or equations from the preparations if they are highly relevant to your discussion of the experiments. If you do not refer to the theory/equations you don't have to include it in your report.
  - Any Matlab code or Simulink diagram sections if they are of particular interest, or is needed for reference in your discussion in the report.

The report is graded **PASS/FAIL** and is based on the quality of the experimentation, discussion, and theoretical understanding.

Ensure that all plots in the presentation are readable and understandable. The plots need units, axis, and legends. If you want to compare two graphs, make sure they are plotted on top of each other. Avoid having black backgrounds on the plots. You won't get any points for a good conclusion if we cannot understand the discussion or the plots the discussion is based on. The best way to make good plots is to save the data obtained in the lab on a portable hard drive, and post-process the data in Matlab, Python or similar.

It is highly recommended to complete the relevant parts of the report before starting your next lab assignment. When writing the report you will realize that you should have checked something else, this can then easily be done at the start of the next lab-day.

1.2 Practical information

- There are four lab assignments.
  1. Monovariable control, which concerns (non-optimal) tuning of a PD controller.
  2. Multivariable control, which concerns (optimal) tuning of a multivariable controller, LQR.

[1] https://github.com/ntnu-itk/labreport

3. Luenberger observer, which concerns the implementation of an (non-optimal) estimator using measurements from an interial measurement unit (IMU).
4. Kalman filter, which concerns the implementation of an (optimal) estimator using measurements from the IMU.

- To complete all four lab assignments you will work in groups of 3 students. The groups will have a total of 48 hours spread over 8 time slots of 6 hours each. This means you will have 2 time slots of 6 hours each for every lab assignment. The morning time slot is between 08:00 and 14:00. The evening time slot is between 14:00 and 20:00. You can reserve two morning slots, two evening slots, or a combination[2].
- Ten helicopters are found in "Elektrobygget", rooms B117 (Helicopter 1, 2, 3 and 4), B121 (Helicopters 5, 6 and 7) and B125 (Helicopters 8, 9 and 10). You need your key card to access the room[3]. You will choose 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. Note that helicopters 1 through 4 is in the same room and it can become crowded when the groups consists of three students each. Consider this when you choose your time slots.
- A student assistant will be present for 4 hours each day, between 10:00 and 12:00 and between 15:00 and 17:00.
- The lab computers are not connected to the internet, which means that 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 portable hard drive 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 1.5.4 for more details. Please remove your external harddrive from the lab-computers after file-transfer as the building of the Simulink code may cause buffer overflow if the code is run externally.

1.3 The helicopter

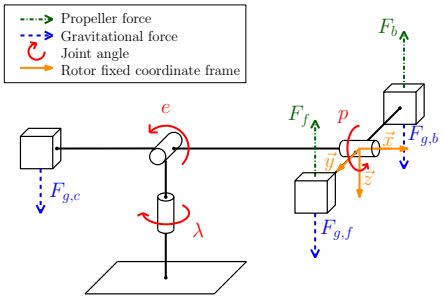
The helicopter consists of a base on which a long arm is attached; see Fig. 1.1a. The helicopter head, with two motors and propellers, is attached to one side of the arm. On the other side, a counterweight is placed. The helicopter has three rotational joints. The helicopter arm can rotate about the 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 be 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.

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.

[2] See `https://www.itk.ntnu.no/fag/TTK4115/lab/` for registration of time slots.

[3] The access should already be given to you if you are registered for the course. If you are denied access, talk to Lill Hege Pedersen who works in the Department's reception (D-block, first floor in "Elektrobygget"), or contact the teaching assistant.

(a) Helicopter



(b) Model

1.4 Safety

- Before you start, read the safety instructions attached to your work station.
- Keep your fingers (and other 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 poorly. It is also expensive and a bit fragile, so avoid crash landings. One person in the group should always be prepared to switch off the system when it is running (stop-switch), while another one should be standing prepared to catch the counterweight of the helicopter to avoid crash landing. **Do not grab the end with the 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.**

1.5 User guide

1.5.1 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, start Matlab.

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, an initialization file containing the physical parameter values for the helicopter, `init_heli.m`, has been prepared. **Note that there are different initialization files for helicopter 1-2 than for helicopter 3-10.** The initialization must be run before your Simulink program can be built as it contains calibration data for the

encoders and other configuration parameters. These files can be downloaded from "Blackboard"[4].

