Introduction to control systems
2020.1

TI0118 – Homework 2

Student name: Emmanuel Victor Barbosa Sampaio
Student number: 417180

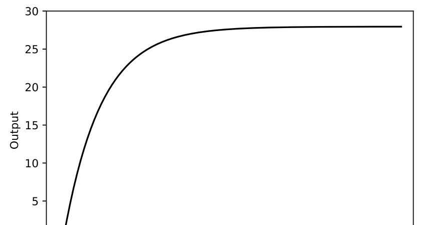
Exercise 1

The unit steps responses of two systems A and B are recorded and reported in the files attached to the homework assignment. In each file, the first column gives the time vector, t, and the second column provides the output response, y(t), for the systems A in `HW2_ex1_dataA.txt` and B in `HW2_ex1_dataB.txt`.

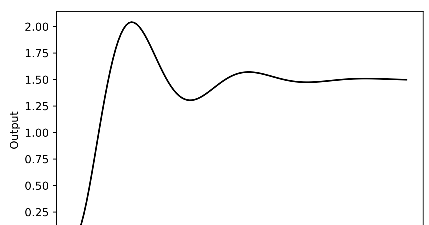
1. Load the data and plot the responses for systems A and B.
2. Identify the order of the systems. Based on the plots, estimate the transient response characteristics, such as time constant, settling time, rise time, peak time and percentage of overshoot. Write the corresponding transfer functions $T_A(s)$ and $T_B(s)$ for the systems A and B, respectively.
3. Plot and compare the step response of the systems $T_A(s)$ with the data provided in `HW2_ex1_dataA.txt`.
4. Plot and compare the step response of the systems $T_B(s)$ with the data provided in `HW2_ex1_dataB.txt`

Solution

Using the code 1 the data was loaded and plotted. The following plots show the result of this operation:



(a) Plot of the system A data.



(b) plot of the system B data.

The graphs provide the information that system A is a first order system and system B is a second order system. Let's start analysis by data from system A.

System A analysis

- From the data, the final value of the system A output is: 27.94.
- So 63% of its final value is: 17.60.
- Looking into the data 63% of the final value should happened in the following time range:

Time	Output
4.1585	17.3237
4.3565	17.8020

Table 1: This values defies a range to search the time constant

  So, the best time constant going to be found between this two values of time. The constant τ going to represent the time constant.

- A first order system has the following type:

$$G(s) = \frac{K}{\tau s + 1}$$

- So let's verify how the MSE of the data compare to the simulation of the transfer function varies when τ changes in the time range between 4.1585 and 4.3565.

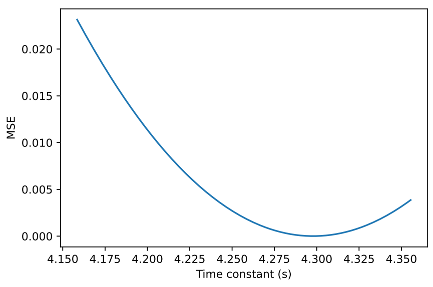


Figure 2: How the error varies as the time variant changes

  The time constant associated with the lowest error is: $\tau = 4.2975$. The code 2 was used to find the time constant associated with he lowest error and plot 2. Now some time constants can be calculated.

- The settling time: $T_s = \frac{4}{1/\tau} = 17.60s$
- The rise time: $T_r = \frac{2.2}{1/\tau} = 9.45s$

  The transfer function of the system is:

$$T_A(s) = \frac{6.503}{s + 0.2327} \tag{1}$$

  The code 3 was used to calculate the constants and calculate the transfer function of the system.

  The following plot compare the step response of the transfer function in 1 and the data:

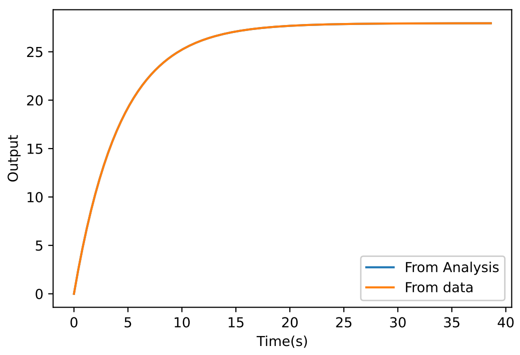


Figure 3: Comparing the step response of the transfer function and the data.

The MSE is: 7.32×10^{-6}, so we can verify that we have a good approximation of the system behavior.

For the system B

- From the plot, it's possible to assume that the system B is a uderdamped second order system. The general form for this system is:

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

- Searching in the data the peak time found was: $T_p = 1.0391$
- Using the formula:
%OS=(max output- final output value)*100/final output value
the overshoot found was: $OS = 36.256\%$
- Now that we have the overshoot we can calculate the damping ratio as:

$$\zeta = -\frac{\ln(\%OS/100)}{\sqrt{\pi^2 + (\ln(\%OS/100))^2}}$$

The damping ration found was $\zeta = 0.307$

- So the natural frequency can be calculated as:

$$\omega_n = \frac{\pi}{T_p\sqrt{1-\zeta^2}}$$

The natural frequency found was: $\omega_n = 3.17$

- Now we can use the code 4 to calculate the following time constants of the system B.
- The gain of the system: $K = 1.4985$
- The rise time of the system is: $T_r = 0.423$
- The settling time of the system is: $T_s = 4.09$

In conclusion, the transfer function of the system B:

$$T_B(s) = \frac{15.13}{s^2 + 1.953s + 10.09} \tag{2}$$

The code 6 was used to plot the step response of the $T_B(s)$ and compare it with the data:

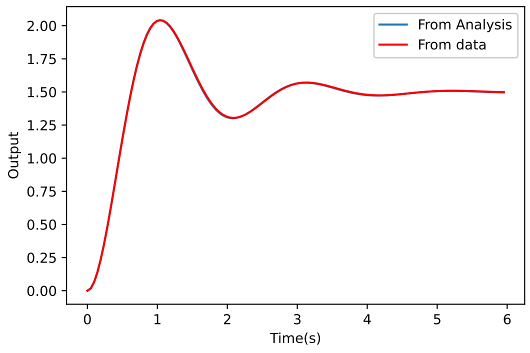


Figure 4: Comparing the step response of the transfer function and the data.

The mean square error is: 1.827×10^{-5}. Based on this, we have a good approximation of the system behavior.

Exercise 2

Consider a system with the transfer function in (3)

$$G(s) = \frac{\tau s + 1}{(0.1s + 1)(0.002s^2 + 0.02s + 1)(s^2 + 0.1s + 1)} \tag{3}$$

1. Consider $\tau = 0.5$:
    - Find the poles and zeros of the system. Plot them in a zero-pole map and draw some conclusions.
    - Use a dominant-pole argument to find an equivalent second-order transfer function. Can the zero be neglected in this case? Plot and compare the step responses for system as in 3 and the second order equivalent one. Explain the differences between the two responses
2. Consider $\tau = 20$:
    - Find the poles and zeros of the system. Plot them in a zero-pole map and draw some conclusions.
    - Use a dominant-pole argument to find an equivalent second-order transfer function. Can the zero be neglected? Plot and compare the step responses for system as in 3 and the second order equivalent one. Explain the differences between the two responses.

Solution

Consider $\tau = 0.5$

The zeros and poles of the transfer function are:

- For $(0.1s + 1) = 0$, $s = -10$ is a pole.

- For $(0.002s^2 + 0.02s + 1) = 0$
  The discriminant is $\Delta = -7.6 \times 10^{-3}$, so we have two complex poles: $p_1, p_2 = -5 \pm 21.79j$
- For $(s^2 + 0.1s + 1) = 0$
  The discriminant is $\Delta = -3.99$, so we have two complex poles: $p_1, p_2 = -0.05 \pm 0.99j$
- For $(0.5s + 1) = 0$, $s = -2$ is a zero.

