EE4307 Control Systems Design and Simulation

Project 1

Philippe Andrin Brigger

A0248834U

Contents

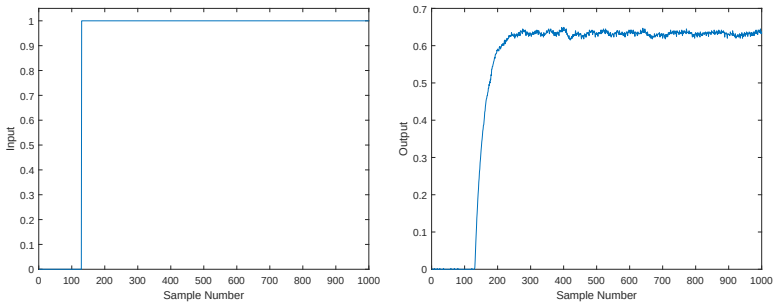
1 Introduction

This project aimed at getting hands-on experience in control systems design and simulation. In particular, non-parametric system identification methods were explored with tools in Matlab and Simulink to examine how well different methods work. Controllers were designed in continuous and discrete time and the effects of pole placement or sampling period were seen. Finally, the controllers were simulated using Simulink and Python for a robotic arm where insights into neural networks were provided. In this report, I discuss my results and learning experiences of the project.

2 Exercise 2

2.1 Exercise 2.1

For this exercise, the data file *part4_1* was loaded into the workspace. In a first step, the input vs. sample number and the output vs. sample number were plotted (see Figure 1).



(a) Input vs. sample number (b) Output vs. sample number

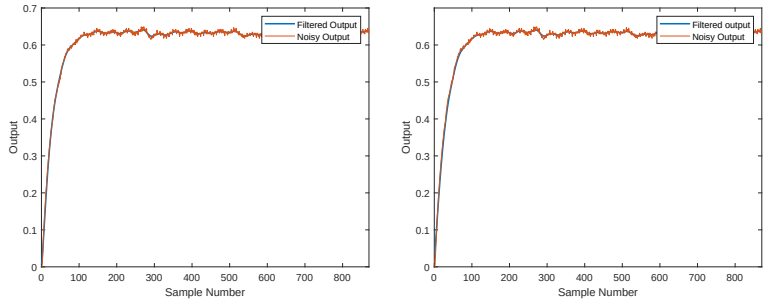
Figure 1: Response of system to a step input.

It can be seen that the first sample numbers still have zero input. The step input only starts at sample number 130 and ends at sample number 1000. The range of the output data was adapted to these sample numbers. Additionally, the high frequency noise was removed with the Matlab command *idfilt(Y(:,2), N, w, 'noncausal')*. w was set to 0.05 while the filter order N (either first or second order) did not play a major role as Figure 2 shows.
Plot Figure 3 suggests that the system at hand is a system of first order. A system of first order has the form:

$$G(s) = \frac{K}{1+Ts} e^{-\tau s} \tag{1}$$

The steady state gain K is found by dividing the steady state output with the steady state input. To find the steady state output, the mean of the output values from the last one hundred sample numbers was taken. This resulted in:

$$K = \frac{y_{ss}}{u_{ss}} = \frac{0.631}{1} = 0.631$$



(a) Output data filtered with filter of first order. (b) Output data filtered with filter of second order.

Figure 2: Filtering of output data with different filter order.

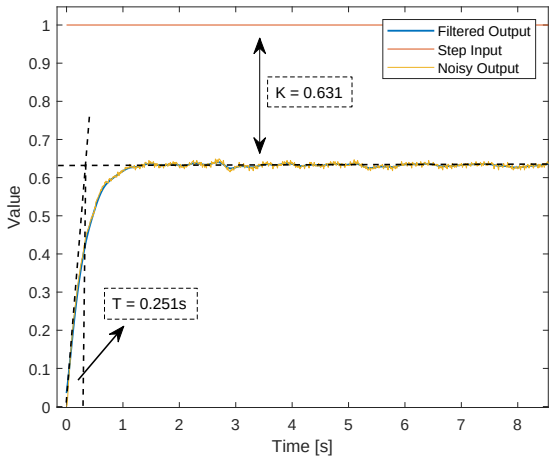


Figure 3: Output versus input data.

As Figure 3 suggests, there is no delay in the output ($\tau = 0$). To determine T, the system can be converted into the time domain and an output coordinate point $(t*, y*)$ can be inserted into the equation to obtain the time constant T.

$$y(t) = \mathcal{L}^{-1}\big(Y(s)\big) = K(1 - e^{-\frac{t}{T}})$$

Inserting the values for a data point $(y(t^*), t^*)$ before the output reaches its steady state, and solving the above equation for T gives:

$$T = -\frac{t^*}{ln(1 - \frac{y^*}{K})} = 0.251s$$

To conclude, the system from this exercise was identified as:

$$G(s) = \frac{0.631}{1 + 0.251s}$$

2.2 Exercise 2.2

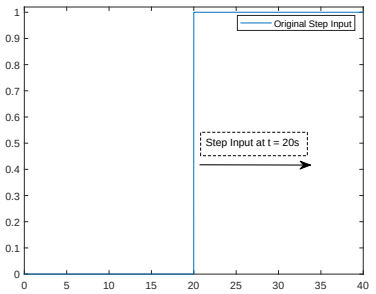
For exercise 2.2, data was provided in the file *cond.dat*. The initial plots (see Figure 4) of the data showed that there was no offset in the output and that the relevant data starts at $t = 20s$ as that is where the step input starts. Therefore, only the data after $t = 20s$ was conditioned and analyzed (fig. 5).
As in exercise 2.1, the high frequency noise of the output data was filtered out using the command *idfilt(Y(:,2), N, w, 'noncausal')* where a filter order of 1 and $w = 0.05$ was chosen. The resulting signal is plotted in Figure 6.

2.3 Exercise 2.3

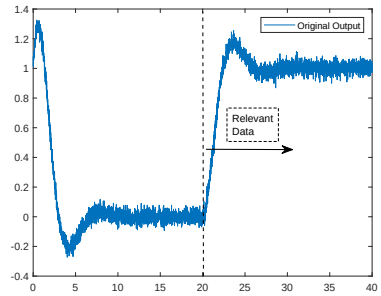
First, the impulse response of a digital system $G(z)$ was plotted in Matlab (see fig. 7).

$$G(z) = \frac{z}{z + 0.5}$$

The impulse response was found with the command $X = impulse(G, [0 : 1 : 15])$ where a sampling time of $T = 1s$ was used. Figure 7 shows the sampling points of the impulse response of the digital system (the system is described in discrete time).

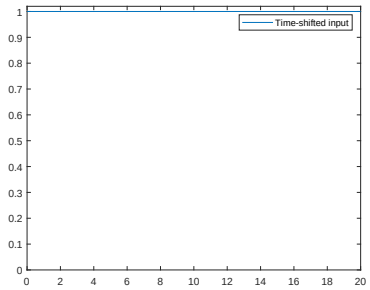


(a) Step input. Start at $t = 20s$.

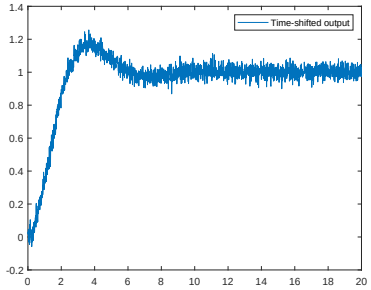


(b) Noisy output. No offset present (The output starts at 0 when the step input is applied at $t = 20s$.)

Figure 4: Plots of original, unconditioned data.



(a) Time-shifted step input.



(b) Time-shifted, noisy output.)

Figure 5: Time-shifted plots of original, unconditioned data.