You can add new blocks into the Simulink diagram by opening the "Library browser" (the symbol left of the gear symbol), or by single-clicking in the diagram and writing the name of the block you need.

For a significantly faster building procedure and to prevent buffer overflow, copy the Matlab/Simulink files to your folder on the hard drive of the computer:

`C:\Users\username`

where `username` 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.

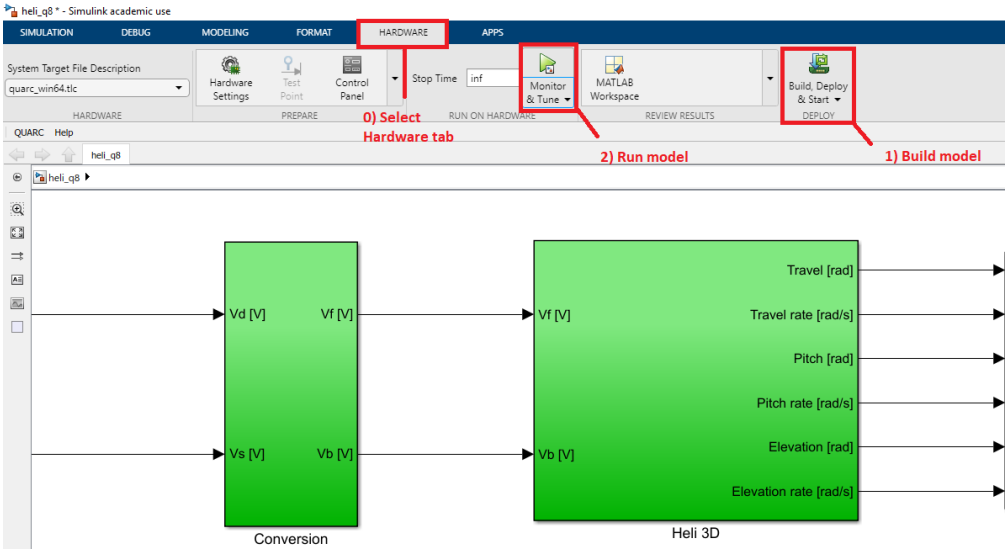


Figure 1.2: Overview of initial Simulink window.

1.5.2 Running

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. There are several ways to start your model. First the implemented logics needs to be built using the *build* button. See Fig. 1.2 for reference. Select it from the dropdown menu under *deploy*. Next, the model can be run by clicking deploy and start under the same menu or by clicking *Monitor & Tune*. The model can also be started through the scope interface. When the source code has been built, Simulink is used to run the implementation in real-time. Before running the build+start sequence you have to power on the helicopter using the power switch on top of the table. It is important that the helicopter is powered on before executing the program. Some drivers might crash if the helicopter does not have power requiring a re-boot of Windows. Press the stop button in the Simulink interface (either under the hardware tab or in the scope

[4]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.

interface) to stop the execution. In some cases, when stopping the model using the stop-switch on the table, the model will restart when power is turned back on.

When you make changes to the Simulink diagram, by e.g. adding new blocks, you will have to `build` again before you can start the model. 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` nor a restart of the program.

1.5.3 Recording data

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. 1.3. 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. 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: `Hardware` → `Control Panel` → `Signal & Triggering` → `Trigger options`, you should increase the duration to an appropriate value.

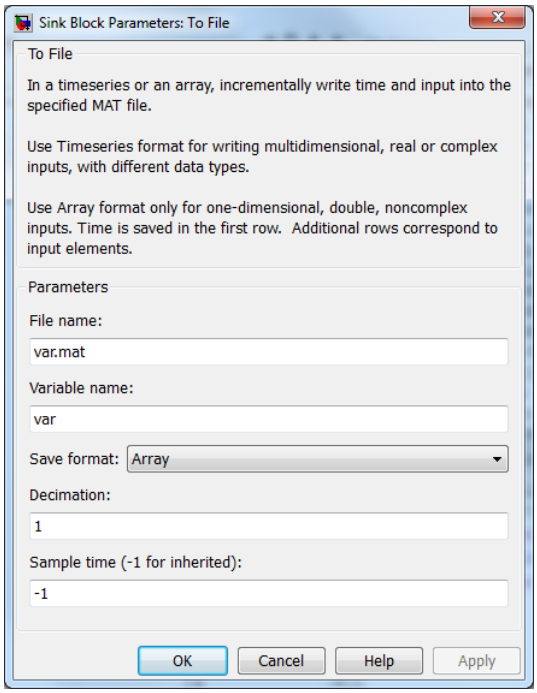


Figure 1.3: Sending data to a file

1.5.4 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 **dont 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 the 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).