  Therefore, this is the pole zero map:

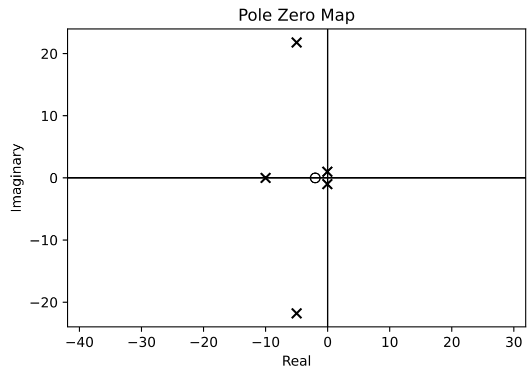


Figure 5: The poles and zeros of the system

Poles	$-5. + 21.79j$	$-5. - 21.79j$	-10	$-0.05 + 0.99j$	$-0.05 - 0.99j$
Zeros	-2				

Table 2: Table of the poles and zeros of the system.

There are two poles that are dominant, $p_1, p_2 = -0.05 \pm 0.99j$, the other three are more than 5 times bigger, so their impact on the system response is no so big, for this reason they can neglected. Also, the zero can be consider only as a simple gain. To verify what was said, the code 7 was used to simulate the system step response, the result is in the plot bellow.

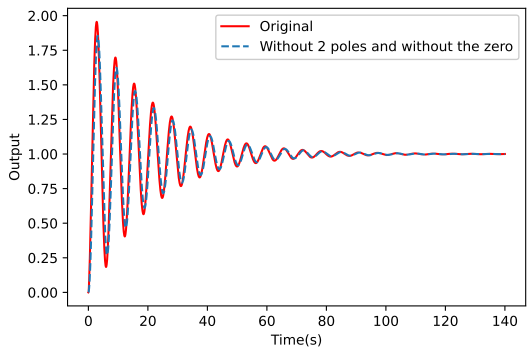


Figure 6: Compare the original system and the system without the poles.

The MSE is 0.005. Based on this, we can say that we can approximate the system transfer function to the following second order system:

$$G(s) = \frac{1}{s^2 + 0.1s + 1}$$

Consider $\tau = 20$

The poles are the same of the previous system, but there is a different zero:

- $(20s + 1) = 0$, $s = 0.05$ is a zero.

  Using the code 8 the following poles and zero plot were found:

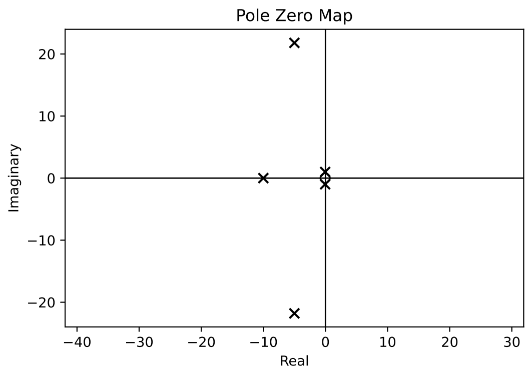


Figure 7: The pole and zero map of the transfer function.

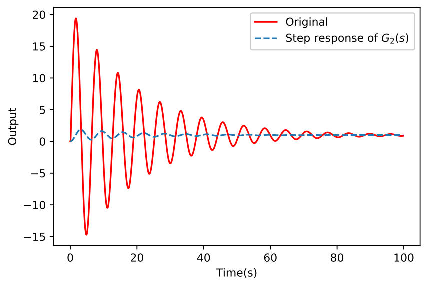
Poles	$-5. + 21.79j$	$-5. - 21.79j$	-10	$-0.05 + 0.99j$	$-0.05 - 0.99j$
Zeros	-0.05				

Table 3: Table of the poles and zeros of the system.

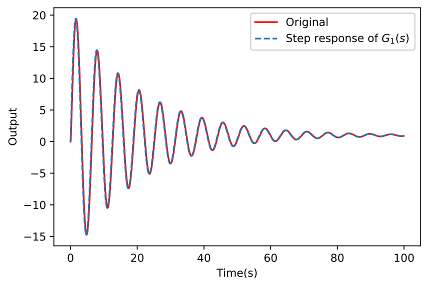
The zero is close to the dominant poles, we going to see the impact of this in next simulations, and there are 3 poles far from the dominant ones, so they can be neglected, as was presented in the first item. In this sense, we can compare the response of the original system with the response of the following transfer functions:

$$G_1(s) = \frac{20s + 1}{s^2 + 0.1s + 1}$$
$$G_2(s) = \frac{1}{s^2 + 0.1s + 1}$$

$G_1(s)$ is the transfer function of the system with a zero and the 3 non dominant poles removed. $G_2(s)$ is the transfer function without the 3 non dominant poles and also without the zero. The code 8 was used to get the following plots:



(a) Comparing $G_2(s)$ step response against the step response of the original transfer function.



(b) Comparing $G_1(s)$ step response against the step response of the original transfer function.

The plots reflect the relation between the zero near to the origin and the system response, this comes from the fact that the zero near to the dominant poles has an greater impact on the system amplitude compare to the zero far from them, which can be considered as simple gain as was showed in the first item.

In conclusion, $G_1(s)$ represents a better approximation of the system behavior, for this reason we can represent $G(s)$ as:

$$G(s) = \frac{20s+1}{s^2+0.1s+1} \tag{4}$$

Exercise 3

Given the state-space model in (5):

$$\begin{cases} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \end{cases} \tag{5}$$

1. Define the transfer function W (s) of the given system and evaluate the its BIBO stability. Briefly explain the concept of BIBO stability.
2. Define the Lyapunov's stability of the given system. Briefly explain the concept of stability according to Lyapunov. How does it differ from the BIBO stability?

Solution

The transfer function can be calculated as:

$$W(s) = C(sI-A)^{-1}B \tag{6}$$

Calculating the inverse:

$$(sI-A)^{-1} = \frac{Adj(sI-A)}{Det(sI-A)}$$

Where $Adj(sI-A)$ is the adjunct matrix of $(sI-A)$. The adjunct matrix is the transpose of the cofactor matrix:

$$Adj(sI-A) = \begin{bmatrix} s+2 & -5 \\ 1 & s \end{bmatrix}$$

The inverse of $(sI-A)^{-1}$ is calculated as follows:

$$(sI-A)^{-1} = \frac{1}{s^2+2s+5} \begin{bmatrix} s+2 & -5 \\ 1 & s \end{bmatrix} \tag{7}$$

Using (7) in (6):

$$\frac{1}{s^2+2s+5} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s+2 & -5 \\ 1 & s \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \frac{2s}{s^2+2s+5}$$

In conclusion:

$$C(sI-A)^{-1}B = W(s) = \frac{2s}{s^2+2s+5}$$

We can find the same answer using the code 9.
Calculating the poles of the system:

$$s^2 + 2s + 5 = 0$$
$$\Delta = 4 - 20 = -16$$
$$p_1,\ p_2 = -1 \pm 2j$$

Using 10 we get the following:

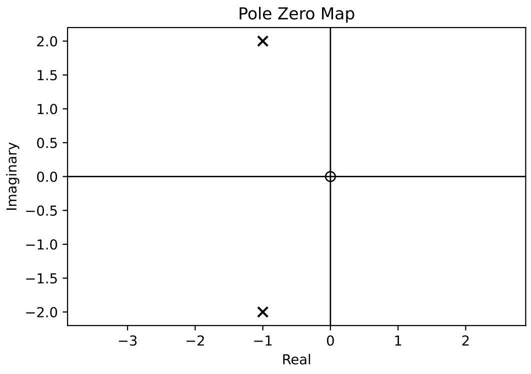


Figure 9: The poles and zeros of the transfer fucntion. Look the each poles is in the left hand side of the system.

