Manual for Double-Tank Lab

Several contributors*

Department of Electrical Engineering,
Chalmers University of Technology

November, 2022

*First version, 1995: Magnus Jansson.
Updated, 2002: Fredrik Bruzelius, Knut Åkesson.
Updated, 2006: Lars Johannesson, Avenir Kobetski, Veronica Olesen.
Adapted to Simulink, 2006: Johan Larsson, Bo Egardt.
Updated, 2007: Goran Čengić.
Updated, 2007: Lars Johannesson, Johan Rönnberg, Mikael Thor
Updated, 2008: Lars Johannesson, Mikael Thor
Updated, 2009: Magnus Nilsson.
Updated, 2022: Torsten Wik, Albert Škegro.

1 Introduction.

In this assignment (and subsequent laboratory exercise), a controller will be designed on the basis of a non-mathematical description of a double-tank process. Typical steps in such a project are:

- Physical system modeling.
- Model analysis and experimental validation.
- Model-based controller design.
- Experimental evaluation.

To be sufficiently prepared for the experimental part, it is therefore **an absolute prerequisite** that Exercise 1 to 11 are solved before the start of the laboratory exercise.

1.1 Purpose of `Part I: Preparatory assignments`.

The purpose of preparatory assignments is to

- Apply a general methodology to formulate a mathematical model based on fundamental physical laws and simplifications.
- Use mathematical software to analyze the model in terms of transient responses, bandwidth, stability margins, controllability and observability.
- Use mathematical software for controller design.
- Provide an introduction to the experimental part.

1.2 Purpose of `Part II: Experiments`.

The purpose of experiments is to

- Learn how different controllers perform and relate observations to properties such as stability margins, sensitivity to disturbances, robustness, etc.
- Experimentally find controller parameters in a case where simulation and implementation are closely integrated.
- Learn how different disturbance models affect the estimations of a stationary Kalman filter.
- Illustrate the importance of instrumentation, particularly in cases when the controlled variable cannot be measured.

1.3 Outline of preparatory assignments and experiments.

The purposes described above are reached by solving a number of exercises. What follows is an outline of the main tasks that are to be accomplished.

Part I: Preparatory assignments (executed *before* the lab).

1. **Process modeling.** Using fundamental laws of physics, balance equations are derived. In the case of the double-tank process, this results in two nonlinear coupled differential equations. In order to be able to proceed with linear controller design, the process is linearized around a given operating point.

2. **Analysis and simulations.** The process models (both original nonlinear and the linearized model) are analyzed. `Simulink` model is built for the nonlinear model. In order to validate the linearized model, the transient response is simulated for both models.
3. **Controller design.** A SISO PI-controller is synthesized and tested in `Simulink`. Thereafter, a MIMO-controller is designed based on the minimization of a quadratic criterion.

Part II: Experiments (executed *in* the lab).

1. **Real-time experiment.** The process model is validated and the controllers designed in `Part I: Assignment` are tested on the double-tank process.
2. **State estimation.** A Kalman filter is designed to estimate the water levels in the tanks for the case when only one level sensor is available.
3. **Integral action.** The process model is augmented with additional states to allow for integral action.
4. **Real-time experiment.** The performance of the resulting MIMO-controller paired with the Kalman filter is tested on the double-tank process.

Note. Within `Part I: Preparatory assignments` (Exercise 1 to 11), all calculations and simulations assume the continuous-time framework.

However, within `Part II: Experiments` (Exercise 12 to 18), all calculations assume a discrete-time framework. `MATLAB` will be used for translating calculations to a discrete-time framework. This is a good occasion to refresh on the similarities and differences between the two frameworks. See also Exercise 11.

2 Process modeling.

There are a number of approaches to process modeling.

By first principles or *physical modeling* (also: *white-box modeling*), the derivation of a mathematical model of a process is understood based on laws of physics, physical relationships and detailed insight in the process.

The contrary approach deals with deriving a model without knowledge of the process - solely the measurement data is used. This is referred to as *black-box modeling.*

Mixtures between pure physical modeling and black-box modeling can be referred to as *gray-box modeling.*

2.1 A control engineer's approach to physical modeling.

Physical modeling is a craft. Derivation of an appropriate model requires an intuition for the control problem. There are, however, some guiding principles for how to approach the modeling work. In the process of formulating a suitable state-space model, three main steps can be identified:

1. Set up the problem **structure**.
2. Express the relevant laws of physics and physical relationships in terms of mathematical equations (**basic equations**).
3. Reformulate the equations to achieve a **state-space model**.

The first phase is the most abstract. It requires understanding of a problem and ability to split the system into smaller parts (subsystems). The work becomes more systematic towards the final phase, which mainly consists of manipulating the equations. Note that it might be necessary to repeat the process several times until a satisfying model is found.

2.1.1 Structure.

The first and the most important question in the entire physical modeling phase is:

What will the model be used for?

A model which is to be used for investigating main dynamic features, such as the dominating time constant, stationary gain, time delays etc. allows for much rougher approximations than a model which is to be used for detailed studies.

When the degree of precision has been decided upon, the control engineer should split the system into smaller subsystems. Furthermore, it is important to divide the system variables into two categories. The variables which fall in the first category are the ones through which the subsystems interact with each other. On the contrary, the variables which fall in the second category are the external variables affecting the subsystems. This results in a model which is in the form of a block diagram. The main blocks might not yet contain any equations or transfer functions. For large systems, it is a good idea to thoroughly document the specifications (inputs/outputs) for each subsystem model.

2.1.2 Basic equations.

In the next phase, the physical laws and relationships applicable to each subsystem are listed. The relationships can be divided into *conservation laws* and *constitutive relationships.*

Conservation laws relate *quantities of the same kind.* Common examples are:

- Kirchhoff's laws: $\sum$ currents = 0 or $\sum$ voltages = 0.
- Material balances: inflow − outflow = accumulated matter per time.
- Force balance: $\sum$ external forces = kinetic power (= ma).
- Energy balance: power in − power out = accumulated energy per time.

Constitutive relationships relate *quantities of different kinds* to each other. Common examples are:

- Ohm's law: relates current to voltage over a resistance.
- Geometry: relates geometry to volume and mass.
- Air resistance: relates force to geometry and speed.
- Heat capacity: relates temperature to energy.

The most intuitive approach begins with applying the conservation laws, followed by constitutive relationships to express the laws of conservation in suitable variables. As a final check, it is recommended that dimensions (and physical units) of all terms are analyzed.

2.1.3 State equations.

Although the mathematical model of the system is now available, the equations are likely unstructured. It can be difficult to simulate the model directly, and even more difficult to use it for controller design. A standard form for a dynamic model, suitable for simulation and controller design, is the *state-space form.* If possible, the following general form should be aimed for:

$$\dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \tag{1a}$$

$$\mathbf{y}(t) = \mathbf{g}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \tag{1b}$$

where $\mathbf{x}(t)$ are the states, $\mathbf{u}(t)$ the inputs, and $\mathbf{y}(t)$ the outputs.

