Automatic Control, Advanced Course

Laboratory Session 3

Model-Predictive Control of the Quadruple Tank[1]

Department of Automatic Control
Lund University

1. Introduction

The purpose of this lab session is to study some different aspects of constrained predictive control. The ideas are illustrated on a quadruple-tank process with two inputs and two outputs. The process can be modified to have different dynamics: In one configuration the process is minimum phase, and in another configuration the process is non-minimum phase. In the lab we will study model-predictive control and connect it to fundamental limitations introduced by a right-half-plane transmission zero.

The different assignments are labeled ***Preparatory***, ***Task*** and ***Reflect***. All ***Preparatory*** assignments should be done before the lab session, the ***Task*** assignments will involve actual experiments to do, while at the ***Reflect*** assignments you should think about what you just did. Write down your results and conclusion on a separate piece of paper during the lab session. When you are finished let the lab supervisor know and be ready to present your findings from the different tasks.

Intended learning outcomes

When you complete this laboratory exercise you will be able to:

- Use `MATLAB`'s Model Predictive Control Toolbox to solve constrained control problems.
- Make design choices from process knowledge.
- Use simulations and process experiments to tune controller parameters.
- Tackle measurement noise and process disturbances with disturbance models.

Pre-lab assignments

Read this manual carefully. On the lab you will start at Section 5 and you are expected to have a good understanding of the sections leading up to that. Solve all assignments marked ***Preparatory***. You will need to complete the `MATLAB` function `setup_lqg.m` before the lab. Bring this code to the lab session. In order to be allowed to perform the lab you must have completed all preparatory assignments. You must also do the **pre-lab quiz** on Canvas before the lab.

[1] Written by Olle Kjellqvist 2020; updated by Johan Lindberg September 23, 2022.

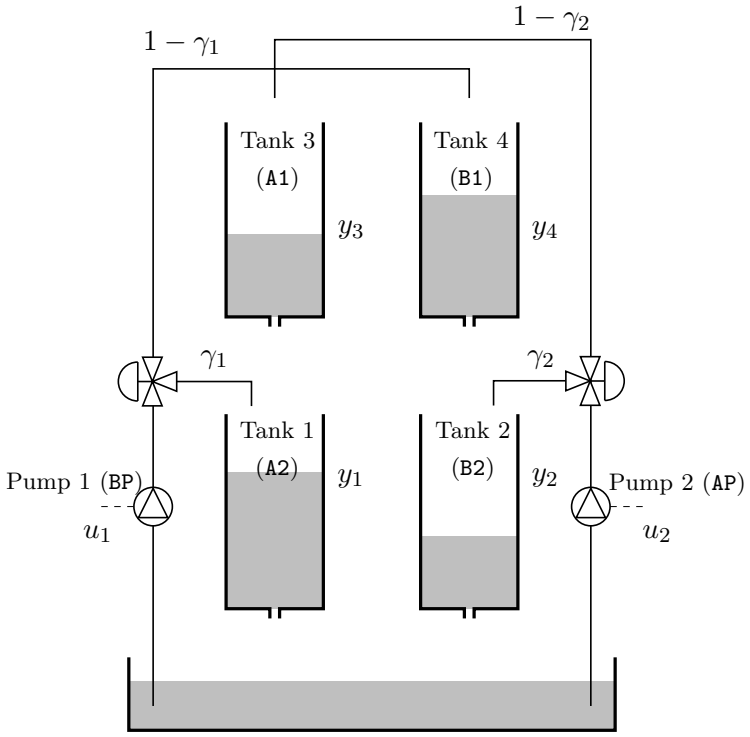

Figure 1 The quadruple-tank process.

2. The process

The quadruple-tank process is shown in Figure 1. The goal is to control the water level in the lower tanks using the two pumps.

The process inputs u_1, u_2 are the voltages (0–10 V) applied to the two pumps. The process outputs y_1, y_2 are voltages (0–10 V) representing the levels in the lower tanks. It is assumed that we cannot measure the levels in the upper tanks.

Nonlinear model

In this section, a nonlinear model of the quadruple-tank process is derived. For each tank $i = 1 \dots 4$, mass balance and Torricelli's law give that

$$A_i \frac{dh_i}{dt} = -a_i \sqrt{2gh_i} + q_{in_j}$$

where A_i [cm^2] is the cross-section of the tank, h_i [cm] is the water level, a_i [cm^2] is the cross-section of the outlet hole, g [cms^{-2}] is the acceleration of gravity, and q_{in_i} [cm^{3}s^{-1}] is the inflow to the tank.

