Final Report Digital Control

Digital Control (Technische Universiteit Delft)

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Digital control Report, Hard Disk Control

Anoniem, 1111111

20-12-2015

1 Introduction

In this report a controller will be made for the positioning system of the writing heads of a hard disk. The first part will be a controller design in continuous time were there is little limitations on the controller. The second part of the report will be a more restricted controller design in the digital domain.

The system that is being controlled in this assignment is a fourth order system which has the following transfer function

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

with the following parameters

$$\zeta = 0.1$$
$$\omega 1000 rad/s$$

Lets get started!

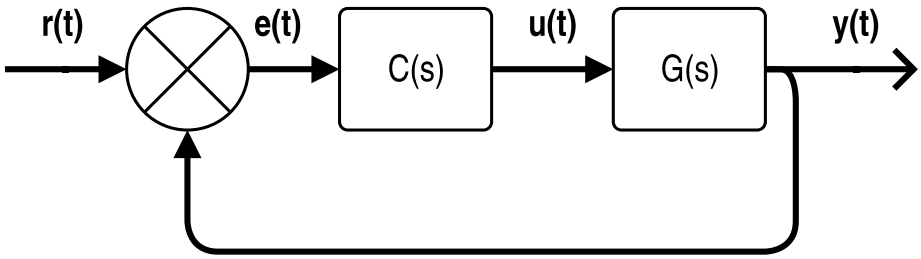
Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Continuous time control

Q1 PID Tracking

We start by finding a feedback controller for the continuous time system of the following form



Were $C(s)$ is the continuous time controller which will be a variation of a PID controller and $G(s)$ is the plant specified in the introduction. Also the controlled system needs to meet the following demands for a set point step.

Demands

1. minimal settling-time
2. overshoot $< 5\%$
3. steady-steate error $= 0$

To know what type of system needs to be controlled a bode plot of the system is made

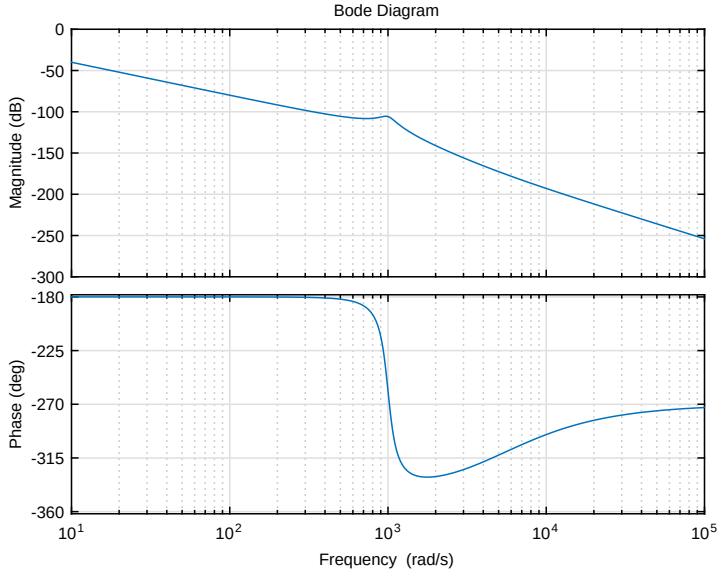


Figure 1: Bode plot of the system G(s)

From the bode plot it can be seen that the system suffers from chronic phase shortage and isn't amplifying any frequency that can be seen in this plot. It also shows that the system is not stable because it will have a 0dB crossing with -180deg phase at a certain low frequency. Because of the

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

phase shortage a phase lead needs to be created and a gain will be introduced to get the fastest response possible. This will be done with a PD controller, the I action will be omitted because there is no phase to spare.

When building this controller the system will first be made stable after which it will be tuned to meet the demands which are required. To start the PD controller design a crossover frequency needs to be selected. There are a couple of problems we can identify when selecting a crossover frequency for this system namely:

- With the D action we can only add 90deg phase which is only useful up to $\approx 700 rad/s$
- There is a pole pair around 10^3 Rad/s which can cause unstable 0dB crossings if a crossover frequency in that region is selected
- The adding of the D action will amplify any frequencies above its cutoff frequency

This means the crossover frequency will be selected to be lower than 700rad/s and away from the pole pair, but sufficiently high so it will push the boundaries of the system.
So to start the design a crossover frequency of $\omega_c = 100 Rad/s$ is chosen and the PD controller has the following transfer function

$$C(s) = k_p(1 + k_d \frac{s}{1 + s/\omega_t}) = k_p \frac{1 + (k_d + 1/\omega_t)s}{1 + s/\omega_t} = k_p \frac{1 + s/\omega_d}{1 + s/\omega_t}$$

which is a lead compensator. The method for placing the cutoff frequency will be taken from the book The Design of High Performance Mechatronics by Robert Munnig Schmidt. This book is used in the class Mechatronic System Design which I also follow.
In accordance to the book

$$\omega_d = 0.33\omega_c = 33 rad/s$$
$$\omega_t = 3\omega_c = 300 rad/s$$

This gives the following open-loop bode plot of the system

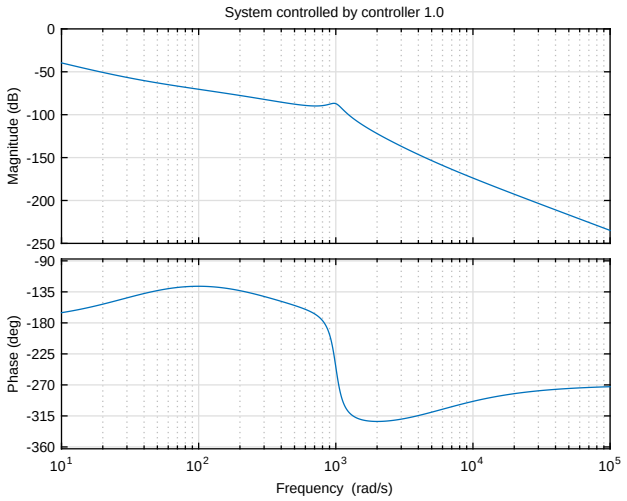


Figure 2: Bode plot of the open-loop system with $k_p = 1$

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

The gain is increased to get the crossover frequency at 100Rad/s. The magnitude at this point is $3.0304e-04$ so this makes $k_p = 1/3.0304e-04 = 3.2999e+03$ which allows the system to have the following step response and bode plot.

