FRTN55 Automatic Control, Advanced Course

Handin Assignment on

Control Design by Convex Optimization[1]

Department of Automatic Control
Lund University

Administrative Information

You may work alone or in pairs. Submit the Matlab code (i.e., the completed `servo_optimize.m`) via Canvas no later than October 19. (If you work in a pair, you only need to submit the solution once.) The code should answer all the questions in the assignment and produce an optimal controller that satisfies all the stated constraints. Add comments in the code to answer specific questions in the tasks.

Part 1 of the assignment can be performed after Lecture 13: The Youla Parametrization. It is part of Exercise 12 and requires Matlab.

Part 2 of the assignment can be performed after Lecture 14: Control Design Using Convex Optimization. It constitutes Exercise 13 and requires Matlab with the CVX toolbox. CVX is already installed on the lab computers—type `setup_cvx` in Matlab to activate it. If you want to run CVX on your own computer, it can be downloaded from `http://cvxr.com`.

This assignment is a Pass or Fail. Discussing theory or Matlab/CVX issues with your classmates is fine but the actual tasks must be performed by you (two) alone.

If you need any help with the assignment, Matlab-related or otherwise, ask in the exercise sessions or on Discord

PART 1: The Control Problem. Youla Parametrization

The Process

In the assignment, we revisit the flexible linear servo process from Lab 1, see Figure 1. The goal is to control the position p_2 of the second mass using the motor force F_m acting on the first mass. Balance equations for the forces in the system give the following continuous-time model:

$$m_1 \frac{d^2 p_1}{dt^2} = -d_1 \frac{dp_1}{dt} - k(p_1 - p_2) + F_m$$

$$m_2 \frac{d^2 p_2}{dt^2} = -d_2 \frac{dp_2}{dt} + k(p_1 - p_2)$$

Introducing the state vector $x = [\, p_1 \quad \dot{p}_1 \quad p_2 \quad \dot{p}_2 \,]^T$, the control signal $u_0 = \frac{F_m}{k_m}$, and the controlled output $z_0 = k_y p_2$, the system can be written in state-space form as

$$\dot{x}(t) = Ax(t) + Bu_0(t)$$
$$z_0(t) = Cx(t)$$

[1] Written by Anton Cervin

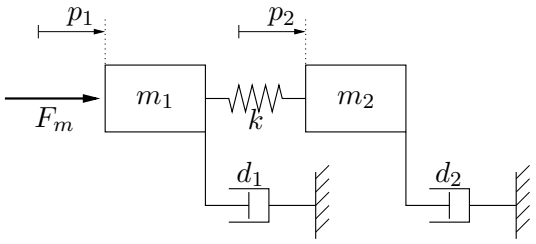

Figure 1 The flexible linear servo process

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m_1} & -\frac{d_1}{m_1} & \frac{k}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_2} & 0 & -\frac{k}{m_2} & -\frac{d_2}{m_2} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{k_m}{m_1} \\ 0 \\ 0 \end{bmatrix}, \quad C = [\,0 \quad 0 \quad k_y \quad 0\,]$$

In the assignment, we will use the following constants and coefficients:

$$\begin{aligned} m_1 &= 2.3 \text{ kg} & m_2 &= 2.1 \text{ kg} \\ d_1 &= 3.2 \text{ N/m/s} & d_2 &= 8.6 \text{ N/m/s} \\ k &= 400 \text{ N/m} & k_m &= 0.443 \text{ N/V} \\ k_y &= 280 \text{ V/m} \end{aligned}$$

With the parameters above, the transfer function from u_0 to y_0 is given by

$$P_0(s) = \frac{10272}{s(s + 2.695)(s^2 + 2.791s + 362.6)}$$

Controller Structure and Generalized Plant Model

We want to design a two-degree-of-freedom controller for the flexible servo according to the structure shown in Figure 2(a). Our main objective is to minimize the influence of the load disturbance d, which represents both external forces and unmodeled nonlinear phenomena such as friction. At the same time, we want the controlled output z_0 to follow the reference r while not using excessive motor commands u. Finally, the measurement noise n should be sufficiently attenuated.