The transfer function of the system has 2 poles in the left hand side of the Argan-Gauss plan. As a result, the exponential terms in the system time response going to have an decreasing behavior, so as the time pass the system response goes to tend to a equilibrium point. For this reason, a bounded input can only deliver the system response for a bounded response, This is also called a BIBO stable system. We can verify this by seen the step response of the system. The code used to perform this plot was 11.

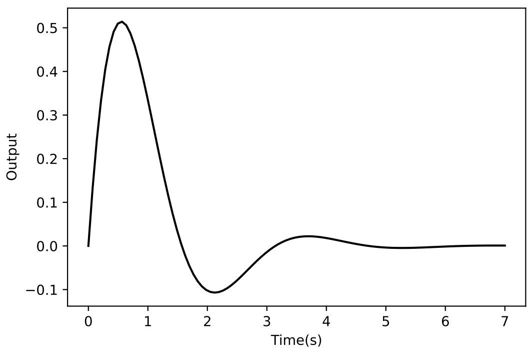


Figure 10: The system step response.

Lyapunov stability of the system

For a linear system we can divide the system response in the input response and the state response. The first one is related on how the system behavior changes due to the interaction of some manipulated variable. The other one can be seen as the effect of the initial states variables values in the system response. This is exemplify by the following equation:

$$x(t) = e^{At}x_0 + \int_0^{\tau} e^{A(t-\tau)}Bu(\tau)d\tau \tag{8}$$

Therefore, the state response can be considered to be only:

$$x(t) = e^{At}x_0 \tag{9}$$

Based on the previous equation, the system can go from some initial position to some $x(t)$, that could or not be near to an equilibrium point of the system. This is the context that the Lyapunov stability take place, because Lyapunov verifies if a system can naturally goes from some point in the space back to the equilibrium point. In this context, naturally means that the system is driving by the relation of the states. There are many examples of this kind of system, for example the equilibrium in natural chemical reactions, where there are reagents that are consumed and other are produced until an stabilization. In the other hand there are some system that are not naturally drive, for example the stick up is an equilibrium position for a stick, but it can not be drive to the up position naturally it tends to move far from this point.

In this context, Lyapunov call a system that can goes from an initial state to the equilibrium point as asymptotically stable, if the system tends to a point that is not the equilibrium the system is said to be marginally stable. Mathematically the Lyapunov stability can be introduced as follows:

- The system is marginally stable if and only if all eigenvalues of A have zero or negative real parts and those with zero real parts are simple roots of the minimal polynomial of A .
- The system is asymptotically stable if and only if all eigenvalues of A have negative real parts.

In the system represented by (5) the code 12 was used to verify the system eigenvalues, the result are in the table below:

Eigenvalue	Real part	Imaginary part
λ_1	-1	2i
λ_2	-1	-2i

Table 4: The eigenvalues of the system

In conclusion, based on the previous table the system is said to be asymptotically stable. The code 13 creates the following plot, that plot show how the system goes from a different initial state to zero when there is no input applied.

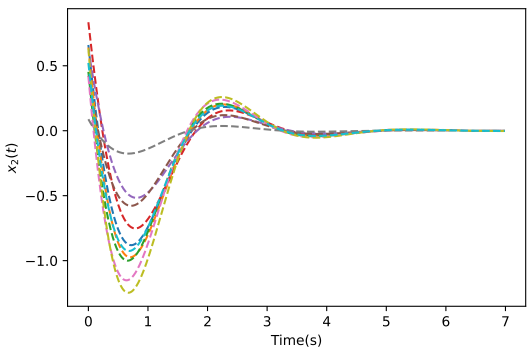


Figure 11: The system state response for different initial conditions

Exercise 4

For the unit feedback system in figure , G(s) is given as:

$$G(s) = \frac{s+4}{s(s^2+3.2s+2.4)}$$

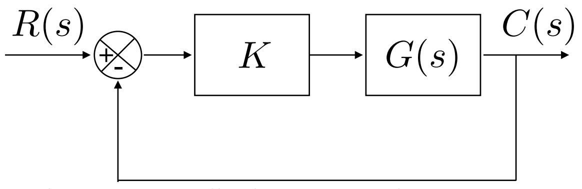


Figure 12: Feedback system used in Exercise 4.

1. Use the Routh-Hurwitz criterion to define:
    - The range of K that provides stability to the closed-loop system.
    - The range of K that makes the system oscillate.
2. Plot and discuss the step responses for stable and oscillatory values of K.

Solution

The transfer function of the closed loop system can be calculated as:

$$\frac{R(s)}{C(s)} = T(s) = \frac{KG(s)}{1+KG(s)}$$

Replacing the G(s), the following transfer function was found:

$$T(s) = \frac{K(s+4)}{s^3+3.2s^2+2.4s+K(s+4)}$$

Let's verify how is the Routh table of the system:

s^3	1	$(2.4+K)$
s^2	3.2	4K
s	$(3.2(2.4+K)-4K)/3.2$	0
s^0	4K	0

Table 5: Routh table of the transfer function

To avoid the poles in the right hand side of the Argan-Gauss plane the table can't have signal changes in the first column, so:

- $\frac{(3.2(2.4+K)-4K)}{3.2} > 0$. Then, $9.6 > K$. So, the value of K for the system be stable is: $0 < K < 9.6$.
- The following plot show how the poles of the system varies as the value of the K keeps on the range and when the values of the K goes out of the range:

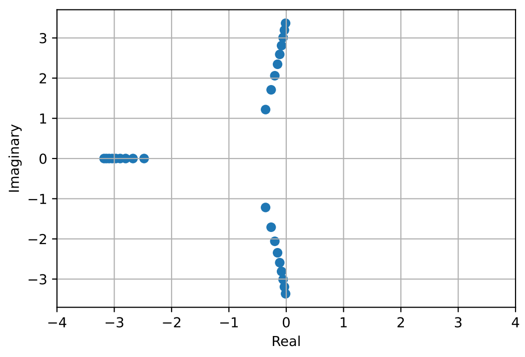


Figure 13: Poles when 9.6>K>0

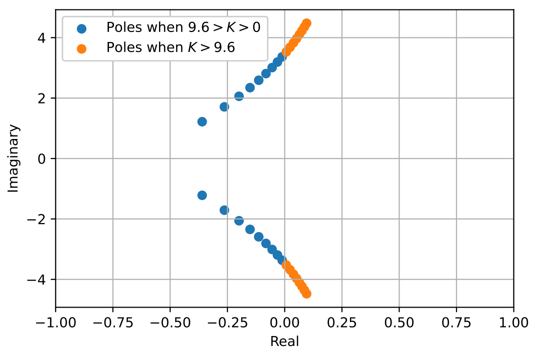


Figure 14: Showing the poles when $K > 9.6$

When $K = 9.6$ we going to have the following situation in the Routh table:

s^3	1	$(2.4 + K)$
s^2	3.2	4K
s	0	0
s^0	?	0

Table 6: Routh table of the transfer function

To by-pass this situation it's necessary to use the derivative of the even polynomial:

$$p(s) = 3.2s^2 + 4 * (9.6) = 0$$
$$\frac{dp(s)}{ds} = 6.4s$$

So, we replace the 0 for 6.4:

s^3	1	$(2.4 + K)$
s^2	3.2	4K
s	6.4	0
s^0	4*9.6	0

Table 7: Routh table of the transfer function

The row of zeros is caused by an even polynomial of degree 2, which has symmetric roots. The missing of signal changes bellow the row of zeros gives the information that the even polynomial has no roots in the RHP, so to keep the symmetry of its roots they need to be in the imaginary axis. To show this the code 15 was used to plot the poles and zero map of the system.