The form Equation (1a) contains only derivatives of time. Hence, it is an *ordinary differential equation* (ODE). Detailed physical descriptions, however, often result in *partial differential equations* (PDE) originating from, for example, the heat equation, diffusion equation, dispersed flow equation, Navier-Stokes equations etc. They are much more difficult to handle and are not suited for controller design. Therefore, it is the task of a control engineer task to make appropriate approximations to avoid having to cope with the actual solution of a PDE. For example, using *finite differences* when calculating space derivatives in a time-dependent PDE transforms the time-dependent PDE into an ODE.

In most cases, an ODE for the system description can be achieved. However, sometimes it happens that that purely algebraic expressions have to be included in the model as well. Such models are named *ordinary differential-algebraic equation* (ODAE). Correspondingly, PDE systems with algebraic equations are referred to as *partial differential-algebraic equation* (PDAE).

2.1.4 State variables.

Candidates for state variables can be found among the variables having a time derivative in the resulting ODE. Generally, it is desirable to use as few state variables as possible. The model is then said to be of *minimum order*. For a linear system of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2a}$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{2b}$$

the model is of minimum order if the system is both *controllable* and *observable*.

One may also get an idea of the number of states by counting the number of variables that represent an accumulation of matter or energy: a moving mass accumulates kinetic energy, a spring stores potential energy, an inductor stores magnetic field energy, a capacitor stores electric field energy, a liquid vessel stores potential energy etc. After the state variables have been selected, the constitutive relationships are to be used to bring the model to the form of Equation (1) or Equation (2).

2.1.5 Linearization.

Most methods for controller design and analysis (such as Equation (2)) require a linear model of the system. If the state-space model is nonlinear, a natural step is to linearize the model around an expected operating point. As long as the nonlinearities are reasonably smooth, the linearized model will describe the dynamics well in a region near the operating point.

The process of linearization is described in most textbooks on basic control. It is merely a Taylor expansion around an operating point with all terms of order two or higher ignored as they are small in the vicinity of the operating point. If differentiable, the nonlinear system Equation (1) can thus be linearized into the form Equation (2). The operating point is typically chosen to be a *stationary* point $(\mathbf{x}_0, \mathbf{u}_0, \mathbf{y}_0)^T$ which means

$$\begin{aligned} 0 &= \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) \\ \mathbf{y}_0 &= \mathbf{g}(\mathbf{x}_0, \mathbf{u}_0) \end{aligned}$$

After determining this stationary point, the matrices **A**, **B**, **C**, and **D** are determined from

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_\mathrm{n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\mathrm{n}}{\partial x_1} & \cdots & \frac{\partial f_\mathrm{n}}{\partial x_\mathrm{n}} \end{bmatrix}_{(\mathbf{x}_0,\mathbf{u}_0)} \qquad \mathbf{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_\mathrm{m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\mathrm{n}}{\partial u_1} & \cdots & \frac{\partial f_\mathrm{n}}{\partial u_\mathrm{m}} \end{bmatrix}_{(\mathbf{x}_0,\mathbf{u}_0)}$$
$$\mathbf{C} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_\mathrm{n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_\mathrm{n}}{\partial x_1} & \cdots & \frac{\partial g_\mathrm{n}}{\partial x_\mathrm{n}} \end{bmatrix}_{(\mathbf{x}_0,\mathbf{u}_0)} \qquad \mathbf{D} = \begin{bmatrix} \frac{\partial g_1}{\partial u_1} & \cdots & \frac{\partial g_1}{\partial u_\mathrm{m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_\mathrm{n}}{\partial u_1} & \cdots & \frac{\partial g_\mathrm{n}}{\partial u_\mathrm{m}} \end{bmatrix}_{(\mathbf{x}_0,\mathbf{u}_0)}$$

where the partial derivatives in the Jacobians are evaluated using the operating point values $\mathbf{x}_0$ and $\mathbf{u}_0$.

2.2 Process description.

The process, schematically depicted at Figure 1, consists of two water tanks with a pipe in between that allows water to flow in both directions. The two water levels (h_1 and h_2) are to be controlled. The process is multiple-input multiple-output system (MIMO-system) since each tank has an individual inflow q_i and an individual water level measurement h_i. **The cross-coupling between the tanks may motivate controllers based on state-space descriptions.**

To examine the system's robustness to disturbances, individual disturbance inflows are also available for each tank.

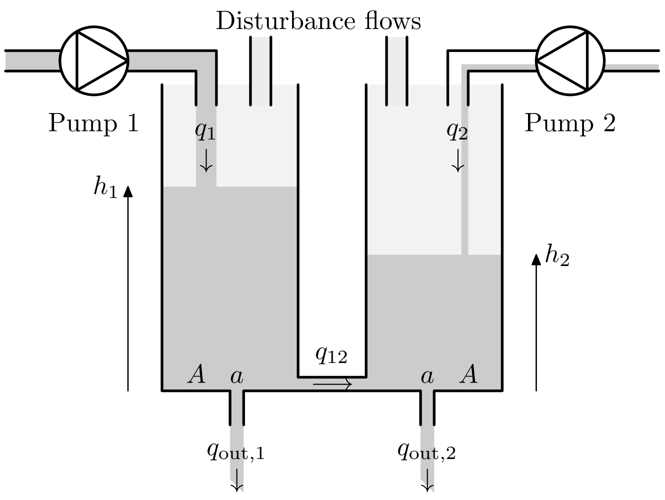

Figure 1: Schematic depiction of the double-tank process.

2.2.1 Instrumentation and manual open-loop control.

The process is controlled with a standard personal computer equipped with software to handle real-time control. The outputs consist of electrical voltages directly proportional to the pressure the liquid exerts to the bottom of each respective tank. The pressures are, in turn, directly proportional to the water level in respective tank. The voltage levels from the sensors are registered by the computer through the analog-to-digital converter (A/D-converter).

The computer delivers control signals to the process through digital-to-analog conversion (D/A-conversion). Control signals are pump voltages and they determine the inflows. Even though control is normally handled through the computer, a console exists for the manual control of the process as well. Note that the disturbance flows can only be controlled from this console.

2.3 Process model.

We should now apply the guiding principles from Section 2.1 to find an appropriate model of the process. Unfortunately, most assumptions of the process behavior have to be preset, since the tasks you will carry out further on presumes a particular model structure.

2.3.1 Structure.

The model should be used for analysis of the main dynamics and for controller design and synthesis. Therefore, we aim for a simple model of low order. If the model turns out insufficient, more details are added. For the controller design, a linear state-space model needs to be derived.

2.3.2 Equations.

A simplified illustration of the process is shown in Figure 1. We must first decide what physical properties we should consider. On a macro scale, we can divide the system into the electrical part, the mechanical part, and the fluid part.