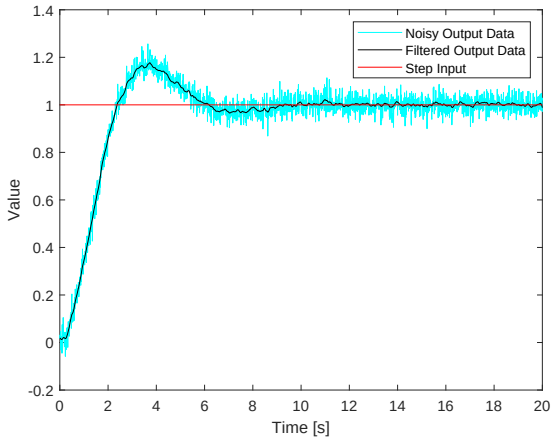


Figure 6: Filtered output vs. noisy, unfiltered output data.

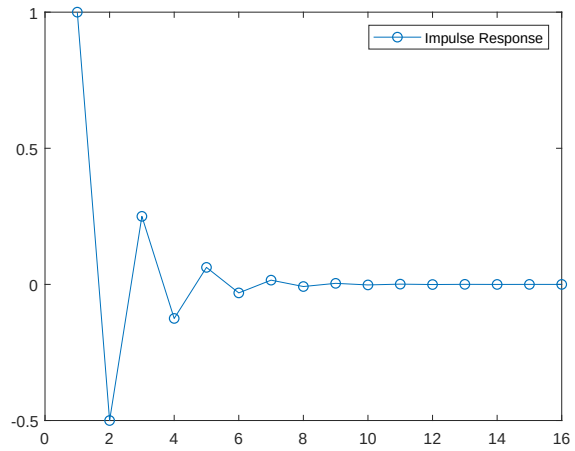


Figure 7: Impulse response of digital system G(z).

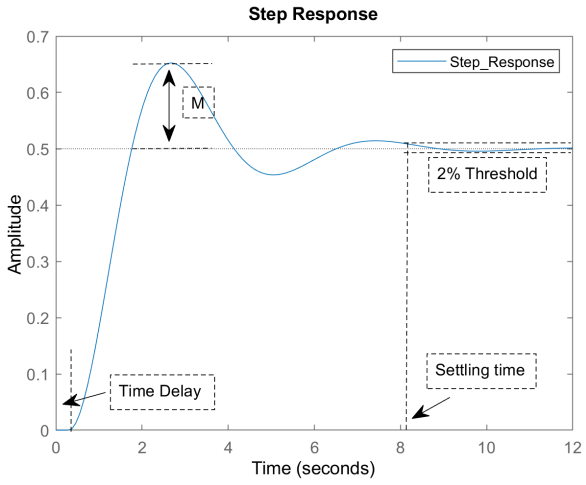


Figure 8: Step response of continuous system G(s).

Next, the time delay τ, the settling time t_s and overshoot M were derived from the system $G(s)$.

$$G(s) = \frac{e^{-0.3s}}{s^2 + s + 2}$$

The step response of the system, plotted with the command $step(G)$, can be seen in Figure 8.
Using the command stepinfo(G), gives various parameters of the system.

- Rise time $t_r = 0.987s$
- Settling time $t_s = 8.042s$
- Peak $P = 0.653$
- Peak time $t_p = 2.671s$

The rise time was defined as the time the system takes to rise from 10% to 90% of the way from the initial value to the steady-state value. The settling time was

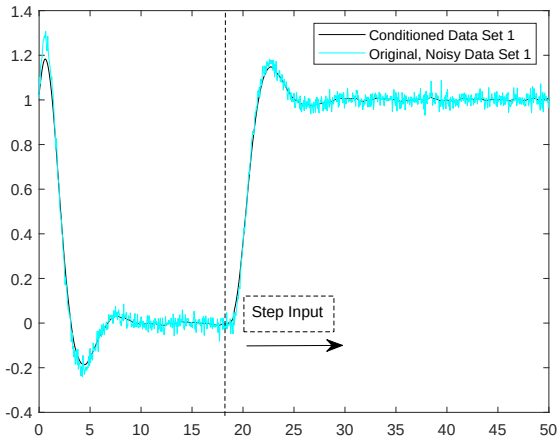


Figure 9: Conditioned data set 1 plotted over original noisy data set.

defined as the time it takes for the response to stay within a 2% threshold of the steady-state value and found to be $t_s = 8.042s$.
The overshoot M is defined as the peak value minus the steady state value of the step response. The steady state-value of the system was 0.5. Therefore, $M = 0.653 - 0.5 = 0.153$.
To find the time delay, an algorithm was implemented which found the time at which the output starts to rise. The result was a time delay $\tau = 0.298s$, which, if rounded, gives back $\tau = 0.3s$ as in the original system definition.

2.4 Exercise 2.4

Two different data sets of the same system were given and corrupted with high frequency noise. The goal of the exercise was to identify the system with the impulse response and step response of the conditioned data. The resulting systems were then compared with each other to see which method was more accurate. The more accurate system was translated into state space description.
In the following, the results for data set 1 are illustrated. Afterwards, data set 2 is treated before comparing the results.
The first data set was conditioned with the same command as in Section 2.1 with *idfilt(data(:,3), N, w, 'noncausal')*. A filter order $N = 1$ was chosen and $w = 0.05$ was set. Figure 9 shows the conditioned versus noisy data set 1.
Judging by the plot, this is the step response of the system. However, the step response only starts after 18 seconds which can be seen when analyzing the values of the data set. Therefore, for the analysis, the data of the first eighteen seconds was removed. Figure 10 shows the conditioned data which was used for the analysis.
The shape of the response in Figure 10 suggests that the system is of second order. Therefore, the transfer function has the form:

$$G(s) = \frac{k \cdot \omega_0^2 \cdot e^{-\tau s}}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{2}$$

Looking at Figure 10, it can be said that the system has a time delay. Analyzing the data in the conditioned data set showed that the system responds with a delay of $\tau = 0.65s$.
The steady state gain k was found by dividing the steady state output by the steady state input value. The steady state output $y_{ss} = 1.002$ was found by taking the

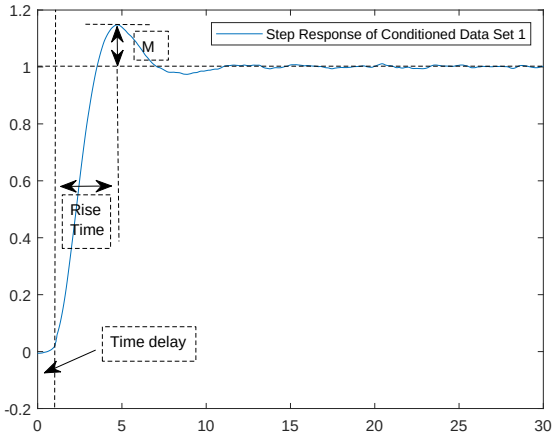


Figure 10: Step response of conditioned data set 1 only with relevant data after 18 seconds.

mean of the output values of the last five seconds. Since this value is basically equal to the steady state input value $u_{ss} = 1$, a steady state gain $k = 1$ was assumed. Next, the overshoot $M = 0.148$ of the system was calculated by subtracting the steady state output value from the maximum value of the conditioned output. This value could be used to determine the damping coefficient ζ of the system as follows:

$$\begin{aligned} M &= e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \\ \big(ln(M)\big)^2 &= \frac{\zeta^2\pi^2}{1+\zeta^2} \\ \big(ln(M)\big)^2 &= \zeta^2 \cdot \Big(\pi^2 + \big(ln(M)\big)^2\Big) \\ \Rightarrow \zeta &= \sqrt{\frac{\big(ln(M)\big)^2}{\Big(\pi^2 + \big(ln(M)\big)^2\Big)}} \\ \zeta &= 0.520 \end{aligned}$$

The rise time to the first peak of the step response was found to be $t_p = 4.1s$ by subtracting the time instance at the peak with the time delay of the system. The peak time can be used to calculate the natural frequency ω_0 of the system:

$$\begin{aligned} t_p &= \frac{\pi}{\omega_0\sqrt{1-\zeta^2}} \\ \Rightarrow \omega_0 &= \frac{\pi}{t_p\sqrt{1-\zeta^2}} \end{aligned}$$

This gave a value of $\omega_0 = 0.897\frac{rad}{s}$. Therefore, the system identified from conditioned data set 1 is described by the transfer function:

$$G(s) = \frac{0.805}{s^2 + 0.933s + 0.805} \tag{3}$$

After finding a transfer function for the system with data set 1, the same was done with data set 2. Figure 11 shows the conditioned versus unconditioned data set.

