Automatic Control Laboratory, ETH Zürich
Prof. J. Lygeros

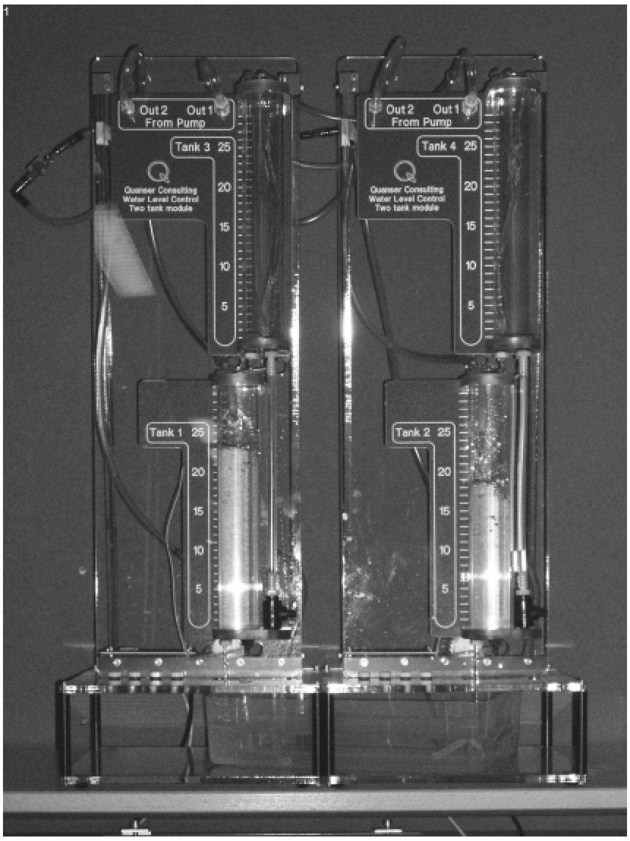
Manual prepared by: X Guidetti and A. Karapetyan
(based on manual by S. Richter, J. Felder and C. Hersberger)
Revision from: April 14, 2023

IfA Fachpraktikum - Experiment 3.4 :

Quad Tanks

In this Fachpraktikum you are controlling a setup of four coupled water tanks. This multiple input multiple output (MIMO) system can be made minimum and non-minimum phase. We want to show you the following aspects of control theory:

- Effects of coupling in a system
- Meaning of the relative gain array (RGA)
- Concept of LQR control
- Control of minimum and non-minimum phase systems



Contents

Chapter 1

Problem Setup and Notation

1.1 Setup

The quadruple tank experiment consists of four water tanks and two pumps (see Fig. 1.1 for a system schematic). The aim is to control the water levels in the lower tanks (tanks 1 and 2) with the two pumps. The inputs of the process to control are the input voltages of the pumps v_1 and v_2. The outputs are the corresponding water levels in the tanks h_1 and h_2. The pump and valve symbols used in Fig. 1.1 are explained in more detail in Fig. 1.2.

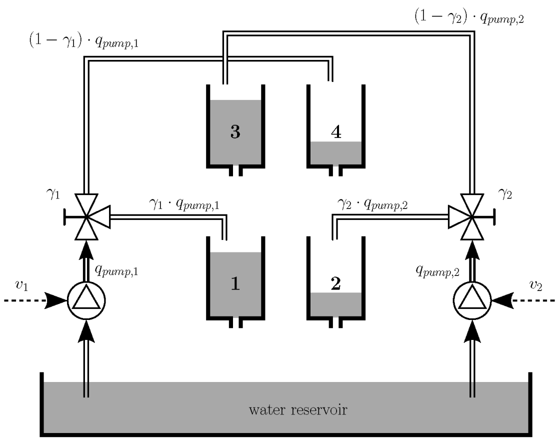
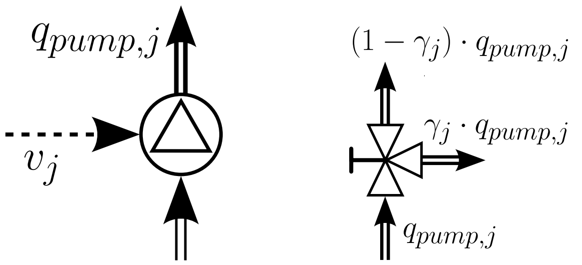


Figure 1.1: System schematic of the quadruple tank.



(a) Symbol of a pump with input voltage v_j generating flow $q_{pump,j}$

(b) Symbol of valve splitting up pump generated flow ($\gamma_j \in (0, 1)$)

Figure 1.2: Symbols used in Fig. 1.1.

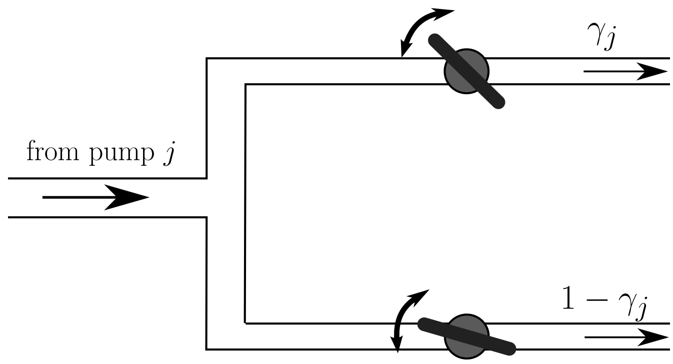


Figure 1.3: Sketch of a valve section. The flow from pump j is split up in a part proportional to γ_j and a part proportional to $1-\gamma_j$.

The flows of the pumps are split up by valves. The flow of pump 1 goes into tanks 1 and 4 whereas pump 2 feeds tanks 2 and 3.

There are two valve sections in the setup each consisting of two manually operated valves (see Fig. 1.3). Changing the flow ratios of the valve sections makes the system minimum or non-minimum phase.

1.2 Variables and Constants

The following table gives you a reference of the most important variables used in this experiment:

Variable in Text	Variable in Matlab	Description	Units / Value
h_i	`hi`	water levels in the tanks	cm
$\overline{h_i}$	`hi_ss`	steady state water levels	cm
$\overline{h_1}, \overline{h_2}$	`h1_ss`, `h2_ss`	steady state water levels of tanks 1,2	15 cm
x_i	`xi`	deviations of water levels, $x_i := h_i - \overline{h_i}$	cm
q_i	-	flows of the pumps to tanks	cm^3/s
v_j	`vj`	voltages to the pumps	V
$\overline{v_j}$	`vj_ss`	steady state pump voltages	V
u_j	`uj`	deviation of pump voltages, $u_j := v_j - \overline{v_j}$	V
$q_{pump,j}$	-	total flow of a pump	cm^3/s
γ_j	`gammaj`	ratio of the flows (see Table A.1 on page 17)	0 to 1

Constant in Text	Constant in Matlab	Description	Units / Value
a	`a`	cross-section of the tanks	15.52 cm^2
o	`oi`	cross-section area of an outlet	0.178 cm^2
g	`g`	acceleration due to gravity	981 cm/s^2
k_p	`kp`	pump flow constants	3.3 cm^3/sV

Indices used: $i = 1, \ldots, 4$ and $j = 1, 2$.