Electrical and mechanical system. The inflows to each tank are controlled by pumps driven by small DC motors. Because of their fast dynamics, the pump behavior can be approximated by the static linear relation,

$$q_{\mathrm{i}} = K_{\mathrm{q}} u_{\mathrm{i}} \qquad i = 1, 2 \tag{3}$$

where u_{i} are the input voltages to the DC motors, q_{i} are the inflows to the tanks, and $K_{\mathrm{q}} = 10\,\mathrm{cm}^3/\mathrm{sV}$.

Fluid system. Bernoulli's law describes the flow of an incompressible fluid along a stream line without losses as

$$p + \rho\frac{w^2}{2} + \rho g z = \text{constant},$$

where p is the static pressure [Pa], ρ is the density of the fluid [kg/m^3], w is the flow velocity [m/s], g is the gravity constant [m/s^2], and z is the height of the liquid [m]. Applying this on the two respective water tank outlets we get

$$q_{\mathrm{out,i}} = a\sqrt{2gh_{\mathrm{i}}} \qquad i = 1, 2 \tag{4}$$

where $a = 0.238\,\mathrm{cm}^2$ is the outlet area and $g = 981\,\mathrm{cm/s}^2$ is the Earth's gravity constant. Because of non-ideal conditions we cannot apply Bernoulli's equation on the flow between the tanks. However, it can be reasonably approximated by

$$q_{12} = K_{12}(h_1 - h_2)\,, \tag{5}$$

where $K_{12} = 6\,\mathrm{cm}^2/\mathrm{s}$. The sensors deliver voltages (V_1 and V_2) that are approximately proportional to the water levels, i.e.,

$$h_{\mathrm{i}} = K_{\mathrm{h}} V_{\mathrm{i}} \qquad i = 1, 2, \tag{6}$$

where $K_{\mathrm{h}} = 2.0\,\mathrm{cm/V}$. The bottom area of each tank is $A = 63.6\,\mathrm{cm}^2$.

Exercise 1 Physical modeling delimitations.

Denote at least 4 applicable properties or relationships that have been ignored in the physical modeling process.

Exercise 2 Nonlinear state-space model.

Derive a nonlinear state-space model with inputs (u_1, u_2) and outputs (h_1, h_2) when water levels are chosen as state variables. Use Equations (3) to (5) together with the volume balance equation.

Due to the nonlinear expressions originating from Bernoulli's law, the state equations have to be linearized to allow for linear control design. In this case, linearization corresponds to approximating the square roots by a line (tangent) as exemplified in Figure 2.

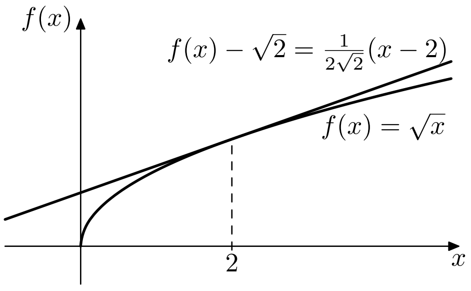

Figure 2: Linear approximation of $f(x) = \sqrt{x}$ around $x = 2$.

Exercise 3 Linearization of the nonlinear model.

Linearize the nonlinear model from Exercise 2. Choose the operating point $h_{10} = h_{20} = 14$ cm. Useful `MATLAB` commands: `solve`, `jacobian`, `subs`.

2.3.3 State-space transformations.

Normally, it is desirable to use state variables that have a physical interpretation, as in Exercise 2 and Exercise 3, where the water levels were selected as state variables. However, there exists no set of state variables that is unique for a given model. Another set of state variables $z_1, \ldots, z_n$, can be achieved from combinations of the original states $x_1, \ldots, x_n$. Particularly, to obtain a new set of state variables a linear transformation can be used:

$$\mathbf{z} = \mathbf{T}\mathbf{x} \tag{7}$$

In order to have a one-to-one mapping between the new and the original states, the *transformation matrix* $\mathbf{T}$ must be non-singular (i.e., there must exist an inverse $\mathbf{T}^{-1}$). From Equation (7), the following can be obtained:

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{z}$$

The new state-space model can then be written as:

$$\begin{aligned} \dot{z} &= \mathbf{A}_z\mathbf{z} + \mathbf{B}_z\mathbf{u} \\ y &= \mathbf{C}_z\mathbf{z} \end{aligned}$$

Exercise 4 State-space transformation: an example.

Determine $\mathbf{A}_{\mathrm{z}}$, $\mathbf{B}_{\mathrm{z}}$, and $\mathbf{C}_{\mathrm{z}}$ in terms of $\mathbf{T}$, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. What happens to the poles and zeros of the system? Justify your answer!

Suppose we want to use the voltages from the sensors as states instead of h_1 and h_2. What will $\mathbf{A}_{\mathrm{z}}$, $\mathbf{B}_{\mathrm{z}}$, and $\mathbf{C}_{\mathrm{z}}$ be in that case?

3 Analysis and simulations.

The remaining exercises are solved using computer software. The recommended software is MATLAB and Simulink in combination with Control System Toolbox. Specific MATLAB commands are displayed on the form

```
>> command
```

If explanations on how to use the command are desired, MATLAB's help can be consulted. As an example, if you need help on how to use the command `plot`, you type

```
>> help plot
```

or use the help function under RESOURCES. It is strongly recommended that you create a MATLAB script file where you keep all parameters and calculations, since the parameter values and calculations in your script will be utilized in the actual experiments in the lab. The MATLAB editor can be opened by issuing the command

```
>> edit
```

Within the editor, parameter names can be assigned values, as exemplified in Figure 3. If you save your script file as (for example) `Lab_WaterTanks_Params.m`, all parameters in the file can be set by issuing the command

```
>> Lab_WaterTanks_Params
```

```
% Pars for nonlin model
Kq  = 10;    % cm^3/sV
a   = 0.238; % cm^2
At  = 63.6;  % cm^2
g   = 981;   % cm/s^2
K12 = 6;     % cm^2/s
h10 = 14;    % cm
h20 = 14;    % cm
```

```
% Decoupled lin. model
A = [-a/At*sqrt(g/(2*h10)), 0;...
     0, -a/At*sqrt(g/(2*h20))];
B = [Kq/At, 0;...
     0, Kq/At];
C = eye(2);
D = zeros(2);
tanklindecoupl = ss(A,B,C,D);
```

Figure 3: Example of reasonable code snippet to include in a script.

3.1 Controllability and observability

Exercise 5 Minimal realization.