Each pump $j = 1, 2$ gives a flow proportional to the control signal,

$$q_{pump_j} = k_j u_j$$

where k_j [cm^{3}V^{-1}s^{-1}] is a pump constant. The flows from the pumps are divided according to the two parameters $\gamma_1, \gamma_2 \in [0, 1]$. The flow to Tank 1 is $\gamma_1 k_1 u_1$ and the flow to Tank 4 is $(1 - \gamma_1) k_1 u_1$. Symmetrically, the flow to Tank 2 is $\gamma_2 k_2 u_2$ and the flow to Tank 3 is $(1 - \gamma_2) k_2 u_2$.

The measured level signals are $y_1 = k_c h_1$ and $y_2 = k_c h_2$, where k_c [Vcm^{-1}] is a measurement constant.

Considering the flow in and out of all tanks simultaneously, the dynamics of the quadruple-tank process are given by

$$\begin{aligned}\frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}u_1 \\ \frac{dh_2}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}u_2\end{aligned} \tag{1}$$

$$\begin{aligned}\frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}u_2 \\ \frac{dh_4}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}u_1\end{aligned} \tag{2}$$

Stationary points

For a stationary operating point (h_1^0, h_2^0, h_3^0, h_4^0, u_1^0, u_2^0), the equations above imply that

$$\begin{aligned}\frac{a_3}{A_3}\sqrt{2gh_3^0} &= \frac{(1-\gamma_2)k_2}{A_3}u_2^0 \\ \frac{a_4}{A_4}\sqrt{2gh_4^0} &= \frac{(1-\gamma_1)k_1}{A_4}u_1^0\end{aligned} \tag{3}$$

and thus

$$\begin{aligned}\frac{a_1}{A_1}\sqrt{2gh_1^0} &= \frac{(1-\gamma_2)k_2}{A_3}u_2^0 + \frac{\gamma_1 k_1}{A_1}u_1^0 \\ \frac{a_2}{A_2}\sqrt{2gh_2^0} &= \frac{(1-\gamma_1)k_1}{A_4}u_1^0 + \frac{\gamma_2 k_2}{A_2}u_2^0\end{aligned} \tag{4}$$

If we choose stationary levels in the lower tanks, h_1^0, h_2^0, we can obtain the stationary control signals u_1^0, u_2^0 by solving the linear system of equations (4). The stationary levels in the upper tanks, h_3^0, h_4^0, are then obtained from (3).

Linear model

Let $\Delta u_i = u_i - u_i^0$, $\Delta h_i = h_i - h_i^0$, and $\Delta y_i = y_i - y_i^0$. Introducing

$$u = \begin{pmatrix}\Delta u_1 \\ \Delta u_2\end{pmatrix}, \quad x = \begin{pmatrix}\Delta h_1 \\ \Delta h_2 \\ \Delta h_3 \\ \Delta h_4\end{pmatrix}, \quad y = \begin{pmatrix}\Delta y_1 \\ \Delta y_2\end{pmatrix}$$

and linearizing around a stationary point gives the linear system

$$\begin{aligned}\frac{dx}{dt} &= \begin{pmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{pmatrix} x + \begin{pmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{pmatrix} u, \\ y &= \begin{pmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{pmatrix} x,\end{aligned}$$

where the time constants T_i, $i = 1 \ldots 4$, are given by

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}}$$

The transfer matrix from u to y is given by

$$G(s) = \begin{pmatrix} \dfrac{\gamma_1 c_1}{1+sT_1} & \dfrac{(1-\gamma_2)c_1}{(1+sT_1)(1+sT_3)} \\ \dfrac{(1-\gamma_1)c_2}{(1+sT_2)(1+sT_4)} & \dfrac{\gamma_2 c_2}{1+sT_2} \end{pmatrix}$$

where $c_1 = T_1 k_1 k_c / A_1$ and $c_2 = T_2 k_2 k_c / A_2$.

