TTK4135 Optimization and Control
Helicopter Lab

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Trondheim
Updated: January 2025

Contents

1 Practical Information

If you've taken the course TTK4115 Linear System Theory, then you've implemented simple feedback controllers to control the helicopter with a joystick. In this course you will implement open-loop predictive control, which uses optimization to predict optimal input/state trajectories a certain amount of time into the future. You will then stabilize this optimal trajectory using feedback control (LQ).

Some general information about the LAB:

- The assignment is compulsory, and should be performed in groups of two students.
- The problem formulation and MATLAB files can be downloaded from Blackboard.
- The helicopters are in Elektroblokk B, rooms B113, B117, B121 and B125 (at least in that hallway, the exact rooms may vary). The helicopters are numbered 1 to 10. Helicopter 2 is used as a spare, please do not use this.
- If you can't access the building/lab-rooms, contact the teaching assistant (or lecturer).
- If you can't log on to the lab computers, you should get a user id for the computer network at Department of Engineering Cybernetics. This can be obtained by consulting the department office at The Department of Engineering Cybernetics.
- The project work is based on using MATLAB/Simulink and the Optimization Toolbox for optimal control.
- In the lab, MATLAB version R2020b is installed (as of last update of this document), including Simulink and QUARC. Be aware of what version you are using for your prepareatory work, as there may be compatibility issues.
- Hints on how to solve parts of the exercises might be posted on Blackboard and updated as it becomes clear what most students find difficult.
- Each group should hand in only one report. If you cooperate with other groups, which is allowed, each group should still do all tasks separately, and write a separate unique report. The tentative deadline for handing in the project work is April 06 2025, in Blackboard by 23:59.

2 Health, Safety, and Environment ("HMS")

The first three items on this list should be obvious, but to be on the safe side:

- Keep fingers off the propellers while they are running.
- Remember to turn off the power-module when you leave the lab.
- Stand by to catch the helicopter when it is flying. The system is rather unstable and quite sensitive, so avoid crash landings. Also, stand by to either turn the power supply off or to stop the program.

Furthermore, you are required to

- Familiarize yourself with the general safety instructions for the department (`https://www.itk.ntnu.no/english/about/hse`), including escape routes, fire extinguishers, and first aid.
- Familiarize yourself with safety instructions for lab.
- Report all equipment errors.
- Think safety: Minimize both the probability and consequences of unwanted events,

$$\min risk = frequency \times consequence \tag{1}$$

3 Objective

The purpose of this exercise can be summarized by the following items:

- The students should get practice in formulating a dynamic optimization problem, as well as discretizing and solving the resulting problem using a computer.
- The exercise should illustrate the implementation and practical use of optimal control with and without feedback.
- A modern environment for real-time system development based on automatic generation of program code from Simulink block diagrams is used.

The students are expected to spend about 24 hours working on this project.

4 Lab report

A lab report describing what has been done in this project work should be handed in after the project work is completed. For each part of the exercise the report should include

- printouts of data from relevant experiments (plots),
- relevant model derivations, calculations and parameter values (remember to include units and scaling),
- discussion and analysis of the results,
- Simulink diagrams and MATLAB code that shows what has been done. (do not include code that is the same as in the previous task or that was provided via template and has not been edited)

When the report is evaluated, documented calculations, experimental results, analysis and discussion of the results, skills in MATLAB/Simulink, and the quality of the written documentation of the results will be emphasized.

The report is graded with "approved" or "not approved", and must be approved to take the exam in TTK4135.

4.1 Plots

When experimenting at the lab, save all data as it's gathered! Do not plot the data, then only save the plot. If you don't save the data itself, you cannot replot it later if you change your mind on how the data is presented, or if you forget to add labels, etc. Another trick is to save the figure as a "matlab figure", to easily open the figure at a later time, without having to rerun the code. Finally, when adding figures to the report, make sure to use vector graphics rather than pixel based images. Vector based images can be zoomed in on while remaining crisp/sharp, which is a neat way to check is your figure is vector based. A simple way making vector figures from your MATLAB plots is to save the matlab figure as a ".pdf", then simply adding this pdf to in you latex document (may need to be cropped). Search the interent if you want to find a more streamlined way of adding vector images of your plots to your report.

Also, do not add large figures to your report! Use "subplot(3,1,p)" or similar for the typical trajectories you get in this report. One does not want to scroll through pages of plots to get to the next section! General rule: Figures should be as compact/small as possible, while remaining clear, tidy and readible.

All plots must contain/be:

- Title
- Labels on axes
- Units
- Desciption of what all graphs represent (f.ex. legends)
- Properly scaled
- Relevant to the discussion
- Vector graphics
- Clear (the reader must not misunderstand what is presented)
- Tidy (the plots should be organized such that the data is presented in a logical/reasonable and clean/pretty manner, such that reading them is easy)
- Readable (the numeric values should be somewhat interpretable)

5 System Description

The helicopter consists of a base with an arm attached, as shown in Figure 1. The arm has the helicopter body at one end and a balance weight at the other end, see Figure 2. The arm can be moved up and down around an elevation axis and rotate around a vertical axis (travel). The helicopter body itself can also rotate around an axis normal to the arm (pitch). These three angles are measured with optical position sensors.

Two manipulated variables are available. These are the two DC motors with propellers attached. The force from the propellers is assumed to be proportional to the voltage applied. The counter balance is adjusted so that the weight to be lifted by the propellers is approximately 50 g. If you apply a voltage of approximately 1.5 V to each motor, the helicopter will take off from the ground.

The physical data (parameter values) can be found in Table 1 (page 12) and in the MATLAB file initF.m which are posted on Blackboard.

5.1 QuaRC, Simulink, and Realtime Workshop

QuaRC is Quanser's new, state-of-the-art rapid prototyping and production system for real-time control. QuaRC integrates seamlessly with Simulink to allow Simulink models to be run in real-time on a variety of targets, such as Windows and Linux.