To apply the Youla parametrization, we first reformulate the controller structure in Figure 2(a) into the general feedback loop in Figure 2(b). Define the exogenous

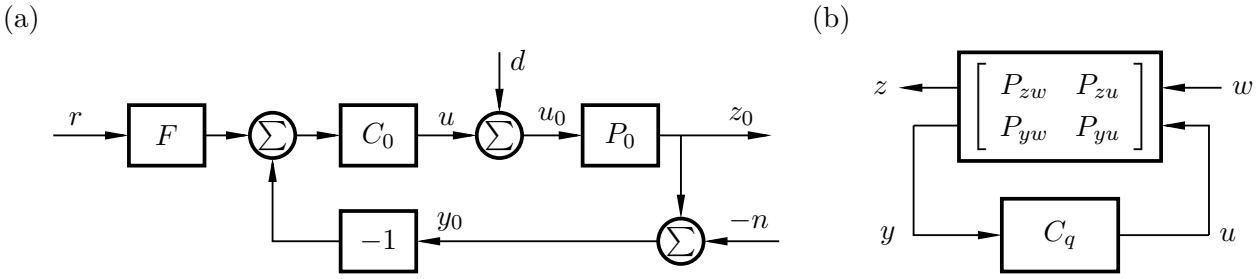

Figure 2 (a) Two-degree-of-freedom controller structure, (b) general feedback loop with controller C_q and generalized plant model P.

signal vector $w = \begin{pmatrix} r & n & d \end{pmatrix}^{\mathrm{T}}$, the performance output vector $z = \begin{pmatrix} z_0 & u \end{pmatrix}^{\mathrm{T}}$ and the controller input vector $y = \begin{pmatrix} r & y_0 \end{pmatrix}^{\mathrm{T}}$. (The scalar control signal u is the same in both diagrams.)

Task 0. Open the Matlab script `servo_optimize.m` and fill in your name(s) at the top.

Task 1. We will work in discrete time in this assignment. At the start of the script we define the continuous-time process model by executing `servo_model`. Following this, insert code to calculate an equivalent discrete-time system

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_{0,k} \\ y_{0,k} &= C x_k \end{aligned}$$

assuming zero-order hold sampling and the interval $h = 0.05$. Use the command `c2d` and store the result in the variables `Phi` and `Gam`. (The C matrix is the same in continuous and discrete time.)

Task 2. As explained in Lecture 13, a state-space description of the generalized plant model $P = \left(\begin{smallmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{smallmatrix}\right)$ for the system in Figure 2 is given by

$$\begin{aligned} x_{k+1} &= \Phi\, x_k + \begin{pmatrix} \Gamma_w & \Gamma_u \end{pmatrix} \begin{pmatrix} w_k \\ u_k \end{pmatrix} \\ \begin{pmatrix} z_k \\ y_k \end{pmatrix} &= \begin{pmatrix} C_z \\ C_y \end{pmatrix} x_k + \begin{pmatrix} D_{zw} & D_{zu} \\ D_{yw} & D_{yu} \end{pmatrix} \begin{pmatrix} w_k \\ u_k \end{pmatrix} \end{aligned}$$

where

$$\Gamma_w = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \Gamma \end{pmatrix} \qquad \Gamma_u = \Gamma$$

$$C_z = \begin{pmatrix} C \\ \mathbf{0} \end{pmatrix} \qquad D_{zw} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad D_{zu} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$C_y = \begin{pmatrix} \mathbf{0} \\ C \end{pmatrix} \qquad D_{yw} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \qquad D_{yu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Enter the generalized plant as the discrete-time state-space system `Pgen` in Matlab. Use the command `ss`. The resulting system should have four states, four inputs, four outputs, and the sampling time $h = 0.05$.