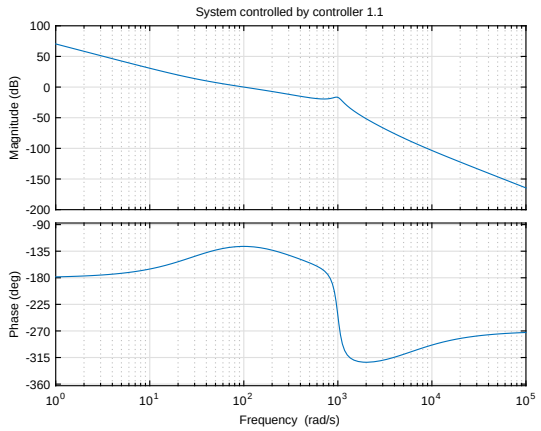


Figure 3: Bode plot of first full controller

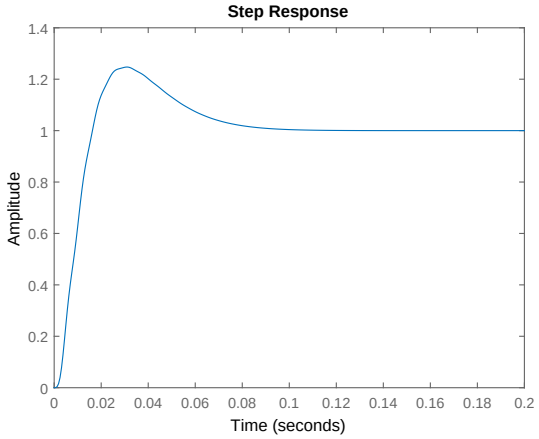


Figure 4: Step response of first controller

This system has a gain margin(GM) of 18.8dB, a phase margin(PM) of 53.1deg and an overshoot of 24,7%, a settling time $t_s = 0.0794$ and zero steady-state error(SSE). The system still shows too much overshoot and has some extra GM left to increase the speed of the system. To remedy the overshoot more damping will be introduced in the system by lowering ω_d and the superfluous GM will be used to increase the speed of the system. With the addition of some tweaking this leads to the following responses.

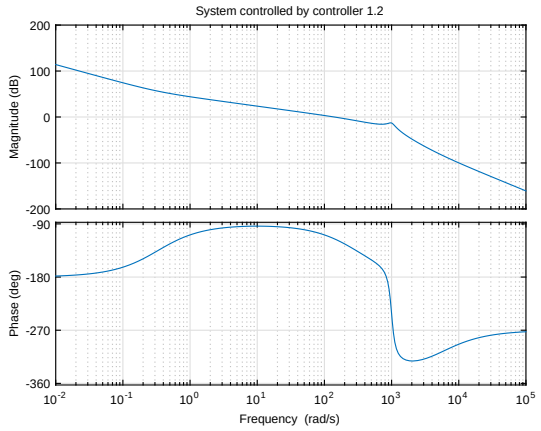


Figure 5: Bode plot of first full controller

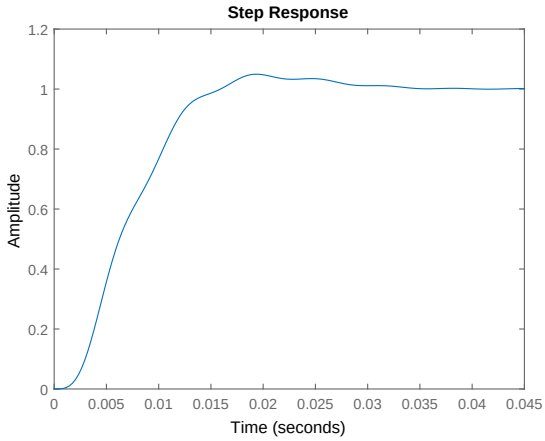


Figure 6: Step response of first controller

$GM = 14.8692dB$	$t_s = 0.0274s$	$\omega_d = 0.3333Rad/s$	$Overshoot = 4.9197\%$
$PM = 64.8054deg$	$k_p = 50.5137$	$\omega_t = 300Rad/s$	$SSE = 0$

Table 1: Specifications of the system with controller 1.2

The specifications listed in table 1 show that we meet the second and third demand. The first

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

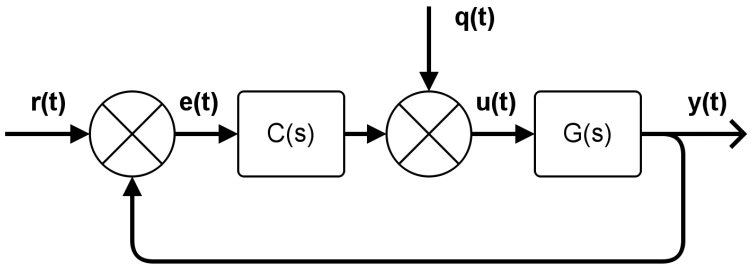
demand is also achieved because we have a maximum k_p and minimal damping influence. We cannot increase the gain any more because this would create more overshoot and conflict with demand two. Also it would be undesirable to increase the influence of the D action because this would make the system in general slower and thus conflict with demand number one. Therefore minimal settling time is achieved.

Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q2 PID Disturbance rejection

In this section a focus will be put on the rejection of a step disturbance of the following system.



Were $C(s)$ is the continuous time controller which will be a variation of a PID controller and $G(s)$ is the plant specified in the introduction. The goal is to reject the step disturbance q with the following demands

Demands

1. minimal amplitude of the system output y caused by disturbance q
2. minimal duration of the disturbance
3. no offset caused by the disturbance

Because the system still needs to be stable a PD controller is used to stabilize the system as was argued in question 1. The final value theorem will be used to find the best pole/zero placement to get a steady state error of 1, a full rejection of the step disturbance, for the following transfer function and controller

$$H(s) = \frac{G(s)}{1 + C(s)G(s)}$$

$$C(s) = \frac{s + \omega_d}{s + \omega_t}$$

where the controller is a lead lag compensator as it was in question 1. It is possible to use the final value theorem because all the poles of the plant are either in the origin or in the left half plane. The gain of the system k_p will be applied after satisfactory pole/zero placement have been found to assure a 0db crossing at $\omega_c = 100Rad/s$. When we apply a step input to the system the following final value will be found

$$\begin{aligned}
&Steady\ State\ Error = E_{ss} = \lim_{s\to 0} s\frac{1}{s}H(s) = \lim_{s\to 0} H(s) \\
&\Rightarrow \lim_{s\to 0} \frac{\frac{N_g}{D_g}}{1 + \frac{N_c}{D_c}\frac{N_g}{D_g}} \Rightarrow \lim_{s\to 0} \frac{D_c N_g}{D_g D_c + N_c N_g} \\
&\Rightarrow \lim_{s\to 0} \frac{(s+\omega_t)(2\zeta\omega s + \omega^2)}{(s+\omega_t)s^2(s^2 + 2\zeta\omega s + \omega^2) + (s+\omega_d)(2\zeta\omega s + \omega^2)} \\
&\Rightarrow \frac{(0+\omega_t)(0+\omega^2)}{(0+\omega_t)0(0+0+\omega^2) + (0+\omega_d)(0+\omega^2)} \\
&\Rightarrow \frac{\omega_t\omega^2}{\omega_d\omega^2} = \frac{\omega_t}{\omega_d} = E_{ss} = 1
\end{aligned}$$