Is the linear model from Exercise 3 a minimal realization? Motivate your answer! Some useful MATLAB commands are

```
>> obsv(A,C) % Observability matrix
>> ctrb(A,B) % Controllability matrix
>> rank(M)   % Rank of matrix M
```

3.1.1 Open-loop dynamics.

A linear state-space model can be created in `MATLAB` using the command `ss`, whilst the step response can be simulated using the command `step`. The linear model from Exercise 3 can thus be simulated with commands by issuing commands such as

```
>> sysc = ss(A,B,C,D); step(sysc); hold on; % Cont. step resp.
>> sysd = c2d(sysc,Ts); step(sysd);         % Discr. step resp.
```

where the second line exemplifies how a continuous-time system can be translated in `MATLAB` into its discrete-time counterpart, given the sampling time T_s. In order to simulate the *nonlinear* system from Exercise 2, however, we construct a `Simulink` model as in Figure 4.

Start `Simulink` by issuing the command

```
>> simulink
```

in `MATLAB`. After `Simulink` has opened, choose `Blank Model` to start creating a new model. Objects from the `Library Browser` can now enter the new model by dragging objects using the mouse. The blocks you need reside in common `Simulink` libraries. Blocks are connected by left clicking and dragging the pointer from the outlet of a block to the inlet of an another block.

Note. `Simulink` has access to `MATLAB`'s workspace, so it is recommended to keep all numerical constants assigned to parameters in a script, and write the parameter names in the blocks (see for example the gain block in Figure 4 where K_q/A is used instead of the corresponding numerical value).

Exercise 6 `Simulink` model of the total (nonlinear) system.

Construct a `Simulink` model of the nonlinear system according to Figure 4 and save it to the local hard drive.

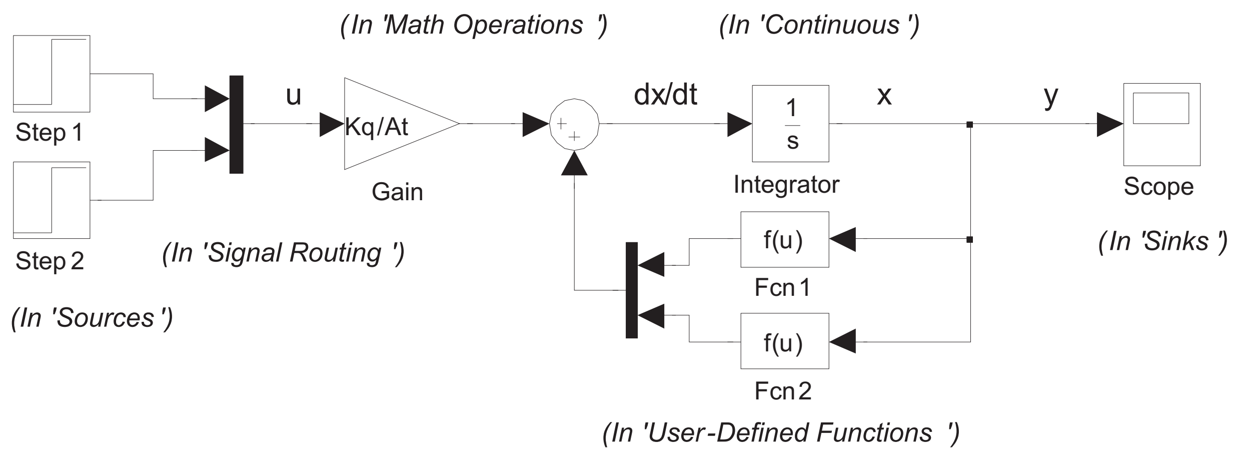

Figure 4: An example of how the nonlinear state-space model can be modeled in `Simulink`. The tank area is denoted with At instead of A to avoid the name clash with the matrix **A** from the linearized model. The italicized text within parentheses tells in which `Simulink` library the blocks reside.

Double-click on the function blocks and enter the appropriate functions. Double-click on step blocks to set their initial values to u_{10} and u_{20} you calculated for the operating points, and their final values to something different with an appropriate step time. Also, double-click on the integrator to set the initial value to the operating point levels (`[h10;h20]`) from Exercise 3. Set the simulation stop time to 300 s (the default value is 10.0 s). Run the simulation by pressing the `Play` button and study the output by double-clicking the `Scope` block.

Make a sketch of the step response in Figure 5 and denote the approximate time constant of the system. Compare with the time constant of the linear model.

Figure 5: Sketch a step response from Exercise 6 in this figure and denote the time constant of the system.

4 Controller design.

You have created a nonlinear and a linear model of the process and have verified some properties of the models using simulations. You are now ready to design controllers for the process. For this purpose, a `Simulink` model with the appropriate control structure is available on the course home page, named controlstructureforassignment.mdl. It is recommended that you download it to save time.

4.1 Simple PI design.

Exercise 7 PI controller for the linearized system model.

Assume that the flow q_{12} between the tanks can be neglected so that the process consists of two identical SISO processes. Determine the transfer function of a linearized model of the resulting process around the operating point $h_\mathrm{i} = 14\,\mathrm{cm}$.

Design a PI controller according to the *lambda-tuning method* from Appendix A. Use $\lambda = \tau/10$. (A rather small selected value for λ is motivated by the time constraints in the experimental part.)

Write down your parameter values and test the resulting controller on the nonlinear process in `Simulink`. Comment on the results.

4.2 MIMO design with linear quadratic control (LQR).

The purpose of the stationary linear quadratic control is to minimize the cost criterion

$$J = \int_0^\infty \mathbf{x}^T(t)\mathbf{Q}_\mathrm{x}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{Q}_\mathrm{u}\mathbf{u}(t)\,dt \tag{8}$$

with respect to the input signal $\mathbf{u}(t)$, where $\mathbf{Q}_\mathrm{x} \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{Q}_\mathrm{u} \in \mathbb{R}^m \times \mathbb{R}^m$ are called *penalty or cost matrices*. The penalty matrices, which are chosen by the control designer, determine the performance of the closed-loop system. Therefore, $\mathbf{Q}_\mathrm{x}$ and $\mathbf{Q}_\mathrm{u}$ are sometimes referred to as *design parameters*. The individual values of $\mathbf{Q}_\mathrm{x}$ and $\mathbf{Q}_\mathrm{u}$ have little significance. It is instead their relation to each other that determines the process behavior. (Why?) We therefore assume the matrices to be of the form

$$\mathbf{Q}_\mathrm{x} = \rho \mathbb{I}_\mathrm{n} \,, \qquad \mathbf{Q}_\mathrm{u} = \mathbb{I}_\mathrm{m} \,. \tag{9}$$

Hence, $\rho \geq 0$ becomes the only design parameter.

The criterion Equation (8) is natural to minimize when the reference is zero, and the resulting control law then becomes a state feedback, $\mathbf{u} = -\mathbf{L}\mathbf{x}$.

However, when the reference is non-zero, it seems natural to add a term to the control signal such that $\mathbf{u} = -\mathbf{L}\mathbf{x} + \mathbf{K}_\mathrm{r} r$. The value of the prefilter gain matrix $\mathbf{K}_\mathrm{r}$ is then chosen such that the static gain of the closed-loop system becomes unity.