Chapter 2

Preparation@Home

2.1 Theory

Task 1: Modeling, LQR and computation of flow ratios

Get familiar with the system and its mathematical model (Appendix A on pages 16ff). Read the introduction to the linear quadratic regulator (LQR) in Appendix B on pages 20ff and the section about the flow ratios (Appendix ?? on pages ??ff).

2.2 PI Design

The hands-on part of your homework is the design of two PI controllers for the minimum phase system. As derived in Section A.2 on pages 16ff, the system has the transfer function matrix:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{T_1\gamma_1 k_p}{a(1+sT_1)} & \frac{T_1(1-\gamma_2)k_p}{a(1+sT_3)(1+sT_1)} \\ \frac{T_2(1-\gamma_1)k_p}{a(1+sT_4)(1+sT_2)} & \frac{T_2\gamma_2 k_p}{a(1+sT_2)} \end{bmatrix} \cdot \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \tag{2.1}$$

We want to control this MIMO system with two PI controllers. Because the system is only slightly coupled (in the minimum phase setting), we ignore the off-diagonal elements of the transfer function matrix. Thus we ignore the transfer functions $U_1(s) \dashrightarrow Y_2(s)$ and $U_2(s) \dashrightarrow Y_1(s)$ in (2.1). We are left with two decoupled SISO systems which are schematically illustrated in Fig. 2.1 by the dotted lines in the block denoted by $G(s)$.

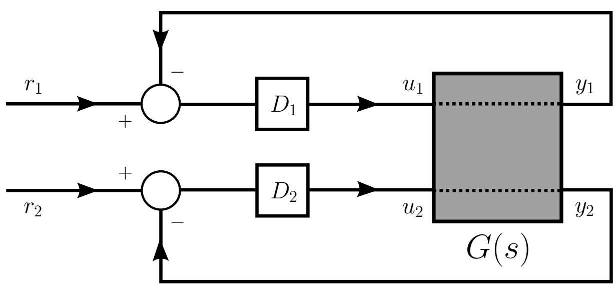


Figure 2.1: MIMO system $G(s)$ controlled by two PI controllers $D_j(s)$ (ignoring the coupling).

The transfer functions of these SISO systems follow from the diagonal elements of (2.1):

$$g_j(s) = \frac{T_j\gamma_j k_p}{a(1+sT_j)}, \quad \text{where} \quad Y_j(s) = g_j(s) \cdot U_j(s). \tag{2.2}$$

Note that the PI controllers $D_j(s)$ in Fig. 2.1 have the form:

$$D_j(s) = K_j\left(1 + \frac{1}{\tau_j s}\right) \tag{2.3}$$

Task 2: Compute the Closed Loop Transfer Functions

Compute the closed loop transfer functions $H_j(s) = \frac{Y_j(s)}{R_j(s)}$, $j = 1, 2$. Bring them to the standard form of a second order system:

$$H_j(s) = \frac{\cdots}{s^2 + 2\zeta_j\omega_{n,j}s + \omega_{n,j}^2} \tag{2.4}$$

Such a system has a step response as shown in Fig. 2.2.

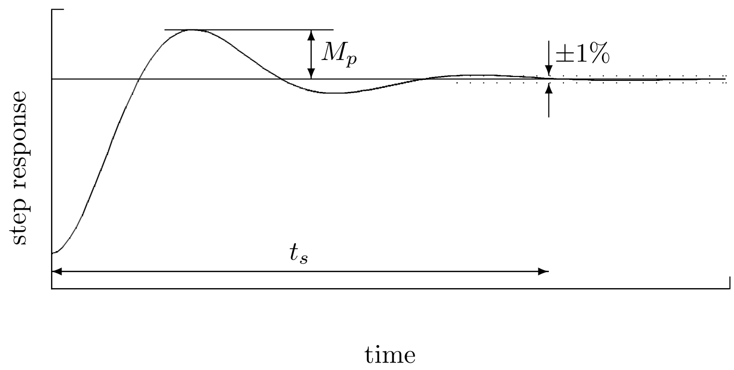


Figure 2.2: Step response of a second order system with overshoot M_p and settling time t_s.

Task 3: Compute the PI Parameters

Now compute K_j and τ_j such that the specifications

- settling time $t_s < 40s$
- overshoot $M_p < 9\%$

are fulfilled. As K_j and τ_j will depend on system parameters γ_1 and γ_2 you will get symbolic expressions at home only. In the lab, you will need these expressions to compute numeric values for the PI parameters (see Task 6).

The following relations should help you to obtain the PI parameters. They relate the *damping ratio* ζ_j and the *undamped natural frequency* $\omega_{n,j}$ to settling time and overshoot:

$$t_s \approx \frac{4.6}{\zeta\omega_n} \quad \text{and} \quad M_p = \exp\left(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}\right) \tag{2.5}$$

Note that these relationships hold for a second order system of the form

$$P(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.6}$$

only whereas you will find in Task 2 that the actual transfer functions $H_1(s)$ and $H_2(s)$ have an additional (real) zero. However, for a first design controller design these relationships can be used.

Chapter 3

Lab Session Tasks

3.1 Arduino Interface

You will use an Arduino Uno microcontroller as an interface between the computer and the lab equipment.

> **Important:** If at any time the experiment looks out of control and water is about to overflow and touch the ultra-sonic sensor mounted on top of the tanks, the best way to stop it is to **push the reset button on the Arduino board.** It is the only button you will find, hidden between the main board and the shield. Try it now!

3.2 Minimum Phase System

First you will control the minimum phase system with two PI controllers and compare the step responses to those of an LQR controller.

Task 4: Get the system ready

- Make sure the water reservoirs are filled with water (approx. 3/4 of max. height) and that the hoses to the pumps are well placed in the reservoirs.
- Also check if the big hose connecting the two reservoirs is completely filled with water and that there is no air in it! Only if there is no air in it, a balancing effect between the two reservoirs can be reached. If this is not the case, then ask your colleague to close one end of the hose with her/his fingers while you are pouring water into the other end. Then close the other end too and put both ends back into water without letting any air into the hose (can be a bit tricky!). It might help to shift the reservoirs a bit to the front so that it is easier to put the hose back into water.
- Switch on the two power supplies above the monitor. They will provide power to the water pumps.
- On the desktop, extract the lab files from the respective zip file.
- Make sure the Arduino board is properly connected to the computer through a USB port and then flash the required firmware on it. To do so, open the file `Arduino_QuadTanks.ino` with Arduino IDE and then load it on the board by

clicking on the arrow shaped button. You should also be able to see four other tabs, each corresponding to a different control scheme. If the upload button does not do anything and it says "No board selected" in the bottom right, proceed as follows: Press the "Select Board" Tab on the top. Then choose "COM4" or "COM5". In the next popup, search for the "Arduino Mega or Mega 2560". Select it and press "OK".