Task 3. To find the Youla parametrization, we shall first stabilize the generalized plant using an observer-based controller as shown in Figure 3. Since we will later optimize the closed loop using the Youla parameter Q, we can in principle apply *any* stabilizing LQG controller in this step.

- Calculate a Kalman filter gain vector `L` for the generalized plant using `dlqe`. Assume an identity matrix for the noise input matrix G (matching the size of x) and identity matrices of appropriate dimensions for R_1 and R_2 (matching the sizes of x and y, respectively). The resulting `L` should be of size 4×2.

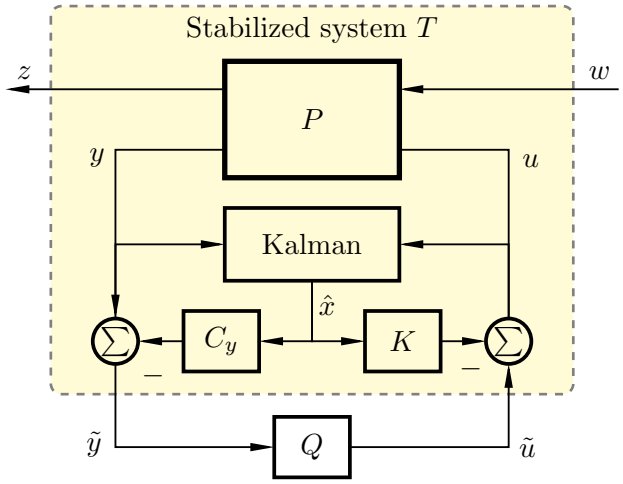

Figure 3 The unstable generalized plant P can be stabilized using an LQG controller. The Youla parametrization can then be applied to the stabilized system T.

- Calculate a state feedback gain vector `K` for the generalized plant using `dlqr`. Let Q_1 be an identity matrix of appropriate size and let $Q_2 = 10^{-6}$. The resulting `K` should be of size 1×4.

Next we formulate the stabilized system $T = \begin{pmatrix} T_{zw} & T_{z\tilde{u}} \\ T_{\tilde{y}w} & T_{\tilde{y}\tilde{u}} \end{pmatrix}$ in the yellow box in Figure 3 as a discrete-time state-space system (see Lecture 13—this step is already prepared in the script). Check how well the stabilized system is able to meet the closed-loop time-domain specifications by executing `plotspecs(Tzw)`. This command plots all the step responses of T_{ij}, from each exogenous input w_j to each performance output z_i. Design specifications similar to the ones in Lab 1 are shown as dashed red lines.

Task 4. Play around with some constant Youla parameters `Q = [q1 q2]`, where `q1` and `q2` are scalars, and see how the resulting closed-loop system $G_{zw} = T_{zw} + T_{z\tilde{u}}\, Q\, T_{\tilde{y}w}$ changes. Check the results against the time-domain specifications using `plotspecs(Gzw)`. Does it seem to be possible to meet the constraints using this type of simple "manual optimization"? **Discuss your results by writing comments in the code.**

PART 2: Convex Optimization

As you probably saw in Task 4, a constant Q parameter is not sufficient to solve the design problem at hand. Instead, we will use optimization and search for the best Youla parameter in the form of an FIR filter (see Lecture 14),

$$Q(z) = \sum_{i=0}^{N_q-1} \begin{pmatrix} \frac{q_{1,i}}{z^i} & \frac{q_{2,i}}{z^i} \end{pmatrix}$$

where N_q is the number of basis functions and $q_{j,i}$ are the scalar parameters to be optimized by CVX. We start with $N_q = 30$ terms in Q but will experiment with this value later. Given a $Q(z)$, a number of interesting properties of the closed-loop system $G_{zw} = T_{zw} + T_{z\tilde{u}}\, Q\, T_{\tilde{y}w}$ can now be evaluated, for instance,