Exercise 8 Prefilter calculation.

Determine $\mathbf{K}_\mathrm{r}$ as a function of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{L}$, such that the closed-loop system's static gain becomes unity. **Hint:** Set up the state-space equations for the closed-loop system and solve for $\mathbf{K}_\mathrm{r}$ given that the system is at steady state and that $y = r$ is desired.

Exercise 9 LQR design.

Study the help documentation for the `MATLAB` command `lqr`. Then carry out the design of a linear quadratic controller by issuing a command of the type

```
>> Lc = lqr(ss(A,B,C,D),Qx,Qu);
```

Test different values for the penalty ($\rho = 1, 2, 5, 10, 40$) and choose a value which gives a good compromise between settling time and control activity. Subsequently, determine the approximate time constant of the closed-loop system from a step-response graph. The closed-loop step response can be simulated with (Why?)

```
>> step(ss(A-B*Lc,B*Kr,C-D*Lc,D*Kr)) % step resp.: output signals
>> step(ss(A-B*Lc,B*Kr,-Lc,Kr))      % step resp.: control signals
```

Finally, test the resulting controller in `Simulink` on the nonlinear model and comment on its performance.

4.3 Linear quadratic control with integral action (LQI).

The model which the LQ control law is based on does not perfectly describe the nonlinear model, which results in minor steady-state errors. Disturbance flows are not completely compensated for either. It is therefore suitable to include integral action in the linear quadratic controller.

Exercise 10 LQI design.

Carry out the design of a linear quadratic controller with integral action according to Appendix B by issuing commands of the type

```
>> Atot = [A zeros(2);...
                -C zeros(2)];
>> Btot = [B;...
            zeros(2)];
>> Qtot = blkdiag(rho*eye(2),eta*eye(2));
>> Ltot = lqr(Atot, Btot, Qtot, Qu);
```

Test different penalties for ρ (`rho`) and η (`eta`). **Hint:** η needs to be relatively small to maintain robustness.

Test the resulting controller on the nonlinear model by augmenting your `Simulink` model according to Figure 6. For a faster reference response, introduce a prefilter gain $\mathbf{K}_\mathrm{r}$ in your `Simulink` model according to Figure 9 in Appendix D.

Exercise 11 `MATLAB` commands for discretization.

In preparation for the experimental part, read the help documentation concerning the following `MATLAB` commands: `c2d`, `dlqr`, `lqrd`, `lqed`, `dlqe`. Explain the differences between `dlqr` and `dlqe` and also between `lqrd` and `lqed`.

5 Experiments.

5.1 Startup.

1. Log on to a computer connected to the double-tank process.
2. Start MATLAB.
3. Issue the command

   ```
   >> mimotanklab
   ```

   in MATLAB. It sets you up for the experiments by opening an appropriate Simulink model. Follow the instructions displayed in the MATLAB command window!

   Note that, for your calculations, you need to set the parameter values in the MATLAB file `mimotanklabinit.m`.

5.2 Model and controller verification.

To start out, you will test the accuracy of the model you have been working on at home as well as some of the controllers you have designed. **Remember to translate their parameters to fit a discrete-time framework before you test them.**

Exercise 12 Accuracy of (nonlinear) process model.

Make sure your Simulink model is set to open-loop mode (How?). Test the accuracy of your process model by performing a series of simple step responses near your desired operating point. Compare with your results from Exercise 6 and comment on the model's accuracy.

Exercise 13 Discretization of controllers. Robustness test.

Choose a suitable sampling time T_s and assign it to `Ts` in MATLAB's workspace. Translate and test both your PI-controller and LQ-controller (without integral action) on the real process.

Introduce disturbances from the manual control panel to test the robustness of each controller. Comment and compare with your results from `Part I`.

Note. You need to press *stop* and *play* in `Simulink` in order for new parameter values to take effect.

5.3 Kalman filter design.

In practice, it is rare that all state variables of interest can be measured. A state estimator or *observer* can be used in cases when some state variables cannot be measured, or measurements are unreliable. A `Kalman filter` is an observer which is designed to minimize the variance of the prediction error, $\mathrm{E}\,(\mathbf{x}-\widehat{\mathbf{x}})(\mathbf{x}-\widehat{\mathbf{x}})^T$, based on the model

$$\mathbf{x}(k+1) = \boldsymbol{\Phi}\mathbf{x}(k) + \boldsymbol{\Gamma}\mathbf{u}(k) + \mathbf{v}(k) \tag{10a}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{w}(k)\,, \tag{10b}$$

where $\mathbf{v}$ and $\mathbf{w}$ are Gaussian-distributed process and measurement noise terms with covariances

$$\mathbf{R}_1 = \mathrm{E}\,\mathbf{v}\mathbf{v}^T\,, \qquad \mathbf{R}_2 = \mathrm{E}\,\mathbf{w}\mathbf{w}^T \tag{10c}$$

respectively. The result of the minimization is the Kalman gain $\mathbf{K}$ which determines the dynamics of the optimal one-step ahead predictor

$$\widehat{\mathbf{x}}_{\mathrm{k+1\,|\,k}} = \boldsymbol{\Phi}\widehat{\mathbf{x}}_{\mathrm{k\,|\,k-1}} + \boldsymbol{\Gamma}\mathbf{u}_{\mathrm{k}} + \mathbf{K}(\mathbf{y}_{\mathrm{k}} - \mathbf{C}\widehat{\mathbf{x}}_{\mathrm{k\,|\,k-1}})\,. \tag{11}$$

In Equation (11), the first two terms constitute a prediction of the state based on the process model, previous state estimate, and current input. The last term can be seen as a correction of the prediction based on the measured outputs. The term $\mathbf{y}_{\mathrm{k}} - \mathbf{C}\widehat{\mathbf{x}}_{\mathrm{k\,|\,k-1}}$ is sometimes called the *innovation*, since it holds the new information arising from random components.

The Kalman gain can be calculated in `MATLAB` using the function `lqe` for the continuous-time case, and `dlqe` or `lqed` for the discrete-time case. The covariance matrices in Equation (10c) are then commonly interpreted as *design parameters* which determine the dynamics of the observer.

Exercise 14 Kalman filter design.

Design a Kalman filter under the assumption that only the level in the first tank can be measured. Denote and justify your choices of $\mathbf{R}_1$ and $\mathbf{R}_2$ for the design.

Exercise 15 LQG control.

Test your Kalman filter together with your LQ controller (without integral action) and comment on the results.

Remark. The abbreviation LQG in the title of this document stands for *Linear-Quadratic-Gaussian*. LQG-control concerns minimizing a quadratic criterion under the model Equation (10). Your controller in Exercise 15 is the unique solution to this fundamental optimal control problem! The separation principle guarantees that the observer and regulator can be designed and computed independently.

5.4 Linear quadratic control with integral action.

The model which the linear quadratic control law is based on does not perfectly describe the real process, which results in minor steady-state errors. Disturbance flows are not completely compensated for either. It is therefore suitable to include integral action.