1.6 Debugging guide

During your work in the helicopter lab you will face scenarios where things can break, hardware might act up, and pesky bugs can pop up. Even though it might be frustrating, remember that **it's all part of the learning process**. This hands-on experience will equip you to handle unexpected issues that often crop up when working with hardware and algorithms. Just like in your future professional work, these challenges are stepping stones in your growth as an engineer in a real-world industry setting.

1.6.1 Helicopter

These errors/bugs are relevant for all four tasks.

- No power from one or both of the rotors. Check that you have voltage signals entering the helicopter block, initially named "Heli 3D", see Fig. 1.4. If the signals are connected the error is most likely a loose, broken or disconnected power cable. Contact the teaching assistant. Do not try to repair this yourself.

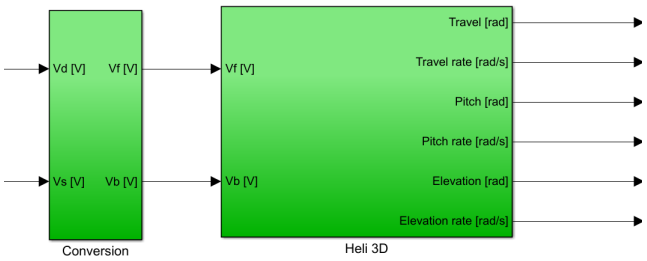


Figure 1.4: Voltage signals entering helicopter block.

1.6.2 QUARC and Simulink

These errors/bugs are relevant for all four tasks.

- If you recieve and error with the text "buffer overflow" it is most likely due to the program being built from files stored on an external hard drive (memory stick).
  1. Store your files

2. Close your files
3. Copy the files from the memory stick/external hard drive over to the local hard drive on the computer.
4. re-open files.
5. Clean and re-build.

This is the most common bug in this section.

- The error "algebraic loops" tells you that you have a loop in your program making a value directly dependent on itself at the current time-step. You are only allowed to loop if there is a time-delay, memory block, integrator, transfer function, or other dynamic block in the loop.
- If you receive an error with the following text: "An operating system specific kernel-level driver for the specified card could not be found". Turn the power to the helicopter off for 30 seconds (stop-switch). Then make sure that the helicopter is turned on (stop-switch) and try to clean and re-build the program. This error might be caused by a Quarc driver on the computer that has crashed, this often happens if you attempt to run the program while the helicopter is off. To restart the driver you must restart the computer.

It can be useful to verify the signal dimensions. This can be done by pressing `Debug` → `Information Overlays` → `Signal Dimensions`. Note that it may need to be toggled off and on again to work. The most common mistake causing the wrong dimension is to have "Element-wise(K.*u)" multiplication in gain blocks instead of "Matrix(K*u) (u vector)".

It can sometimes help to clean and re-build the program. Cleaning can be done by pressing `Clean all...` in the `QUARC` menu in Simulink.

1.6.3 IMU

These errors are only relevant for task 3 and 4.

- If you receive an error with the following text: "It was not possible to connect to the specified URI". This is either caused by having specified the wrong PORT for the IMU application, or by the Arduino not being detected by Windows. Open device manager and check that you have the correct port. The PORT number to be used is the number after "COM" under Ports in Device manager. The name of the port may vary from helicopter to helicopter and the number of the PORT may change when the Arduino is powered on and off. If the Arduino does not show up, contact the teaching assistant. The port may change between sessions. **This is the most common bug in this section.**
- If you do not receive an error but there is no new data from the IMU. The Arduino might have crashed and requires a re-start. Turn off the power supply through the "stop" swith on the helicopter table. Wait a few seconds and then turn it on again. If you continue to not receive data, contact the teaching assistant. A typical symptom of a crashed Arduino is that your estimated state values increase or decrease to $\pm\infty$.
- If the values that come from the IMU-block are unreasonable, either constant or super high, restart the Arduino. **Make sure you use the correct IMU block for your helicopter. There is one for helicopter 1-2 and one for 3-10**. If this does not work contact the teaching assistant. Make sure to look at the values that come out of the IMU block to ensure that the problem is not caused by your code/estimator. Remember that