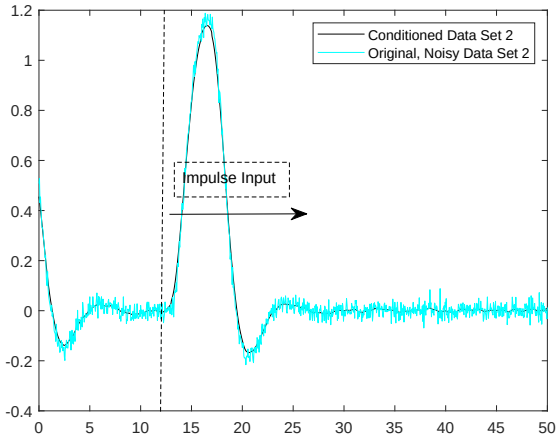


Figure 11: Conditioned data set 2 plotted over original noisy data set.

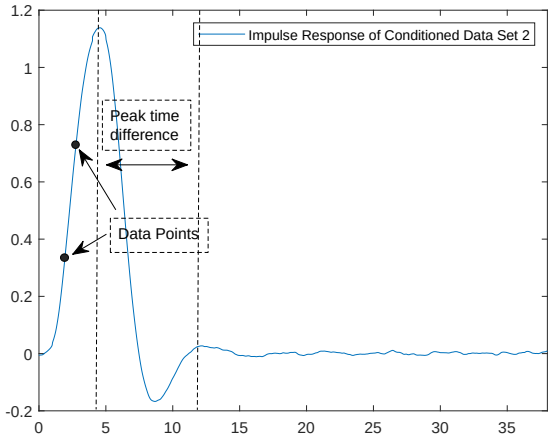


Figure 12: Impulse response of conditioned data set 2 only with relevant data after 12 seconds.

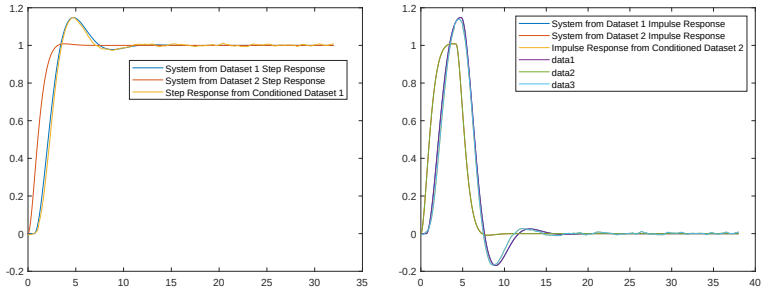
As for data set 1, the data set was filtered using the command *idfilt(data(:,3), N, w, 'noncausal')* with the same parameters $N = 1, w = 0.05$.
Figure 12 shows the impulse response of the system. Looking at the data set more precisely, it can be seen that the impulse is given to the system at $t = 12s$. Therefore, only the data after 12 seconds is relevant for the analysis.
An impulse response in the time domain has the form:

$$y(t) = \left(\frac{\omega_n e^{-\delta\omega_n t}}{\sqrt{1-\delta^2}}\right) \sin(\omega_d t) \tag{4}$$

To find the unknown parameters, the time difference t_d between the first and second peak is needed as well as two arbitrary data points. The time difference evaluated to $t_d = 12.2s - 4.55s = 7.65s$. From this, ω_d is calculated:

$$\omega_d = \frac{2\pi}{t_d} = 0.821\frac{rad}{s}$$

The remaining unknowns δ and ω_n can be calculated with the following formulas by taking two arbitrary data points:



(a) Step responses of conditioned data set 1 versus step responses of identified systems.

(b) Impulse responses of conditioned data set 1 versus impulse responses of identified systems.)

Figure 13: Step and impulse response plots of identified systems versus recorded data.

$$\delta = \left[\left((t_3 - t_4) \ln \left(\frac{y(t_3) \sin(\omega_d t_4)}{y(t_4) \sin(\omega_d t_3)} \right) \right)^2 + 1 \right]^{-0.5}$$

$$\omega_n = \frac{\omega_d}{\sqrt{1 - \delta^2}}$$

The results were $\delta = 0.836$ and $\omega_n = 1.496$. This gives the impulse response:

$$y(t) = \left(2.726 e^{-1.246t} \right) \sin(0.821t)$$

The impulse response in the time domain can be translated to the frequency domain with the following formula:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2}$$

Therefore, the transfer function of the system identified using the impulse response of data set 2 is:

$$G(s) = \frac{2.238}{s^2 + 2.501s + 2.238}$$

After obtaining the transfer functions of the same system with two different data sets, already just observing the coefficients shows that the two methods produced considerably different results. In particular, the impulse response system identification method using data set 2 neglects the time delay of the system entirely.
Figure 13 clearly shows that the system identified with data set 1 and the step response is a lot more accurate and resembles the true system responses accurately. The system identified with data set 2, however, behaves very differently than the original system and hence shouldn't be used.
Therefore, only a state space description of the system identified from data set 1 should be used for control purposes. Using the command *tf2ss* in Matlab converts a transfer function to a state space model. For the system identified with data set 1, the state space description is:

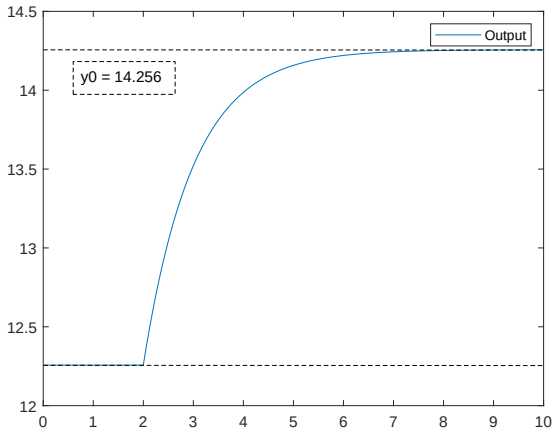


Figure 14: Identifying y_0 from the first operating point $u_0 = 3$.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -0.9330 & -0.8050 \\ 1.0000 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t - \tau_0)$$

$$y(t) = \begin{bmatrix} 0 & 0.8050 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

It is important to note that there is a delay of $\tau_0 = 0.65s$ acting on the input of the system defined above. This state space description could now be used to design a controller etc.