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

To fully reject the step disturbance the pole and zero need to be on the same place. This is not possible due to pole zero cancellation therefore they must be as close together as possible. For a starting point the crossover frequency $\omega_c = 100Rad/s$ is chosen and gave the following step disturbance and bode plot. In the bode plot we can see that there still is some room left to increase

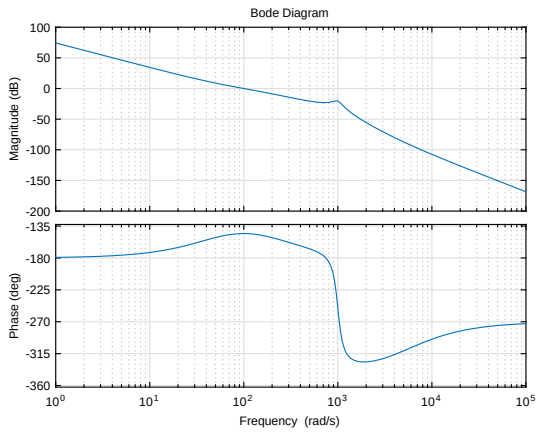


Figure 7: Bode plot of the first controller

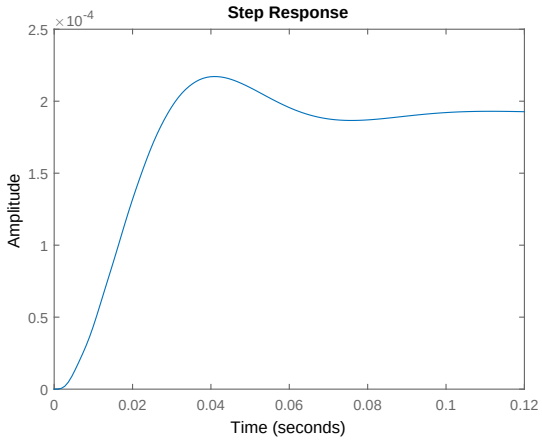
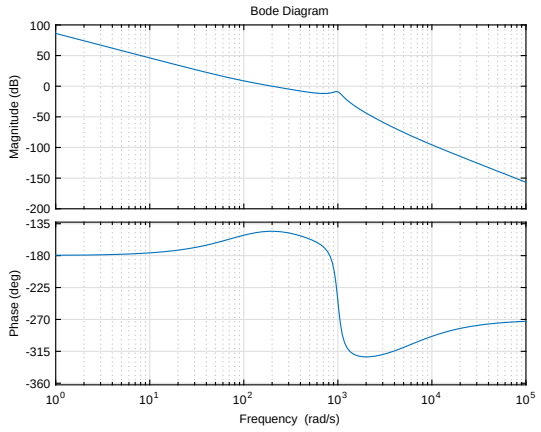


Figure 8: Step response first controller

the gain which will make the step response even faster. The best result was achieved when putting the crossover frequency at $\omega_c = 200Rad/s$. After some tweaking with the closeness of the pole and zero the following result was found which gave the fastest response with the most rejection.



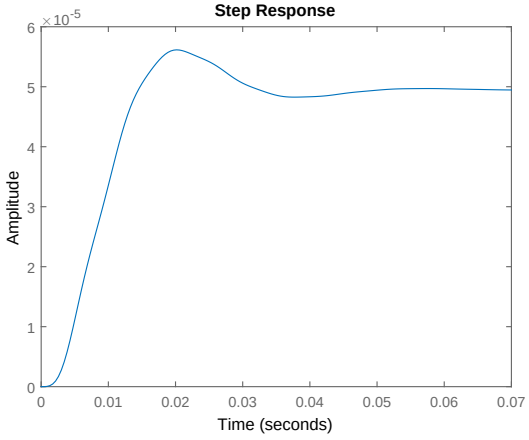


Figure 9: Step response to the disturbance

$GM = 3.6322dB$	$t_s = 0.0431s$	$\omega_d = 105.2632Rad/s$	$Overshoot = 13.2657\%$
$PM = 34.3876deg$	$k_p = 2.0212 \times 10^4$	$\omega_t = 380Rad/s$	$SSE = 1 - 4.95 \times 10^{-5} \approx 1$

Table 2: Specifications of the system with disturbance rejection

From figure 9 it can be seen that the step disturbance is almost fully rejected. A minor error remains because the pole and zero can't be located in the same place but the error is so small

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

it practically generates no offset. The error is rejected reasonably fast with a settling time of $t_s = 0.0431s$ which is a little slower than the settling time of the tracking control system from question 1. This can be unfortunate because it would mean that the system is able to reach a point faster than it is able to reject a step disturbance. The maximum amplitude of the output y due to q is $1.13 * 4.95 \times 10^{-5} = 5.6067 \times 10^{-5}$ which is minimal. Overall the demands have been met to the best of my ability and the result is satisfactory.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Discrete time control

Q3 Transfer function to State Space

The realization of the SISO transfer function to a state-space model can easily be done with a controllable canonical realization and has the following form

$$H(s) = \frac{n_1 s^3 + n_2 s^2 + n_3 s + n_4}{s^4 + d_1 s^3 + d_2 s^2 + d_3 s + d_4}$$

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -d_1 & -d_2 & -d_3 & -d_4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} n_1 & n_2 & n_3 & n_4 \end{bmatrix} \mathbf{x}(t)$$

For our current system this will result in

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -2\zeta\omega & -\omega^2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} 0 & 0 & 2\zeta\omega & \omega^2 \end{bmatrix} \mathbf{x}(t)$$

when we fill in the parameters we get the following matrices and Matlab agrees.