The control program is made in MATLAB and Simulink, see the example in Figure 3. There are separate Simulink blocks for communication

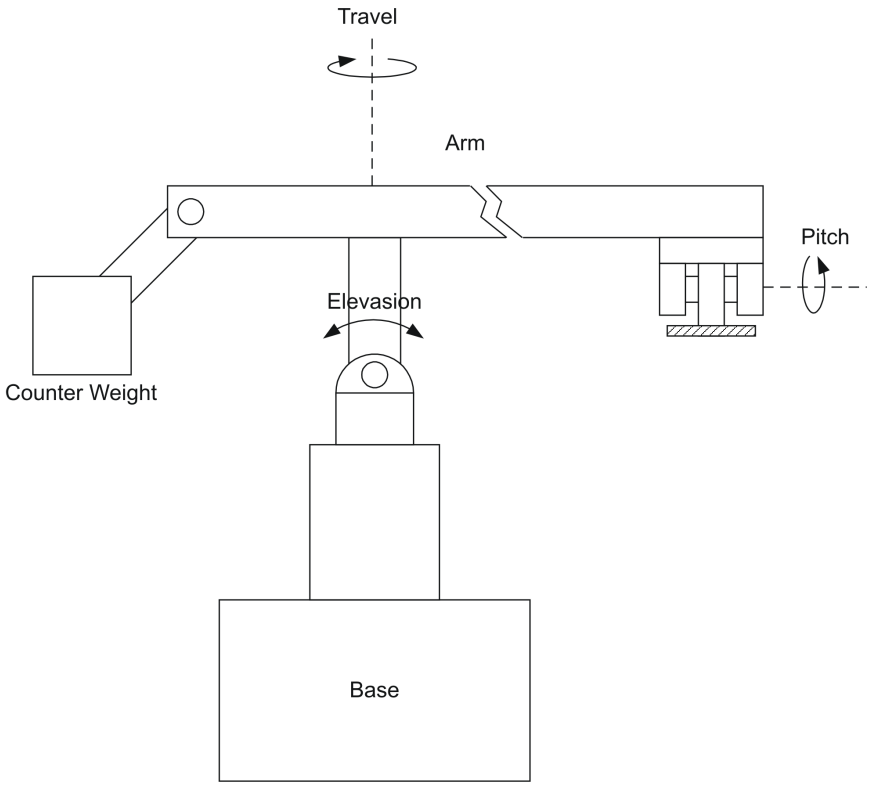

Figure 1: Cross section of the Helicopter system.

through the IO-card. After a block diagram for the controller is made in Simulink, QuaRC and Realtime Workshop are used to run the system. If these programs are installed, the QuaRC menu is available in Simulink. Choose "Monitor & Tune" to build, deploy, connect and start the code, see Figure 4. You can also do this step by step, as can be seen in the figure. If you only want to try to build the code (without starting the helicopter), choose "Build for Monitoring". Note that the button "Build, Deploy & Start" (with a red cross) should not be used because plotting and storing data is disabled! The following happens when the build command is executed:

- Real-Time Workshop converts the Simulink diagram to C code.
- The code is compiled and linked using Visual C++ into a QUARC real-time executable.

After this, the program is ready to be deployed. Now the power module can be turned on, and the program is started by doing the last steps (Deploy, Connect and Start), or alternatively clicking the "Monitor & Tune" button.

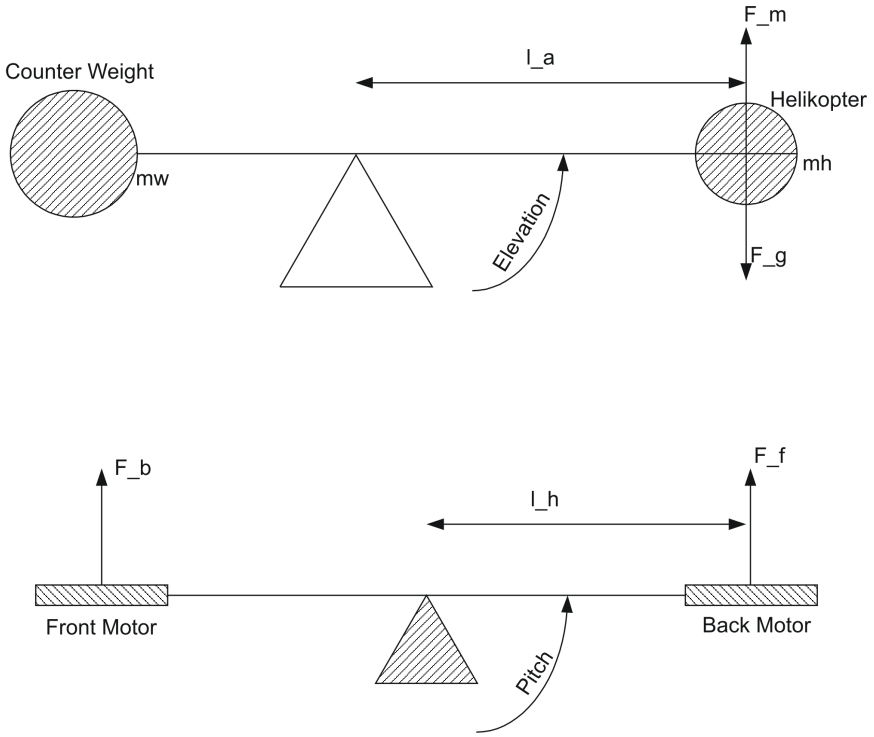

Figure 2: Sketch of the helicopter.

To carry out measurements on the system, the sampled data must be transferred to MATLAB. The "To Workspace"-block in Simulink does not work in real-time, so real-time plots must be used. This is the usual "Scope" plot in Simulink.

Before you start a measurement series, the following parameters must be set in the plot window:

- Buffer size. The length of the measurement series that the plot will "remember" in seconds
- Sampling frequency.
- The time window that will be displayed in the plot. You should set this parameter equal to the buffer size. If the option for Maximize Plotting Frequency is selected, the highest possible sampling frequency will be implemented.

After a measurement series has been recorded, you can export the data to the MATLAB workspace or to a file. Go to File $\rightarrow$ Save in the plot of interest. The best thing to do is probably to save the data as a mat-file, and keep them for later use.

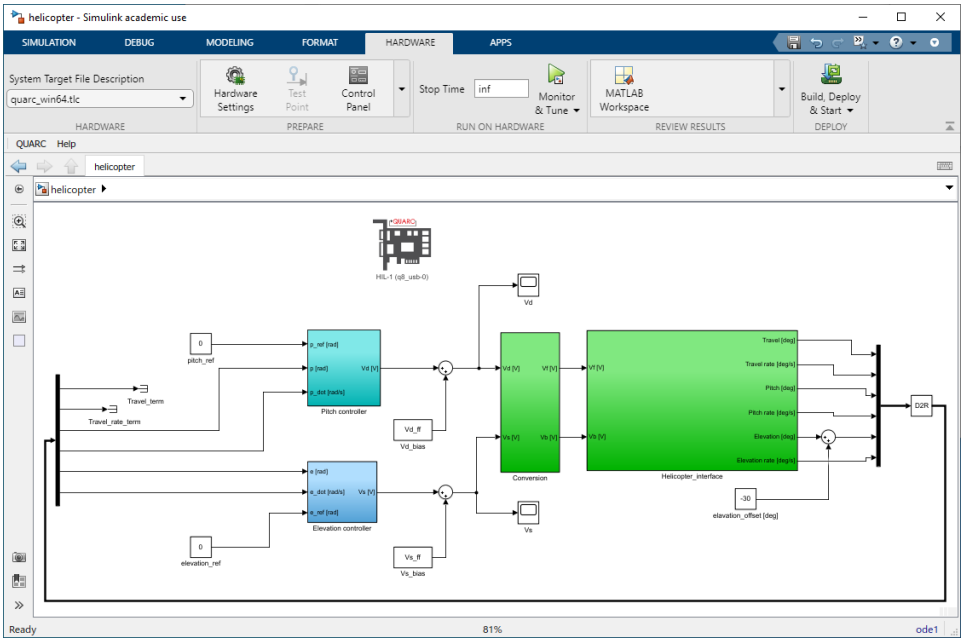

Figure 3: An example of SIMULINK diagram with QuaRC.