2.5 Exercise 2.5

Plotting the output of the first operating point $u_0 = 3$ applied to the unknown system showed that the corresponding output is $y_0 = 14.256$.
This value was added as a constant in the second block diagram. To allow the system to reach a steady state at the operating point before performing the step response, a step time of $t = 10s$ was used as that is where the output of the first system reaches its steady state.
Figure 15 shows that the system must be of first order. It can be seen that the system has a delay of $\tau_0 = 2s$. Since the steady state value $y_{ss} = 1 = u_{ss}$, the steady state gain of the system is $k = 1$. The last thing missing is the time constant T of the system.
The easiest way to find T is to insert a data point shortly after the output starts to rise into the time domain step response function:

$$y(t) = k \cdot u(t) \cdot \left(1 - e^{\frac{-(t-\tau_0)}{T}}\right)$$
$$\Rightarrow T = \frac{\tau_0 - t}{ln(1 - \frac{y(t)}{k \cdot u(t))}}$$

This approach yielded $T = 1s$. The unknown system can therefore be described with the transfer function:

$$G(s) = \frac{e^{-2s}}{1 + s}$$

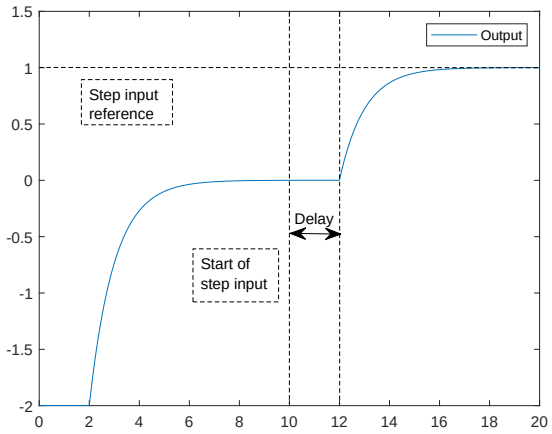


Figure 15: System identification from step input.

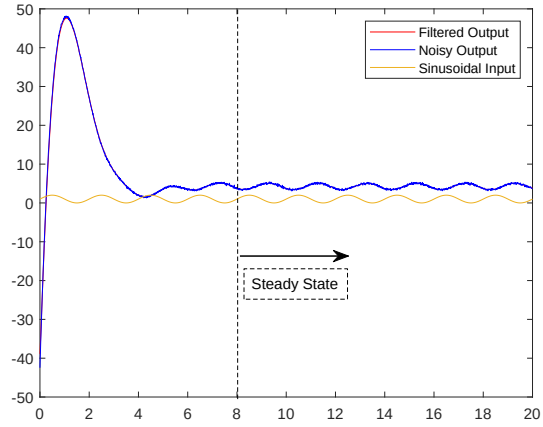


Figure 16: Response of system to sinusoidal input. The steady state behavior is a phase- and amplitude-shifted sinusoidal signal.

2.6 Exercise 2.6

Plotting the data (see Figure 16) from set 3 shows that a sinusoidal input is given to the system. For linear systems, the steady state response of the system will also be sinusoidal with the same frequency but potentially gain- and phase-shifted.
To determine the gain and phase shift, the conditioned output was replotted with the input from $t = 8s$ onwards.
The input data has a constant offset of 1 from zero, and a period of $T_0 = 2s$ (The frequency of the signal is therefore $\omega_0 = 2\pi/T_0 = \pi \cdot rad/s$). There is no phase shift. The input signal therefore is described as:

$$u(t) = A\sin(\omega)t + A_0 = \sin(\pi t) + 1$$

It can be seen from fig. 17 that the output signal has the same frequency ω_0 as the input signal. The amplitude $A^* = 0.83$ of the output signal was found by subtracting the mean from the maximum value of the signal. Therefore, the gain shift between the input and output signal is also 0.83. The phase shift $\phi = -0.82\pi$ between the two signals was calculated by finding the time offset between two signal peaks of the output and input signal. Finally, the constant offset between an output

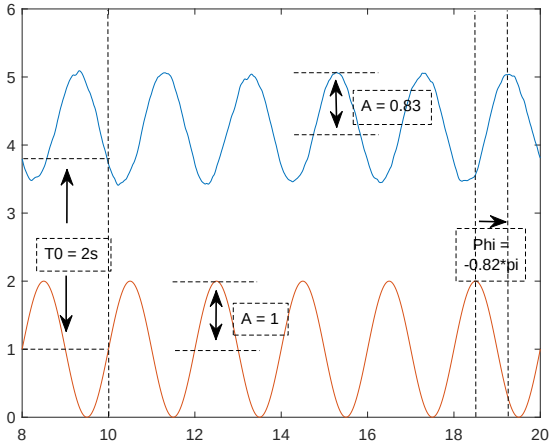


Figure 17: Phase and gain shift of output data.

and input peak is $A_0^* = 3.09$. To conclude, the steady state output signal can be written as:

$$y(t) = 0.83 \sin\big((t - 0.82)\pi\big) + 4.09$$

3 Exercise 3

3.1 Exercise 3.1

A continuous time controller $D(s)$ was designed to stabilize the following plant $G(s)$:

$$G(s) = \frac{1}{s^2 + 2s + 1} \tag{5}$$

The gains of the PID controller $D(s)$ were chosen such that the dominant roots of the plant would lie at $s = -5$. To achieve this, the pole placement technique was used. To find appropriate PID gains, the closed loop characteristic equation for the system $G(s)$ and a general PID controller $D(s)$ needs to solved:

$$\begin{aligned} 1 + G(s)D(s) &= 0 \\ 1 + \frac{1}{s^2 + 2s + 1}(k_p + k_d s + \frac{k_i}{s} &= 0 \\ s^2 + 2s + 1 + k_p + k_d s + \frac{k_i}{s} &= 0 \\ s^3 + (2 + k_d)s^2 + (k_p + 1)s + k_i &= 0 \end{aligned}$$

Since the highest order of s is 3, the closed-loop transfer function will have three poles. In order to have a stable system, all of these poles must have a negative real part. Since the dominant roots should have $s = -5$, the poles were placed at $s = -p_1 = -p_2 = -p_3 = -5$. The PID gains can be calculated by reformulating the closed cloop characteristic equation:

$$\begin{aligned} &s^3 + (2 + k_d)s^2 + (k_p + 1)s + k_i = (s + p_1)(s + p_2)(s + p_3) = 0 \\ &s^3 + (2 + k_d)s^2 + (k_p + 1)s + k_i = s^3 + (p_1 + p_2 + p_3)s^2 + (p_1p_2 + p_1p_3 + p_2p_3)s + p_1p_2p_3 = 0 \\ &\Rightarrow k_d = p_1 + p_2 + p_3 - 2 \\ &\Rightarrow k_p = p_1p_2 + p_1p_3 + p_2p_3 - 1 \\ &\Rightarrow k_i = p_1p_2p_3 \end{aligned}$$

Inserting the pole values gives the PID gains $k_d = 13$, $k_p = 74$ and $k_i = 125$. The continuous time controller $D(s)$ with these gains is defined as:

$$D(s) = 74 + 13s + \frac{125}{s} \tag{6}$$

Simulating the system $G(s)$ and $D(s)$ with a unit step input at $t = 1s$ showed that the controller successfully brings the system to the reference value (see Figure 18). To increase the performance of the system, different PID gains can be used. In general, if the poles are placed farther away from the origin (in the left-hand plane), then the system will respond much quicker. This is easily seen when resimulating and replotting the system response for $p_1 = 5$, $p_2 = p_3 = 50$.

3.2 Exercise 3.2

In this exercise, the system is given in state-space as:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} d \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \end{aligned}$$

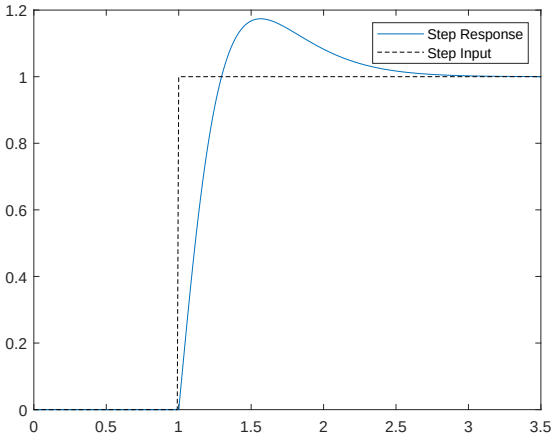


Figure 18: Step response of system with PID controller.