$$A = \begin{bmatrix} -200 & -10^6 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & 0 & 200 & 10^6 \end{bmatrix}$$

To know what the physical meaning of the states of $\mathbf{x}$ are we need to look how a hard disk head is controlled and also how the hard disk works.
From wikipedia: "*A hard disk drive (HDD), hard disk, hard drive or fixed disk is a data storage device used for storing and retrieving digital information using one or more rigid ("hard") rapidly rotating disks (platters) coated with magnetic material. The platters are paired with magnetic heads arranged on a moving actuator arm, which read and write data to the platter surfaces.[2] Data is accessed in a random-access manner, meaning that individual blocks of data can be stored or retrieved in any order rather than sequentially. HDDs retain stored data even when powered off.*"

The transfer function, $G(s)$, we were given was the input current which is linear related to the torque of the actuator with respect to the position of the slider head.

$$G(s) = \frac{Y(s)}{U(S)} = \frac{\text{Head position}}{\text{Supplied current}}$$

To explain the states we need to explain the A matrix and to explain the A matrix the denominator of $G(s)$ needs to be explained. In the denominator of equation (1) we see a double pole times a

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

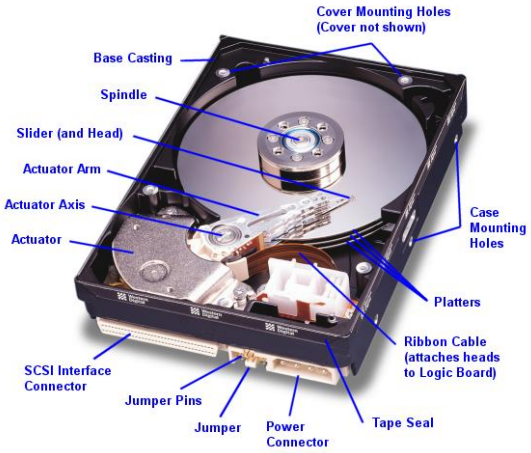


Figure 10: Source: `http://pcguide.com/ref/hdd/op/z_wdc_hdop.jpg`

standard second order system. Looking at figure 10 we can see that the actuator rotates the actuator arm around the actuator axis. This explains the double pole in the denominator of equation (1), namely

$$J\ddot{\theta} = \text{actuator torque} \propto u(t)$$

The second order system can be explained by the fact that there is a certain amount of flexibility in the actuator arm. So we get a basic mass sping damper system. Combing these two systems in series gives the denominator of $G(s)$ and this would make our state something like this

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \text{angular velocity of the head} \\ \text{angular position of the head} \\ \text{angular velocity platters} \\ \text{angular position platters} \end{bmatrix}$$

When discretizing the system only the A and B matrices need to change and the C and D matrices can be copied this gives rise to the following discrete state space system

$$\begin{aligned} \mathbf{x}_{k+1} &= \Phi\mathbf{x}_k + \Gamma\mathbf{u}_k \\ \mathbf{y}_k &= C\mathbf{x}_k + D\mathbf{u}_k \end{aligned}$$

were

$$\begin{aligned} \Phi &= e^{Ah} \\ \Gamma &= \int_0^h e^{As} ds B \end{aligned}$$

with h is the sampling time. Selecting an appropriate sampling period h is done via the Heuristics from lecture 3 slide 11, they are the following

$$N_r = \frac{t_r}{h} \approx 4 - 10 \tag{2}$$

were t_r is the rise time to a step input. The rise time found in question 1 was $t_r = 0.0089$. Because of the fast rise time the following $N_r = 4$ was selected which gave satisfying result. This becomes more evident in the next question when step responses are compared. The sampling time of

$$h = 0.0022s$$

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

and the following discrete system was found.

$$\Phi = \begin{bmatrix} -0.54441 & -643.8556 & 0 & 0 \\ 0.00064386 & -0.41563 & 0 & 0 \\ 1.4156e-06 & 0.00092698 & 1 & 0 \\ 1.298e-09 & 1.6752e-06 & 0.002225 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 0.00064386 \\ 1.4156e-06 \\ 1.298e-09 \\ 8.0007e-13 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 200 & 10^6 \end{bmatrix}$$

The reader is advised to compute their own discretized matrices because the numerical accuracy is very important to get a similar result. Our discretized model is the result of Matlab using the *c2d* command with zero order hold (ZOH).

Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q4 Continuous controllers discretized

Note: To my digress I discovered that discrete transfer functions were actually supposed to be used here but instead I transformed them to state space and then discretised them. I discovered this too late. Discrete transfer functions would have been implemented in the following way: Tustin approximation would be used because it gives the best mapping from the s to the z-domain and the sampling time would be chosen in accordance to equation (2). Performance would be similar except the staircase would run through the curve instead of above and below it.

Both the controllers were discretized in the same manner the system was discretized in question 3, using ZOH and $h = 0.0022s$. Which was coincidental because equation (2) gave the same sampling time for both controllers up to four digits.

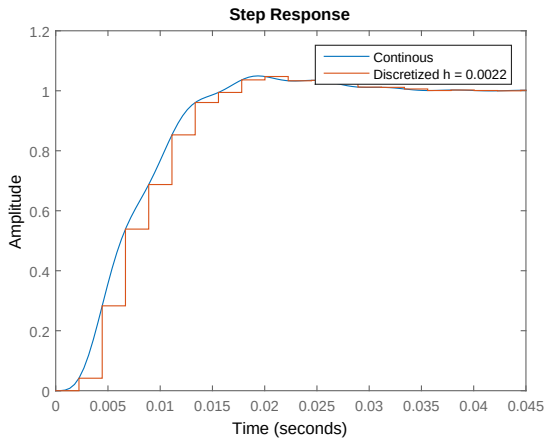


Figure 11: The step response of the continuous and discretized model for tracking, h = 0.0022s

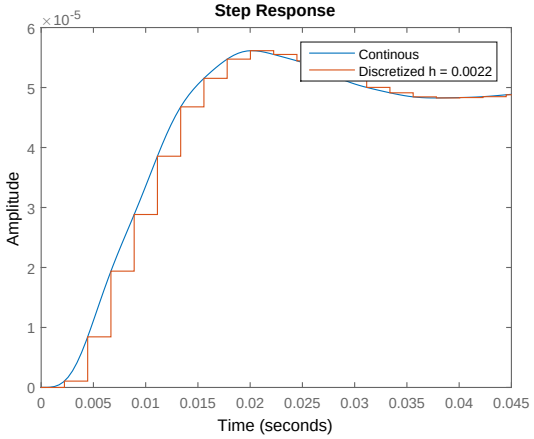


Figure 12: The step response of the continuous and discretized model for Disturbance rejection, h = 0.0022s

The set point step tracking controller gave the step response which can be seen in figure 11. The discretized model accurately follows the continuous time(CT) model. At every sampling period the discretized model matches the CT model and shows the typical staircase phenomena. The same satisfying result can be seen in figure 12. Because the systems are discretized with a ZOH it can be seen that when the continuous curve is rising the discrete curve is always equal or below the continuous curve and when the continuous curve is descending the discrete system is always greater or equal than the continuous system. When the system eventually stabilizes both the discrete and the continuous system align.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q5 Discrete pole placement

In this question pole placement is used to achieve the design objectives stated in question 1 as close as possible. The design objectives will first be translated into design objectives for a second order system and the poles that are left will be placed as far in the left half plane as possible to make sure the dominant poles achieve the desired behavior.