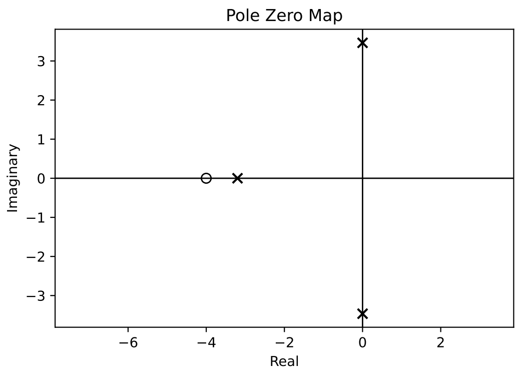


Figure 15: Showing the poles when $K = 9.6$.

The following table summary the result found:

$K \in (-\infty, 0)$	The system is unstable.
$K \in (0, 9.6)$	The system is stable
$K = 9.6$	The system is marginally stable
$K \in (9.6, +\infty)$	The system is unstable

Table 8: How the stability varies as K varies.

To exemplify what was said in the last sections the following plot was made varying the value of K in the range of (0, 9.6). It's is possible to verify that the system response goes to stabilize, but when K increases in the range the amplitude and time for stabilization increases. This is a reflect of the approximation of the poles to the imaginary axis which represents a reduction of the negative real part and a increases of the imaginary part.

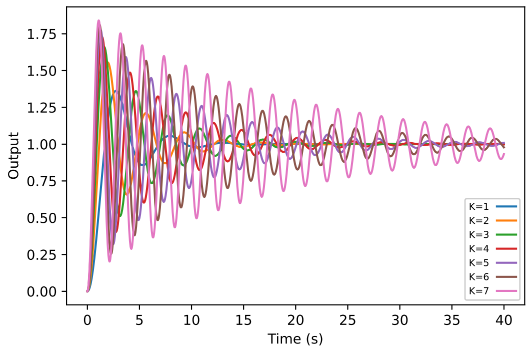


Figure 16: Step response of the closed loop system for different values of K.

Now when K is equal to 9.6, the response does not goes to stabilize, but it not going to infinity. The response is a pure oscilation with a constant amplitude, this reflects the poles that are in the imaginary axis.

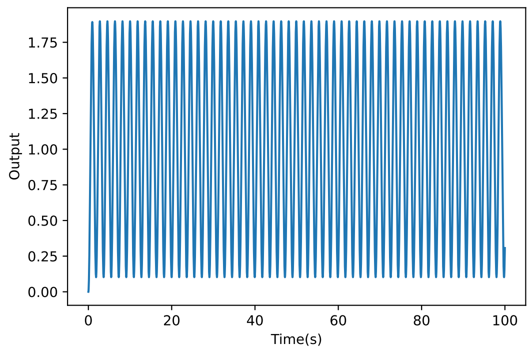


Figure 17: Step response of the system when K=9.6

Exercise 5

A unity feedback system has the forward transfer function given by:

$$G(s) = \frac{K(s+7)}{s(s^3+25s^2+196s+480)} \tag{10}$$

1. Evaluate the system type.
2. Find the value of K to yield a 1% error in steady-state for an input of 0.1t.
3. Find the static error constants for the value of K found in point (2) of this exercise.
4. Verify the stability of your system and plot its step response.

Solution

The system is type 1, because the number of integrators in the forward path is 1.

The error in the s domain is given by:

$$\begin{aligned}
\mathcal{L}\{e(t)\} = E(s) = Y(s) - R(s) \\
E(s)G(s) = Y(s) \\
\frac{R(s)G(s)}{1+G(s)} = Y(s) \\
E(s) = R(s)\frac{1}{1+G(s)}
\end{aligned}$$

For a stable system, the final value theorem can be used to calculate the stationary error. I going to suppose that the system is stable, and them calculate K, after find it I going to check using the Routh Table if for this value the system is stable.

$$e(\infty) = \lim_{s\to 0} sE(s) \tag{11}$$

So, for $r(t) = 0.1t$:

$$e(\infty) = \lim_{s\to 0} s\frac{0.1}{s^2}\frac{1}{1+G(s)} = \frac{0.1}{\lim_{s\to 0} sG(s)} = \frac{480 * 0.1}{7K}$$

So for $e(\infty) = 0.01$. $K = \frac{480*0.1}{0.01*7} = 685.71$.
Now, let's certify that the system is stable:

$$T(s) = \frac{G(s)}{1+G(s)} = \frac{K(s+7)}{K(s+7) + s(s^3+25s^2+196s+480)} = \frac{685.71(s+7)}{s^4+25s^3+196s^2+1165.71s+4799.97}$$

After that we can create the Routh table:

s^4	1	196	4799.97
s^3	25	1165.71	0
s^2	149.37	4799.97	0
s^1	365.34	0	0
s^0	4799.97	0	0

Table 9: The Routh table of the system. There are no signals change in the first column, so the system is stable

From the Routh table we can verify that the system is stable, because there is no signal change in the first column. To showing this explicit the pole zero map was plot:

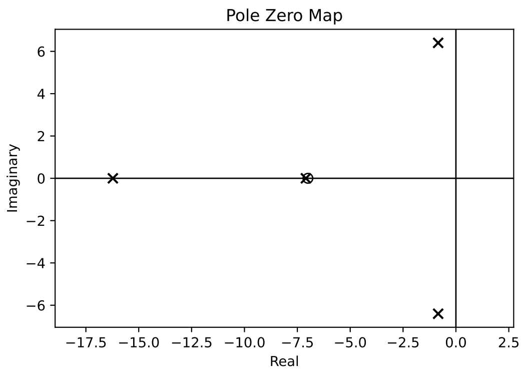


Figure 18: The poles of the closed loop system

Using 17 we got the following plot for the system step response.

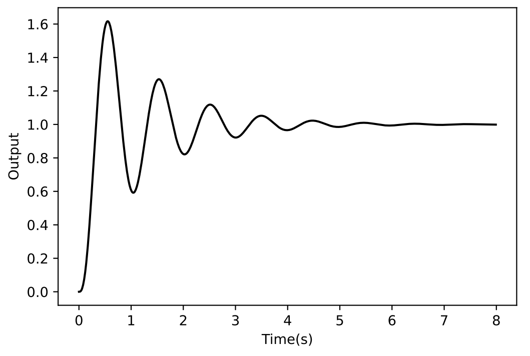


Figure 19: The step response of the system

Then, the static error constants were calculated.