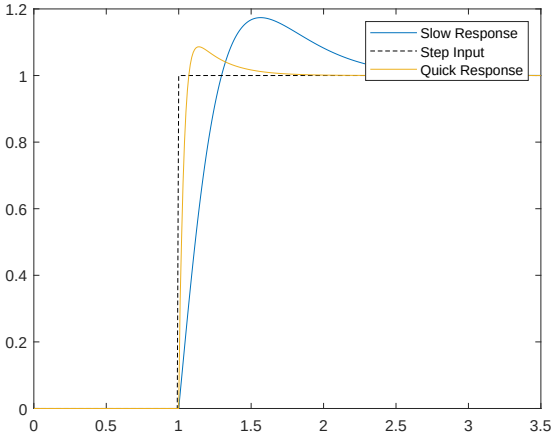


Figure 19: Choosing poles with very large negative real part will result in a quicker step response.

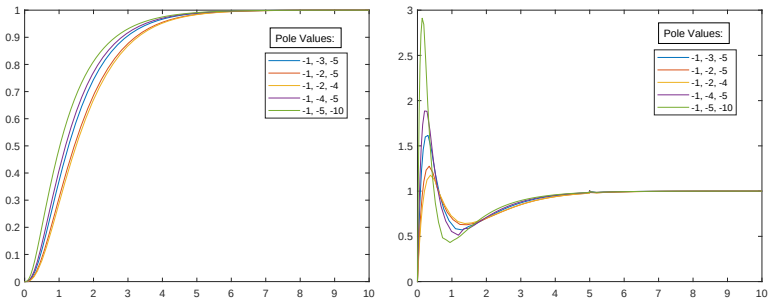
To be able to reject the external disturbance d and track a reference r, an integral of the tracking error is incorporated into the state-feedback control law.

$$u = -k_1 x - k_2 \dot{x} - k_3 v$$
$$v = \int_0^t (r - y) d\tau$$

Inserting this control law gives the augmented state-space description of the system:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 - k_1 & -1 - k_2 & -k_3 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} d + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} r$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

In order for the controller to stabilize the system and follow a reference, the matrix A^* of the augmented system must have only stable poles p_1, p_2, p_3. Hence, the poles



(a) Reference tracking with different pole values.

(b) Control input with different pole values.

Figure 20: Effect of pole placement on control input and reference tracking.

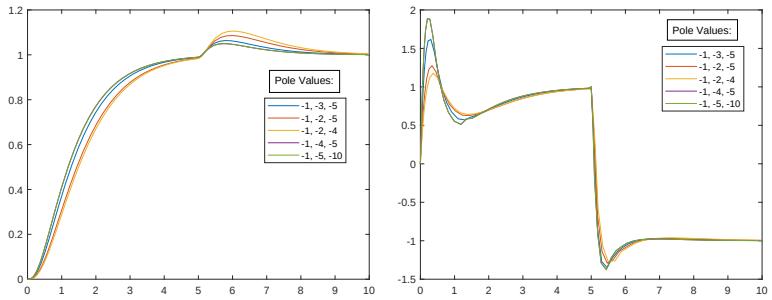
of A^* must all have negative real part. The poles are calculated with the following equation.

$$
\begin{aligned}
det(A^* - p_i) = det\left(\begin{bmatrix} -p_i & 1 & 0 \\ -1-k_1 & -1-k_2-p_i & -k_3 \\ -1 & 0 & -p_i \end{bmatrix} \right) &\stackrel{!}{=} 0 \\
-p_i \cdot det\left(\begin{bmatrix} -1-k_2-p_i & -k_3 \\ 0 & -p_i \end{bmatrix} \right) - det\left(\begin{bmatrix} -1-k_1 & -k_3 \\ -1 & -p_i \end{bmatrix} \right) &\stackrel{!}{=} 0 \\
-p_i \cdot (p_i^2 + k_2 p_i + p_i) - (p_i + k_1 p_i - k_3) &\stackrel{!}{=} 0 \\
p_i^3 + (1+k_2)p_i^2 + (1+k_1)p_i - k_3 &= 0
\end{aligned}
$$

As can be seen, the controller gains depend on the pole placement. The appropriate controller gains can be calculated once the values of the three desired poles are inserted into the above equations for $i = 1, 2, 3$.

$$
\begin{aligned}
\boldsymbol{A_P \cdot k} = \begin{bmatrix} p_1 & p_1^2 & -1 \\ p_2 & p_2^2 & -1 \\ p_3 & p_3^2 & -1 \end{bmatrix} \cdot \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} &= \begin{bmatrix} -p_1(p_1^2 + p_1 + 1) \\ -p_2(p_2^2 + p_2 + 1) \\ -p_3(p_3^2 + p_3 + 1) \end{bmatrix} = \boldsymbol{B_P} \\
\Rightarrow \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \left(\begin{bmatrix} p_1 & p_1^2 & -1 \\ p_2 & p_2^2 & -1 \\ p_3 & p_3^2 & -1 \end{bmatrix} \right)^{-1} &\cdot \begin{bmatrix} -p_1(p_1^2 + p_1 + 1) \\ -p_2(p_2^2 + p_2 + 1) \\ -p_3(p_3^2 + p_3 + 1) \end{bmatrix}
\end{aligned}
$$

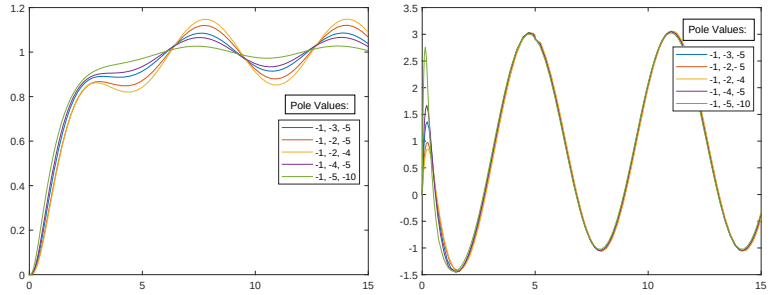
For the matrix A_p to be invertible, it is not allowed to be singular. In other words, it must have full rank. For this to be fulfilled, the poles must each have a unique value. If two or more poles have the same value, the matrix would need to pseudo-inverted. To avoid this case, unique poles were set for the subsequent analysis.
The exercise description asks for $p_1 = -\zeta\omega_n = -0.5 \cdot 2 = -1$ and that p_2, p_3 are between two to five times faster than the dominant pole p_1.
AS the plots in Figure 26 show, choosing poles with greater negative real parts will lead to a better reference tracking and will increase the control input. The pole with the largest negative real part will influence the response the greatest. A pole with a large negative real part will lead to a quick system response but require a greater control effort to do so. Therefore, when placing the poles, a tradeoff between control effort and quick reference tracking must be made depending on what hardware is being used.



(a) Reference tracking with different pole values and a step disturbance of magnitude 2 at time $t = 5s$.

(b) Control input with different pole values and a step disturbance of magnitude 2 at time $t = 5s$.

Figure 21: Effect of pole placement on control input and reference tracking when a step disturbance of magnitude 2 at time $t = 5s$ acts on the system.



(a) Reference tracking with different pole values and a sinusoidal disturbance $d = 2\sin(t)$.

(b) Control input with different pole values and a sinusoidal disturbance $d = 2\sin(t)$.

Figure 22: Effect of pole placement on control input and reference tracking when a sinusoidal disturbance $d = 2\sin(t)$ acts on the system.