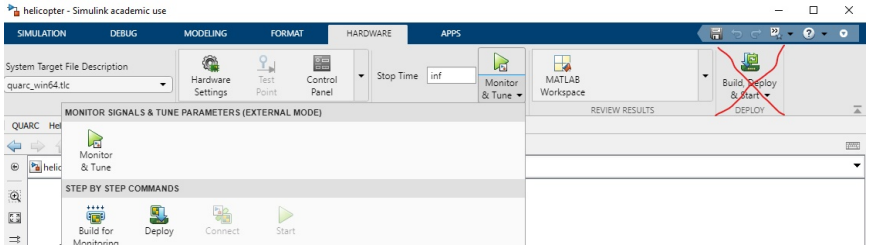

Figure 4: Build QuaRC.

Notice: If it turns out that the data you have transferred has a much higher sampling frequency than the one you chose, try to record a new measurement series in the same plot-window and transfer the data again.

Note: If the above method of recording data does not work, see Section 11.

5.2 Hardware

5.2.1 Quanser Q4

The Q4 is an innovative HIL ("hardware in the loop") control board with an extensive range of input and output support, see Figure 5. A Quanser Q4 IO card is connected to the PC bus. The Quanser Q4 card measures the physical signals coming from the helicopter and converts them into digital

signals that can be read by the computer. The card also converts the signals from the computer that is used as input to the helicopter.

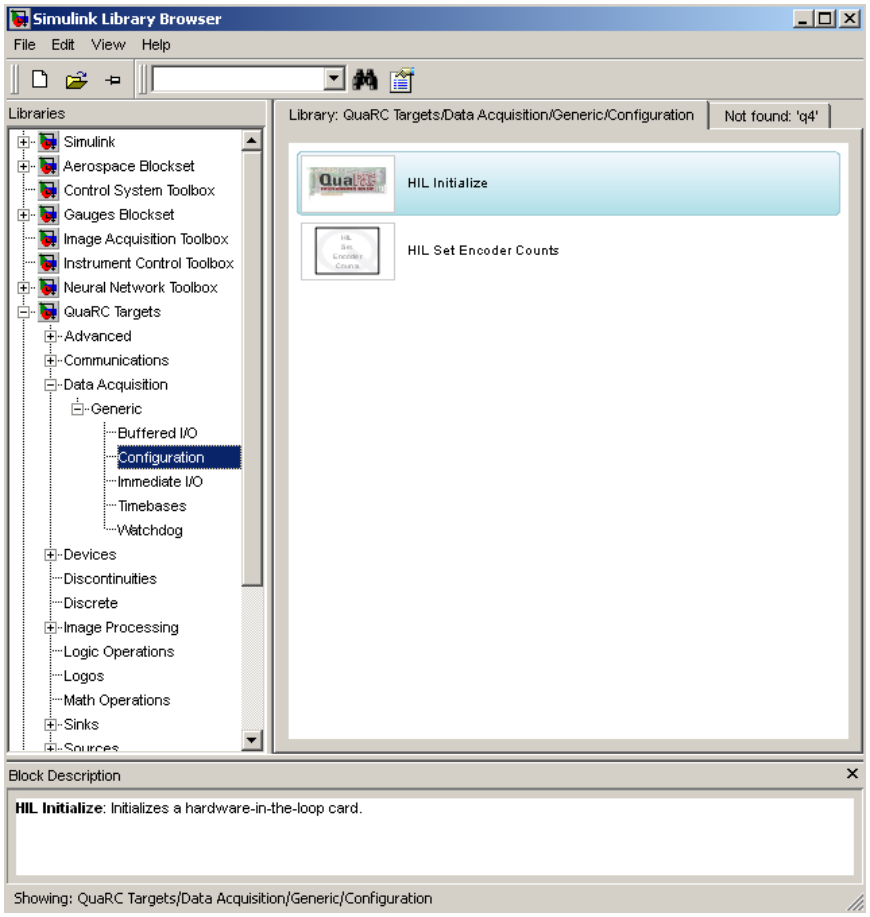

Figure 5: HIL in Simulink Library Browser.

The following blocks are available:

- Analog input.
- Analog output.
- Digital input.
- Digital output.
- Encoder input. (Gives out an integer that indicates the condition of a position sensor.)

5.2.2 Power Module

The power that operating the motors comes from one or two Universal Power Modules.

6 Some Advice

- Set up a work folder locally under: `C:\Users\yourUsename`, when you start the day. This will help avoiding many unknown errors. **This folder may be deleted until next time**, so you must save your work to a USB stick or use the "Home directory"/"hjemmeområde" functionality, before leaving the lab for the day. Running the code from a USB stick or the "hjemmeområde" directly is a bad idea and will quite often(/always) fail. The lab computers do not have access to the internet, so you may not use any other cloud based services.
- Errors may arise during compilation. It is a good idea to wait to start the helicopter, until the compilation is finished. It is also important that the compilation is done by "Build for Monitoring".
- If inexplicable error messages arise during compilation, there are a couple of things that may help. First, erase all compiled code and start over again. If this does not help, try restarting the computer.
- If something is not working properly or is broken, notify the student or teaching assistant.

7 Starting Point for the Programming

A ready-made Simulink block diagram should be used as the basis for your program. You will need the following files:

- helicopter: Simulink-diagram of helicopter, half-made.
- init0i.m: contains the physical data for the helicopter number i.

These files can be downloaded from Blackboard. Change the names of the init0i.m file to init.m. No backup is taken of your own files, so remember to do this yourself.

Some files that may be helpful for solving the project work are gen_aeq.m, gen_q.m, and gen_constraints.m. See the files for documentation/explanation.

Table 1: Parameters and values.

Symbol	Parameter	Value	Unit
l_a	Distance from elevation axis to helicopter body	0.63	m
l_h	Distance from pitch axis to motor	0.18	m
K_f	Force constant motor	0.25	N/V
J_e	Moment of inertia for elevation	0.83	$\mathrm{kg\,m^2}$
J_t	Moment of inertia for travel	0.83	$\mathrm{kg\,m^2}$
J_p	Moment of inertia for pitch	0.034	$\mathrm{kg\,m^2}$
m_h	Mass of helicopter	1.05	kg
m_w	Balance weight	1.87	kg
m_g	Effective mass of the helicopter	0.05	kg
K_p	Force to lift the helicopter from the ground	0.49	N

Table 2: Variables.