- Within the Arduino IDE, press *CTRL+Shift+M* to open a console with the sensor readings. Make sure that that the sensor readings (the second, fourth, sixth and eighth value) of all tanks are within ±1cm. If one sensor is reading high values, try to tap it or move it gently until it reads good values. If the console just prints weird symbols, press the button on the right of the console that contains a number and the word "baud". It will open a list, from which you pick "115200 baud". The console should now print numbers.
- Set the valves to minimum-phase. The valves feeding the bottom reservoirs should be completely open (parallel to the flow direction) while the valves feeding the top reservoirs should be approximately 3/4 open. This will approximately result in a $\gamma_1 \approx \gamma_2 \approx 0.7$, which you can assume throughout the next tasks.

Task 5: RGA and zeros of the transfer function matrix

Run the Matlab script `physical_model.m` to compute the mathematical model of the system together with the relative gain array (RGA) and the zeros of the transfer function matrix according to the equations in Appendix A on pages 16ff. Save the RGA as a new variable: `RGA_min=RGA`. You will need it later.

In the arduino IDE, in the fill-in part of the `Arduino_QuadTanks` tab, change the values of the steady-state references (`h1_ss, h2_ss`) and voltages (`v1_ss, v2_ss`) so that they correspond to the ones that MATLAB has just calculated and displayed in the command window. Load this updated code on the Arduino.

The RGA has entries ≈ 1.1 on the diagonal, whereas the elements on the off-diagonal are small. This means: input 1 is coupled with output 1 and input 2 with output 2. The coupling between input 1 and output 2 and vice versa is only small. From this we conclude that it makes sense to control tank 1 with pump 1 and tank 2 with pump 2. Which signs do you expect (and hopefully see) for the zeros?

Task 6: Decentralized PI control

Now you will regulate the system with two decentralized PI controllers. The term 'decentralized' herein means that PI controller D_1 does not coordinate its control actions with PI controller D_2, i.e. the (existing) coupling in the plant is neglected when it comes to controlling. You need the values for the PI controller parameters K_j, τ_j, you calculated as a homework now. Go to the tab "pi_controller.ino" in the Arduino IDE.

- Use your PI parameters computed for the PI controller parameters (from Task 3) to define the parameters `K1`, `tau1` and `K2`, `tau2` respectively. Then flash the code to the Arduino board. The microcontroller now contains everything it needs to control the quad tanks. The computer connection will only be used to visualize the system's behavior

- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'p'` and execute the script. If needed, you can interrupt the experiment by pushing the Arduino reset button and - after that - hitting ctrl+C on matlab. If MATLAB prints an error message that the COM port cannot be opened, close the Serial Monitor in the Arduino IDE (where you verified that the sensor readings are okay) and try again. The red lines with "KF-OFF" are the measurements and "KF-ON" a smoothed variant of these measurements.
- Once the experiment is completed, save the figure it produced

Task 7: How is the control performance?

Compare the plot of the step responses with the specifications on overshoot and settling time given in Task 3. As you can see, the system does not react as desired. What is the reason in your opinion? Try to think of an explanation and discuss it with the supervisor.

Task 8: Compare with PI Controller *and* dynamic decoupling

Now we want to compare the control performance of the coupled system from before with the one of a decoupled system. Decoupling is possible for some plants by means of a dynamic element (the *decoupler*) located between the PI controllers and the actual plant. We have already prepared the code calculating the effect of a decoupler in the script `pi_controller_decoupled.m`.

- Open the script `pi_controller_decoupled.m` and execute it, this will calculate additional controller gains
- Note the controller gains that you have just calculated, then move to the Arduino IDE. Insert new the gains in the fill-in section of the `PI_controller_dec` tab, then flash the code to the Arduino board.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'d'` and execute the script. If needed, you can interrupt the experiment by pushing the Arduino reset button and - after that - hitting ctrl+C on matlab
- Once the experiment is completed, save the figure it produced
- Compare the two experiments you just conducted: with and without the dynamic decoupler

Task 9: Manually tune PI controller parameters

Decoupling the system as done in the previous task is not always possible (luckily, here it is). It might happen that the transfer function of the dynamic decoupler cannot be realized. You will now return to the coupled system but manually tune the PI controller parameters to get better control performance. The tuning of the controller is done by multiplying or dividing the previous controller parameters K_j, τ_j.

In the `pi_controller` tab of the Arduino firmware this has already been prepared: `K_p1`, `K_p2`, `tau_1` and `tau_2` are the values that are actually used in the PI controller implementation in arduino. By replacing the factors "`1*`" in `K_p1=1*K1` etc. by other factors you can define multiples of your previously calculated PI controller

PROPORTIONAL PART

Symptom	Solution
Slow response	Increase K_j
High Overshoot or Oscillations	Decrease K_j

INTEGRAL TIME

Symptom	Solution
Slow response	Decrease τ_j
Instability or Oscillations	Increase τ_j

Table 3.1: PI tuning guide: What you have to do when your system does not react as desired

parameters. Table 3.1 illustrates how you can improve your step response in a systematic way.

A first improvement can be achieved by doubling the integral times and leaving the proportional constants:

- In the Arduino IDE open the tab `pi_controller`.
- Scroll down to the "tuning" section and change the values used for the PI controllers: `K_p1=1*K1` and `tau_1=2*tau1`. Do the same for `K_p2` and `tau_2`.
- Flash the modified code on the arduino board.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'p'` (to disable the decoupler again) and execute the script.

If you want to try to improve the step response further, you can try to find better values for the controllers. If you need further help with tuning the PI controller, look on page 23. There you can see how the proportional and the integral part of the PI controller affect the step response. In general, a trade-off between overshoot and settling time is necessary. When you are satisfied with your results, save figure containing the step responses of your optimized controller for comparison.

Task 10: Design an LQR controller

Now you will compare your optimized PI step responses with those of an LQR controller.

The LQR controller is a linear, static state-feedback controller, i.e. $u = -Kx$. So, for its implementation we first have to compute x, the deviations of the states from their steady state levels. Then u is computed using the linear control law. After that, the steady state values of the pump voltages are added to the "deviation voltages" u, i.e. $v = u + \bar{v}$ is then the vector of actually applied pump voltages. These operations are carried out in Arduino IDE tab `LQR_controller`.