As the plots in Figure 21 show, if a step disturbance of magnitude 2 at time $t = 5s$ acts on the system, the controller is able to stabilize the system and deal with the disturbance. The different pole values show that choosing pole values with a large negative real part compensate the disturbance quicker. At the same time, the control input is only slightly greater. Therefore, if the reference input stays constant, it would be desirable to increase the PID gains as they will deal with disturbances quicker without increasing the control effort too much.
The plots in Figure 22 show that the controller no longer can reduce the steady state error to zero when a sinusoidal (or simply put time-varying) disturbance acts on the system. This is also to be expected, as in steady state the following equation holds:

$$\dddot{x} + (1+k2)\ddot{x} + (1+k1)\dot{x} + k3(r-x) = \dot{u} + \dot{d}$$
$$\Rightarrow x = \frac{2\cos(t) + \dot{u}}{k_i}, \quad \dddot{x} = \ddot{x} = \dot{x} = 0, \quad \textit{steady} \quad \textit{state}$$

Furthermore, it can be seen that with poles that have larger negative real parts, the amplitude of the oscillating steady-state of the output is larger. Once the reference signal has been reached, the pole values don't have any significant impact on the

control effort. Therefore, if it is known that a time-varying disturbance acts on the system, it could be advantageous to choose poles that are closer to the origin and the magnitude of the negative real part is not too large.

3.3 Exercise 3.3

In this exercise, the continuous time controller $D(s)$ from Section 3.2 was converted into a discrete time controller $D(z)$. There are many methods that can be used to convert a continuous time system into a discrete time system. Here, the bilinear or Tustin's transformation was used:

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{7}$$

To find $D(z)$, the controller $D(s)$ first had to be defined. In Section 3.2, the following control law was used:

$$u(t) = -k_1 x - k_2 \dot{x} - k_3 \int_0^t (r - x) d\tau$$

Writing the control law in the frequency domain via Laplace transform gives:

$$U(s) = -k_1 X(s) - k_2 X(s) s - k_3 \frac{R(s) - X(s)}{s}$$

To write the control law in discrete time, the variable s must be replaced with the transformation method defined in eq. (7):

$$\begin{aligned} U(z) &= -k_1 X(z) - k_2 X(z) \frac{2}{T}\frac{z-1}{z+1} - k_3 \frac{R(z) - X(z)}{\frac{2}{T}\frac{z-1}{z+1}} \\ \Rightarrow U(z) &= -k_1 X(z) - k_2 X(z) \frac{2}{T}\frac{z-1}{z+1} - k_3 \frac{T}{2}\frac{z+1}{z-1}\Big(R(z) - X(z)\Big) \end{aligned}$$

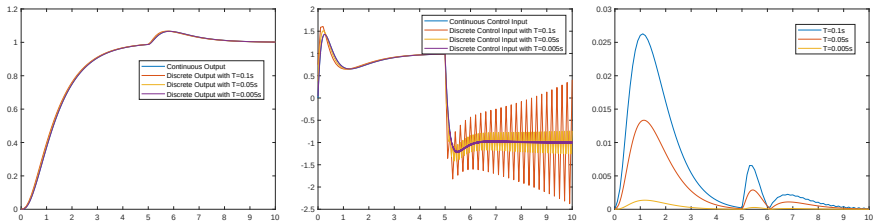
T is the sampling period. For this exercise, the PID controller $D(s)$ from section 3.2 with the poles placed at $p_1 = -1, p_2 = -3, p_3 = -5$ was discretized. The gains were calculated with the approach described in section 3.2. This gave the following discretized PID controller $D(z)$:

$$\begin{aligned} D(s) &= 22 + 8s - \frac{15}{s} \\ \Rightarrow D(z) &= 22 + \frac{16}{T}\frac{z-1}{z+1} - \frac{15T}{2}\frac{z+1}{z-1} \end{aligned}$$

As Figure 24 shows, if the sampling time is small enough, the difference between the continuous and discrete time signals are minor. It can be observed that the control input from the discrete time controller is very oscillatory if the sampling time is too large. In the worst case, the system becomes unstable because the sampling time is too large.

3.4 Exercise 3.4

Instead of designing a controller $D(s)$ in continuous time and then discretizing the controller to discrete time controller $D(z)$ with the z-transform and e.g. Tustin's transformation, the plant $G(s)$ can be discretized to $G(z)$ and then the controller $D(z)$ can be designed directly in discrete time based on the discretized plant $G(z)$. This is what was done in this exercise.

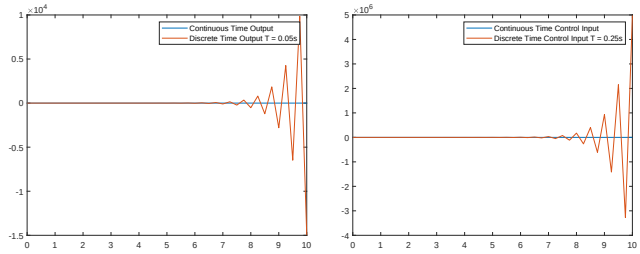


(a) Output of discrete versus continuous time signal with different sampling times T.

(b) Control input of discrete versus continuous time signal with different sampling times T.

(c) Error between discrete and continuous time output with different sampling times T.

Figure 23: Results of discrete versus continuous time controller with different sampling times T.



(a) Output blows up if sampling time T is too large.

(b) Control input blows up if sampling time T is too large.

Figure 24: If sampling time T is too large, the system becomes unstable.

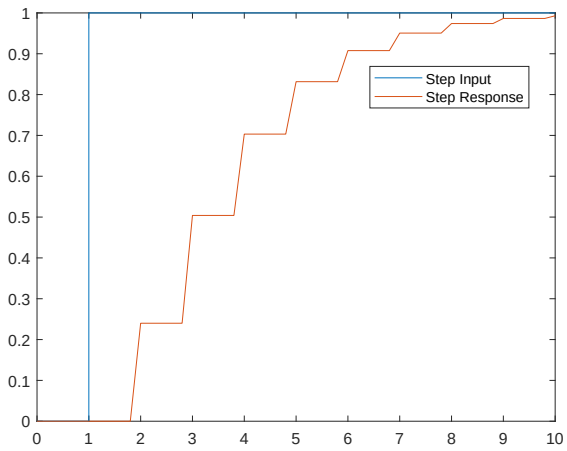


Figure 25: Step response with discretized plant $G(z)$ and controller $D(z)$ designed in discrete time. A sampling time of $T = 1s$ was used and the poles were placed at $p_1 = -0.2, p_2 = -0.5, p_3 = -0.4$.

For this purpose, a sampling time of $T = 1s$ was used. This gave the discretized plant $G(z)$:

$$G(z) = (1 - z^{-1})Z\big(\frac{G(s)}{s}\big) = \frac{z}{z^2 - 1.6z + 0.64}$$

The goal of the exercise was to design a discrete PID controller $D(z)$ such that the dominant pole of the closed loop system lies at $z = -0.2$. By placing the poles of the closed loop characteristic equation at the desired positions, the PID gains of $D(z)$ were inherently given.

$$\begin{aligned} H(z) &= \frac{D(z)G(z)}{1 + D(z)G(z)} \\ &= \frac{\big(k_p + k_i \frac{z}{z-1} + k_d \frac{z-1}{z}\big)\big(\frac{z}{z^2-1.6z+0.64}\big)}{1 + \big(k_p + k_i \frac{z}{z-1} + k_d \frac{z-1}{z}\big)\big(\frac{z}{z^2-1.6z+0.64}\big)} \\ \Rightarrow 1 + D(z)G(z) &= (z + p_1)(z + p_2)(z + p_3) = 0 \end{aligned}$$

Inserting the poles $p_1 = -0.2, p_2 = -0.5, p_3 = -0.4$ into the above equation and with the use of a Matlab script, the resulting discrete controller was:

$$D(z) = 0.66 + 0.24\frac{z}{z-1} + 0.6\frac{z-1}{z}$$

As Figure 25 shows, the discrete controller successfully brings the output of the discretized plant to the step input value. However, the signal is not very smooth since a very large sampling time of $T = 1s$ was used.