Symbol	Variable
p	Pitch
p_c	Setpoint for pitch
λ	Travel
r	Speed of travel
r_c	Setpoint for speed of travel
e	Elevation
e_c	Setpoint for elevation
V_f	Voltage, motor in front
V_b	Voltage, motor in back
V_d	Voltage difference, $V_f - V_b$
V_s	Voltage sum, $V_f + V_b$
$K_{pp}, K_{pd}, K_{ep}, K_{ei}, K_{ed}$	Controller gains
T_g	Moment needed to keep the helicopter flying

8 Model Derivation

To derive a model for the helicopter, one can formulate the moment balances; this is done below. For a list of parameters and variables, see Table 1 and 2.

We start with the derivation of the model for elevation:

$$J_e\ddot{e} = l_a K_f V_s - T_g \tag{2a}$$

so that

$$\ddot{e} = K_3 V_s - \frac{T_g}{J_e}, \quad K_3 = \frac{l_a K_f}{J_e} \tag{2b}$$

The following PD controller is implemented to control the height:

$$V_s = K_{ep}(e_c - e) - K_{ed}\dot{e}, \quad K_{ep},\, K_{ed} > 0 \tag{3}$$

This leads to the system

$$\ddot{e} = -K_3 K_{ed}\dot{e} - K_3 K_{ep} e - \frac{T_g}{J_e} + K_3 K_{ep} e_c \tag{4}$$

In addition, an integral term is added, giving a PID controller. We assume that the integral term counteracts the effect from the constant term $-\frac{T_g}{J_e}$, so that these two terms can be ignored. The resulting system for elevation is then

$$\ddot{e} + K_3 K_{ed}\dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c \tag{5}$$

The same can be done for the pitch angle:

$$J_p\ddot{p} = K_f l_h V_d \tag{6a}$$

so that

$$\ddot{p} = K_1 V_d, \quad K_1 = \frac{K_f l_h}{J_p} \tag{6b}$$

The PD controller

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p}, \quad K_{pp},\, K_{pd} > 0 \tag{7}$$

is used to control the pitch angle. This leads to the following closed-loop system

$$\ddot{p} = K_1(K_{pp}(p_c - p) - K_{pd}\dot{p}) \tag{8a}$$

or

$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \tag{8b}$$

Finally, we need a model for the travel ("vandring"). We start from the moment balance. For small pitch angles, the force needed to keep the helicopter flying is approximately K_p. The horizontal component of the force gives the acceleration around the travel axis:

$$J_t \dot{r} = -K_p l_a \sin p \tag{9}$$

We assume that the angle p is small, so that $\sin p \approx p$. Then,

$$\dot{r} = -K_2 p, \quad K_2 = \frac{K_p l_a}{J_t} \tag{10}$$

Note that a positive pitch angle gives a negative travel acceleration. Also remember that $\dot{\lambda} = r$.

The model we will use can then be summarize by

$$\ddot{e} + K_3 K_{ed} \dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c \tag{11a}$$
$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \tag{11b}$$
$$\dot{\lambda} = r \tag{11c}$$
$$\dot{r} = -K_2 p \tag{11d}$$

All angles in the equations above are given in radians. Remember that the measured angles from the helicopter model are recorded in degrees. The controllers for pitch angle and elevation are tuned, and further tuning of these controllers should not be necessary.

Based on the model, you may get the impression that there are no interactions between elevation and pitch angle/travel. This is a result of the assumptions made in the model derivation. Interactions between these variables are of course present.

9 Before you Start

The arrangement of the helicopters is as shown in Figure 6.

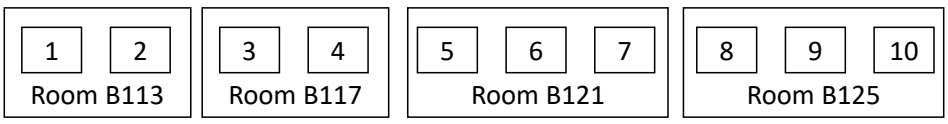

Figure 6: Arrangement of helicopters.

Since the helicopters are different, using the same helicopter for the entire lab project is recommended.

For the report, you will be asked to include MATLAB code and Simulink figures. To make it easier for yourselves, we suggest that you make copies of your work as you go. Do not just have one matlab file and one simulink file. For each subtask that involves running the helicopter, make a seperate folder. This will make it easier when it comes to the report, but it will also make it easier if you will need to re-run a previous task to collect data.

10 Exercises

The assignment is divided in four parts:

1. Repetition/introduction to use Simulink/QuaRC to control the helicopter. Familiarize yourselves with MATLAB/Simulink interaction.

2. Optimal control of pitch/travel with no feedback. An optimal open-loop input sequence that moves the helicopter 180 degrees should be calculated and implemented on the helicopter. (The helicopter mey not behave exactly as the predicted state trajectory, since the pre-calculated input trajectory is 'blindly' applied)

3. Optimal control of pitch/travel with feedback (LQ control). Feedback is now used to stabilize the open-loop state trajectory from part 2. (The helicopter should now behave more closely to what was predicted in part 2)

4. Optimal control of pitch/travel and elevation with feedback. (Similarly to part 2, an open-loop optimal input/state trajectory is calculated. That trajectory is then stabilized by a feedback controller)

The PID controllers modelled in section 8 "Model Derivation" can be assumed to be well tuned. We call these the "inner" control loops. We consider setpoints for the inner loops, (p_c and e_c), as the manipulated variables in ($u(t)$).

For each exercise, make sure you have done the section "Solve the following tasks before coming to the lab" (hereafter: prework) before arriving at the lab. **It is possible to do the prework for all the exercises before finishing the "At the lab" sections**. If you have not yet seen the theory in the lectures, fear not, all the required theory is either provided directly or a reference to the material is given.

10.1 Repetition/introduction to Simulink/QuaRC and MATLAB

This task consists of two introductory tasks. In the first task, you will learn a few key points on how to work with MATLAB and Simulink. In the second task, you will familirize yourself with the lab setup

10.1.1 Solve the following tasks before coming to the lab

Passing and storing data
The goal of this task is for you to familiarize yourselves with how you may pass data from MATLAB to Simulink. We hope that this will save you a good amount of time for the later tasks. In this task you will get experience in how to:

- Pass data from MATLAB to Simulink. This will be needed later to pass the optimal trajectories you find using quadprog/fmincon later on.
- Store recorded data to a file. This will be needed for the step below.
- Load the recorded data and plot it. This will be needed for the report.

1. For this task you may either create your own files to play around, or you may use those handed out at blackboard: "passing_data_training.m" and "dummy_system.slx".
2. Create a suitable object of the hardcoded reference trajectories `x_travel_ref` and `x_elevation_ref` . You may use the class *timeseries* in MATLAB. The first argument is the values you want to pass, i.e., the "trajectory". The second argument is a vector of the time-steps.
3. Load the timeseries in Simulink. You may use the block "From workspace" in Simulink. If you want to pass several vectors, of the same length, to Simulink, then you could either stack them together as matrix of several columns and make a timeseries out of it, or use two different blocks in Simulink.
    - If you use a matrix in the timeseries object, then you may want to use a "demux" block in Simulink to have access to each of the signals. Double click the "demux" block to choose the amount of outputs.
4. Store the recorded data. Here we recommend to use the block "To file". Just like the for the loading process, you may either use a "mux" to store several signals into one file, or you may use several "To file" blocks.