Now you will compute the feedback matrix K and run the LQR:

- Run the script `lqr_controller.m` to compute the controller K.
- Note the controller gains that you have just calculated, then move to the Arduino IDE. Insert new the gains in the fill-in section of the `LQR_controller` tab, then flash the code to the Arduino board.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'l'` and execute the script.
- Once the experiment is completed, save the figure it produced

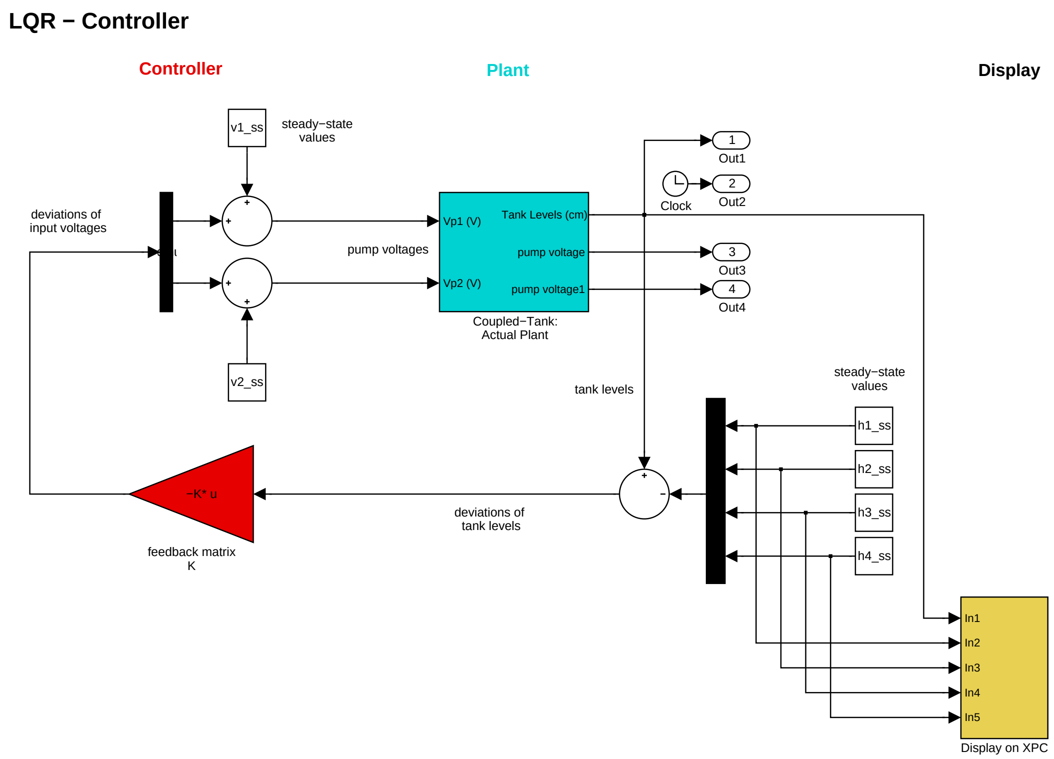


Figure 3.1: Simulink block diagram of the LQR controlled quad tank.

- Compare the experiment you just conducted with one of the optimized PI controllers

How is the control performance now? Why is there a steady state error? Can you imagine a remedy for that?

Task 11: Design an LQR integral controller

In order to get rid of the steady state error of the LQR controller from before, it is possible to combine it with an integral controller. We don't give any further details here regarding how the new controller gains are being computed but just ask you to try it out:

- Run the script `lqr_int_controller.m` to compute the LQR with integral control gains.
- Note the controller gains that you have just calculated, then move to the Arduino IDE. Insert new the gains in the fill-in section of the `LQR_int_controller` tab, then flash the code to the Arduino board.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'q'` and execute the script.
- Once the experiment is completed, save the figure it produced

- Compare the results of the experiment you just conducted with one of the optimized PI controllers

3.3 Non-Minimum Phase System

Now we want to run the same experiment in the non-minimum phase setting to learn about the differences between a minimum and non-minimum phase system.

Task 12: Make the system non-minimum phase

To make the system non-minimum phase you have to change the valve settings:

- Set the valves to non-minimum phase: lower valves should be 3/4 open while upper valves should be fully open. This will change γ_1, γ_2 approximately to 0.3.
- As the system has changed, we have to re-compute the mathematical model of the system. Run the Matlab script `physical_model.m` to compute the mathematical model of the system together with the RGA and the zeros of the transfer function matrix. Which signs do you expect (and hopefully see) for the zeros now?

Task 13: Decentralized PI control for non-minimum phase system

Now you will use the PI controller you developed at home for the control of the *minimum phase* system for the *non-minimum phase* system. As the system has changed (γ_1 and γ_2 are different now), the controller parameters have to be re-computed accordingly.

- Familiarize yourself again with the reset button of the Arduino.
- Evaluate your symbolic expressions for the PI controller parameters (see Task 3) and define the parameters `K1`, `tau1` and `K2`, `tau2` respectively in the Arduino IDE `PI_controller` tab. Make sure to reset the PI tuning section to its original state, i.e. `K_p1=1*K1`, ..., and flash the new code.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'p'` and execute the script. Stop the system with the Arduino reset button (followed by ctrl+C) if the behavior of the system becomes strange, i.e. water levels rise too high.

Task 14: What went wrong?

The system will not react as desired. What is the problem?

Comparing the relative gain array you computed for the non-minimum phase system (`RGA`) with the relative gain array for the minimum phase system (`RGA_min`) might help you to answer this question. When you have arrived at a conclusion, look at the quad tanks system: What should be changed there to get better control performance? Discuss your answer with the supervisor before you proceed.

Task 15: PI tuning for non-minimum phase system

After having made the appropriate changes to the hardware, we need to re-compute the PI controller parameters since the corresponding plant transfer functions have changed: They are now the off-diagonal elements of the transfer function matrix (2.1). Since these SISO transfer functions do not have the same structure as the transfer functions used for the computation of the controller parameters before, the explicit expressions for the PI controller parameters you calculated as a homework have become useless!

By trial-and-error we have found new values for the PI controller parameters:
$K_1 = 1$, $K_2 = 1.5$, $\tau_1 = \tau_2 = 18$.

- Define the given controller parameter values in the Arduino IDE `PI_controller` tab. Make sure to reset the PI tuning section to its original state, i.e. `K_p1=1*K1`, ..., and flash the new code.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'p'` and execute the script.
- Tune the controller parameters with the guide (Table 3.1) or with the help of Appendix C as done in Task 9. As before, change the parameters in Arduino IDE, flash the new controller and then use matlab to make it run.
- When you have finished tuning, save the obtained response
- Compare the step responses of the minimum phase system and the non-minimum phase system

Task 16: Design an LQR controller for the non-minimum phase system

Important: Make sure you undo the hardware changes you made during the previous section and bring the system back to its original state.
Now you will regulate the system with the LQR and compare the step response to the step response of the PI regulated system but also to the LQR response of the minimum phase system.

- Run the script `lqr_controller.m` to compute the controller K.
- Note the controller gains that you have just calculated, then move to the Arduino IDE. Insert the new gains in the fill-in section of the `LQR_controller` tab, then flash the code to the Arduino board.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'l'` and execute the script.
- Once the experiment is completed, save the figure it produced
- Compare the step responses of PI and LQR control solutions
- You will observe a steady state error in the LQR responses. Let's eliminate it again by means of additional integral control: Run the script `lqr_int_controller.m` to compute the LQR with integral control gains.
- Note the controller gains that you have just calculated, then move to the Arduino IDE. Insert new the gains in the fill-in section of the `LQR_int_controller` tab, then flash the code to the Arduino board.
- Come back to matlab and open the script `run_controllers.m`. Set the variable `ctrl_char` to `'q'` and execute the script.
- Once the experiment is completed, save the figure it produced

- Compare the step responses of PI and LQR with integral control solutions
- Finally, compare the LQR with integral control responses of the minimum and the non-minimum-phase system. You should see that the minimum phase setting shows much better control performance than the one of the non-minimum phase setting.

