Homework 1 in EL2450 Hybrid and Embedded Control Systems

Daniel Zanelli
19930125-8779
dezs@kth.se

Michele Petrone
19980904-T610
michele.petrone.automazione@gmail.com

January 29, 2023

Task 1

The gain Tap models the opening of the tap. If it is 0, the tap is closed and water can only flow to the lower tank. If it is 1 the water flows equally to the lower tank and outside.

Task 2

The code declaring the transfer functions for the upper and lower tanks is:

```
uppertank = tf(k_tank,[1 Tau]);
lowertank = tf(gamma_tank,[1 Tau*gamma_tank]);
```

Task 3

The reference signal is a step, delayed by 25 seconds and with amplitude of 10.
Y_{ss} is the Steady State Output and is the value around which the system is linearized.
U_{ss} is the Steady State Input and is the constant input we need to apply our system to get Y_{ss} as an output.

Task 4

```
[K, Ti, Td, N]=polePlacePID(chi, omega0, zeta, Tau, gamma_tank, k_tank);
s=tf('s');
F=K*(1+(1/(Ti*s)+(Td*N*s/(s+N))))
```

With the first command we get the PID values based on the system and step response requirements, then we build the PID transfer function.

Task 5

The rise time,T_r, is the time it takes to get from 10% to 90% of the response signal. The overshoot value is the maximum value the signal reaches with respect to the step. The setting time, T_{set}, is the time it takes from the start of the step signal to the point it is in the 2% of the asymptote.

χ	ζ	ω_0	T_r	M	T_{set}
0.5	0.7	0.1	$8.2s$	14.5%	$45s$
0.5	0.7	0.2	$5s$	34.6%	$37s$
0.5	0.8	0.2	$5.8s$	31.7%	$26s$

Table 1: System parameters

So out of these three options, the one that gives the best results would be the third one: $\chi = 0.5$ $\zeta = 0.8$ and $\omega_0 = 0.2$. This is because it is the only one that fits the criteria of having a T_{set} lower than $30s$.

Task 6

We simply use the following Matlab code:

```
getGainCrossover(F*G,1)
ans = 0.362
```

With $F * G$ being the open-loop system (series of controller and plant) and $1 = 10^0$ being the gain we need to cross over.

Task 7

We arrived to the solution by trying different values of the sampling time, T_s, from 0 to 1 and looking at the overshoot value and checking it was under 35%. We iterated in a binary search style and found that the values of $T_s < 0.72$ give values that fit to the requirements (particularly $M = 34.6\%$ at $T_s = 0.72$), while values higher than 0.72 give an overshoot higher than 35%. So we can consider this value as the critical point, at which the other results from the simulation are $T_r = 4.7s$ and $T_s = 25s$.

Task 8

For this solution we have created the transfer function $F_d = c2d(F, T_s, ZOH)$ and gotten the denominator and numerator by using $tfdata(Fd,' v')$. Then we obtained the discretized values of A, B, C, D by using $tf2ss(num, den)$.
By using the same $T_s = 0.72$ as before, we get better results, with an overshoot of 25.6%, $T_s = 24s$ and $T_r = 5.2s$.

Task 9

The sampling frequency has to be 4 to 10 times the crossover frequency. Considering the crossover time was 0.362, we arrive at a sampling time of $T_s = 0.276s$ at 10 times (conservative) and a sampling time of $T_s = 0.69s$ for 4 times.

Task 10

At the minimum required sampling frequency (4 times the cross frequency), we get an overshoot within the range, but the settling time over the requirements, while using the conservative approach (10 times the cross frequency) we get a settling time $T_{set} = 29.6$ which is just within the requirements of the system. Every sampling time larger than $T_s = 0.276s$ does not meet the requirements.

Task 11

The control performance is very bad and doesn't seem to converge to a final value, even increasing by much the simulation time, as can be seen in figure 1:

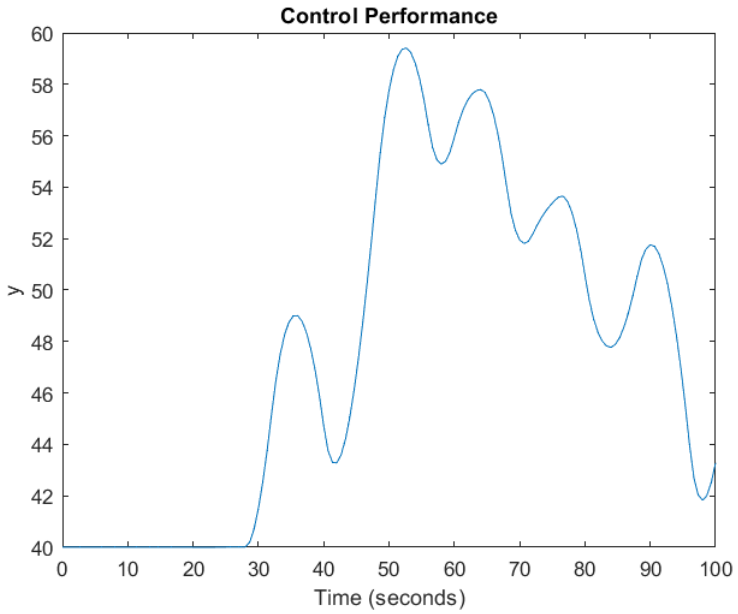

Figure 1: Control Performance

Task 12

The code used is:

```
Gd=c2d(G,4,'ZOH');
[d_num, d_den]=tfdata(Gd,'v');
[Phi,Gamma,C,D] = tf2ss(d_num,d_den);
```

And the matrix values are:

$$\Phi = \begin{bmatrix} 1.45 & -0.52 \\ 1 & 0 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Task 13

Using the commands *ctrb* and *obsv* we can get the controllability (also defined as reachability) and the observability matrix Wc and Wo. Then, by comparing their rank to the matrix Φ dimension, we can determine if the system is completely reachable and observable.

```
Wc = ctrb(Phi,Gamma);
rank(Wc)
Wo = obsv(Phi,C);
rank(Wo)
```

In this case both matrix have rank = 2, which is also the dimension of Phi, so the system is completely reachable and observable.

Task 14

The l_r gain is used to compute the feed forward control action, and is needed to have as a steady state output $y = r$.

Task 15

It is possible to define the dynamic observer as:

$$\Delta\hat{x}(k+1|k) = \phi\Delta\hat{x}(k|k-1) + \Gamma u(k) + K[-C\Delta\hat{x}(k|k-1) + \Delta y(k)]$$

And it is possible to write the state space representation of the closed loop augmented system as:

$$x_a(k+1) = \begin{bmatrix} \Delta x(k+1) \\ \Delta\hat{x}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi & -\Gamma L \\ KC & \Phi - KC - \Gamma L \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \Delta\hat{x}(k) \end{bmatrix} + \begin{bmatrix} \Gamma l_r \\ \Gamma l_r \end{bmatrix} r(k)$$

Task 16

By writing the state depending on the error:

$$\Delta x(k+1) = \Phi\Delta x(k) - \Gamma L\hat{x}(k) + \Gamma l_r r(k)$$

$$= (\Phi - \Gamma L)\Delta x(k) + \Gamma L\Delta\tilde{x}(k) + \Gamma l_r r(k)$$

Which is obtained by adding and subtracting $\Gamma L x(k)$
It's possible to determine the dynamics of the error as:

$$\Delta\tilde{x}(k+1) = \Delta x(k+1) - \Delta\hat{x}(k+1)$$

$$= \Phi\Delta x(k) - \Gamma L\Delta\hat{x}(k) + \Gamma l_r r(k) - (KC\Delta x(k) + (\Phi - KC - \Gamma L)\Delta\hat{x}(k) + \Gamma l_r r(k))$$

$$= (\Phi - KC)\Delta x(k) - (\Phi - KC)\delta\hat{x}(k))$$

$$= (\Phi - KC)\Delta\tilde{x}(k)$$

These equations justify the $z(k+1)$ equation, and the resulting matrix that multiplies $z(k)$ is an upper triangular matrix, so the eigenvalues of the whole matrix are those of the $\Phi - \Gamma L$ and $\Phi - KC$ matrix.

Task 17

By calculating and simplifying the closed loop transfer function in continuous time, it's possible to find the four poles:

```
closed_loop_system=F*G/(1+F*G);
closed_loop_system=minreal(closed_loop_system);
p=pole(closed_loop_system);
```

The poles are as follows:

$$p_1 = p_2 = 0.5 \quad p_{3,4} = -0.16 \pm 0.12i$$

To convert these continuous poles in discrete poles we used the following formula:

$$p_{d,i} = e^{T_s p_{c,i}}$$

The results are:

$$p_1 = p_2 = 0.13 \quad p_{3,4} = 0.46 \pm 0.24i$$

And, by using the Matlab code:

```
for i=1:length(p)
    disc_p(i)=exp(Ts*p(i));
end
observer_p=[disc_p(1) disc_p(2)];
controller_p=[disc_p(3) disc_p(4)];
L=acker(Phi,Gamma,controller_p);
K=acker(Phi',C',observer_p)';
```

K has been designed using the duality principle. By calculating Aa as described in task 16 and finding its eigenvalues with the command "$eig(Aa)$" it's possible to confirm that the poles have been placed correctly.