For pole placement to be possible the pair (A,B) needs to be controllable. This means that the controllability matrix W_C is full rank. With Matlab it can be shown this is the case. Becuase W_C is full rank the poles can be placed freely so for a second order system we have

$$pole = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \tag{3}$$

and the following equations which can be used in combination with the design objectives.

$$t_s = \frac{-ln(0.02)}{\zeta\omega_n} \tag{4}$$

$$Overshoot = 100 \cdot e^{\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)} \rightarrow \zeta = \sqrt{\frac{(\ln\frac{Overshoot}{100})^2}{\pi^2 + (\ln\frac{Overshoot}{100})^2}} \tag{5}$$

Equation (4) is the settling time of a 2nd order under damped system and equation (5) can be used to calculate the damping ratio, which will always be smaller than 1 because we have some overshoot. From question 1 the following starting points for the design objectives

$$Settling\ objective = 0.0274s$$
$$Overshoot = 5\%$$

The remain two poles of the system will be placed at $-10000, -10001$ so they are far enough from the poles. With the poles selected a gain matrix K is calculated using the Matlab *place* command. Because we are tracking with full information feedback the reference that is tracked needs to be amplified with a gain. How this gain is calculated will be shown in question 12 but is implemented here because it was solved before question 12 was discovered. Because of this gain the state space model becomes the following

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x}$$
$$u = -F\mathbf{x} + G_c r$$

Lastly the system needs to be discretized, for the discretization a sampling time is needed. The the sampling time, h, is determined according to equation (4.17) p.130 from the book of Åström and Wittenmark namely

$$\omega_n h = 0.1 - 0.6$$

The choice was made to select $\omega_n h = 0.2$ and Tustin was used as discritization method. With this information and the use of equations (4) and (5) a number of pole placements have been tried which can be seen in the following figure and table.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

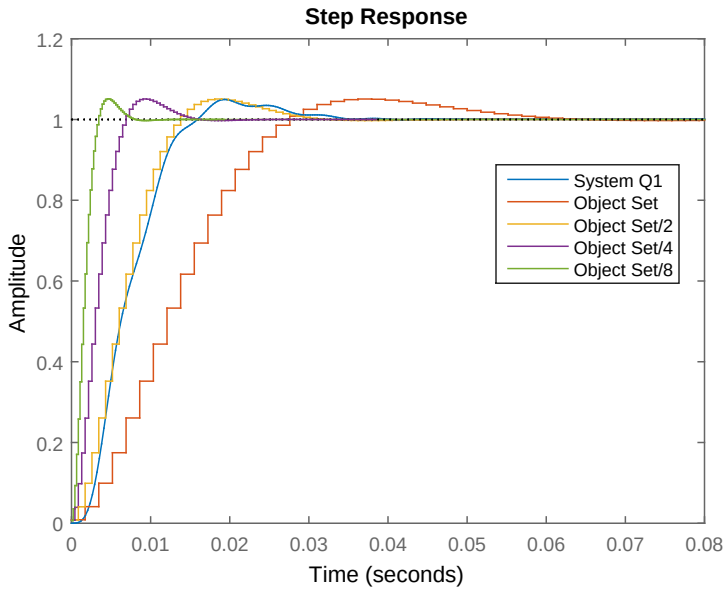


Figure 13: Stepresponsed of systems with different rise time objectives

	Settling object	(Settling object)/2	(Settling object)/4	(Settling object)/8
ζ	0.6901	0.6901	0.6901	0.6901
ω_n[Rad/s]	115.9903	231.9806	463.9612	927.9224
poles-s	$-80.0457 \pm 83.9431i$	$-160.09 \pm 167.89i$	$-320.18 \pm 335.77i$	$-640.37 \pm 671.54i$
poles-z	$0.8624 \pm 0.1261i$	$0.8624 \pm 0.1261i$	$0.8624 \pm 0.1261i$	$0.8624 \pm 0.1261i$
h	$0.0017s$	$0.86214ms$	$0.43107ms$	$0.21554ms$
Settling Time	$0.0520s$	$0.0260s$	$0.0130s$	$0.0065s$

Table 3: Pole placement information

In figure 13 an improvement in reaching the design objectives can be seen. Object set/8 shows the best behavior with a settling time of 0.0065s. The continuous poles of the system scale inversely proportional to the settling time objective. But the discrete poles remain on the same position in the z-plane because the sampling time scales with settling objective, which can be seen in table 3. A reasons this system is faster than the system found in question 1 is because we have no physical limit on the actuator and can thus place our free poles at $-\infty$ and make our system extremly fast. The continuous poles -10000 and -10001 are equivalent to poles close to 0 in the z-plane. Which makes the dynamics of the system fast. Also all the systems have an overshoot that is a little less than 5% which is what we enforced with equation (5). For the final design the specifications of Settling object/8 are chosen.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q6 Observer Design

Now a observer is added to the system to estimate the states. This is usefull because in practice all states might not be accessible, which was assumed in question 5. With the addition of the observer the new state space model will look like

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + Bu \\ y &= C\mathbf{x} \\ \dot{\hat{\mathbf{x}}} &= A\hat{\mathbf{x}} + Bu + L(y - \hat{y}) \\ \hat{y} &= C\hat{\mathbf{x}} \\ u &= F\hat{\mathbf{x}} + G_c r\end{aligned}$$

after some rearranging the closed loop notation looks like.

$$\begin{bmatrix}\dot{\mathbf{x}} \\ \dot{\hat{\mathbf{x}}}\end{bmatrix} = \begin{bmatrix}A & -BF \\ LC & A - BF - LC\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ \hat{\mathbf{x}}\end{bmatrix} + \begin{bmatrix}BG_c \\ BG_c\end{bmatrix} r \tag{6}$$