Multivariable zeros

As shown in Exercise Session 10, the (transmission) zeros of $G(s)$ are given by the equation

$$(1+sT_3)(1+sT_4) - \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1\gamma_2} = 0$$

The system is found to be minimum phase (i.e., all zeros are in the left half-plane) if $1 < \gamma_1 + \gamma_2 \leq 2$ and non-minimum phase (i.e., at least one zero is in the right half-plane) if $0 \leq \gamma_1 + \gamma_2 < 1$.

Process data

The quadruple tank has approximately the following physical constants:

Table 1 Nominal parameter values of the quadruple-tanks process.

Name	Value	Unit
A_i	4.9	cm^2
a_i	0.03	cm^2
k_i	1.6	cm^3/V
k_c	0.5	Vcm^{-1}
g	981	cms^{-1}

The flow from each pump is divided so that roughly 70% goes into one tank and 30% into the other. In the minimum-phase case, the 70% goes into the lower tanks and we have $\gamma_1 = \gamma_2 = 0.7$. In the non-minimum-phase case, the 30% goes into the lower tanks and we have $\gamma_1 = \gamma_2 = 0.3$.

The operating point in the lower tanks are chosen as $h_1^0 = h_2^0 = 10$ cm. In the minimum-phase case the stationary levels in the upper tanks are calculated to be $h_3^0 = h_4^0 = 0.9$ cm and in the non-minimum-phase case the stationary levels are $h_3^0 = h_4^0 = 4.9$ cm. In both cases, the stationary control signals are $u_1^0 = u_2^0 = 2.6$ V.

Discretization

The undisturbed, continuous-time model

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t), \end{aligned}$$

is ZOH-sampled in `Matlab` to a model of the form:

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k \\ y_k &= Cx_k \end{aligned}$$

Table 2 Physical constraints of the tank process.

Name	Min	Max	Unit
u_i	0	10	V
h_i	0	15	cm

Table 3 Description of files used in this laboratory exercise. (*) is implemented by the student.

Name	Description
`configure_lab.m`	The file where you define system and controller parameters.
`LinearTankModel.m`	A class file for the `LinearTankModel` class
`flowCtrl1.m`	PI controller for pump 1.
`flowCtrl2.m`	PI controller for pump 2.
`setup_mpc.m`	Function for creating an MPC object.
`plot_figures.m`	Script for plotting each run to matlab figures.
`quadtank_real_manual.mdl`	Simulink interface for manually running the tank process.
`simulate_process.mdl`	Simulink interface for both simulations and experiments.
`estimate_parameters.m`	Matlab script for parameter estimation.
`estimate_parameters_sim.slx`	Simulink file used in parameter estimation.
`getConstraints.m`	Function - formulates constraints in linearized variables.
(*)`setup_lqg.m`	Function generates an LQG-controller.

Constraints

Since the tanks and pumps have limited capacity, the constraints in Table 2 will be used in this laboratory exercise.

Preparatory 1: The system will be linearized around the a some stationary points h_i^0, given some stationary inflows u_i^0. Express the constraints from Table 2 in the linearized variables $x_i = h_i - h_i^0$ and $\Delta u_i = u_i - u_i^0$. I.e state $x_{i,max}$, $x_{i,min}$, $\Delta u_{i,max}$ and $\Delta u_{i,min}$.

3. The lab interface

The controllers are designed and tested in `Matlab/Simulink`. The `Simulink` file `simulate_process.mdl`, see Figure 2, is used to switch controllers (LQG, Kill, MPC) and systems (linear model, nonlinear model, real process). The controller *Kill* sends $u = 0$ and is used to stop the tanks between experiments. The `Simulink` interface runs the script `configure_lab.m`, where you define all parameters, before each simulation. See Table 3 for a full list of files.