- Position Constant: $K_p = \lim_{s\to 0} G(s) = \infty$. This value comes from the fact that the forward path transfer function has 1 pole in the origin, so when s goes to zero, G(s) goes to tend to the infinity. It gives the information that for a constant input, the error in the stationary goes to be zero, since the error is:

$$e(\infty) = \lim_{s\to 0} \frac{1}{1+G(s)} = 0$$

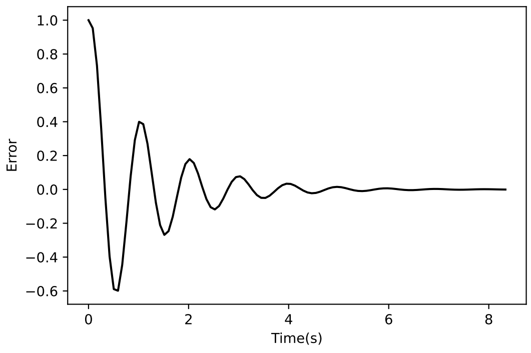


Figure 20: The error for a step input

- Velocity Constant: $K_v = \lim_{s\to 0} sG(s) = \frac{7K}{480} = \frac{0.1}{0.01} = 10$. This values shows that for a input that increases in a linear relation with the time such as the ramp, the system goes to have some constant error. For the $r(t) = 0.1t$

$$e(\infty) = \lim_{s\to 0} \frac{0.1}{sG(s)} = \frac{0.1}{K_v} = \frac{0.1}{10} = 0.01$$

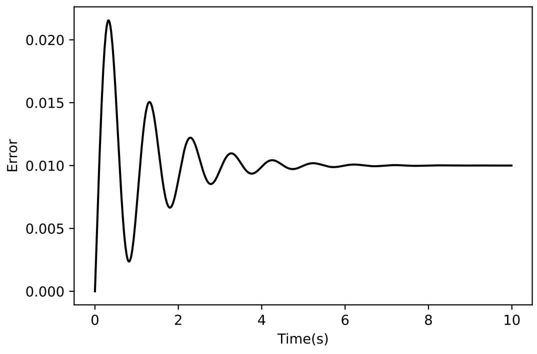


Figure 21: The error for $r(t) = 0.1t$.

- Acceleration constant $K_a = \lim_{s\to 0} s^2G(s) = 0$. This value comes from the fact that G(s) does not have enough poles to cancel the s^2, so this value goes K_a goes to zero. For a $r(t) = t^2$ the error is:

$$e(\infty) = \lim_{s\to 0} \frac{1}{s^2G(s)} = \frac{1}{K_a} = \infty$$

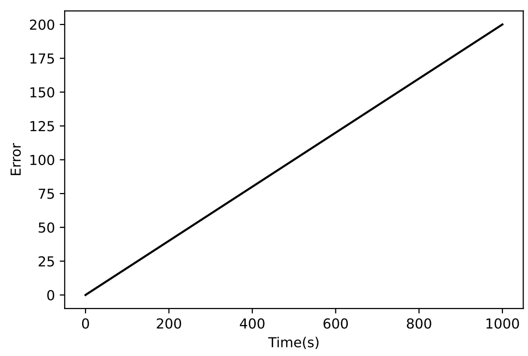


Figure 22: The error for a parabola

Exercise 6

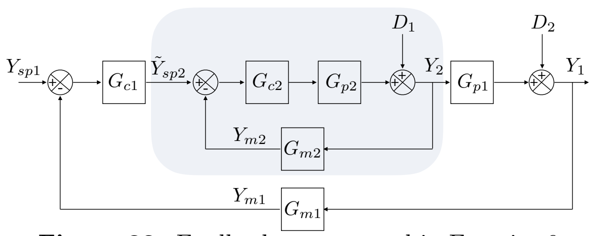


Figure 23: Feedback system used in Exercise 6

The Figure 23 represents a cascade control system of the temperature, Y_1 , in a small room which is achieved by adjusting the hot/cold water flow, Y_2 , adjusted by an inner control loop (coloured in Figure 23).

The relationship between the room temperature and the water low rate can be represented with the transfer function G_{p1} , whereas G_{p2} corresponds to the conditioning unit. G_{c2} represents the water flow rate controller and G_{c1} keeps the room temperature at the desired set-point by defining the needed water flow rate. G_{m1} and G_{m2} correspond to the temperature and flow rate measuring devices, respectively.

The disturbances can influence the system: D_1 represents unwanted changes in the water pressure inside the pipes and might affect Y_2 and D_2 represents possible variations in the external temperature and might affect Y_1.

1. Assuming $D_1 = D_2 = 0$, find the input-output transfer functions for the inner and outer loops: Y_1/Y_{sp1} and Y_2/Y_{sp2}.
2. Assuming $Y_{sp1} = D_1 = 0$, find the disturbance-output transfer functions: Y_1/D_2
3. Assuming $Y_{sp1} = D_2 = 0$, find the disturbance-output transfer functions: Y_2/D_1.
4. Define the characteristic equation of the cascade control system in Figure 2 and analyses the stability of the system.
5. For $K_{c2} = 1$, verify whether is possible or not simplify the input-output $\frac{Y_1(s)}{Y_{sp1}}$ representation found in point (1) of this exercise.
6. Compare the two cases (with and without simplification) by plotting the response of the system to a unit step input. From the plot, identify the steady-state value, the percentage overshoot and settling time in the two cases.

$$\begin{aligned}
G_{p1}(s) =& \frac{K_{p1}}{(1+\tau_{p1,1}s)(1+\tau_{p1,2}s)} = \frac{0.1}{1000s^2+70s+1}\\
G_{p2}(s) =& \frac{K_{p2}}{1+\tau_{p2}s} = \frac{1}{1+10s}\\
G_{c1}(s) =& K_{c1} = 100\\
G_{c2}(s) =& K_{c2}\frac{1+\tau_{c2}s}{s} = K_{c2}\frac{1+100s}{s}\\
G_{m1}(s) =& 1; \quad G_{m2}(s) = \frac{K_{m2}}{1+\tau_{m2}s} = \frac{1}{1+s}\\
G_{d1}(s) =& 1; \quad G_{d2}(s) = 1
\end{aligned}$$

Solution

First, let's have a look of the system inside:

$$\begin{aligned}(Y_{sp2} - G_{m2}Y_2) &= E_2(s)\\ E_2(s)G_{c2}G_{p2} &= Y_2(s)\\ (Y_{sp2} - G_{m2}Y_2) &= \frac{Y_2(s)}{G_{c2}G_{p2}}\\ Y_{sp2}(s)G_{c2}(s)G_{p2}(s) &= Y_2(s)(1 + G_{m2}(s)G_{c2}(s)G_{p2}(sc))\end{aligned}$$

So, the transfer function $H_2(s)$:

$$H_2(s) = \frac{Y_2(s)}{Y_{sp2}(s)} = \frac{G_{c2}G_{p2}}{1 + G_{m2}G_{c2}G_{p2}} = \frac{K_{c2}(1 + 100s)(s + 1)}{s(s + 1)(1 + 10s) + K_{c2}(1 + 100s)}$$

For $K_{c2} = 1$:

$$H_2(s) = \frac{100s^2 + 101s + 1}{10s^3 + 11s^2 + 101s + 1} \tag{12}$$

Now for the system outside:

$$\begin{aligned}Y_{sp1}(s) - G_{m1}Y_1(s) &= E_1(s)\\ E(s)G_{c1}H_2(s)G_{p1} &= Y_1(s)\\ (Y_{sp1}(s) - G_{m1}Y_1(s))G_{c1}H_2(s)G_{p1} &= Y_1(s)\end{aligned}$$

So the transfer function $H_1(s)$

$$\begin{aligned}H_1(s) &= \frac{Y_1(s)}{Y_{sp1}(s)} = \frac{G_{c1}(s)H_2(s)G_{p1}(s)}{1 + G_{m1}(s)G_{c1}(s)H_2(s)G_{p1}(s)}\\ H_1(s) &= \frac{10K_{c2}(1 + 100s)(s + 1)}{(1000s^2 + 70s + 1)(s(s + 1)(1 + 10s) + K_{c2}(1 + 100s)) + 10K_{c2}(1 + 100s)(s + 1)}\\ H_1 &= \frac{10K_{c2}(1 + 100s)(s + 1)}{10^4s^5 + 1.1710^4s^4 + (1780 + 10^5K_{c2})s^3 + (81 + 910^3K_{c2})s^2 + (1 + 1180K_{c2})s + 11K_{c2}}\end{aligned}$$

For $K_{c2} = 1$:

$$H_1(s) = \frac{1000s^2 + 1010s + 10}{10000s^5 + 11700s^4 + 101780s^3 + 9081s^2 + 1181s + 11} \tag{13}$$