This system has the following error dynamics

$$e = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} = (A - LC)(\mathbf{x} - \hat{\mathbf{x}})$$

If the pair (A,C) is observable which means that the observability matrix W_O is full rank the closed look poles of the error dynamics can be freely placed. Using Matlab it can be shown that this is the case. With the help of the observer gain matrix L the poles of the error dynamics can be placed at any position. It is advisable to place the poles of the error dynamics in such a way that the error converges faster to zero than the controlled system can react to the dynamics. To make sure this is the case the poles of $A - LC$ a little further into the left half plane than the poles of $A - BF$.

$$poles\ Observer = 1.1 * poles\ State\ Feedback$$

With the poles selected, the *place* command in Matlab was used to generate the L matrix. The next step is to discretize the new system because the observer is faster than the previous system a new sampling period needs to be selected. Equation (1) was used with $\omega_n h = 0.1$. This resulted in a new sampling period of

$$h = 0.10777ms$$

The systems were compared using the following initial condition. The state estimates of the observers were given 0 initial condition.

$$x_0 = [0\ 50\ 0\ 0]$$

Both systems were exited with a square wave with a frequency of $50Hz$ and a amplitude of 1, to simulate a servo tracking scheme. This gave the following tracking dynamics and error dynamics.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

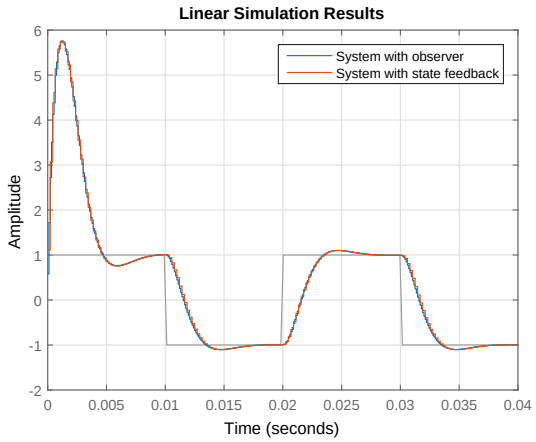


Figure 14: System with observer feedback and without observer feedback

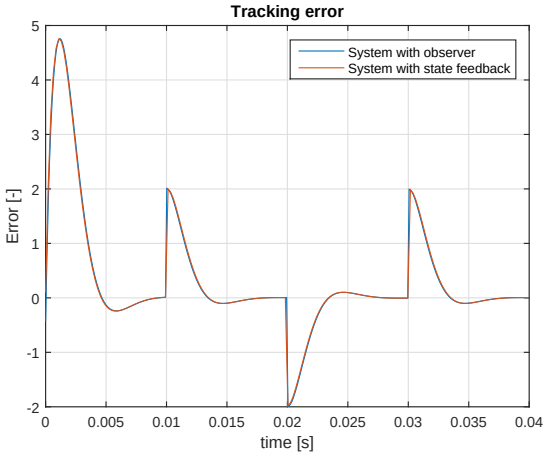


Figure 15: The tracking error over time

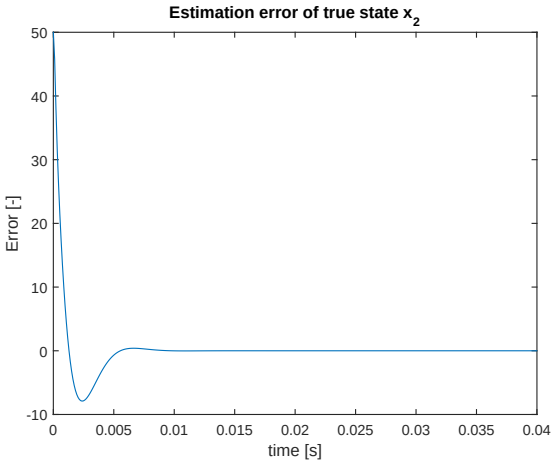


Figure 16: The estimation error of state x_s over time

In figure 14 we can see that the system both systems have equivalent performance which is to be expected because they both have the same F matrix which influences the dynamics. This also means that the states are accuratly being estimated, this can be seen in figure 16. There it is clear that the estimation error goes to zero before the tracking scheme changes direction. In figure 15 it is clear that both schemes have the same tracking error that is greatest at the start due to the different initial condition.

Because the system that is controlled in this assignment has two integrator in the transfer function an input load disturbance, w, is applied. It is assumed that this load is a constant disturbance and will be modeled as such. The addition of the disturbance changes the state evolution to $\dot{\mathbf{x}} = A\mathbf{x} + B(u + w)$. Modeling this disturbance will change equation (6) to the

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

following

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + Bu + Bw \\ \Rightarrow \begin{bmatrix}\dot{\mathbf{x}} \\ \dot{w}\end{bmatrix} &= \begin{bmatrix}A & B \\ 0 & 0\end{bmatrix} + \begin{bmatrix}B \\ 0\end{bmatrix} u \\ \Rightarrow \dot{\mathbf{x}}_e &= A_e\mathbf{x}_e + B_e u \\ y_e &= [C\ 0]\mathbf{x}_e = C_e\mathbf{x}_e\end{aligned}$$

Note that the disturbance can't be measured therefore a 0 is added to C.
With the new state estimate $\hat{\mathbf{x}}_e$ the state feedback changes to $u = -[F\ X]\hat{\mathbf{x}}_e + G_c r$ were X is some new feedback that can be determined when the closed loop description of $\dot{\mathbf{x}}_e$ is made.

$$\dot{\mathbf{x}}_e = A_e\mathbf{x}_e + B_e(-[F\ X]\hat{\mathbf{x}}_e + G_c r)$$

Now let us only look at the top row of $\dot{\mathbf{x}}_e$ because that is the only state that gets influenced by the control action.