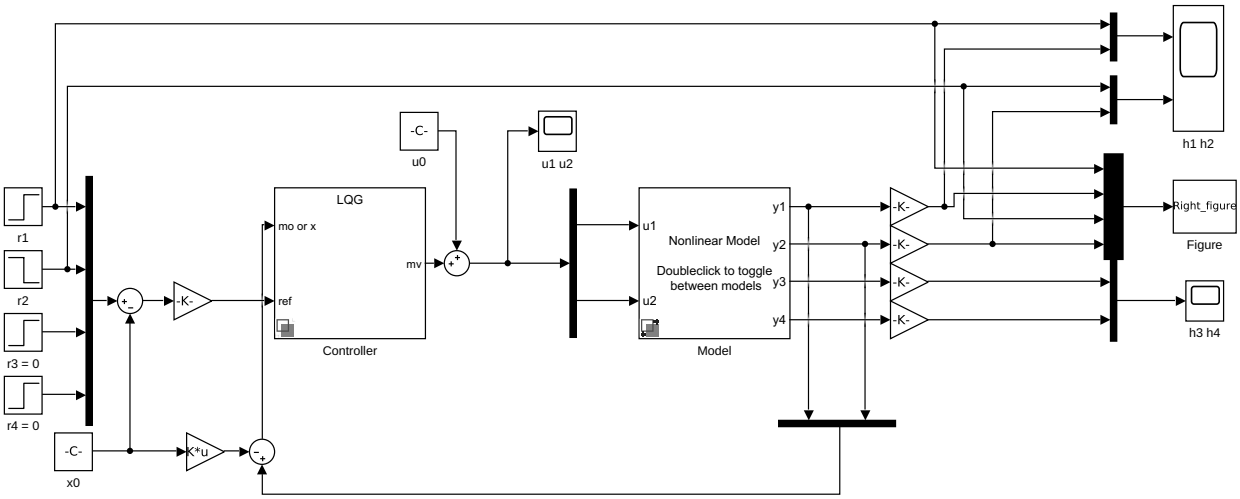

Figure 2 `Simulink` interface to this lab exercise.

4. [THEORY] From LQG to MPC

The purpose of this section is to illustrate the similarites between the Linear-Quadratic-Gaussian regulator with the quadratic-cost model-predictive controller. We will study the case where there are no cross-terms in the cost functionals.

4.1 Linear-Quadratic Regulator (LQR)

Given some initial process state x_0,

$$\begin{aligned} \text{minimize} \qquad & J = \sum_{k=0}^{\infty} \Big(x_k^{\mathrm{T}} Q_1 x_k + u_k^{\mathrm{T}} Q_2 u_k \Big) \\ \text{subject to} \qquad & u_k \in \mathbf{R}^m, \ x_k \in \mathbf{R}^n \\ & x_{k+1} = \Phi x_k + \Gamma u_k \end{aligned} \tag{5}$$

We can solve the LQR problem using dynamic programming, which leads to an algebraic Riccati equation when the optimization horizon goes to infinity. The solution is a static feedback law, $u_k = -Kx_k$.

Preparatory 2: Calculate the optimal feedback K using `MATLAB`'s `dlqr` in `setup_lqg.m`

4.2 Unconstrained Model-Predictive Control (MPC)

Given the initial state x_τ,

$$\begin{aligned} \text{minimize} \qquad & J = \sum_{k=\tau}^{\tau+H_p} \Big((x_k - r_k)^{\mathrm{T}} Q_1 (x_k - r_k) + u_k^{\mathrm{T}} Q_2 u_k \Big) \\ \text{subject to} \qquad & u_k \in \mathbf{R}^m, \ x_k \in \mathbf{R}^n \\ & x_{k+1} = \Phi x_k + \Gamma u_k \end{aligned} \tag{6}$$

The mathematical program above is very similar to the LQR problem. The main difference is that, for MPC, we are only looking H_p (called the prediction horizon) steps ahead. We compute the solution numerically, which will be a control trajectory $(u_\tau, u_{\tau+1}, \ldots, u_{\tau+H_p})$. At time τ we apply u_τ. The next time-step we measure $x_{\tau+1}$ and redo the computation.