- Double click the "To file" block. By default, the data are stored as a timeseries. The lab setup only allows for "Array" option.
- You can also give the array a name, default is "ans". Please choose something more descriptive for your case.
- Please note that you, the users, must keep track of which signal corresponds to which column of the timeseries. The logic is straight forward: the upper input signal to the mux is the first column of the timeseries. However, if you keep changing the Simulink file and the order of the inputs of the mux, this could be hard to keep track of.
- *Additional info: If you want the trajectory passed to Simulink and the recorded data to have the same length, you should store both in Simulink. This may make your lives a bit easier.*

5. Load the recorded data for plotting. Here you may use the `loaded_data = load(name_of_file)` functionality in MATLAB.
   - Try plotting the data and make the figures look good, e.g., use title, legend, grid, axes descriptions, etc...
   - Find a way to store the plots in a format that are suitable for a report. Good choice: "eps". Can you store the figures both through code and through the Graphical User Interface?

When you have performed the steps above, you should be more confident in how to work with data in the setting of Simulink and MATLAB. Hopefully, you will now be able to focus on the more interesting part of the following tasks. Note: You are not supposed to include anything from this task in the report.

10.1.2 At the lab

Repetition/introduction to Simulink/QuaRC
This first exercise is meant to be a repetition for those who have completed the helicopter lab in TTK4115 Linear System Theory and a quick introduction to Simulink/QuaRC to those who have not.

Set up a work folder locally, see Section 6.

Use the files helicopter.slx and initXX.m from Blackboard. Choose the init file that corresponds to your helicopter.

Start MATLAB and go to your work folder. Run init.m and open the helicopter model in Simulink. Go through the different blocks in the Simulink diagram and figure out what they do. To run the helicopter, go to QuaRC $\rightarrow$ Build. This command converts the Simulink block diagram to C-code, compiles, links and downloads the program to QuaRC. When this is done,

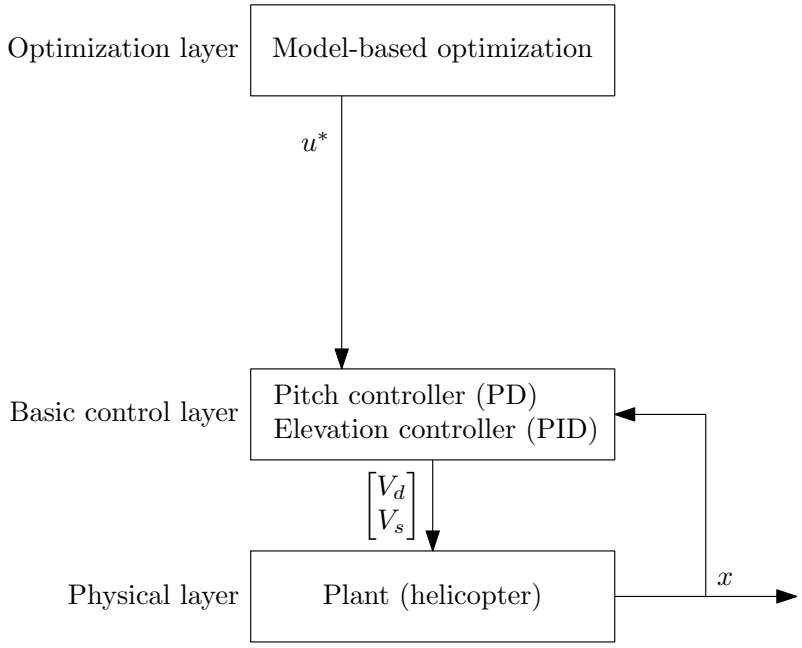

Figure 7: Illustration of the layers in the control hierarchy used in Section 10.2.

the QuaRC will start. Start the real-time program, and verify that the controllers work. Try changing the setpoint for pitch and elevation.

Try opening some of the real-time plots (remember that this must be done from the QuaRC). See for example the plots for pitch angle and elevation. Notice that all the states are not measured. The three angles are measured, but the angular velocities are estimated.

10.2 Optimal Control of Pitch/Travel without Feedback

In this part of the exercise we will disregard elevation, that is, we assume $e = 0$. We will then calculate an optimal trajectory x^* and a corresponding optimal input sequence u^*. This input sequence will be implemented as setpoints for the inner controllers, but we will not feed back the measured state to correct for deviations from the optimal trajectory. This control hierarchy is illustrated in Figure 7.

10.2.1 Solve the following tasks before coming to the lab

1. Write the model on continuous time state space form

$$\dot{x} = A_c x + B_c u \tag{12}$$

with $x = \begin{bmatrix} \lambda & r & p & \dot{p} \end{bmatrix}^\top$ and $u = p_c$. What are we modeling here? Is it just the helicopter? Discuss what the model includes, and how it

relates to Figure 7.

2. Discretize the model using the forward Euler method and write the resulting model on discrete time state space form

$$x_{k+1} = Ax_k + Bu_k \tag{13}$$

3. Calculate an optimal trajectory for moving the helicopter from $x_0 = \begin{bmatrix}\lambda_0 & 0 & 0 & 0\end{bmatrix}^\top$ to $x_f = \begin{bmatrix}\lambda_f & 0 & 0 & 0\end{bmatrix}^\top$ when the elevation angle is assumed to be constant. Use $\lambda_0 = \pi$ and $\lambda_f = 0$. Also implement the constraint

$$|p_k| \leq \frac{60\pi}{360}, \quad k \in \{1, \ldots, N\} \tag{14}$$

on the pitch angle. Why do we need this constraint? Is it necessary to also implement this constraints on the pitch reference angle p_c? We want to minimize the cost function

$$\phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + qp_{ci}^2, \quad q \geq 0. \tag{15}$$

Solve the optimization problem using the MATLAB function "quadprog". Try using the values $q = 0.12$, $q = 1.2$, and $q = 12$ as weights. Plot the manipulated variable and the output. Comment the results with respect to the different weights chosen. Remember that some useful files are posted on Blackboard. Use a sampling time of 0.25 s and $N = 100$.

Hints: To use the quadprog function in MATLAB, you need to formulate the optimization problem as a QP problem in standard form. Start by defining a large vector z containing all the variables that should be optimized:

$$z = \left(x_1^\top, \ldots, x_N^\top, u_0^\top, \ldots, u_{N-1}^\top\right),$$

and use this definition when constructing the objective function and constraints. For more hints read Chapter 3.5 in "Merging Optimization and Control" by Foss and Heirung, which should be on Blackboard. This chapter consists of two examples of "finite horizon optimal open loop optimization", which is exactly what we need! A template and some helper functions to these assignments have also been posted on blackboard. Using this template is optional but try to use a similar structure if you implement everything from scratch. Having a similar structure will make it easier for the student assistants to help you and evaluate your results.