Task 17: Finalizing the Lab-Experiment:

- Please fill out the online feedback-form on the registration-page under `MyExperiments`. Each student/participant should fill out his own feedback-form. This will help us to steadily improve the experiments. Thank you for your inputs.
- Discuss the experiment with your lab-tutor to get the *testat*.

Appendix A

Mathematical Model – System Equations

In this section the mathematical model of the quad tank is derived. We set up the basic equations that hold for each of the tanks and for the two pumps. Then they are put together to obtain the model of the whole system. A word about notation: the time derivative of a value z is denoted with a dot, $\frac{dz}{dt} = \dot{z}$, and its steady state value with $\overline{z}$.

A.1 Basic Equations

Mass Balance Mass balance gives for each of the four tanks:

$$\dot{V} = a \cdot \dot{h} = q_{in} - q_{out} \tag{A.1}$$

with V: volume of water in the tank
a: cross-section area of the tank
h: water level
q_{in}: inflow
q_{out}: outflow

Bernoulli's Law By evaluating Bernoulli's law for incompressible liquids

$$p + \frac{1}{2}\rho v_w^2 + \rho g h = const. \tag{A.2}$$

at the water surface ($v_w = 0$) and at the bottom of each tank ($h = 0$) and subtracting the resulting equations from each other, we obtain for the outflow:

$$q_{out} = o \cdot v_w = o\sqrt{2g} \cdot \sqrt{h} \tag{A.3}$$

with o: cross-section area of an outlet
v_w: speed of water (at the outflow)
h: water level
g: acceleration due to gravity

Pump Generated Flows The pumps generate a flow proportional to the applied voltage: $q_{pump,j} = k_p \cdot v_j$. The flow is split up by the valves according to Table A.1.

A.2 Quad Tank System

Now we want to derive the state space representation and the transfer function matrix of the system.

	to tank 1	to tank 2	to tank 3	to tank 4
from pump 1	$\gamma_1 k_p \cdot v_1$	-	-	$(1-\gamma_1)k_p \cdot v_1$
from pump 2	-	$\gamma_2 k_p \cdot v_2$	$(1-\gamma_2)k_p \cdot v_2$	-

Table A.1: Flows to the tanks generated by the two pumps

Nonlinear System Equations (A.1) and (A.3) and Table A.1 lead to the following nonlinear differential equations for the four tanks:

$$
\begin{aligned}
\dot{h}_1 &= \frac{1}{a}\Big(\overbrace{o\sqrt{2g}\cdot\sqrt{h_3} + \gamma_1 k_p \cdot v_1}^{q_{in}} - \overbrace{o\sqrt{2g}\cdot\sqrt{h_1}}^{q_{out}}\Big)\\
\dot{h}_2 &= \frac{1}{a}\Big(o\sqrt{2g}\cdot\sqrt{h_4} + \gamma_2 k_p \cdot v_2 - o\sqrt{2g}\cdot\sqrt{h_2}\Big)\\
\dot{h}_3 &= \frac{1}{a}\Big((1-\gamma_2)k_p \cdot v_2 - o\sqrt{2g}\cdot\sqrt{h_3}\Big)\\
\dot{h}_4 &= \frac{1}{a}\Big(\underbrace{(1-\gamma_1)k_p \cdot v_1}_{q_{in}} - \underbrace{o\sqrt{2g}\cdot\sqrt{h_4}}_{q_{out}}\Big)
\end{aligned}
\tag{A.4}
$$

Steady State In equilibrium all time-varying variables have settled to some constant value. It holds that $\dot{h}_i = 0$ for each tank, which leads to four equations for the six steady state values $\overline{h_1}$, $\overline{h_2}$, $\overline{h_3}$, $\overline{h_4}$, $\overline{v_1}$, and $\overline{v_2}$. This allows us to choose two of the values. As we want to control the levels of tanks 1 and 2, we choose $\overline{h_1}$ and $\overline{h_2}$ and resolve the system of equations:

$$
\begin{aligned}
0 &= \frac{1}{a}\Big(o\sqrt{2g}\cdot\sqrt{\overline{h_3}} + \gamma_1 k_p \cdot \overline{v_1} - o\sqrt{2g}\cdot\sqrt{\overline{h_1}}\Big)\\
0 &= \frac{1}{a}\Big(o\sqrt{2g}\cdot\sqrt{\overline{h_4}} + \gamma_2 k_p \cdot \overline{v_2} - o\sqrt{2g}\cdot\sqrt{\overline{h_2}}\Big)\\
0 &= \frac{1}{a}\Big((1-\gamma_2)k_p \cdot \overline{v_2} - o\sqrt{2g}\cdot\sqrt{\overline{h_3}}\Big)\\
0 &= \frac{1}{a}\Big((1-\gamma_1)k_p \cdot \overline{v_1} - o\sqrt{2g}\cdot\sqrt{\overline{h_4}}\Big)
\end{aligned}
\tag{A.5}
$$

The last two equations of (A.5) tell us that $o\sqrt{2g}\cdot\sqrt{\overline{h_3}} = (1-\gamma_2)k_p \cdot \overline{v_2}$ and $o\sqrt{2g}\cdot\sqrt{\overline{h_4}} = (1-\gamma_1)k_p \cdot \overline{v_1}$. These expressions together with the first two equations of (A.5) give a system of two linear equations:

$$
\begin{aligned}
\begin{bmatrix} o\sqrt{2g}\cdot\sqrt{\overline{h_1}} \\ o\sqrt{2g}\cdot\sqrt{\overline{h_2}} \end{bmatrix} &= \begin{bmatrix} \gamma_1 k_p & (1-\gamma_2)k_p \\ (1-\gamma_1)k_p & \gamma_2 k_p \end{bmatrix} \cdot \begin{bmatrix} \overline{v_1} \\ \overline{v_2} \end{bmatrix} \iff \\
\begin{bmatrix} \overline{v_1} \\ \overline{v_2} \end{bmatrix} &= \begin{bmatrix} \gamma_1 k_p & (1-\gamma_2)k_p \\ (1-\gamma_1)k_p & \gamma_2 k_p \end{bmatrix}^{-1} \cdot \begin{bmatrix} o\sqrt{2g}\cdot\sqrt{\overline{h_1}} \\ o\sqrt{2g}\cdot\sqrt{\overline{h_2}} \end{bmatrix}
\end{aligned}
\tag{A.6}
$$

It follows that the values for the remaining four steady state variables are[1]:

[1]Note that for $\gamma_1 + \gamma_2 = 1$ the steady state voltages $\overline{v_1}$ and $\overline{v_2}$ cannot be computed with the given expression, because the matrix in equation (A.6) is not invertible, its determinant is equal to 0. This means, that in the case of $\gamma_1 + \gamma_2 = 1$ we cannot choose $\overline{h_1}$ and $\overline{h_2}$ independently of each other.

$$
\begin{aligned}
\begin{bmatrix}\overline{v_1}\\ \overline{v_2}\end{bmatrix} &= \frac{\sqrt{2g}}{k_p(\gamma_1+\gamma_2-1)}\cdot\begin{bmatrix}\gamma_2 & \gamma_2-1\\ \gamma_1-1 & \gamma_1\end{bmatrix}\cdot\begin{bmatrix}o\cdot\sqrt{\overline{h_1}}\\ o\cdot\sqrt{\overline{h_2}}\end{bmatrix}\\
\overline{h_3} &= \left(\frac{(1-\gamma_2)k_p\cdot\overline{v_2}}{o\sqrt{2g}}\right)^2 \quad\text{and}\quad \overline{h_4} = \left(\frac{(1-\gamma_1)k_p\cdot\overline{v_1}}{o\sqrt{2g}}\right)^2
\end{aligned} \tag{A.7}
$$

Linearization Now we want to get the state space representation of the system: $\dot{x} = Ax + Bu,\ y = Cx + Du$.
This includes some matrix algebra. So let us introduce vectors $h = \begin{bmatrix}h_1 & h_2 & h_3 & h_4\end{bmatrix}^{\mathrm{T}}$ and $v = \begin{bmatrix}v_1 & v_2\end{bmatrix}^{\mathrm{T}}$ and analogously $\overline{h}, \overline{v}$.
Now we can write the system as $\dot{h} = f(h, v)$, where f is a general function of the water levels h and the pump voltages v. We see from equations (A.4), that the system contains square roots of state variables - and therefore the function $f(h, v)$ is nonlinear.
If we want the state space representation with matrices A, B, C and D we have to linearize the system around the steady state $(\overline{h}, \overline{v})$. In steady state it holds that $\dot{h} = f(\overline{h}, \overline{v}) \equiv 0$. Thus we get the system matrices with the aid of Taylor series expansion

$$
\begin{aligned}
\dot{x} = \dot{h} \approx \underbrace{f(\overline{h},\overline{v})}_{\equiv 0} + \underbrace{\left.\frac{\partial f(h,v)}{\partial h}^{\mathrm{T}}\right|_{h=\overline{h}}}_{A} \cdot \underbrace{(h-\overline{h})}_{x} + \underbrace{\left.\frac{\partial f(h,v)}{\partial v}^{\mathrm{T}}\right|_{v=\overline{v}}}_{B} \cdot \underbrace{(v-\overline{v})}_{u}\\
\text{with} \quad x = \begin{bmatrix}x_1 & x_2 & x_3 & x_4\end{bmatrix}^{\mathrm{T}} \quad \text{and} \quad u = \begin{bmatrix}u_1 & u_2\end{bmatrix}^{\mathrm{T}},
\end{aligned} \tag{A.8}
$$

where we have introduced new vectors x and u. x contains the deviations of the water levels from their steady state values ($x_i := h_i - \overline{h_i}$). u contains the deviations of the pump voltages from their steady state values ($u_i := v_i - \overline{v_i}$).

State Space Representation Linearizing the nonlinear system has already given us A and B. C can be found easily: The outputs of the system are simply the state variables $y_1 = x_1$ and $y_2 = x_2$. So, we get the state space representation of the system as:

$$
\begin{aligned}
\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} &= \begin{bmatrix}-\frac{1}{T_1} & 0 & \frac{1}{T_3} & 0\\ 0 & -\frac{1}{T_2} & 0 & \frac{1}{T_4}\\ 0 & 0 & -\frac{1}{T_3} & 0\\ 0 & 0 & 0 & -\frac{1}{T_4}\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} + \begin{bmatrix}\frac{\gamma_1 k_p}{a} & 0\\ 0 & \frac{\gamma_2 k_p}{a}\\ 0 & \frac{(1-\gamma_2)k_p}{a}\\ \frac{(1-\gamma_1)k_p}{a} & 0\end{bmatrix}\cdot\begin{bmatrix}u_1\\ u_2\end{bmatrix}\\
\begin{bmatrix}y_1\\ y_2\end{bmatrix} &= \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}\\
&\text{with the time constants } T_i \text{ such that } \quad \frac{1}{T_i} = \frac{o\sqrt{2g}}{a}\cdot\frac{1}{2\sqrt{\overline{h_i}}}
\end{aligned} \tag{A.9}
$$

Transfer Function Matrix The Laplace transform of (A.9) yields the transfer matrix of the four tank system:

$$
\begin{aligned}
G(s) &= C(sI-A)^{-1}B, \quad \begin{bmatrix}Y_1(s)\\ Y_2(s)\end{bmatrix} = G(s)\cdot\begin{bmatrix}U_1(s)\\ U_2(s)\end{bmatrix}\\
G(s) &= \begin{bmatrix}\frac{T_1\gamma_1 k_p}{a(1+sT_1)} & \frac{T_1(1-\gamma_2)k_p}{a(1+sT_3)(1+sT_1)}\\ \frac{T_2(1-\gamma_1)k_p}{a(1+sT_4)(1+sT_2)} & \frac{T_2\gamma_2 k_p}{a(1+sT_2)}\end{bmatrix} =: \begin{bmatrix}g_{11}(s) & g_{12}(s)\\ g_{21}(s) & g_{22}(s)\end{bmatrix}
\end{aligned} \tag{A.10}
$$

A.3 Further Analysis of the System

Relative Gain Array (RGA) The relative gain array reflects how the inputs and outputs of the system are coupled[2]. For 2×2 systems at steady state it has the form:

$$M = \begin{bmatrix} m & 1-m \\ 1-m & m \end{bmatrix} \quad \text{with} \quad m = \frac{g_{11}(0) \cdot g_{22}(0)}{g_{11}(0) \cdot g_{22}(0) - g_{12}(0) \cdot g_{21}(0)} \tag{A.11}$$

Using (A.10) m can be found as:

$$m = \frac{\gamma_1 \cdot \gamma_2}{\gamma_1 + \gamma_2 - 1} \tag{A.12}$$

For big values of m, the dominating elements of the transfer function matrix are the diagonal elements. Output 1 is affected mostly by input 1, output 2 by input 2. If m is small, output 1 depends mainly on input 2, and output 2 on input 1. The knowledge of which input mainly affects which output is important for the design of (decentralized) PI controllers for the MIMO system.