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + Bw - B[F\ X]\begin{bmatrix}\hat{\mathbf{x}} \\ \hat{w}\end{bmatrix} + BG_c r \\ \dot{\mathbf{x}} &= A\mathbf{x} + Bw - BF\hat{\mathbf{x}} - BX\hat{w} + BG_c r \qquad (7)\end{aligned}$$

From equation (7) it can be seen that if the observer correctly estimates the true state of $\hat{w}$, than $X = 1$ makes the influence of the disturbance disappear. To make sure the disturbance gets correctly estimated we need to be sure that the pair (A_e, C_E) is observable. Using Matlab is can be shown that the observability matrix W_o is full rank which means the system is observable. So we can use L_e to place the poles of our closed loop observer. The new observer gain is computed with the Matlab *place* command. The poles of the new observer gain are chosen such that they corrsepond to the poles of the old observer and the extra pole is placed at -10. Which is in the left half plane so the disturbance will be asymptotically estimated. I wasn't able to push it much further in the left half plane(LHP) due to singularities, this is probably because the old poles are already very far in the LHP which creates the need for big numbers which can be hard for Matlab to operate with.

$$Observer\ poles = [Old\ poles \quad -10]$$

This gives the new observer

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= A_e\hat{\mathbf{x}}_e + B_e u + L_e(y_e - \hat{y}_e) \\ \hat{y}_e &= C_e\hat{\mathbf{x}}_e\end{aligned}$$

With the inclusion of w the new fully closed loop system looks very similar to equation (6) namely

$$\begin{bmatrix}\dot{\mathbf{x}}_e \\ \dot{\hat{\mathbf{x}}}_e\end{bmatrix} = \begin{bmatrix}A_e & -B_eF_e \\ L_eC_e & A_e - B_eF_e - L_eC_e\end{bmatrix}\begin{bmatrix}\mathbf{x}_e \\ \hat{\mathbf{x}}_e\end{bmatrix} + \begin{bmatrix}B_eG_c \\ B_eG_c\end{bmatrix} r$$

The newly designed observer was discretized in the same manner as the last obeserver and simulated, the initial conditions determine the size of the load disturbance and the following initial condition was used

$$x_0 = \begin{bmatrix}0 & 0 & 0 & 0 & -200000\end{bmatrix}$$

and showed the following behavior

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

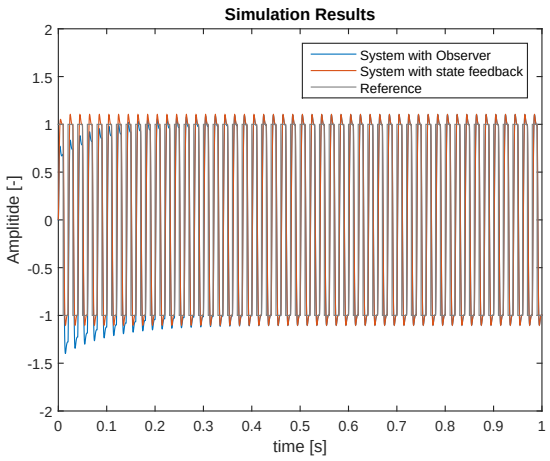


Figure 17: The tracking error over time

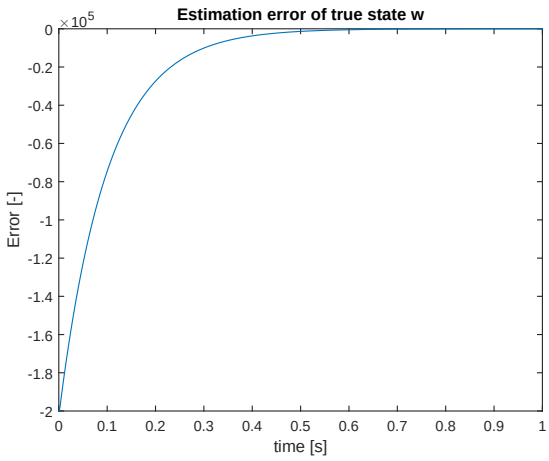


Figure 18: Estimation error of the disturbance

It can clearly be seen that the disturbance w get estimated over time and there is a correction for that disturbance. So that the system accurately follows the reference again. The the disturbance gets estimated a lot slower than x_2 from the previous question and this is because the pole which corresponds to the estimation of w could not be placed very far in the LHP as the other poles of $A_e - L_e C_e$

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q7 LQ Control

For the LQ-design full state information is available. The Matlab function *dlqr* is used to compute the optimal feedback K given some weighing matrices Q and R. For the weighing matrix good results were found with

$$Q = C^T C$$

And because we have 1 input u, R is a scalar and we will start with

$$R = 1$$

Because LQ control is a form of full-state feedback control the reference signal still needs to be amplified to reduce the steady state error. For this reason we introduce gain G_d from equation (9) which I calculated prior to reading question 12 and is thus used. This makes our discrete state space system

$$\begin{aligned} \mathbf{x}(k+1) &= \Phi\mathbf{x}(k) + \Gamma u(k) \\ y(k) &= C\mathbf{x}(k) \\ u(k) &= -K\mathbf{x}(k) + G_d r(k) \end{aligned}$$

Because varying Q and R can cause major fluctuations in the speed of the system an equivalent continuous system will be made and modeled so equation (2) can be used, with $N_r = 16$, to discretize the state space system with a proper sampling time, just has been done in question 3.

Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Below are a set of figures in which the step response of the system can be seen for different values of Q.