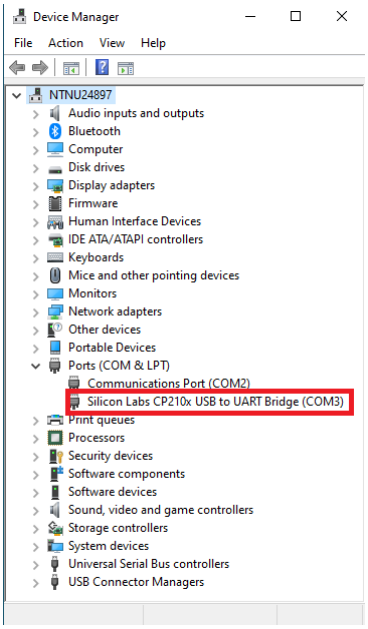


Figure 1.5: Windows device manager showcasing the name and value of the PORT to be used. The name may vary from helicopter to helicopter and the port value may change as the helicopter is powered on and off.

there is a difference between *estimate* values and *observation/measurement* values. If you receive new data, and the data looks good. Then the problem is in your code.

2 Tasks

Make sure you read the introduction before continuing, especially the chapter on safety. Follow the debugging guide before asking for help regarding technical difficulties. A friendly reminder that the student assistants are there to help you with the theory and discussions about the helicopter response, not to debug your code.

Load the Simulink template file (heli_q8.slx) as well as the Matlab initilization file (init_heli_1_2.slx or init_heli_3_10.slx) for your specific helicopter, which are both found on Blackboard (use a portable hard drive as the computers in the lab don't have internet). Any implementation of logics, variables, constants, etc., is meant to be done in Simulink or Matlab. You will mainly control the helicopter using a joystick as input. However, it may be easier to do systematic testing using a step-function as input (see the Matlab and Simulink tips & tricks document on Blackboard).

Before starting the tasks below you should have completed the corresponding helicopter lab preparation tasks.

2.1 Part I – Monovariable control

2.1.1 Task 1 - Manual control

The first attempt to control the helicopter will be done using feedforward. 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. The motors do not need a large voltage difference to increase the pitch angle. It might therefore be a good idea to add a small gain from the output of the joystick to the voltage difference, V_d. Remember to press the build-button in Simulink after changing the logics (adding, removing or connecting blocks). It is rather difficult to control the helicopter this way. Be careful not to crash the helicopter! One person in the group should be ready to catch the counterweight.

2.1.2 Task 2 - Parameter identification

Before you continue, identify that the positive direction given by the encoders matches the positive direction given in the task. Handle any deviations. The identification can be done by manually moving the helicopter with the rotors off (remove any inputs/connections to V_d and V_s in Simulink and rebuild).

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 elevation angle, e. Add constants to the encoder outputs such that the output of the encoder for the elevation is zero when the arm between the elevation axis and the helicopter head is horizontal.

Before we can implement the controller developed in the preparations, we need to determine the motor force constant K_f. By measurement, obtain the value for the voltage V_s which makes the helicopter maintain the equilibrium value $e = 0$ (the arm between the elevation axis and the helicopter head is horizontal). Note that this values is equal to $V_{s,0}$. Use this value to calculate

the motor force constant K_f. You should have an expression for the relationship between $V_{s,0}$ and K_f from the preparatory work. Use the calculated value of K_f for the remainder of the lab.

2.1.3 Task 3 - PD control

In this task, the **elevation** controller already given in the Simulink diagram should be used. The output from the elevation controller is $\tilde{V}_s$. Add $V_{s,0}$ to the output of the controller to get V_s.

Implement the PD **pitch** controller developed in the preparations. Use the x-axis of the joystick as the reference signal, p_c.

Experiment according to your test-plan/hypothesis developed in the preparation. Remember to save data during your testing in order to post-process/plot later for the report. Discuss your findings. You are allowed to change the test-plan if you figure out that there are other tests that are more interesting.

2.1.4 Task 4 - For evaluation

Store different configurations of your Matlab code/Simulink that showcase some of the interesting experiments you have done, which can quickly be loaded in your evaluation day. Make sure there are scopes available for signals of interest. Make sure the helicopter is flyable using the joystick.

2.2 Part II – Multivariable control

2.2.1 Task 1 - LQR implementation and experimentation

Implement the controller developed in part 2, problem 3, in the preparation assignment (no integral effect). Let the x-axis of the joystick provide the reference p_c for the pitch angle, and the y-axis the reference $\dot{e}_c$ for the elevation rate. Note: this is elevation rate, not elevation angle.