For the daring:
The term $(\lambda_{i+1} - \lambda_f)^2$ penalizes the travel angle at all time points, by its deviation from the desired final angle λ_f. As the helicopter is not initiated at the final angle ($\lambda_0 \neq \lambda_f$), this term has large values for at least some of the first angle values ($\lambda_i, i = 1, 2, .., ?$). Is this a good way to make sure that the objective function drives the system to λ_f? How does this term scale compared to the other term, qp_{ci}^2? Could the optimization problem be formulated in another way, and still drive the travel angle to λ_f? Another feature of this objective function is that the helicopter will not accept being at $\lambda_f + 2\pi$, even though this is the same angle in the practical setup. Unless there are wires or similar that are tangled up if the helicopter rotates, the user may be ok with any number of rotations. It might be tempting to use the modulo operation to solve this problem. How might that look? Is there a potential issue with this approach?

4. The optimization in 3) gives the manipulated variable u. It may be a good idea to add some zeros at the beginning of the input vector, so that the helicopter has time to rise to and stabilize at $e = 0$ before the calculated open-loop sequence is applied. Also, adding zeros at the end of the vector is recommended to keep the helicopter stable after the input sequence is over. Add enough zeros to keep the helicopter stable for at least 5 s before and after the optimal input sequence is implemented. This padding (before and after) is already done in the handed out template.

5. The specific implementation of the QP algorithm used here can only weight deviations from origin ($\lambda_f = 0$). The position sensors on the helicopter are relative, which means that the position is reset to zero each time the helicopter is started. To get the helicopter to start in x_0 you have to add a suitable value to the measurement. For example, you can subtract 30° from the elevation measurement to get the helicopter to fly at a reasonable height when $e_c = 0$ (this is done in the given file). Implement the input sequence generated in 3) with $q = 1$.

   Hints: The easiest way to transfer the input sequence to Simulink is to use a "From Workspace" block (which can be found under "Sources" in the Library Browser). Have a look at the first exercise if you have forgotten how to do this.

   Note: You cannot run the simulink file without being at the lab, but try to prepare the simulink file as much as possible.

6. Ask yourselves: Does the input trajectory seem fair/reasonable based upon the goal of the task?

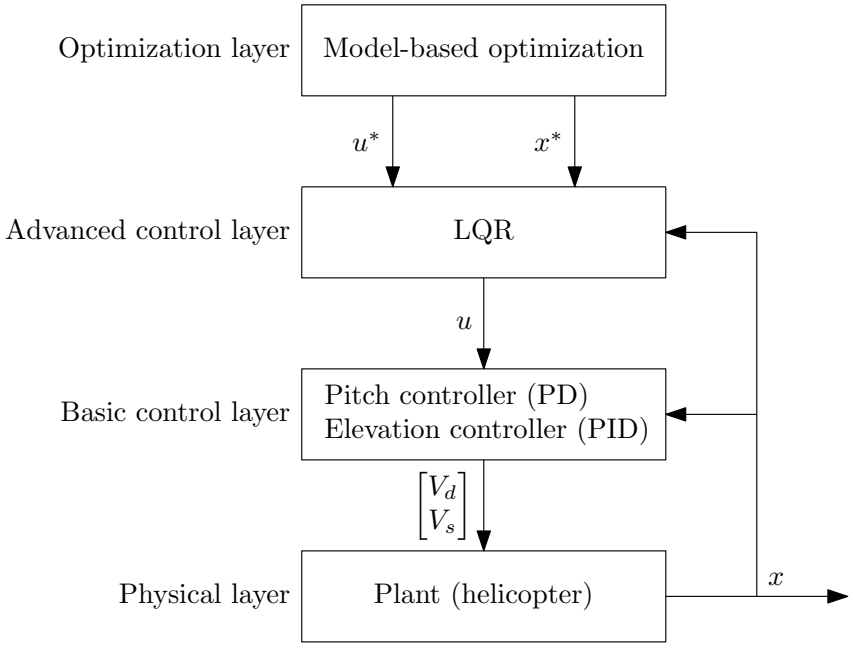

Figure 8: Illustration of the layers in the control hierarchy used in Sections 10.3 and 10.4.

10.2.2 At the lab

7. Run your setup. Does the helicopter end in the desired point x_f? What causes the observed deviation?

10.3 Optimal Control of Pitch/Travel with Feedback (LQ)

10.3.1 Solve the following tasks before coming to the lab

1. We now introduce feedback in the optimal controller. This is done by using an LQ controller. LQ stands for linear quadratic, which means that this controller minimizes a quadratic criteria for a linear model. From the optimal trajectory calculated in exercise 2, we have an optimal input sequence u^* and an optimal trajectory x^*. Using the following manipulated variable will introduce feedback:

$$u_k = u_k^* - K(x_k - x_k^*) \tag{16}$$

   As long as the system is following the calculated optimal trajectory x^* the calculated input sequence u^* is implemented. If a deviation from x^* is observed, the manipulated variable will be modified by the feedback term. This new control hierarchy is illustrated in Figure 8.

   A good choice for the gain matrix K must be found. The gain could be calculated in several ways (for example using pole placement), but

here we will calculate it as an LQ controller. An LQ-controller (in discrete time) minimizes the quadratic objective function

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^{\top} Q \Delta x_{i+1} + \Delta u_i^{\top} R \Delta u_i, \quad Q \geq 0,\ R > 0 \tag{17}$$

for a linear model

$$\Delta x_{i+1} = A\Delta x_i + B\Delta u_i \tag{18}$$

without including inequality constraints. Here, Δx and Δu are deviations from the optimal trajectory,

$$\Delta x = x - x^* \tag{19a}$$

$$\Delta u = u - u^* \tag{19b}$$

In MATLAB you can use the function dlqr to calculate the optimal K matrix. Q and R indicates how much you want to penalize deviation in the states, and how much you want to penalize use of the manipulated variable. Use diagonal matrices for Q and R. A diagonal matrix is easily made in MATLAB using the function diag.

2. Implement the feedback on the helicopter. This is done in Simulink using the following blocks Mux, Demux, Matrix Multiplication, Sum, and "From Workspace".

   Note: You cannot run the simulink file without being at the lab, but try to prepare the simulink file as much as possible. *Hint:* Notice that the "From Workspace" block can import several values (a matrix), meaning only one block is required for x.