Zeros of the System A zero of the transfer function matrix (A.10) is defined as any complex number z_i where the rank of $G(z_i)$ is less than the normal rank of $G(s)$. In our case the normal rank of $G(s)$ is 2 (full rank), so the latter condition is equivalent to $detG(z_i) = 0$. The determinant is given by

$$detG(s) = \frac{T_1 T_2 k_p^2 \gamma_1 \gamma_2}{a^2 \prod_{i=1}^{4}(1+sT_i)} \cdot \left[(1+sT_3)(1+sT_4) - \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1 \gamma_2} \right] \tag{A.13}$$

To find both zeros z_1, z_2, we set the term in the bracket in (A.13) to zero

$$(1+z_iT_3)(1+z_iT_4) - \underbrace{\frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1\gamma_2}}_{\eta} = 0 \ , \quad i \in \{1,2\} \ , \tag{A.14}$$

and solve this quadratic equation for z_1 and z_2. Depending on the flow ratios γ_1 and γ_2, the introduced parameter η takes values in $(0, \infty)$. Qualitatively, the zeros then behave as follows:

- If $\eta < 1$ (because γ_j are big), the two zeros are close to $-\frac{1}{T_3}$ and $-\frac{1}{T_4}$.
- If $\eta = 1$, and thus $\gamma_1 + \gamma_2 = 1$, the zeros are at 0 and $-(\frac{1}{T_3} + \frac{1}{T_4})$.
- If $\eta \to \infty$ one zero tends to $-\infty$ and the other to $+\infty$.

Thus we can say that one of the two zeros is always in the left half-plane, but the other one can be located either in the left or in the right half-plane. The system is minimum phase (both zeros are in the left half-plane) for $\eta < 1$ and thus $\gamma_1 + \gamma_2 > 1$, and the system is non-minimum phase for $\eta > 1$, i.e. $\gamma_1 + \gamma_2 < 1$.

Zeros of the System: Physical Interpretation We want to control the water levels in the two lower tanks. If both flow ratios γ_j are big, most of the water is going directly into the lower tanks. If γ_j are small — and thus η is big — the water is going first to the upper tanks and after that into the lower tanks. Note, that in this case, pump 1 indirectly fills tank 2 and pump 2 indirectly fills tank 1. It is intuitively clear that it is easier to control the lower water levels if the water is going directly into these tanks, instead of going there indirectly via the upper tanks. Briefly: The system is harder to control if it is non-minimum phase than if it is minimum phase.

[2]See for instance the book MULTIVARIABLE FEEDBACK CONTROL, Analysis and Design (2nd Edition), by S. Skogestad and I. Postlethwaite

Appendix B

Introduction to LQR Control

In this section, a short overview of the Linear Quadratic Regulator (LQR) is given. We want to provide the information needed to setup an LQR for the quad tank experiment. The concept is easy to understand, the math will be kept short since Matlab provides us all the functions we need to compute an LQR controller.

B.1 Setup of the Feedback System

The LQR is a controller for dynamic systems, that feeds the system's states back to its inputs via a static feedback matrix K (see Fig. B.1):

$$u = -K \cdot x \tag{B.1}$$

In the case of the quad tank, K is of dimension 2×4, because we want to feed back 4 states to 2 inputs. Setting up an LQR means finding the matrix K.

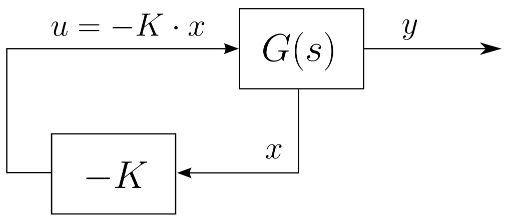


Figure B.1: Diagram of the LQR with feedback matrix K and plant $G(s)$.

B.2 Concept of the LQR

The LQR is a controller that tries to drive the system to its steady state point and to stabilize it there. For our quad tank this means: All the water levels are on their steady state values or equivalently $x = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. This steady state point can be chosen partially (see (A.7)).

B.3 Measuring the Performance Based on the States

As the LQR wants to bring the system to steady state, we can say: A good LQR controller is a controller that keeps the system as close as possible to its steady state. If we want to measure how good the controller performs, we ask: How far away from steady state is the system?
This question can be answered with the norm $||.||$ operator. We know: The bigger $||x||$ is, the

farther away we are from equilibrium.
In a system there are typically states that should have only small deviations from steady state whereas for other states this might not be crucial. Thus it is better to measure the performance of the LQR with a cost function like:

$$\Xi_x = q_1 \cdot (x_1)^2 + q_2 \cdot (x_2)^2 + q_3 \cdot (x_3)^2 + q_4 \cdot (x_4)^2 = \sum_{k=1}^{4} q_k \cdot (x_k)^2 \tag{B.2}$$

Note that Ξ_x is basically a weighted norm.

Example: Weighting Factors q_k: Let's look at our quad tank with x_i being the deviation of the water level from the steady state level in tank i. Our aim is to control the water levels in tanks 1 and 2: We want to keep them close to equilibrium. But we don't care about water level variations in tanks 3 and 4. How do we have to choose the factors q_k if we want to take this into account?
Don't proceed with reading the answer in the footnote until you have thought about it![1]

Note that we could — and will, as we work with Matlab — express Ξ_x also via a diagonal weighting matrix Q as:

$$\Xi_x = \sum_{k=1}^{4} q_k \cdot (x_k)^2 = x^{\mathrm{T}} \cdot Q \cdot x \quad \text{with} \quad Q = diag(q_1, q_2, q_3, q_4) \tag{B.3}$$

B.4 Further Criteria: Considering the Inputs

In the previous section we said: A controller that keeps Ξ_x small is a good controller. But imagine the (practical) situation where we have a system with disturbances. If the controller wants to stabilize the system in its steady state at any cost, it would need huge amounts of energy and probably act very rudely. Usually it is better to have a controller that doesn't want to force the system to its steady state *at any cost*.
Thus we expand our concept of a good controller. We say: A good LQR keeps the system close to its steady state and doesn't act rudely[2] and doesn't use much power.

We expand our cost function and penalize also the deviations of the inputs from their steady state values:

$$\Xi_u = r_1 \cdot (u_1)^2 + r_2 \cdot (u_2)^2 = u^{\mathrm{T}} \cdot R \cdot u \quad \text{with} \quad R = diag(r_1, r_2) \tag{B.4}$$

B.5 Finding the Feedback Matrix K with Matlab

Up to now we have set up criteria for a *good* controller. Now we want to find the *best* one. This means minimizing the sum of the cost functions (B.3) and (B.4) over an infinite horizon into the future

$$J = \int_0^\infty (\Xi_x + \Xi_u)\, dt = \int_0^\infty (x^{\mathrm{T}} \cdot Q \cdot x + u^{\mathrm{T}} \cdot R \cdot u)\, dt \;, \tag{B.5}$$

subject to the system's dynamics $\dot{x} = Ax + Bu$. It turns out that the optimal solution to this problem is a linear, static state-feedback law $u = -Kx$, where matrix K can be found using the Matlab command `K=lqr(A,B,Q,R)`[3].

[1] Answer: q_1, q_2 'large' whereas q_3, q_4 'small' (or zero)
[2] This means: Only small deviations of the input's values from their steady state values are allowed.
[3] Details about the theory on LQR control are taught for instance in the course 'Model Predictive Control' offered by IfA.