Exercise 16 Inversion lemma.

If time allows, verify that

$$\left(\mathbf{C}\left(\mathbf{I}-\mathbf{\Phi}\right)^{-1}\mathbf{\Gamma}\right)^{-1}+\mathbf{L}_{\mathrm{d}}\left(\mathbf{I}-\mathbf{\Phi}\right)^{-1}\mathbf{\Gamma}\left(\mathbf{C}\left(\mathbf{I}-\mathbf{\Phi}\right)^{-1}\mathbf{\Gamma}\right)^{-1}=\left(\mathbf{C}\left(\mathbf{I}-\mathbf{\Phi}+\mathbf{\Gamma}\mathbf{L}_{\mathrm{d}}\right)^{-1}\mathbf{\Gamma}\right)^{-1}$$

for example by applying the matrix inversion lemma which states that

$$\left(\mathbf{T}+\mathbf{U}\mathbf{W}\mathbf{V}\right)^{-1}=\mathbf{T}^{-1}-\mathbf{T}^{-1}\mathbf{U}\left(\mathbf{W}^{-1}+\mathbf{V}\mathbf{T}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}\mathbf{T}^{-1}.$$

This result can be used to appropriately choose the prefilter gain in the Simulink model used in the lab as $K_r=\left(\mathbf{C}\left(\mathbf{I}-\mathbf{\Phi}+\mathbf{\Gamma}\mathbf{L}_{\mathrm{d}}\right)^{-1}\mathbf{\Gamma}\right)^{-1}$.

Exercise 17 Integral action.

Use both measured signals and augment the `Simulink` model with integral action according to Appendix C. Choose an appropriate prefilter $\mathbf{K}_{\mathrm{r}}$ according to the result from Appendix E to speed up the reference response. Test the resulting controller on the process and comment on its performance.

Exercise 18 Kalman filter with integral action.

If time allows: What is a reasonable way to use the Kalman filter from Exercise 14 together with integral action from Exercise 17 if one tank level cannot be measured?

Implement and test this final controller. Does the system achieve integral action for both tank levels?

A Lambda-tuning control design for a first-order process.

The lambda-tuning method is a simple tuning method popular in process industry. It is based on a step response of the open-loop process and works as follows:

1. Model the process as a first-order system with delay L:

   $$G_\mathrm{p}(s) = \frac{K_\mathrm{p} e^{-Ls}}{1 + \tau s}$$

   and identify parameters based on a simple step response.

2. Choose a desired closed-loop behavior:

   $$G_\mathrm{cl}(s) \approx \frac{e^{-Ls}}{1 + \lambda s}$$

   Typically, $0.5\tau \leq \lambda \leq 3\tau$.

3. Choose a PI-controller structure:

   $$G_\mathrm{r}(s) = K\left(1 + \frac{1}{T_\mathrm{i} s}\right)$$

4. Based on the identified parameters, select the PI-controller parameters as

   $$K = \frac{\tau}{K_\mathrm{p}(\lambda + L)}$$

   and

   $$T_\mathrm{i} = \tau$$

   .

B State feedback with integral action in continuous-time.

State feedback with integral action follows the same principles as ordinary state feedback. To reach integral action, the state vector is augmented with the integral of the measurement error

$$x_{\mathrm{I}} = \int_0^t (r - y)\, d\tau$$

The augmented state-space model becomes

$$\frac{d}{dt}\begin{bmatrix} \mathbf{x} \\ x_{\mathrm{I}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}_{\mathrm{tot}}} \begin{bmatrix} \mathbf{x} \\ x_{\mathrm{I}} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}_{\mathrm{tot}}} \mathbf{u} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} r\,.$$

A state-feedback controller can be calculated based on the minimization of

$$\int_0^\infty \begin{bmatrix} \mathbf{x}^T & x_{\mathrm{I}}^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\mathrm{x}} & \mathbf{0} \\ \mathbf{0} & Q_{\mathrm{I}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_{\mathrm{I}} \end{bmatrix} + \mathbf{u}^T \mathbf{Q}_{\mathrm{u}} \mathbf{u}\, dt \tag{12}$$

with respect to $\mathbf{u}$. In MATLAB, this is carried out with a command of the type

```
>> Ltot = lqr(Atot,Btot,blkdiag(Qx,QI),Qu);
```

The resulting control law can be implemented as in Figure 6. For the controller to remain robust, the additional closed-loop poles must be made relatively slow.

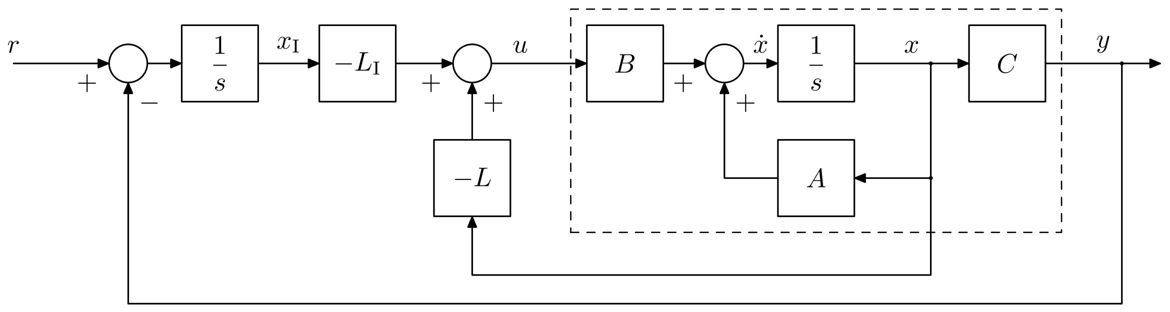

Figure 6: State feedback with integral action. The matrices $\mathbf{L}$ and $\mathbf{L}_{\mathrm{I}}$ form the total feedback gain $\mathbf{L}_{\mathrm{tot}} = [\,\mathbf{L} \quad \mathbf{L}_{\mathrm{I}}\,]$.

C State feedback with integral action in discrete-time.

State feedback with integral action follows the same principles as ordinary state feedback. To reach integral action, the model is augmented with the integral sum of the measurement error

$$x_{\mathrm{I}} = \sum_{j=0}^{k} (r - y)$$

The augmented state-space model becomes

$$\begin{bmatrix} \mathbf{x}(k+1) \\ x_{\mathrm{I}}(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{\Phi} & \mathbf{0} \\ -\mathbf{C} & \mathbf{I} \end{bmatrix}}_{\mathbf{\Phi}_{\mathrm{tot}}} \begin{bmatrix} \mathbf{x}(k) \\ x_{\mathrm{I}}(k) \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{\Gamma} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{\Gamma}_{\mathrm{tot}}} \mathbf{u}(k) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} r(k)\,.$$

A state-feedback controller can be calculated based on the minimization of

$$\sum_{k=0}^{\infty} \left(\begin{bmatrix} \mathbf{x}^T & x_{\mathrm{I}}^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\mathrm{x}} & \mathbf{0} \\ \mathbf{0} & Q_{\mathrm{I}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_{\mathrm{I}} \end{bmatrix} + \mathbf{u}^T \mathbf{Q}_{\mathrm{u}} \mathbf{u} \right) \tag{13}$$

with respect to **u**. In `MATLAB`, this is carried out with a command of the type

```
>> Ldtot = dlqr(Phitot,Gammatot,blkdiag(Qx,QI),Qu);
```

The resulting control law can be implemented as in Figure 7. For the controller to remain robust, the additional closed-loop pole must be made relatively slow.