3. An alternative strategy would be to use Model Predictive Control(MPC), which uses predictive optimization several times during the flight. There are two main ways of implementing an MPC control scheme:

   - (A) Rather than stabilizing the predicted trajectory x^* with an LQ regulator, one can simply re-optimize from the new measured position to get a new prediction x^*_{new}, then apply the newest prediction u^*_{new} to the system.
   - (B) Stabilize the initially predicted trajectory using MPC control rather than with LQ regulation.

   How would you implement the MPC scheme (A)? Discuss advantages and disadvantages with an MPC controller (A) compared to the controller you have implemented (prediction +LQ). Also, what would the structure in Figure 8 look like if you used MPC (A).
   For the daring:
   How would you implement the MPC controller (B), and what would the structure in Figure 8 look like if you used the MPC scheme (B). Here we don't want a detailed description, but only short explanation. What are some advantages and disadvantages of using (B) rather than (A)? Can you somehow combine these two approaches?

10.3.2 At the lab

4. Run the helicopter. Try different values for Q and R used in the LQ controller to generate K-matrix. Justify your choice

5. Discuss and analyze the results and compare them with those obtained in the previous task.

10.4 Optimal Control of Pitch/Travel and Elevation with Feedback

In this task we include the dynamics of elevation, e. The helicopter is moved from x_0 to x_f past a restriction causing the elevation angle to change during the flight. We must therefore add the elevation to the state vector; the setpoint e_c is a new manipulated variable in the system.

10.4.1 Solve the following tasks before coming to the lab

1. Write the system on continuous state space form with the two extra states e and $\dot{e}$. Use $x = \begin{bmatrix} \lambda & r & p & \dot{p} & e & \dot{e} \end{bmatrix}^\top$ and $u = \begin{bmatrix} p_c & e_c \end{bmatrix}^\top$.

2. Discretize the model using the forward Euler method and write the resulting model on discrete state space form.

3. Inequality constraints on the elevation should now be implemented. The constraint is

$$e_k \geq \alpha \exp\left(-\beta(\lambda_k - \lambda_t)^2\right) \ \forall k \in \{1, \ldots, N\} \tag{20}$$

This is a nonlinear constraint. A QP solver can only solve problems with linear constraints, so we must use another function to solve the optimization problem. Instead we will use "fmincon" as this an SQP-type algorithm. See the documentation for a description of the function and how to use it.

The criteria to be minimized is now

$$\phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2 \tag{21}$$

Start with $q_1 = q_2 = 1$ and see if there are other values that give a better result. Remember to include the equality constraints given from the model equations in the optimization problem. Use $\alpha = 0.2$, $\beta = 20$ and $\lambda_t = \frac{2\pi}{3}$. For every point in time a constraint on the form

$$c(x_k) = \alpha \exp\left(-\beta(\lambda_k - \lambda_t)^2\right) - e_k \leq 0 \tag{22}$$

must be implemented and passed to fmincon.

Start with a relatively short time horizon (12–15 time steps). Use $\Delta t = 0.25$ s. Try optimizing over a horizon of approximately 10 s, that is, $N = 40$. We use a shorter horizon in this exercise than in the previous exercises because performing the optimization over the same horizon will take some time.

4. Implement the changes needed in the simulink diagram. Introduce feedback in the same way as in exercise 3, except now the system has more states.

Note: You cannot run the simulink file without being at the lab, but try to prepare the simulink file as much as possible.

5. Ask yourselves: Does the input trajectory seem fair/reasonable based upon the goal of the task?

10.4.2 At the lab

6. Implement the optimal input sequence on the helicopter with an LQ controller active. Does the helicopter follow the trajectory?

7. Notice that the first 4 states in the model are completely decoupled from the last 2. There is clearly a connection between pitch and elevation in the real setup, since when $p = \frac{\pi}{2}$ there is no way for the

helicopter to apply force to neither increase nor maintain elevation. In Section 8, the model is derived, but it starts with: $J_e\ddot{e} = l_a K_f V_s - T_g$. Already here, the elevation dynamics are independent of pitch. The true relationship is sinusoidal in nature: $\ddot{e} = C_p V_s \cos(p) + C_e \cos(e)$. Since the model doesn't 'know' about how pitch affects the elevation dynamics one would think that the controller would crash the helicopter, why does it still work? (there are two good reasons)
If you wanted to improve the controller by accounting for this connection, there are ways one can do that. Below are some suggestions for how one might improve the controller, but some of them are bad suggestions and some are viable. For each suggestion, determine whether they are good or bad suggestions, and explain why. Suggestions:

A Rewrite Equation 2 as: $\ddot{e} = C_p V_s \cos(p) + C_e \cos(e)$, and write the dynamic on state space form: $\dot{x} = f(x, u)$.

B Add "$+ \sum_{k=1}^{N} \left(\dot{e}_k - \dot{e}_{k-1} - \Delta t C_p V_s \cos(p_k) - \Delta t C_e \cos(e_k)\right)^2$" to the objective function.

C For every few time steps during flight, adjust the Q-weight in the LQ controller based on the pitch angle:

$$Q = \begin{bmatrix} q_\lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & q_r & 0 & 0 & 0 & 0 \\ 0 & 0 & q_p & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & q_e & 0 \\ 0 & 0 & 0 & 0 & 0 & q_{\dot{e}} \cdot \cos(p) \end{bmatrix}$$

8. Optional exercise: Try implementing the suggested solutions above, and see if they work. Also try adding more constraints on the states, and play around with various nonlinear constraints. These may be constraints on maximum allowed speed $\dot{e}$ and $\dot{\lambda}$. This task does not give bonus points, but gives a very good overall impression.

11 Additional Information

11.1 Recording Data

If the method for recording data described in Section 5.1 does not work, try the following method instead.

1. Use the "To file" block, see Figure 9.
2. Connect the block to a signal you want to measure. Input some suitable parameters, and note that the format has to be Array — see Figure 10.
3. After running the system, you can access and plot the data using something like the following commands:

   ```
   >> load('e.mat')
   ```

   ```
   >> plot(e_measured(1,:0), e_measured(2,:0))
   ```

   where the first row of `e_measured` is time and the second row is the data. En example result is shown in Figure 11.