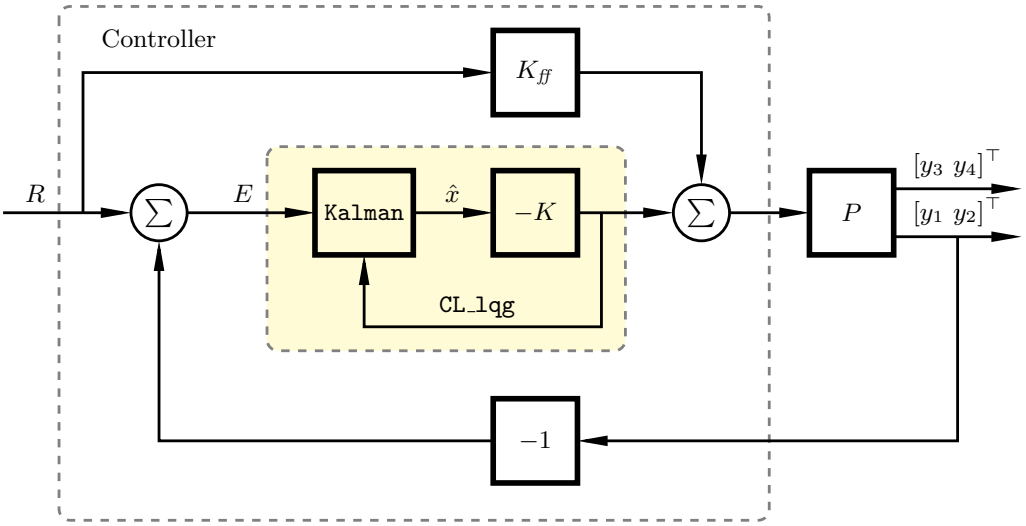

Figure 3 Architecture of LQG controller block. It takes y and r as inputs and gives u as output.

Reflect 1: What happens when you let the prediction horizon go to infinity? I.e., $H_p \to \infty$

Gaussian white noise disturbances and output feedback

Both LQR and MPC as formulated above require full state information. In this lab we will however only measure the water level in the two bottom tanks, you will thus need to design observers for the system. We will assume a uncorrelated white noise on each state and on the outputs, resulting in the extended system

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k + w_k \\ y_k &= C x_k + v_k \end{aligned}$$

where w_k and v_k are white noise, independent from each other, with covariance matrices R_1 and R_2.

Preparatory 3: Design a Kalman estimator in `setup_lqg.m` using `dlqe`. Then use the estimated values in the state feedback, according to Figure 3. Remember that you can only measure y_1 and y_2. How should this be accounted for when determining the C matrix for the extended Kalman system? Also, to get G matrix used in the lecture notes remember that $w_k = Iw_k$. You must design the whole LQG controller, which will be a state-space system. This is the system within the inner dashed box in Figure 3, named `CL_lqg`.

Hint: Use `help` to find out what different functions in `setup_lqg.m` do. Look at what you did in Exercise 8.

Structure and tracking of step signals

We will use the structure in Figure 3, meaning that we do feedback control on the tracking error e. In order to ensure a DC gain of unit magnitude we will use a constant feedforward gain $K_{ff} \in \mathbf{R}^{2\times 2}$.

Preparatory 4: Design the feed forward filter K_{ff}, that guarantees unit stationary gain. Remember to take into account that you only want to control y_1 and y_2. How should this be implemented in `MATLAB`? In `setup_lqg.m` choose the correct calculation of K_{ff}. Motivate your choice.

Table 4 Initial settings for the Minimum-phase simulations

Parameter	**Value**
Model	Nonlinear
γ	0.7
T_s	0.1
H_p	1
`firstPenaltySample`	1
Simulation Time	40s
Q_1	Diag[10 10 0 0]
Q_2	Diag[1 1]
R_1	I
R_2	I

Hints: Remember to take into consideration which states you are trying to control. The LQG block has no integral action. How will this affect the steady-state output of the LQG block?

5. [SIMULATIONS] from LQG to MPC

For this and the next section, you will use a static reference ($r = h_0$). The initial state should be empty tanks. In other words, turning on the simulation will make a reference change from empty tanks to your linearization point $(10, 10)$.

Task 1: Log in on the computer with your student login. Open a terminal (by searching for Terminal in the computer search field). Start version `R2020b` of `MATLAB` by typing

```
VERSION=R2020b matlab
```

in the terminal. Download the lab files from Canvas and set up you simulation according to Table 4. Complete the function `setup_lqg.m` (see Preparatory exercises). Ask the lab supervisor to check that everything is correct in your `setup_lqg.m`. Make sure that everything in `configure_lab` is set according to Table 4. Open `simulate_process.mdl`. The simulatiuon time is set as the "Stop Time" in the `Simulink` model. Select the LQG controller and run the simulation. Switch which figure you are plotting to by double-clicking on the `Figure` block in `Simulink`. The Left figure/Right figure block is for you to be able to look at the results from previous simulations. Switch to the MPC controller and make sure all constraints are $\pm\infty$. Run the simulation. Slowly increase H_p, what do you see?

For what value of H_p does increasing the prediction horizon stop affecting the behavior of the system?

Task 2: Set $\gamma = 0.3$ and repeat the previous experiment.
Hint: If computation becomes too heavy, increase the sample time and decrease the prediction horizon by the corresponding factor.