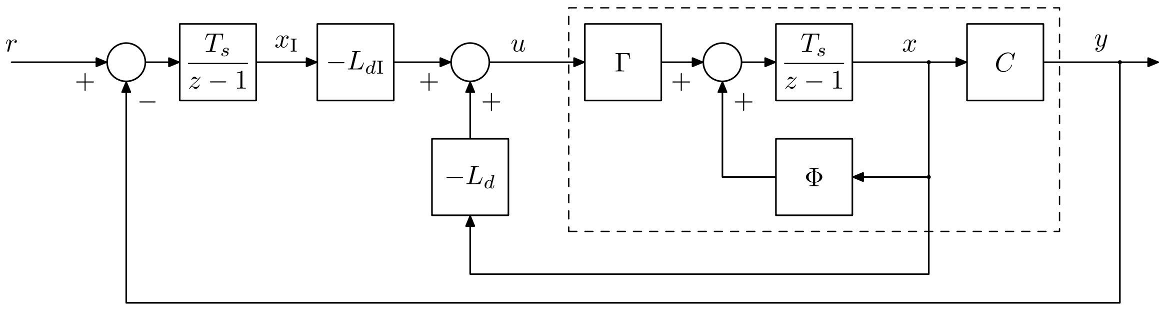

Figure 7: State feedback with integral action. The matrices $\mathbf{L}_{\mathrm{d}}$ and $\mathbf{L}_{\mathrm{d,I}}$ form the total feedback gain $\mathbf{L}_{\mathrm{d,tot}} = [\,\mathbf{L}_{\mathrm{d}} \quad \mathbf{L}_{\mathrm{d,I}}\,]$.

D Integral action with faster reference response - continuous time

The control law arising from minimizing the criterion (12) is optimal for regulation to the origin if the plant has a non-zero initial state. The only reason it manages to track non-zero references is because of integral action. This implies that reference tracking becomes slow.

One natural way to speed up the response is to let each *constant* reference level correspond to a steady-state level $(\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r})$ such that $y = r$ for the linear model before augmentation and then minimize

$$\int_0^\infty \mathbf{x}_\delta^T \mathbf{Q}_\mathrm{x} \mathbf{x}_\delta + x_\mathrm{I}^T Q_\mathrm{I} x_\mathrm{I} + \mathbf{u}_\delta^T \mathbf{Q}_\mathrm{u} \mathbf{u}_\delta \, dt \tag{14}$$

where

$$\mathbf{x}_\delta = \mathbf{x} - \mathbf{x}_\mathrm{r}\,, \qquad \mathbf{u}_\delta = \mathbf{u} - \mathbf{u}_\mathrm{r}\,, \qquad x_\mathrm{I} = \int_0^t (r - y)\, d\tau\,. \tag{15}$$

Minimizing the criterion (14) implies optimal regulation to the desired reference level, making it a more suitable criterion to minimize than (12) when reference tracking is desired. The model $(\mathbf{A}_\mathrm{tot}, \mathbf{B}_\mathrm{tot})$ exploited in the minimization of (14) remains the same, so the calculation of the feedback matrix $\mathbf{L}_\mathrm{tot}$ also remains the same. From (15) we then see that the only thing that happens is that the block diagram in Figure 6 changes into the structure in Figure 8, where

$$\mathbf{K}_{\mathbf{u}_\mathrm{r}} = -(\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\,, \qquad \mathbf{K}_{\mathbf{x}_\mathrm{r}} = \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}$$

have been determined from

$$\mathbf{u}_\mathrm{r} = -(\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} r\,, \qquad \mathbf{x}_\mathrm{r} = \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} r\,,$$

which are required to hold to have $r = y$ at steady state.[1]

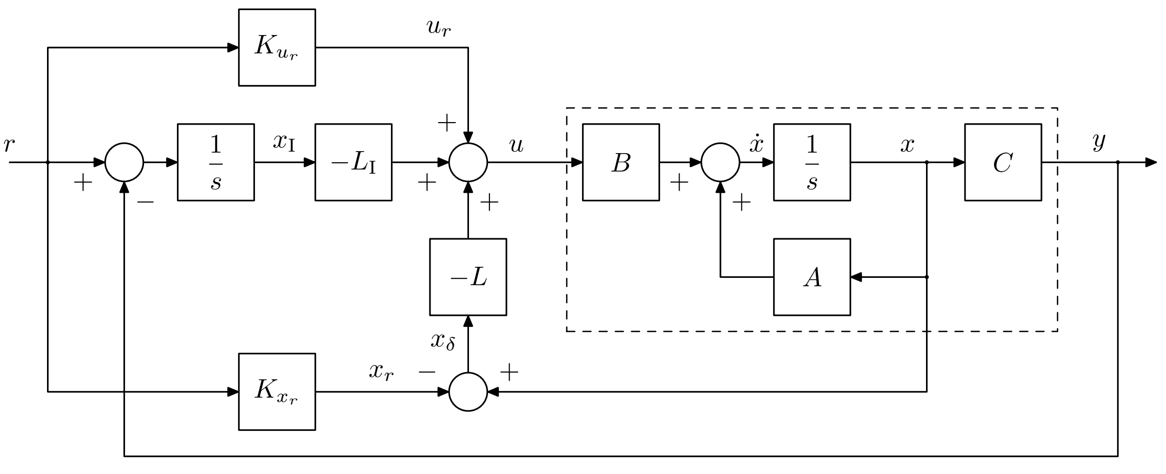

Figure 8: Integral action for faster reference response. Compare with Figure 6.

The calculation of $\mathbf{K}_\mathrm{r}$ in Exercise 8 corresponds to speeding up the reference response *exactly* in the way described above. It is a good exercise to verify this, for example, by applying the *matrix inversion lemma*. With this observation, a faster reference response can simply be achieved by adding a prefilter term $\mathbf{K}_\mathrm{r} r$ to the control signal, as in Figure 9, where $\mathbf{K}_\mathrm{r}$ is calculated in the same manner as in Exercise 8.