The gain matrix $\boldsymbol{K}$ can be computed by using the Matlab command `K=lqr(A,B,`$\boldsymbol{Q}_{\text{LQR}}$`,`$\boldsymbol{R}_{\text{LQR}}$`)`. Experiment with different values for $\boldsymbol{Q}_{\text{LQR}}$ and $\boldsymbol{R}_{\text{LQR}}$ according to your test-plan/hypothesis developed in the preparations day 2 problem 4. Remember to save data during your testing in order to post-process/plot later for the report. Discuss your findings. You are allowed to change the test-plan if you figure out that there are other tests that are more interesting.

2.2.2 task 2- Integral LQR implementation and experimentation

Extend your controller from task 1 to incude the integral action developed in part 2, problem 5, in the preparation assignment. Remember to use the augmented matrices from the preparation assignment.

Experiment according to your test-plan/hypothesis developed in part 2, problem 5, in the preparation assignment. Remember to save data during your testing in order to post-process/plot later for the report. Discuss your findings. You are allowed to change the test-plan if you figure out that there are other tests that are more interesting.

2.2.3 Task 3 - For evaluation

Store different configurations of your Matlab code/Simulink that showcase some of the interesting experiments you have done (with and without integral effect), which can quickly be loaded in your evaluation day. Make sure there are scopes available for signals of interest. Make sure the helicopter is flyable using the joystick.

2.3 Part III – Luenberger observer

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. There is a difference between the IMU.slx file for helicopter 1-2 and 3-10. Make sure you download the correct file for your helicopter. 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. Note that the port might change from one lab day to the next! 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. 1.1b.

NB: Read the debugging guide in the introduction before asking for help!

2.3.1 Task 1 - IMU characteristics

This task is meant to make you familiar with the IMU, its measurements, capabilities and limitations.

In Simulink, disconnect the voltage-signals into the Heli 3D block to make sure we can move the helicopter by hand without the rotors spinning. 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!). Discuss your findings, 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.
- Do you see any limitations on using the gyroscope or accelerometer readings as state feedback in our controllers?

The IMU readings may deviate over time such that there's an offset between the IMU measurements and the encoder measurements. If this is the case for your helicopter, add constant offsets to correct for these biases. It might be easier to detect these biases after doing the transformations in the next task.

You should spend a fair bit of time on this task to make sure you fully understand the IMU readings and how they vary when you rotate the helicopter around different axes.

2.3.2 Task 2 -Transformations

We wish to use the accelerometer measurements to indirectly measure the orientation of the helicopter. Implement the equations for pitch and elevation you developed in part 3, problem 3, in the preparation assignment, as a MATLAB function block in Simulink. During start-up the measurements from the IMU will be zero, making the denominators in the expressions for

the angles zero. Implement some logic in the MATLAB function block to ensure that the block always outputs a valid number.

You probably noticed that we could not use the gyro-measurements directly as state feedback in the controller 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 pitch and elevation derived from the accelerometer as the euler_angle inputs You can also test using only the encoder angles as input to the euler_angles. What is the difference?

Compare the transformed gyro output with the rates from the encoder, and the transformed accelerator measurements with the elevation and pitch angle from the encoder. Verify that they match, and discuss any discrepancies. Remember to add constant offsets for any biases you might observe.

2.3.3 Task 3 - State estimator

Implement the linear observer presented in part 3 problem 4 in the preparation assignment. Let $\hat{\mathbf{x}}$ be the estimate of the states we want to consider. Use $\mathbf{A}$ and $\mathbf{B}$ as derived in the preparation. Use the latest well-working controller you developed in the previous days. It is preferred that you use the LQR with integral action if it worked well. **NB:** We are allowed to estimate states even though we are not controlling the same states, as in the case here. The state vector for the controller and the state vector for the estimator are different, and hence the state matrices are different. Keep the controller problem and estimator problem separate. A common mistake is to utilize, for instance, the same $\mathbf{A}$ matrix in the estimator as in the controller, even though the state vector is different.

Change $\mathbf{C}$ such that $\mathbf{y}$ contain all the measurements from the IMU, that is:

$$\mathbf{y} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} \tag{2.1}$$

Use the *relevant* estimated states as state-feedback to the controllers.