Now for $Y_{sp1} = D_1 = 0$, the transfer function of the disturbance-output $Y_1(s)/D_2(s) = H_3(s)$.

$$\begin{aligned}-Y_1(s)G_{m1}G_{c1}H_2(s)G_{p1}(s) + D_2 &= Y_1(s)\\ \frac{Y_1(s)}{D_2(s)} &= \frac{1}{1 + G_{m1}G_{c1}H_2(s)G_{p1}(s)}\end{aligned}$$

For the $K_{c2} = 1$ the transfer function $H_3(s)$ is:

$$H_3(s) = \frac{10^4s^5 + 1.17 \times 10^4s^4 + 1.018 \times 10^5s^3 + 8081s^2 + 171s + 1}{10^4s^5 + 1.17 \times 10^4s^4 + 1.018 \times 10^5s^3 + 9081s^2 + 1181s + 11} \tag{14}$$

Now, the disturbance-output transfer function $Y_2(s)/D_1(s) = H_4(s)$:

$$\begin{aligned} D_1(s) - Y_2(s)G_{m2}G_{c2}G_{p2} &= Y_2(s) \\ D_1(s) &= Y_2(s)G_{m2}G_{c2}G_{p2} + Y_2(s) \\ H_4(s) &= \frac{Y_2(s)}{D_2(s)} = \frac{1}{G_{m2}G_{c2}G_{p2} + 1} \end{aligned}$$

For $K_{c2} = 1$:

$$H_4(s) = \frac{10s^3 + 11s^2 + s}{10s^3 + 11s^2 + 101s + 1} \tag{15}$$

From $H_1(s)$, the characteristic polynomial

$$p(s) = 10^4 s^5 + 1.1710^4 s^4 + (1780 + 10^5 K_{c2})s^3 + (81 + 910^3 K_{c2})s^2 + (1 + 1180K_{c2})s + 11K_{c2}$$

To simplify a little bit the analysis, I choose to make the following approximations:

$$\begin{aligned} (1780 + 10^5 K_{c2}) &\approx 10^5 K_{c2} \\ (81 + 910^3 K_{c2}) &\approx 910^3 K_{c2} \\ (1 + 1180K_{c2}) &\approx 1180K_{c2} \end{aligned}$$

s^5	10^4	$10^5 K_{c2}$	$1180K_{c2}$
s^4	11700	$910^3 K_{c2}$	$11K_{c2}$
s^3	$92307.69K_{c2}$	$1170.59K_{c2}$	0
s^2	$910^3 K_{c2} + 148.37 = \sigma_1$	$11K_{c2}$	0
s^1	$(\sigma_1 * 1170.59K_{c2} - 11K_{c2} * 92307.69K_{c2})/\sigma_1$	0	0
1	$11K_{c2}$	0	0

Table 10: Routh Table of the exercise.

To not have signal changes in the first column it's necessary that:

$$\begin{aligned} 910^3 K_{c2} + 148.37 > 0 \therefore K_{c_2} > -0.0164 \\ (\sigma_1 * 1170.59K_{c2} - 11K_{c2} * 92307.69K_{c2}) > 0 \therefore K_{c2} > -0.01824 \end{aligned}$$

From this approximate simple analysis, it's possible to say that for all positive values of K_{c2} the system is stable.

For $K_{c1} = 1$

The transfer function is presented in the equation 13. The poles of the system are:

Poles	Real	Imaginary
s_1	-0.5405	+ 3.1270i
s_2	-0.5405	- 3.1270i
s_3	-0.0395	0.1356i
s_4	-0.0395	- 0.1356i
s_5	-0.0100	+ 0.0000i

Table 11: The poles of the system

Zeros	Real	Imaginary
Z_1	-1.0000	0
Z_2	-0.0100	0

Table 12: The zeros of the system

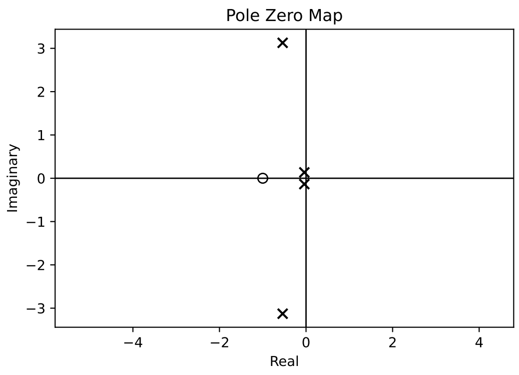


Figure 24: Pole zero map of the system

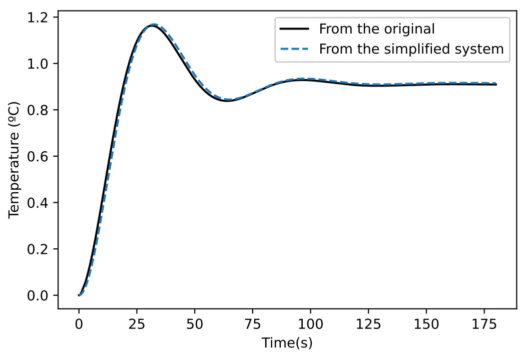
There is a zero pole cancellation, an also there are some poles that are much far from the origin as the dominant poles,$-0.0395 \pm -0.1356j$, we can verify this looking into the poles of the system. So the system can be represented by a second order transfer function:

$$T(s) = \frac{K}{(s + 0.0395 + 0.0968j)(s + 0.0395 - 0.0968j)}$$

The value of K can be calculated using the final value theorem and the original system simulation, using the code 18 we can verify that $K = 0.0099$. Then it's possible to approximate the system to the following transfer function:

$$T(s) = \frac{0.01}{(s + 0.0395 + 0.0968j)(s + 0.0395 - 0.0968j)} \tag{16}$$

The plot below shows how the step response of the system is near to the step response of $T(s)$.



The MSE is 0.0026, so we can have this as system approximation. Using the same formulas of the second question and with the data from the step response the code 20 was used to give the information ahead.

For the transfer function $H_1(s)$:

- Settling time: 101.17463342822805
- Peak Time: 32.45
- The natural frequency: 0.10457478294286249
- The damping ratio: 0.3780605657969098
- Overshoot 27.722376800944076
- The gain of the system: 0.9150664921824592

Using the same code for $T(s)$

- Settling time: 98.85555378922112
- Peak Time: 31.52
- The natural frequency: 0.10757013178822733
- The damping ratio: 0.37615532761901993
- Overshoot 27.93202729653574
- The gain of the system: 0.9091545682955544

Appendix

Homework Notebook

There is a python notebook for the homework: Notebook with the answer. The notebook has the following advantages:

- Each code is separated for each exercise.
- The notebook already has all modules used in this homework installed.