[1]If $\mathbf{A}$ is not invertible, it is natural to desire that a unique solution exists to $\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}\begin{bmatrix}\mathbf{x}_\mathrm{r}\\ \mathbf{u}_\mathrm{r}\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ r\end{bmatrix}$ from where $\mathbf{K}_{\mathbf{u}_\mathrm{r}}$ and $\mathbf{K}_{\mathbf{x}_\mathrm{r}}$ can be calculated. One also gets $\mathbf{K}_\mathrm{r} = \begin{bmatrix}\mathbf{L} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}$.

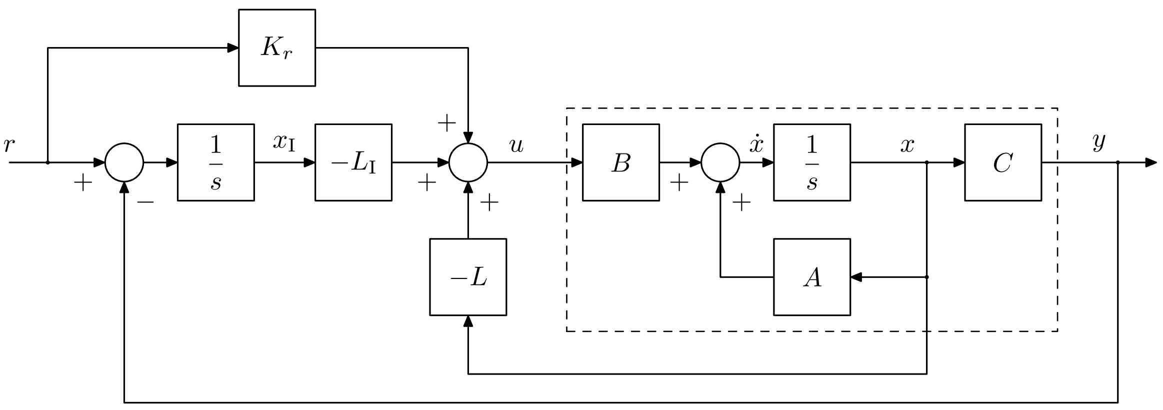

Figure 9: Alternative implementation to reach a faster reference response, where $\mathbf{K}_{\mathrm{r}}$ is calculated in the same manner as in Exercise 8. Compare with Figure 6 and Figure 8.

E Integral action with faster reference response - discrete time

The control law arising from minimizing the criterion (13) is optimal for regulation to the origin if the plant has a non-zero initial state. The only reason it manages to track non-zero references is because of integral action. This implies that trajectory following becomes slow.

One natural way to speed up the response is to let each *constant* reference level correspond to a steady-state level $(\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r})$ such that $r = y$ for the linear model before augmentation and then minimize

$$\sum_{k=0}^{\infty} \left(\mathbf{x}_\delta^T \mathbf{Q}_\mathrm{x} \mathbf{x}_\delta + x_\mathrm{I}^T Q_\mathrm{I} x_\mathrm{I} + \mathbf{u}_\delta^T \mathbf{Q}_\mathrm{u} \mathbf{u}_\delta \right) \tag{16}$$

where

$$\mathbf{x}_\delta = \mathbf{x} - \mathbf{x}_\mathrm{r}\,, \qquad \mathbf{u}_\delta = \mathbf{u} - \mathbf{u}_\mathrm{r}\,, \qquad x_\mathrm{I} = \sum_{j=0}^{k} (r - y)\,. \tag{17}$$

Minimizing the criterion (16) implies optimal regulation to the desired reference, which means it is a more suitable criterion to minimize than Equation (13) when reference tracking is desired. The model $(\mathbf{\Phi}_\mathrm{tot}, \mathbf{\Gamma}_\mathrm{tot})$ exploited in the minimization of (16) remains the same, so nothing new happens for the calculation of the feedback matrix $\mathbf{L}_\mathrm{tot}$. From (17) we then see that the only thing that happens is that the block diagram in Figure 7 changes into the structure in Figure 10, where

$$\mathbf{K}_{\mathbf{u}_\mathrm{r}} = -(\mathbf{C}\,(\mathbf{I} - \mathbf{\Phi})^{-1}\,\mathbf{\Gamma})^{-1}\,, \qquad \mathbf{K}_{\mathbf{x}_\mathrm{r}} = (\mathbf{I} - \mathbf{\Phi})^{-1}\,\mathbf{\Gamma}\left(\mathbf{C}\,(\mathbf{I} - \mathbf{\Phi})^{-1}\,\mathbf{\Gamma}\right)^{-1}$$

have been determined from

$$\mathbf{u}_\mathrm{r} = -\left(\mathbf{C}\,(\mathbf{I} - \mathbf{\Phi})^{-1}\,\mathbf{\Gamma}\right)^{-1} r\,, \qquad \mathbf{x}_\mathrm{r} = (\mathbf{I} - \mathbf{\Phi})^{-1}\,\mathbf{\Gamma}\left(\mathbf{C}\,(\mathbf{I} - \mathbf{\Phi})^{-1}\,\mathbf{\Gamma}\right)^{-1} r\,,$$

which are required to hold to have $r = y$ at steady state.

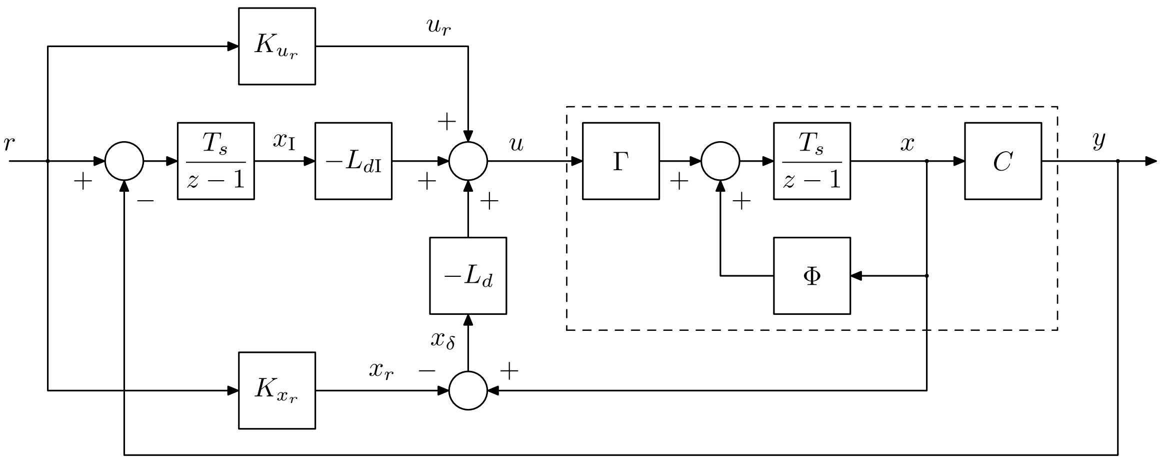

Figure 10: Integral action for faster reference response. Compare with Figure 7.