- step responses of G_{zw} for the time points $t = [0, h, \ldots, N_t h]$. In the script, these are calculated and stored in the three-dimensional variable `Gzw_stepresp`. The first index is the performance output, the second index is the exogenous input, and the third index is the time step. We will use $N_t = 120$.
- frequency responses of G_{zw} for the frequency points $\omega = [0, \pi/h/N_\omega, \ldots, \pi/h]$. In the script, these are calculated and stored in the three-dimensional variable `Gzw_freqresp`. The first index is the performance output, the second index is the exogenous input, and the third index is the frequency step. We will use $N_\omega = 120$.

The response at each time/frequency point is an affine function of the optimization parameters $q_{j,i}$, which allows us to formulate various convex optimization problems involving the responses in the objective function or in the constraints.

Task 5. Our primary design goal is to reduce the effect of process disturbances on the controlled output. For this purpose, we instruct CVX to optimize the sum-squared step response from d to z_0:

```
minimize sum_square(Gzw_stepresp(?,?,:))
```

Replace the question marks by the appropriate indexes into `Gzw_stepresp` to capture our design objective. Run the optimization script and verify that the optimized controller is able to (almost) zero out the load disturbance response.

Note: Check the CVX printouts for the finishing status of the solver. After a successful run, the value of the objective function (i.e., the sum-squared load disturbance response) is returned in the variable `cvx_optval`. Besides the closed-loop step responses plotted in figure 1, you can also inspect the Bode magnitude diagrams of the closed-loop system and the Bode diagram of the optimized controller in figures 2 and 3. The optimized controller is stored in the variable `ctrl_opt`.

Task 6. As seen in the responses, the optimal controller in Task 5 is extremely aggressive. Add (i.e., uncomment) a constraint that puts an upper limit on the sum-squared step response from the measurement noise n to the control signal u:

```
sum_square(Gzw_stepresp(?,?,:))*h <= LIMIT
```

Replace the question marks by the appropriate indexes into `Gzw_stepresp` to activate the relevant constraint. Experiment with different values of `LIMIT` and rerun the optimization until the magnitude of the step response from n to u does not exceed 5 (approximately).

Task 7. Now we will add time-domain constraints similar to the ones specified in the manual to Lab 1. The upper and lower limits of four different input–output responses are given as vectors and can be activated by adding the appropriate lines. For example,

```
constr.Gz0r_lower <= squeeze(Gzw_stepresp(?,?,:)) <= constr.Gz0r_upper
```

specifies constraints on the step response from r to z_0. Again, you need to replace the question marks by the appropriate indexes into `Gzw_stepresp` for each constraint. The optimization should be able to find a solution that satisfies all the constraints.

Task 8. To make the control system more robust (and also to reduce some of the oscillations), we want to add a constraint on the maximum sensitivity. The sensitivity function S is however not available as one of the closed-loop transfer functions of G_{zw}. Using the fact that $S = 1 - T$, where T is the complementary sensitivity function, this constraint can instead be expressed in the form

```
abs(1 - Gzw_freqresp(?,?,:)) <= MS
```

Insert the correct indexes to express the sensitivity constraint and experiment with different values of `MS`. How small can you make `MS` without making the optimization problem infeasible? What drawback can you notice in the results when making `MS` smaller (think about Bode's sensitivity integral)? **Answer these questions by writing comments in the code.** When done experimenting, set the sensitivity limit to 1.6.

Task 9. Finally, we want to experiment with N_q—the number of basis functions in the Q parameter. How small can you make `Nq` and still obtain a solution that satisfies all of the stated constraints? How much worse is the solution in this case (look at the time responses and the value of `cvx_optval`)? Compared to $N_q = 30$, can you notice any difference if you try $N_q = 60$ or $N_q = 120$? Why is it pointless to try even larger values for the current setup? **Answer these questions by writing comments in the code.**