A Source code for exercise 1

Listing 1: Code to load the data and plot.

```python
import pandas as pd
import matplotlib.pyplot as plt
## Loading the data.
sys_A_df = pd.read_csv('sysA.csv')# You need to have sysA.csv.
sys_B_df = pd.read_csv('sysB.csv')# You need to have sysB.csv.

## Plot the response for the first system
plt.plot(sys_A_df['Tempo'],sys_A_df['Output'],'k')
plt.grid('on')
plt.xlabel('Time(s)')
plt.ylabel('Output')
plt.ylim(top=30)
plt.xlim(right=40)
plt.show()

### Plot the response for the second system.
plt.plot(sys_B_df['Tempo'],sys_B_df['Output'],'k')
plt.grid('on')
plt.xlabel('Time (s)')
plt.ylabel('Output')
plt.savefig('EXE_1_SYS_B.eps',format='eps')
plt.show()
```

Listing 2: Code to verify the MSE for different values of time constant.

```python
import numpy as np
from sklearn.metrics import mean_squared_error as mse
import matplotlib.pyplot as plt
import pandas as pd
sys_A_df = pd.read_csv('sysA.csv')# You need to have sysA.csv.
Final_Value=sys_A_df['Output'].values[-1]
time_constant_output = 0.63*Final_Value

initial_value_time = (sys_A_df.loc[sys_A_df['Output']<=time_constant_output]['Tempo'].
    values[-1])
final_value_time  = (sys_A_df.loc[sys_A_df['Output']>=time_constant_output]['Tempo'].
    values[0])

time_constant_range=np.arange(initial_value_time,final_value_time,0.001)
error_vector =[]
for time_constant in time_constant_range:
    a = 1/time_constant
    sys_A_tf = tf([Final_Value*a],[1,a])
    y_out,T=step(sys_A_tf,np.arange(0,196*0.198,0.198))
    error= mse(sys_A_df['Output'],y_out)
    error_vector.append(error)

time_constant=time_constant_range[error_vector.index(min(error_vector))]
a = 1/time_constant
```

```

plt.plot(time_constant_range,error_vector)
plt.ylabel('MSE')
plt.xlabel('Time constant (s)')
plt.savefig('EXE_1_MSE_TIME_CONSTANT.eps',format='eps')
plt.show()
```

Listing 3: Code to find the system A transfer function and constant values.

```
sys_A_tf = tf([Final_Value*a],[1,a])

step_info_sys_A = stepinfo(sys_A_tf)
settling_time = 4/a
rise_time = 2.2/a

print("Final values (Gain): ",Final_Value)
print("Value in the time constant: ",time_constant_output)
print("Value of the constant:",a)
print("Time constant: ",time_constant)
print("Settling time (from stepinfo): ",step_info_sys_A['SettlingTime'])
print("Rise time (from stepinfo): ",step_info_sys_A['RiseTime'])
print("Settling time: ",settling_time)
print("Rise time: ",rise_time)

print("The transfer function: ",sys_A_tf)
```

Listing 4: Code to find system B transfer function and constant values.

```
final_value = sys_B_df['Output'].values[-1]
max_value = max(sys_B_df['Output'].values)
peak_time = sys_B_df.loc[sys_B_df['Output']==max_value]['Tempo'].values[0]
OS = ((max_value-final_value)/final_value)*100


aux = np.log(OS/100)
damping_ratio = -aux/(np.pi**2+aux**2)**0.5

omega_n = np.pi/(peak_time*(1-damping_ratio**2)**0.5)

settling_time_b = 4/(damping_ratio*omega_n)


K = final_value

sys_B_tf = tf([K*omega_n**2],[1,2*omega_n*damping_ratio,omega_n**2])
step_info_sys_B = stepinfo(sys_B_tf)

print("Rise Time (from stepinfo):",step_info_sys_B['RiseTime'])
print("Settling time:",settling_time_b)
print("Peak Time:",peak_time)
print("The natural frequency: ",omega_n)
print("The damping ratio:",damping_ratio)
print("Overshoot",OS)
print("The gain of the system: ",K)
print(sys_B_tf)
```

Listing 5: Code to plot the system response and the data

```
y_out,T=step(sys_A_tf,np.arange(0,196*0.198,0.198))
plt.plot(T,y_out,label='From Analysis')
plt.plot(sys_A_df['Tempo'],sys_A_df['Output'],label='From data')
plt.legend()
plt.grid('on')
plt.xlabel('Time(s)')
plt.ylabel('Output')
plt.savefig('EXE_1_COMPARE_SYS_A.eps',format='eps')
plt.show()
```

Listing 6: Code to plot the system response and the data

```
y_out_b,T=step(sys_B_tf,np.arange(0,6,6/127))
plt.plot(T,y_out_b,label='From Analysis')
plt.legend()
plt.plot(sys_B_df['Tempo'],sys_B_df['Output'],label='From data',color='r')
plt.legend()
plt.xlabel("Time(s)")
plt.ylabel("Output")
plt.savefig('EXE_1_COMPARE_SYS_B.eps',format='eps')
plt.show()
```

B Source code for exercise 2

Listing 7: Code for the $\tau = 0.5$

```
from control.matlab import tf,pzmap
import matplotlib.pyplot as plt
from numpy import arange

## Set the original poles and zero map.
sys_original = tf([0.5,1],[0.0002,0.00402,0.1206,1.016,0.22,1])
poles_1,zeros = pzmap(sys_original)
plt.savefig('EXE_2_POLE_ZERO_MAP.eps',format='eps')

## Set the s
sys_2 = tf([1],[1,0.1,1]) # System without poles and zeros.

## Simulation of the system:
T_aux = arange(0,140,0.01)
y_original,T=step(sys_original,T_aux)
y_simplified,T = step(sys_2,T_aux)

## Plot of the simulation:

plt.plot(T,y_original,color='r',label='Original')
plt.plot(T,y_simplified,'--',label='Without 2 poles and without the zero')
plt.xlabel('Time(s)')
plt.ylabel('Output')
plt.legend()
plt.savefig('EXE_2_ITEM_2.eps',format='eps')
plt.show()
```

Listing 8: Code for $\tau = 20$.

```
from control.matlab import tf,pzmap
import matplotlib.pyplot as plt
from numpy import arange


## The original system transfer function and poles and zero plot.
sys = tf([20,1],[0.0002,0.00402,0.1206,1.016,0.22,1])
poles,zeros = pzmap(sys_2q_item_2)
plt.savefig('EXE_2_POLE_ZERO_MAP_2.eps',format='eps')

## System with poles removed and with zero.
sys_2 = tf([20,1],[1,0.1,1])

## System with poles removed and without the zero.
sys_3 = tf([1],[1,0.1,1])

## Simulations of the different systems.
y_out,T=step(sys_2q_item_2,T_aux)
y_out_2,T = step(sys_2,T_aux)
y_out_3,T = step(sys_3,T_aux)

## Plotting of the simulations
```

```

plt.plot(T,y_out,color='r',label='Original')
plt.plot(T,y_out_2,'--',label='With poles removed and with the zero removed')
plt.xlabel('Time(s)')
plt.ylabel('Output')
plt.legend()
plt.savefig('EXE_2_ITEM_3.eps',format='eps')
plt.show()

plt.plot(T,y_out,color='r',label='Original')
plt.plot(T,y_out_3,'--',label='With poles removed and with the zero removed')
plt.xlabel('Time(s)')
plt.ylabel('Output')
plt.legend()
plt.savefig('EXE_2_ITEM_4.eps',format='eps')
plt.show()
```

C Source code for exercise 3

Listing 9: Code to convert from state space to transfer function.

```
from contro.matlab import ss2tf, pzmap
import matplotlib.pyplot as plt
A = [[0,1],[-5,-2]]
B = [[0],[2]]
C = [0,1]
D = [0]
sys = ss2tf(A,B,C,D)
poles,zero=pzmap(sys)
plt.savefig('EXE_3_POLE_ZERO_MAP.eps',format='eps')
print("The poles of the system: ",poles)
```

Listing 10: Code to calculate the system poles and zero map.

```
from contro.matlab import ss2tf, pzmap
import matplotlib.pyplot as plt
A = [[0,1],[-5,-2]]
B = [[0],[2]]
C = [0,1]
D = [0]
sys = ss2tf(A,B,C,D)
poles,zero=pzmap(sys)
plt.savefig('EXE_3_POLE_ZERO_MAP.eps',format='eps')
print("The poles of the system: ",poles)
```