4 Exercise 4

4.1 Exercise 4.2

In this section a linear PID controller is developed to control the nonlinear dynamics of a robot arm. In order to control the nonlinear system, the system first must be defined in state space form. Then the system can be linearized around the equilibrium point. From the resulting linear system, a PID controller can be designed via pole placement.
The nonlinear dynamics of the robotic arm are:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + d \tag{8}$$

In eq. (8) the nonlinear matrices $M(q), C(q,\dot{q}), G(q)$ are defined as:

$$\begin{aligned}
M(q) &= \begin{bmatrix} m_1 r_1^2 + m_2(L_1^2 + 2L_1 r_2 \cos\theta_2 + r_2^2) & m_2 r_2(r_2 + L_1\cos\theta_2) \\ m_2 r_2(r_2 + L_1\cos\theta_2) & m_2 r_2^2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} -m_2 r_2 L_1 \sin\theta_2 (2\dot{\theta_1}\dot{\theta_2} + \dot{\theta_2}^2) \\ m_2 L_1 r_2 \dot{\theta_1}^2 \sin\theta_2 \end{bmatrix} \\
G(q) &= \begin{bmatrix} m_1 g r_1 \cos\theta_1 + m_2 g(L_1\cos\theta_1 + r_2\cos(\theta_1+\theta_2)) \\ m_2 g r_2 \cos(\theta_1+\theta_2) \end{bmatrix}
\end{aligned}$$

where q is the generalized coordinate, τ consists of the two joint torque inputs and d contains the two joint disturbance torques.

$$\begin{aligned}
q &= \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^T \\
\tau &= \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^T \\
d &= \begin{bmatrix} d_1 & d_2 \end{bmatrix}^T
\end{aligned}$$

By defining a state vector $x = \begin{bmatrix} q & \dot{q} \end{bmatrix}^T$, eq. (8) can be written in state space form as:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(q)\Big(-C(q,\dot{q})\dot{q} + G(q) + \tau + d \Big) \end{bmatrix} = f(x,\tau,d) \tag{9}$$

The state space form in *eq.* (9) has the states appearing nonlinearly. This is evident when any of the matrices $M(q), C(q,\dot{q}), G(q)$ are looked at. To be able to design a PID controller with pole placement, the system first needs to be linearized. In order to this, the equilibrium point x_{eq} and equilibrium joint torque τ_{eq} is needed. The equilibrium points were given in this exercise as:

$$\begin{aligned}
x_{eq1} &= \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \\
x_{eq2} &= \begin{bmatrix} \frac{pi}{2} & 0 & 0 & 0 \end{bmatrix}^T \\
x_{eq3} &= \begin{bmatrix} -\frac{\pi}{2} & 0 & 0 & 0 \end{bmatrix}^T
\end{aligned}$$

The equilibrium joint torque can be calculated with eq. (9) by setting $\dot{x} = 0$:

$$\tau_{eq} = G(q_{eq})$$

The system can then be linearized:

$$\Delta\dot{x} = \left.\frac{\partial f}{\partial x}\right|_{x_{eq},\tau_{eq},d=0} \cdot \Delta x + \left.\frac{\partial f}{\partial \tau}\right|_{x_{eq},\tau_{eq},d=0} \cdot \Delta u$$

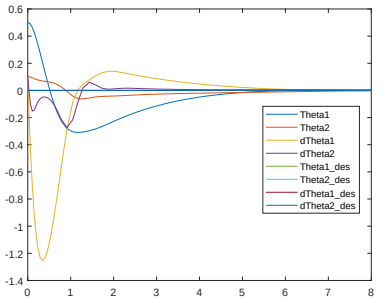
where $\Delta x = x - x_{eq}$ and $\Delta u = \tau - \tau_{eq}$. The system was augmented with the new state:

$$\theta = \int_0^t (r - \Delta q)dt$$

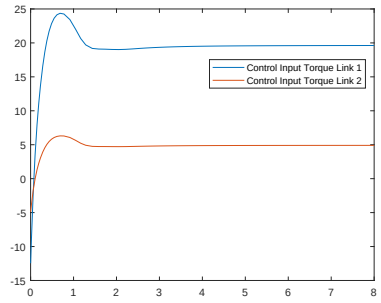
This then made it possible to calculate the control gains with the function $K = place(A, B, p)$ where the poles p could be chosen arbitrarily and the augmented system matrices A, B, resulting from the linearization around the equilibrium point and torque input, were used.
Figure 26 shows the simulation results.
It can be seen that the controller can stabilize the robot arm and reach an equilibrium point $x^1_{eq}, x^2_{eq}, x^3_{eq}$ from an initial state x^1_0, x^2_0, x^3_0. To reach the equilibrium state, the control input torques must change rapidly at the beginning due to the initial state differing from the desired equilibrium state. This is because initially τ_{eq} is given as a control input to the actuators. Since the initial state is not at the equilibrium state, this control input isn't yet the appropriate control input. The controller then adapts the joint torque inputs to bring the robot arm back into the equilibrium position. Once the arm has been stabilized, a new constant, equilibrium torque input τ_{eq} is observable.

4.2 Exercise 4.3

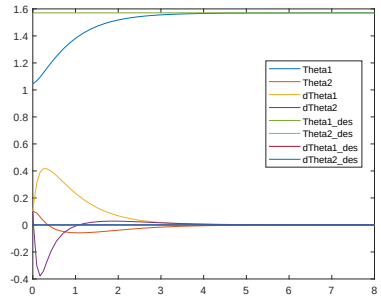
Since the controllers developed in the previous exercises were designed for a specific system model and a system will always have identification errors, a more accurate alternative approach would be to have adaptive control gains. One way to automatically tune the control gains based on the tracking errors is with a neural network (NN). In this exercise the effects of different mass links, learning rates and output activation functions was plotted and analyzed. Additionally, the parameter defining the amount of hidden layers was varied.
Figure 27 shows that using too many hidden layers can make the output of the joint angles oscillate greatly around the reference angle. Figure 28 shows that changing the mass links affects the mean loss of the training function as well as the joint angle output. Choosing a finer learning rate made the output angle signals less smooth. Figure 29 shows that choosing relu instead of sigmoid as an output activation function leads to a very noisy output which wouldn't work well for control purposes.
To conclude, the parameters of the NN have a very strong influence on the system's behavior. Therefore, it is very important to train and evaluate the NN with many different parameter sets to find an optimal configuration. Once a reliable NN has been set up, however, the performance of the controller should remain the same even if the system doesn't perfectly match the identified system as the control gains are automatically adapted online. In fact, since a system will never match its model, e.g. due to wear, noisy measurement data or different external conditions, it makes a lot of sense to work with adaptive controller gains and update the gains online since this will in the end optimally control and stabilize the system.



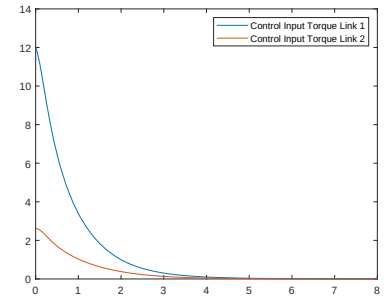
(a) The controller brings the states to the desired equilibrium point x_{eq}^1.



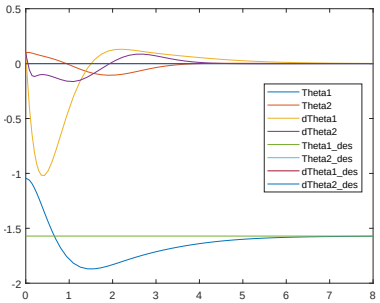
(b) The input control torques used to stabilize the robot arm for x_0^1.



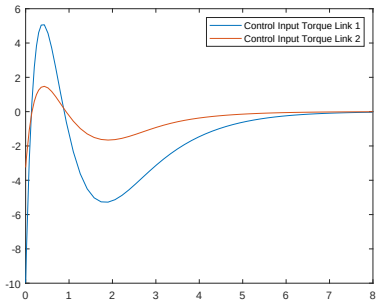
(c) The controller brings the states to the desired equilibrium point x_{eq}^2.



(d) The input control torques used to stabilize the robot arm for x_0^2.