Task 18

The output of the signal improved a lot, and is now converging in a finite time towards a steady-state value, which is not the desired one, leaving a steady-state error of ≈ 0.25. This (small) error may be due to the approximations used to calculate the sampled plant model, which had then been used to tune the controller, propagating the calculation error.

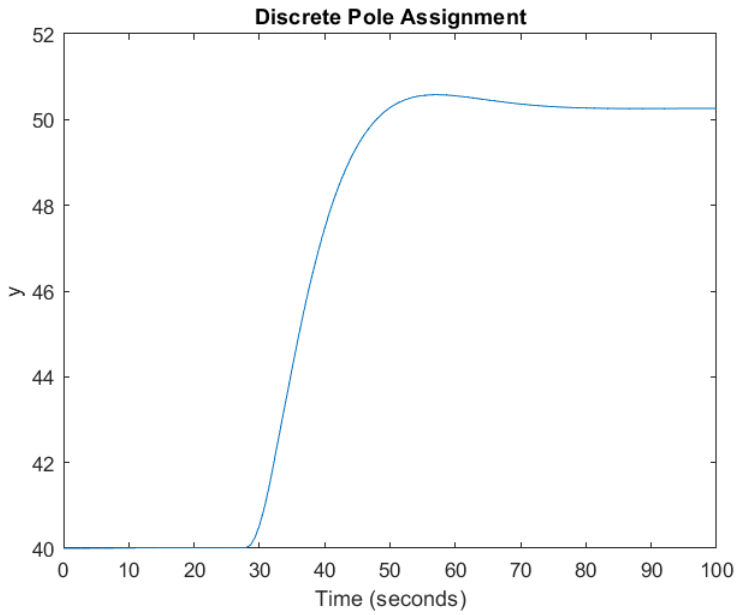

Figure 2: Control Performance With discrete pole assignment

Task 19

10 bit accuracy leads to a resolution of:

$$\frac{1}{2^{10}} \approx 0.0009765 = 0.09765\%$$

Considering the range of the signal is $0-100$, the quantization level of the signal would be:

$$\frac{100}{2^{10}} \approx 100 * 0.0009765 = 0.0976 = 9.765\%$$

Task 20

The quantization block was constructed in simulink by connecting the Quantizer block to a Saturation block, as shown in figure 3:

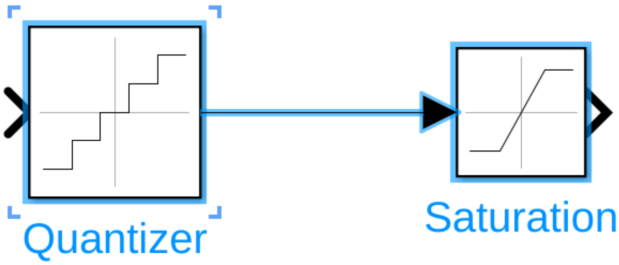

Figure 3: Quantization block

In this setup the Saturation block is used because the Quantizer block does not have any upper or lower limit, hence to create a signal limited to these upper and lower values, the saturation block is required.

Task 21

The control performance starts degrading at 6 bits (1.56 quantization level) but becomes completely different at 4 bits (6.25 quantization level) when the system output does no longer converge to a finite value but instead oscillates around the reference value.

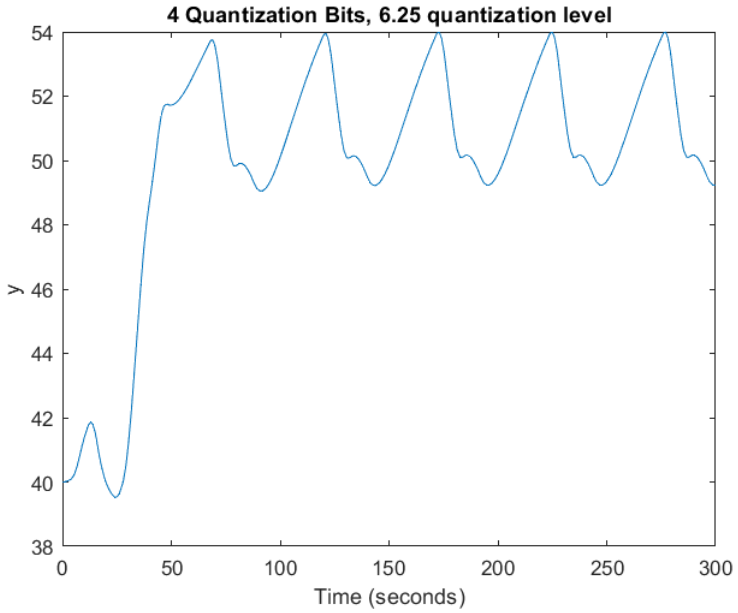

Figure 4: Quantization with 4 bits