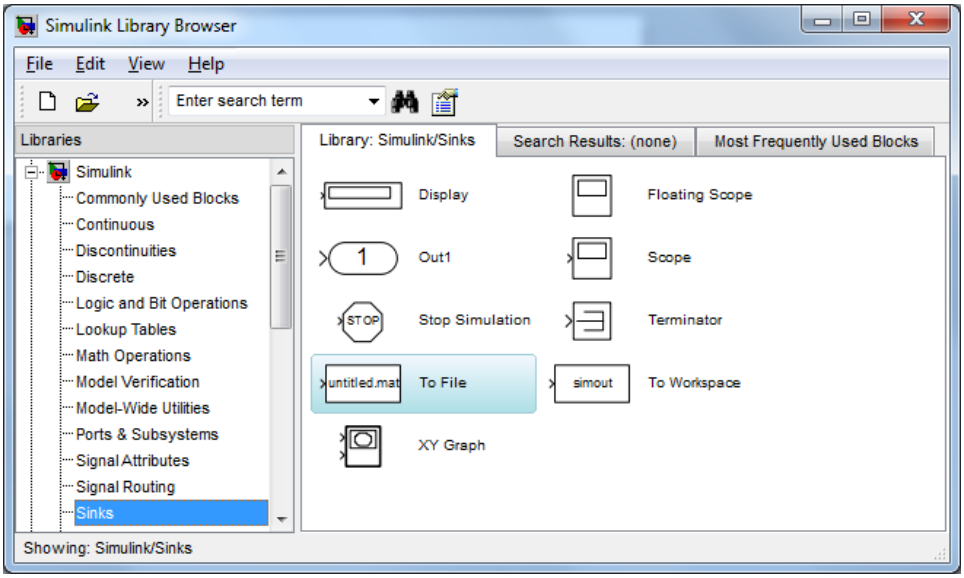

Figure 9: Location of the "To file" block in the Simulink library browser.

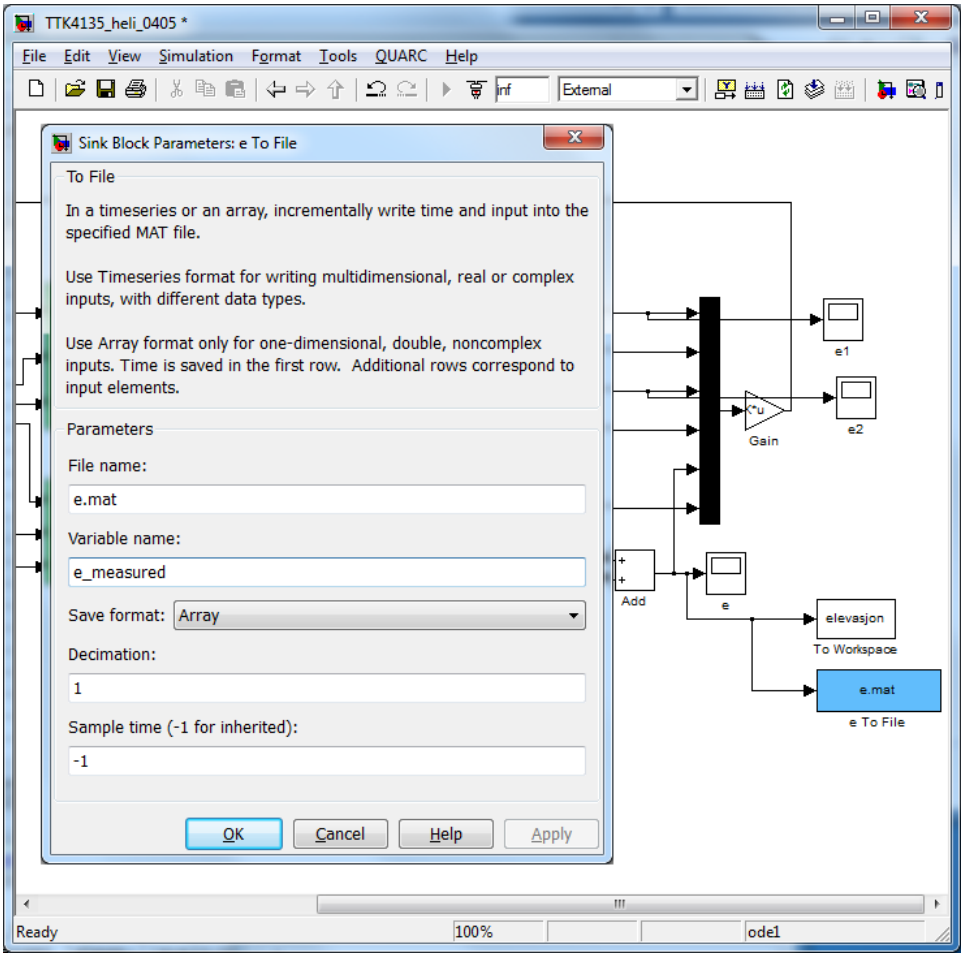

Figure 10: The block (in light blue) connected to the elevation measurement, and the parameters dialogue. Note that format has to be "Array".

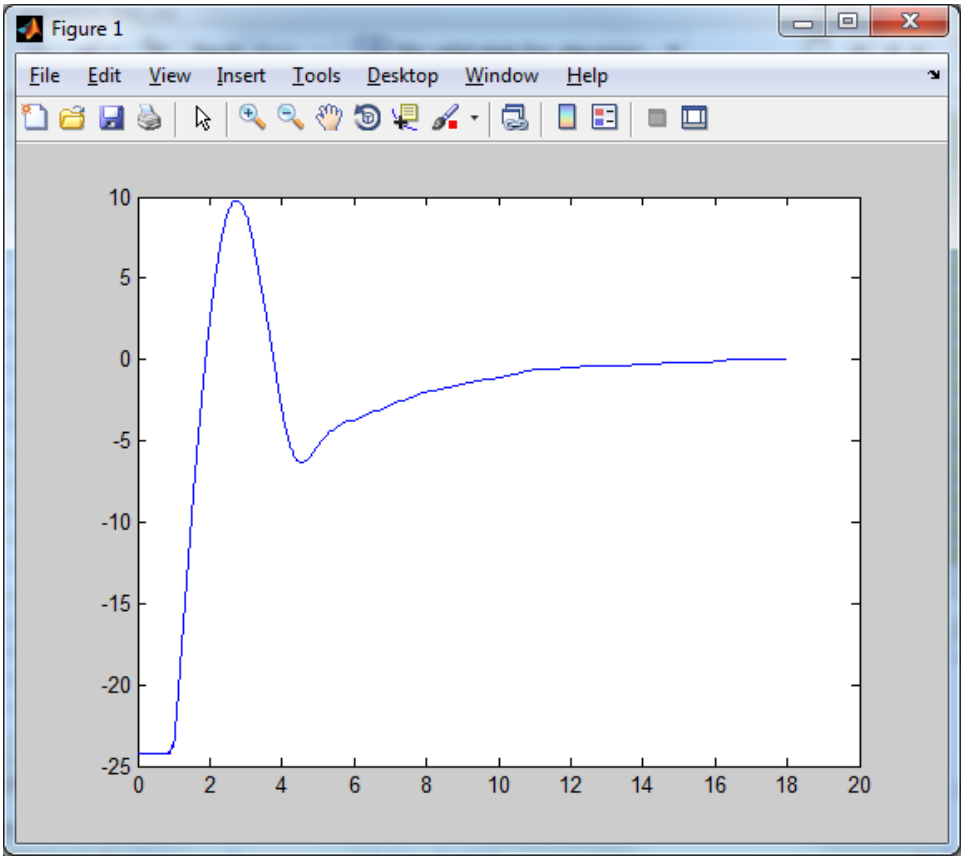

Figure 11: Resulting plot.