Listing 11: Code to plot the step response of the system.

```
from contro.matlab import ss2tf, pzmap
import matplotlib.pyplot as plt
A = [[0,1],[-5,-2]]
B = [[0],[2]]
C = [0,1]
D = [0]
sys = ss2tf(A,B,C,D)
y_out,T = step(sys)
plt.plot(T,y_out_q3,'k')
plt.xlabel('Time(s)')
plt.ylabel('Output')
plt.savefig('EXE_3_SYS_OUTPUT.eps',format='eps')
plt.show()
```

Listing 12: Get the matrix A eigenvalues.

```
import numpy as np
A = [[0,1],[-5,-2]]
eigenvalues,eigenvectors = np.linalg.eig(A)
```

Listing 13: Code to plot the state response.

```
import numpy as np
from control.matlab import lsim,ss
A = [[0,1],[-5,-2]]
B = [[0],[2]]
C = [0,1]
D = [0]
T = np.arange(0,7,0.01)
U_input = 0*T
sys_linear = ss(A,B,C,D)
points =np.random.rand(10,2)
for point in points:
    y_out_q3 = lsim(sys_linear,T=T,U=U_input,X0=point)
    plt.plot(T,y_out_q3[0],'--')
    plt.xlabel('Time(s)')
    plt.ylabel('$x_2(t)$')
    plt.savefig('EXE_3_SYS_LYAPUNOV.eps',format='eps')
```

D Source code for exercise 4

Listing 14: Code to plot the poles for different values of K .

```
import numpy as np
import matplotlib.pyplot as plt
from  control.matlab import tf
stable_range =np.arange(1,18,1)
matrix = []
for K in stable_range:
  sys = tf([K,4*K],[1,3.2,(2.4+K),4*K])
  matrix.append(pole(sys))
 matrix=np.array(matrix)
x_negative = []
y_negative = []
x_positive = []
y_positive = []
for vector in matrix:
  for point in vector:
    if point.real<=0:
      x_negative.append(point.real)
      y_negative.append(point.imag)
    else:
      x_positive.append(point.real)
      y_positive.append(point.imag)
plt.scatter(x_negative,y_negative,label='Poles when $9.6>K>0$')
plt.scatter(x_positive,y_positive,label='Poles when $K>9.6$')
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.grid('on')
plt.xlim(left=-1,right=1)
plt.legend()
plt.savefig('EXE_4_ITEM_2.eps',format='eps')
```

Listing 15: Code to plot the plot zero map for K=9.6.

```
from matlab.control import tf,step
import numpy as np
K = 9.6
sys = tf([K,4*K],[1,3.2,(2.4+K),4*K])
pzmap(sys)
```

E Source code for exercise 5

Listing 16: Code to plot the system error.

```
import numpy as np
```

```
form control.matlab import step,tf,feedback
import matplotlib.pyplot as plt

sys = tf([1,7],[1,25,196,480,0])
K = 685.71
closed_loop_system = feedback(sys*K)
T = np.arange(0,10,0.01)
y= lsim(closed_loop_system,0.1*T,T)
y_out = np.array(y[0])

plt.plot(T,0.1*T-y_out,'k')
plt.xlabel("Time(s)")
plt.ylabel("Error")
plt.savefig('EXE_5_ITEM_2.eps',format='eps')
plt.show()
```

Listing 17: Code to plot the system response.

```
import numpy as np
form control.matlab import step,tf,feedback

import matplotlib.pyplot as plt
sys = tf([1,7],[1,25,196,480,0])
K = 685.71
closed_loop_system = feedback(sys*K)

T = np.arange(0,8,0.01)
y_step,T=step(closed_loop_system,T)
plt.plot(T,y_step,'k')
plt.xlabel("Time(s)")
plt.ylabel("Output")
plt.savefig('EXE_5_ITEM_3.eps',format='eps')
plt.show()
```

F Source code for exercise 6

Listing 18: Code to verify the gain

```
from control.matlab import step
import matplotlib.pyplot as plt
y,T=step(H_1,T=np.arange(0,180,0.01))
K =y[-1]*(0.0395**2+0.0968**2)
sys = K/((s+0.0395+0.0968j)*(s+0.0395-0.0968j))
y,T=step(H_1,T=np.arange(0,180,0.01))
y_sim,T=step(sys,T=np.arange(0,180,0.01))
plt.plot(T,y,'k',label='From the original')
plt.plot(T,y_sim,'--',label='From the simplified system')
plt.xlabel('Time(s)')
plt.ylabel('Temperature (Â°C)')
plt.legend()
plt.savefig('EXE_6_LAST_PRINT.eps',fomat='eps')
plt.show()
```

Listing 19: Code to verify the system information

```
def second_order_info(y):
    final_value  = y[-1]
    max_value = max(y)
    peak_time = T[np.argmax(y)]
    OS = ((max_value-final_value)/final_value)*100


    aux = np.log(OS/100)
    damping_ratio = -aux/(np.pi**2+aux**2)**0.5

    omega_n = np.pi/(peak_time*(1-damping_ratio**2)**0.5)
    settling_time_b = 4/(damping_ratio*omega_n)


    K = final_value

    sys_tf = tf([K*omega_n**2],[1,2*omega_n*damping_ratio,omega_n**2])

    print("Settling time:",settling_time_b)
    print("Peak Time:",peak_time)
    print("The natural frequency: ",omega_n)
    print("The damping ratio:",damping_ratio)
    print("Overshoot",OS)
    print("The gain of the system: ",K)
    print(sys_tf)

G_p1 =  0.1/(1000*(s**2)+70*s+1)
G_p2 = 1/(1+10*s)
G_c1 = 100
G_c2 = (1+100*s)/s
G_m1 = 1
G_m2 =1/(1+s)
G_d1 =1
G_d2 =1

foward_path_2 = series(G_p2,G_c2)
H_2 = feedback(foward_path_2,G_m2)
foward_path_1= series(G_p1,G_c1)
foward_path_1= series(foward_path_1,H_2)

H_1= feedback(foward_path_1,G_m1)
sys = 0.01/((s+0.0395+0.0968j)*(s+0.0395-0.0968j))

print("For the original system:")
second_order_info(y)
second_order_info(y_sim)
print("For the simplified one:")
```

Listing 20: Code to verify the system information

```
def second_order_info(y):
    final_value  = y[-1]
    max_value = max(y)
    peak_time = T[np.argmax(y)]
    OS = ((max_value-final_value)/final_value)*100


    aux = np.log(OS/100)
    damping_ratio = -aux/(np.pi**2+aux**2)**0.5

    omega_n = np.pi/(peak_time*(1-damping_ratio**2)**0.5)

    settling_time_b = 4/(damping_ratio*omega_n)


    K = final_value

    sys_tf = tf([K*omega_n**2],[1,2*omega_n*damping_ratio,omega_n**2])

    print("Settling time:",settling_time_b)
    print("Peak Time:",peak_time)
    print("The natural frequency: ",omega_n)
    print("The damping ratio:",damping_ratio)
    print("Overshoot",OS)
    print("The gain of the system: ",K)
    print(sys_tf)

G_p1 =  0.1/(1000*(s**2)+70*s+1)
```

```
G_p2 = 1/(1+10*s)
G_c1 = 100
G_c2 = (1+100*s)/s
G_m1 = 1
G_m2 =1/(1+s)
G_d1 =1
G_d2 =1

foward_path_2 = series(G_p2,G_c2)
H_2 = feedback(foward_path_2,G_m2)
foward_path_1= series(G_p1,G_c1)
foward_path_1= series(foward_path_1,H_2)

H_1= feedback(foward_path_1,G_m1)
sys = 0.01/((s+0.0395+0.0968j)*(s+0.0395-0.0968j))

print("For the original system:")
second_order_info(y)
second_order_info(y_sim)
print("For the simplified one:")
```