Reflect 2:

- What differed between the two configurations?
- Were your answers different? If so, why?
- Is this the optimal solution?

6. [EXPERIMENTS] from LQG to MPC

We will configure the physical process such that 70% of the flow from Pump 1 goes into Tank 4, and that 70% of the flow from Pump 2 goes into Tank 3, i.e., $\gamma_1 = \gamma_2 = 0.3$. This is achieved with the following valve configuration: `AV1, BV1` and `BV2` should be pulled out while `AV2` should be pushed in. Ensure that `AV3, AV4, BV3, BV4` and `V5` are all pushed down.

When using `Simulink` to control the tanks the final value sent out from the controller will keep being sent to the pumps. To avoid the pumps continuing running there is a Kill setting in the Controller block. Once the simulation time has run out, change the controller to Kill and run. You can imitatively stop the simulation. Now the pumps will be shut off and you can continue with your experiments by selecting the controller you want again. However, the plots generated in the `Simulink` scopes will be lost when Kill is run.

Task 3: Now switch Model to `Real Process`, Controller to `LQG`. Sample time should be 1 second and the tanks should be configured in the non-minimum phase setting described above. Simulation time should be 80 seconds. Run the simulation to perform a reference step. Plot the control signals. You can this by double clicking on the scope icons called `u1 u2` and `h1 h2`. How did it go?

Reflect 3: We calculate the optimal feedback law using a linear model. How well does this model match reality? Where do errors come from? Which of these errors have a large-enough impact to lead to a stationary error?

Preparatory 5: After years of use, the nominal parameters no longer gives an accurate model. Using equations 1, 3, 4, try to figure out a set of experiments to run on the process to determine all the process parameters that can change with wear and tear. You don't have to express the parameters explicitly, but rather come up with different experiments and state which measurements you will need. You should get enough equations to be able to derive all the process parameters that can change with time. Since the cross section area of the tanks don't change with time you can assume that all A_i are known. However, the outflow areas a_i can get clogged up with time and thus need to be estimated through experiments. Which other parameters can change with time? In total there are 8. Due to splashing in the lower tanks the measurements will be very noisy. Therefore, the derivatives in equations 1 will be hard to estimate. How can you come around the noisiness of the measurements in the lower tanks in equation 1? In this preparatory task you can assume that you can measure the levels in both the upper and lower tanks. [1]

[1] In the processes there are sensors in both the upper and lower tanks, even though we only use the measurements from the lower tanks. If there wouldn't have been any sensors in the upper tank it would have been possible to read the heights manually for a few experiments, to calculate the parameters by hand.

Task 4: Run the script `estimate_parameters.m` to start a parameter identification experiment (the experiment completes in about 3 minutes). Can you figure out how the script is deriving the process parameters? Was it similar to how you would have done the experiments? `estimate_parameters.m` will create a file `estimData.mat`. In the `configure_lab.m` file, uncomment the line `lt.estimData = load('estimData.mat').estimData;` to pass the estimated parameters to the `LinearTankModel` object. Also uncomment `lt.useEstimData = ...` in `configure_lab.m`. To toggle between nominal parameters and estimated parameters use:

```
lt.useEstimData = 0; % Use nominal parameters, default
lt.useEstimData = 1; % Use estimated parameters.
```

Task 5: Your new task is to ensure that $u_i \leq 10$ when the system tries to go to the initial reference. Change the weight matrices Q_1 and Q_2 to make sure that you do not violate this constraint when using the LQG controller. How fast can you make the system? I.e. what is the lowest rise-time you can get? Spend at most 10 minutes on this. To speed things up, feel free to use simulations. If you are using the real tanks, make sure that the tanks has time to empty between tests. To make the real simulations quicker, set the simulation time to 10 seconds.

We will now solve the same problem by letting the model-predictive controller enforce constraints.

Task 6: Return to your initial weights. Set the prediction horizon to $H_p = 40$ steps. Change the constraints so that $0 \leq u_i \leq 10$. Run the experiment. Remember to activate the MPC-controller. What do you see?

Task 7: Increase simulation time to 200 seconds. At $t = 100$s we make a reference change, where we increase the amount in Tank 1 and decrease it in Tank 2 by the same amount (± 2). What behavior do you see?