Before starting your experimentation, remember to take extra care when flying the helicopter while testing your estimator. A bad choice of poles can make your normally well behaved helicopter misbehave. Make sure one person is ready to grab the counterweight.

Experiment according to your test-plan/hypothesis developed in part 3 problem 4 in the preparation assignment. Remember to save data during your testing in order to post-process/plot later for the report. Discuss your findings. You are allowed to change the test-plan if you figure out that there are other tests that are more interesting.

In your preparations you found the minimal set of states that had to be measured to make the system observable. Change your estimator to only use these states as input and discuss your findings.

2.3.4 Task 4 - For evaluation

Store different configurations of your Matlab code/Simulink that showcase some of the interesting experiments you have done (with varying $\mathbf{C}$ matrix), which can quickly be loaded in your evaluation day. Make sure there are scopes available for signals of interest. Be prepared to set

up new scopes with other signals on our request during the evaluation. Make sure the most noisy signal is in the back such that all the signals can be seen. Make sure the helicopter is flyable using the joystick.

2.4 Part IV – Kalman filter

2.4.1 Task 1 - Noise estimate

To begin with, we need an estimate of the measurement noise, $\boldsymbol{R}_{\mathrm{d}}$. 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. Analyze the noise in both cases. Compare the covariance for the two cases. Explain the difference. Use the experimental covariance while flying as your $\boldsymbol{R}_{\mathrm{d}}$ matrix. Note that MATLAB has functions for covariance and autocorrelation.

Hint: Dimension of the matrix to be send into Matlab's cov(...) function; one column for each state, One row for each measurement in the time-series.

2.4.2 Task 2 - Implementation

Implement the prediction step as a Matlab function in Simulink. Implement the correction step as a separate Matlab function that also includes the new-data signal.

Data from the IMU arrives at a slower rate than the update frequency of the Simulink program. This can be identified by looking at the "New data" output from the IMU. This results in the output from the IMU being constant over multiple time-steps. In the Luenberger observer we ignored this fact, in this task we will take this into account by only correcting our predictions when new data has arrived. When there is no new measurement available, make the corrected state estimate and error covariance be equal to the predicted state estimate and error covariance.

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} \tag{2.2}$$

$$\hat{\boldsymbol{P}} = \bar{\boldsymbol{P}} \tag{2.3}$$

Nb: Take extra care with regards to the time-steps of the predictions and corrections. The unit-delay block in Simulink can be used to delay the signal one time-step.

If you managed to model the measurement biases from part 4, bonus task 2 in the preparation assignment, implement this model. Test out the implementation and document it in the report. If it worked then use this version for the rest of the lab. Use the corrected estimated states as state-feedback to the controllers.

2.4.3 Task 3 - Experimentation

Experiment according to your test-plan/hypothesis developed in part 4 problem 2 in the preparation assignment. Remember to save data during your testing in order to post-process/plot later for the report. Discuss your findings. You are allowed to change the test-plan if you figure out that there are other tests that are more interesting.

Add a manual switch to the new-data signal, such that you can artificially disconnect it. Experiment with disconnecting and connecting the signal while the program is running. Look at the corrected state estimate, $\hat{\boldsymbol{x}}$, and the covariance of the estimate, $\hat{\boldsymbol{P}}$. Discuss your findings with basis in theory.

Discuss similarities and differences between the Luenberger observer and the Kalman filter.

2.4.4 Task 4 - For evaluation

Store different configurations of your Matlab code/Simulink that showcase some of the interesting experiments you have done (with and without integral effect), which can quickly be loaded in

your evaluation day. Include scopes that plot the raw IMU data, the estimates from the Kalman filter, and the encoder values on top of each other. Make sure the most noisy signal is in the back such that all the signals can be seen. Include the manual-switch that lets you disconnect the new-data signal in the presentation copy.

2.4.5 Task 5 - Oral evaluation

Ensure that you have all the files that you need ready for the oral evaluation. The schedule at the evaluation is tight, so you need to be able to get the helicopter up and running as quick as possible. Make sure you don't save your Simulink diagrams in a newer version of MATLAB than the one at the lab. This will save the simulink diagram in a newer format that cannot be opened on the lab. You can make a file compatible for earlier versions of MATLAB by pressing *file - Export Model to - Previous version...*

Bring a stabled color printed, **A4**, copy of the report to the oral presentation. The printers are guaranteed to fail when you are in a hurry, so don't attempt to print the report 15 min before the evaluation.