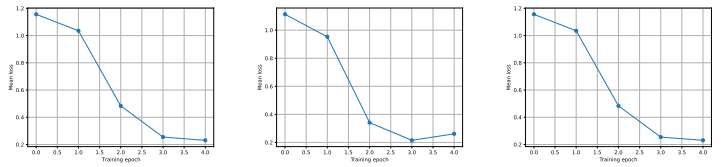


(e) The controller brings the states to the desired equilibrium point x_{eq}^3.

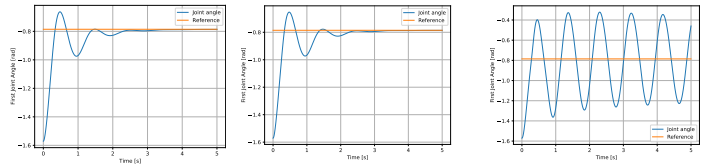


(f) The input control torques used to stabilize the robot arm for x_0^3.

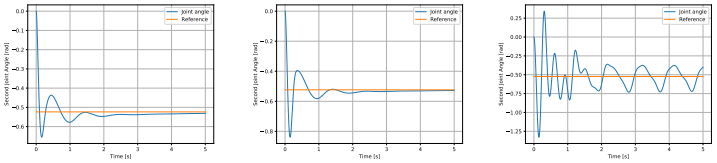
Figure 26: The input control torques manage to move the robot arm from the initial states x_0^1, x_0^2, x_0^3 to the equilibrium states $x_{eq}^1, x_{eq}^2, x_{eq}^3$.



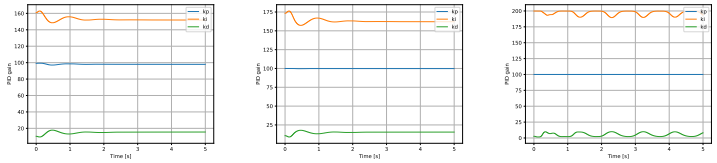
(a) 50 hidden layers (b) 100 hidden layers (c) 150 hidden layers



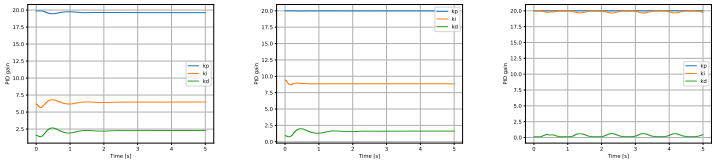
(d) 50 hidden layers (e) 100 hidden layers (f) 150 hidden layers



(g) 50 hidden layers (h) 100 hidden layers (i) 150 hidden layers

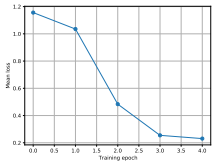


(j) 50 hidden layers (k) 100 hidden layers (l) 150 hidden layers

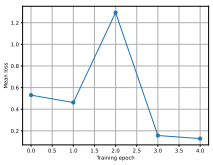


(m) 50 hidden layers (n) 100 hidden layers (o) 150 hidden layers

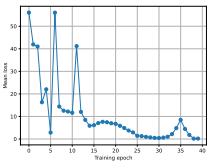
Figure 27: Effects on mean loss, joint angle output and PID gain adaptation when varying the amount of hidden layers present in the NN.



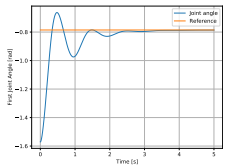
(a) 50 hidden layers, default mass, 1e-4s learning rate



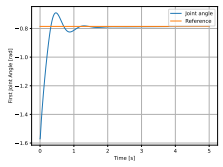
(b) 50 hidden layers, true mass, 1e-4s learning rate



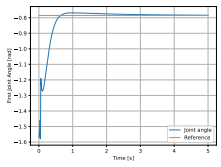
(c) 50 hidden layers, true mass, 1e-5s learning rate



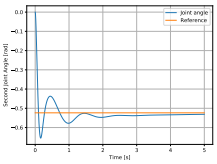
(d) 50 hidden layers, default mass, 1e-4s learning rate



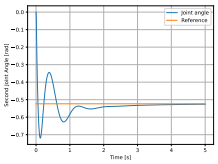
(e) 50 hidden layers, true mass, 1e-4s learning rate



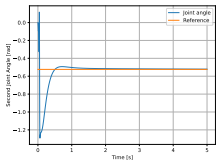
(f) 50 hidden layers, true mass, 1e-5s learning rate



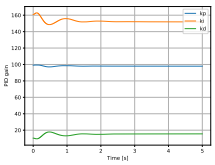
(g) 50 hidden layers, default mass, 1e-4s learning rate



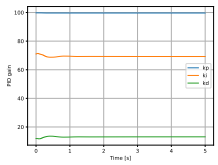
(h) Output of second joint angle for NN with 50 hidden layers, true mass and 1e-4s learning rate.



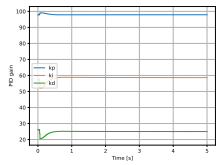
(i) 50 hidden layers, true mass, 1e-5s learning rate



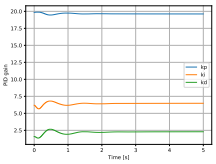
(j) 50 hidden layers, default mass, 1e-4s learning rate



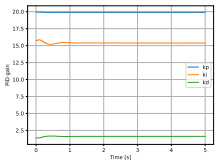
(k) 50 hidden layers, true mass, 1e-4s learning rate



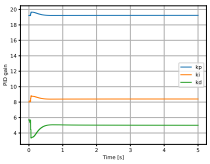
(l) 50 hidden layers, true mass, 1e-5s learning rate



(m) 50 hidden layers, default mass, 1e-4s learning rate

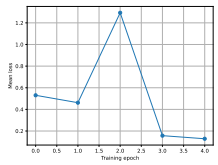


(n) 50 hidden layers, true mass, 1e-4s learning rate

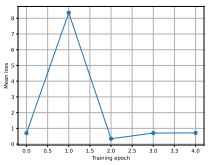


(o) 50 hidden layers, true mass, 1e-5s learning rate

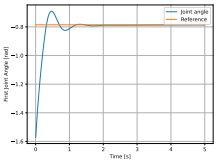
Figure 28: Effects on mean loss, joint angle output and PID gain adaptation with varying link mass and learning rate.



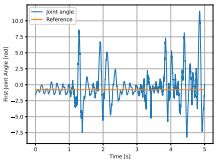
(a) 50 hidden layers, true mass, sigmoid output activation function



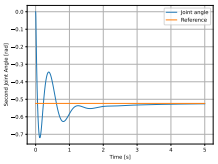
(b) 50 hidden layers, true mass, relu output activation function



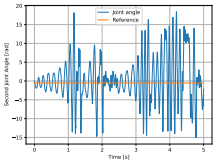
(c) 50 hidden layers, true mass, sigmoid output activation function



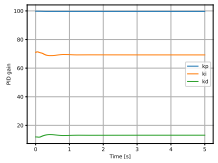
(d) 50 hidden layers, true mass, relu output activation function



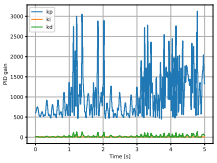
(e) 50 hidden layers, true mass, sigmoid output activation function



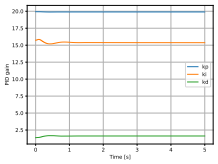
(f) 50 hidden layers, true mass, relu output activation function



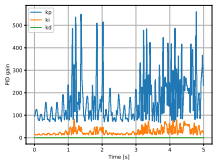
(g) 50 hidden layers, true mass, sigmoid output activation function



(h) 50 hidden layers, true mass, relu output activation function



(i) 50 hidden layers, true mass, sigmoid output activation function



(j) 50 hidden layers, true mass, relu output activation function

Figure 29: Effects on mean loss, joint angle output and PID gain adaptation with different output activation function.