7. [SIMULATIONS] Prediction Horizon

Select the nonlinear model, set $T_s = 1$ and $\gamma = 0.3$ (non-minimum phase configuration) and the simulation time to 200 seconds. Remember to set `lt.useEstimData = 0`.

Task 8: Recall the preparatory quiz on Canvas. Rerun the simulation if you have to. Let t_1 be the time where the step response stops going in the opposite direction and starts going in the right direction.

Select $H_p \cdot T_s = H_p << t_1$ and run the simulation. What happens?

Reflect 4: Why did this happen now, and not previously?

Task 9: Increase the prediction horizon to $H_p = 100$ and run the simulation. Now slowly decrease the prediction horizon. How low can you go and still track the reference change decently? How low can you go before the controller starts doing strange things?

Reflect 5:

- Why do you need a much longer prediction horizon when trying to create a difference between the tanks, compared to when you make a uniform reference change?
- Does the minimum viable prediction horizon agree with your observation of the time constant you found in the quiz for a reference difference? Why?

8. [THEORY] Time-varying weights

The cost function J is something we choose as engineers. Our goal is to choose a cost function which leads both to desired behavior when minimized, as well as a tractable problem to solve. Quadratic functions is a class which presents tractable problems, and has corresponds to penalizing the energy of the states and input signal. Previously we had a time-invariant cost function, but there are cases where it makes sense to vary the penalites dependin on how far we are predicting. The structure of a mathematical program with time-varying weights is shown in (7).

$$\begin{aligned} \text{minimize} \quad & J = \sum_{k=\tau}^{\tau+H_p} \Big((x_k - r_k)^{\mathrm{T}} Q_{1,k-\tau} (x_k - r_k) + u_k^{\mathrm{T}} Q_{2,k-\tau} u_k \Big) \\ \text{subject to} \quad & u_k \in \mathcal{U},\ x_k \in \mathcal{X} \\ & x_{k+1} = \Phi x_k + \Gamma u_k \end{aligned} \tag{7}$$

In the upcoming tasks we will study the case where

$$Q_{1,k} = \begin{cases} Q_1, & k \geq \texttt{firstPenaltySample} \\ 0, & \text{else,} \end{cases}$$

meaning that we do not penalize the first couple of samples and put equal weight on the samples starting from `firstPenaltySample`.

9. [SIMULATIONS] Time-varying weights and Model errors

Task 10: Experiment with increasing the `firstPenaltySample` and decreasing the prediction horizon, what is the smallest H_p you can get away with? How low can you go before the controller starts doing strange things?

Reflect 6: Why does it help to not penalize the first couple of time-steps?

Task 11: Switch to simulating on the linear model. Do you see any differences? Pick one of your previous winning combinations of H_p and `firstPenaltySample`. Change the references to

$$r_1(t) = \begin{cases} 2 & \text{if } t \geq 0 \\ 3 & \text{if } t \geq 100 \end{cases}$$

$$r_2(t) = \begin{cases} 2 & \text{if } t \geq 0 \\ 1 & \text{if } t \geq 100 \end{cases}$$

This can be done by double clicking on the two step icons called `r1` and `r2`.

Task 12: Run the experiment on the linear model, switch figure and run it on the nonlinear model. Can you explain the differences?

In order to to account for model errors and constant disturbances the standard disturbance model in `MATLAB`'s MPC toolbox is integrated white noise as output disturbance. To enable this comment out the lines

- `setoutdist(mpcobj, 'Model', ss(zeros(4,1)))`
- `setEstimator(mpcobj, ssdata(lt.DtSys)*L, L)`

in the file `configure_lab.m`.

Task 13: Simulate against the nonlinear model again. Do you see any difference?

Reflect 7: Is it always better to include integrated white noise, like `MATLAB`'s MPC toolbox does by default?

10. [EXPERIMENTS] Prediction Horizon and time-varying weights

Task 14: If you have time, repeat tasks 8–10 for the real process. Do not start this if there is less than 30 minutes left of the lab session. Remember use `lt.useEstimData = 1`. How does this compare to the simulations?

11. Conclusion

In this laboratory exercise we have experimented with design choices leading to a right-half-plane zero. Poor process design can lead to poor performance under control, we must always respect the fundamental limitations. What are some considerations you will bring into the process-design process?

When you are finished let the lab supervisor know and be ready to present your findings from the different tasks.