B.6 How to choose weighting matrices Q and R

Weighting matrices Q and R are used to define the cost function as shown before. The topic of this section is to provide some general guidelines on how to choose Q and R in order to get a linear quadratic regulator that acts according to the specifications. Note that these are simplified rules only!

- We restrict ourselves to diagonal weighting matrices, i.e.
  $Q = diag(q_1, q_2, q_3, q_4)$ and $R = diag(r_1, r_2)$. Q has four entries because we have four states and R has two entries because we have two inputs.

- Q and R are weighted relatively to each other, so multiplying both with a scalar gives the same controller.

- With $q_k \geq 0$ we can set how deviations of state x_k from zero or steady state – depending on how the states are defined – are penalized. If q_k is big, we do not allow for big deviations of state x_k. If $q_k = 0$ we don't care about deviations of this state at all.[4]

- With $r_k > 0$ we can set how much energy our input u_k is allowed to use. If r_k is big, it means that the controller should not act too forcefully on input u_k. On the other hand, making r_k small means that the controller can use more energy on this input. $r_k = 0$ is not reasonable at all since this would allow the controller to use infinite energy.

- If we want to have little deviations of the states from zero (or steady state), we have to choose the values of R small compared to Q, thus making control cheap. But if control is made too cheap, the controller is going to react very brusquely, even on the smallest deviation of the states.

- If we want to make control expensive (e.g. because we have a battery that does not provide much power) we have to choose the values of R big compared to Q. But if control is made too expensive, the controller's reaction could get too sluggish and slow.

- In general one has to find a good ratio between the values q_k and r_k. For the quad tank experiment we have found out (by trial-and-error method) that a good ratio of q_k and r_k is 100, i.e.
  $Q = diag(1, 1, 0, 0)$ and $R = diag(0.01, 0.01)$.

B.7 Summary - The LQR

An LQR controller implements a static state-feedback control law $u = -K \cdot x$. The matrix K can be found by minimizing the cost function given in (B.5) subject to the system's dynamics. Since the cost function is a quadratic function in both the states' evolution and the inputs' evolution over time and the system dynamics are expressed in terms of a linear differential equation, the resulting controller is termed "Linear Quadratic Regulator" .

For the implementation of an LQR we need information about *all* the states of the system. In general, not all the states can be measured so they must be estimated. This complicates the implementation of an LQR. Fortunately in the quad tank experiment the states (which are the water levels of the individual tanks) can be easily measured.

To compute matrix K the following steps have to be performed:

- Compute the system matrices A and B.

- Choose Q and R following the rules in Section B.6.

- Compute the controller with Matlab: `K=lqr(A,B,Q,R)`.

[4] The choice $q_k = 0$ could cause problems when state x_k reflects an unstable mode, but for our quad tank this is not the case.

Appendix C

PI Controller Parameters and the Step Response

The table on page 24 shows the effects of the proportional constant K and the integral time τ on the step response. The plot located in the middle of this table corresponds to "untuned values". These untuned values were then doubled and halved to get the other surrounding step responses. The measurements were done with the minimum phase system.

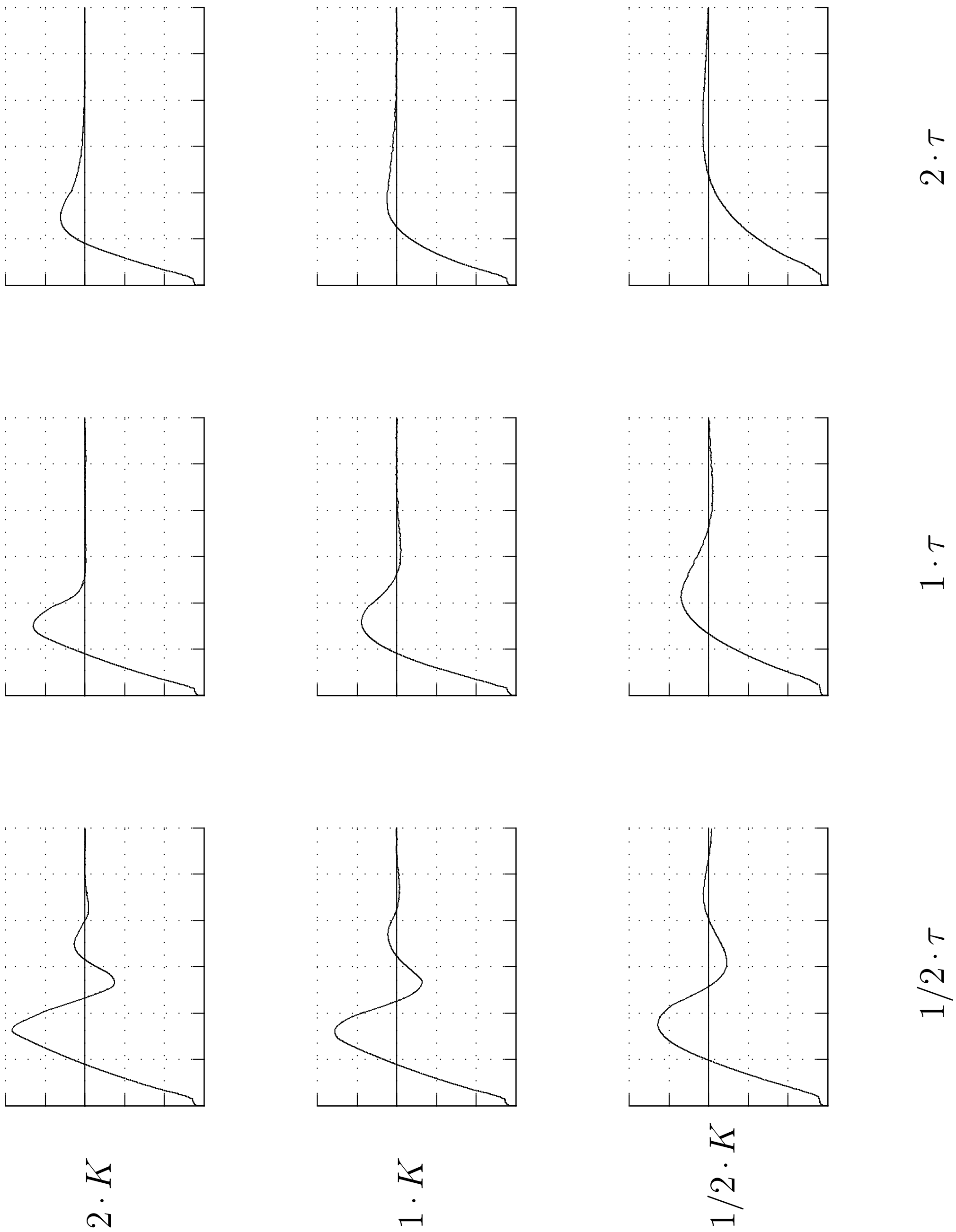


Table C.1: Abscissa: time $[0 \ldots 60s]$; ordinate: step response $[0 \ldots 25\text{cm}]$.