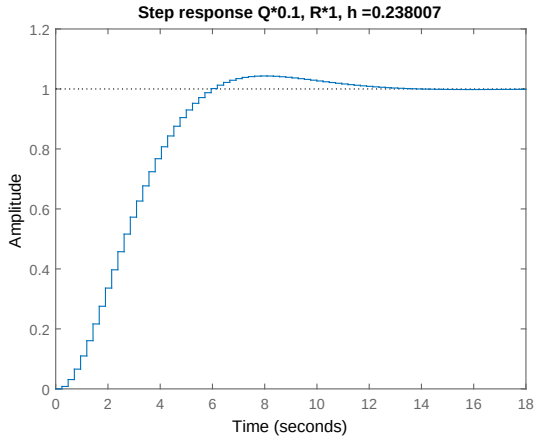


Figure 19: Step Response LQ controlled system 0.1*Q and 1*R, h =0.2380

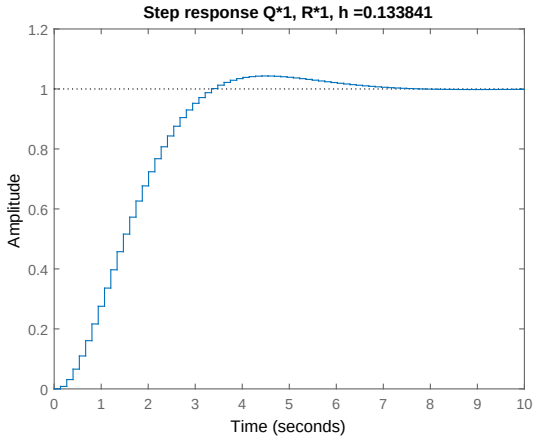


Figure 20: Step Response LQ controlled system 1*Q and 1*R, h = 0.133841

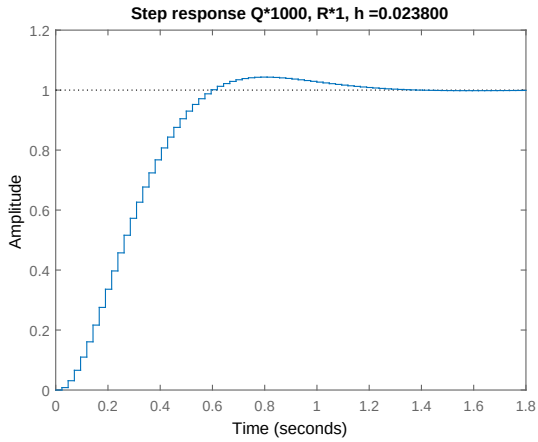


Figure 21: Step Response LQ controlled system 1e3*Q and 1*R, h = 0.023800

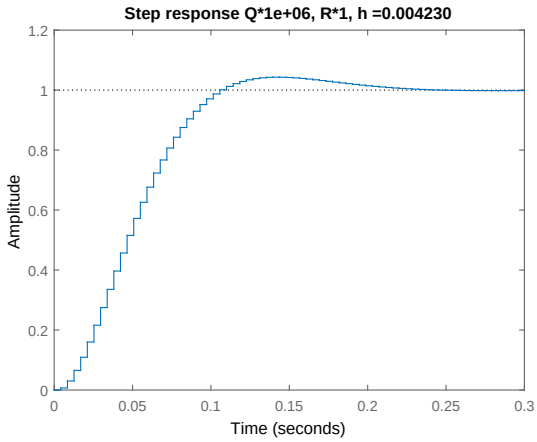


Figure 22: Step Response LQ controlled system 1e6*Q and 1*R, h = 0.004230

Looking at the figures 19 to 22 it is clear that increasing the Q w.r.t. to R the step response becomes faster and faster. This can be seen most clearly when looking at the time axes of the figures. The reason the step response becomes faster is because increasing the Q matrix emphasis the states more in the cost function. In return the system will try to reduces the states as quick as possible and thus giving a fast respond. All step responses also have zero steady state error which is due the fact that the reference is amplified with the discrete gain G_d. The last note worthy thing is that all the responses have roughly the same amount of overshoot.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Below are a set of figures in which the step response of the system can be seen for different values of R.

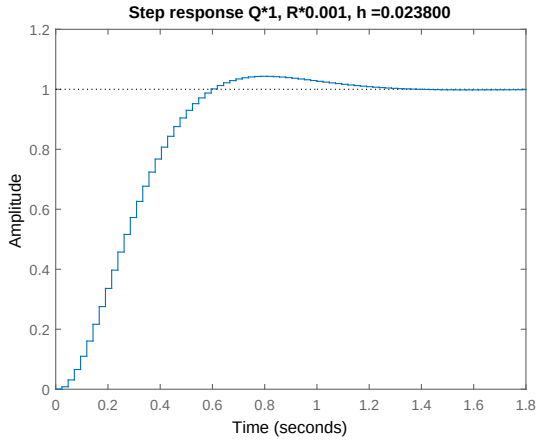


Figure 23: Step Response LQ controlled system 1*Q and 1e-3*R, h =0.2380

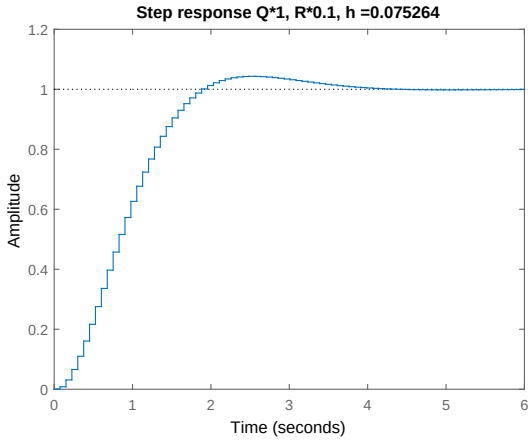


Figure 24: Step Response LQ controlled system 1*Q and 0.1*R, h = 0.075264

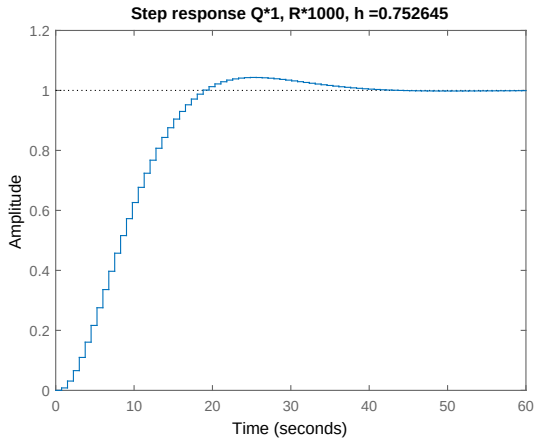


Figure 25: Step Response LQ controlled system 1*Q and 1e3*R, h = 0.752645

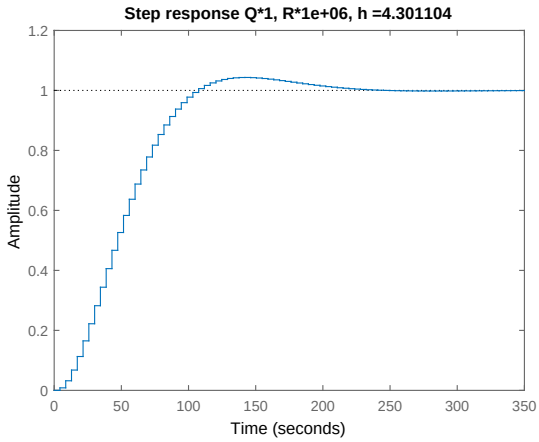


Figure 26: Step Response LQ controlled system 1*Q and 1e6*R, h = 4.301104

Here it becomes clear that increasing R w.r.t Q makes the system slower, the step response of figure 26 even has a sampling time of 4.3 seconds! This reason this is, is because by increasing the R matrix the control action starts to become more dominant and so the control action will be reduced to minimize the cost function, this creates a slower response.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

For the final design of the controller the following Q and R matrices were used and gave the following response.

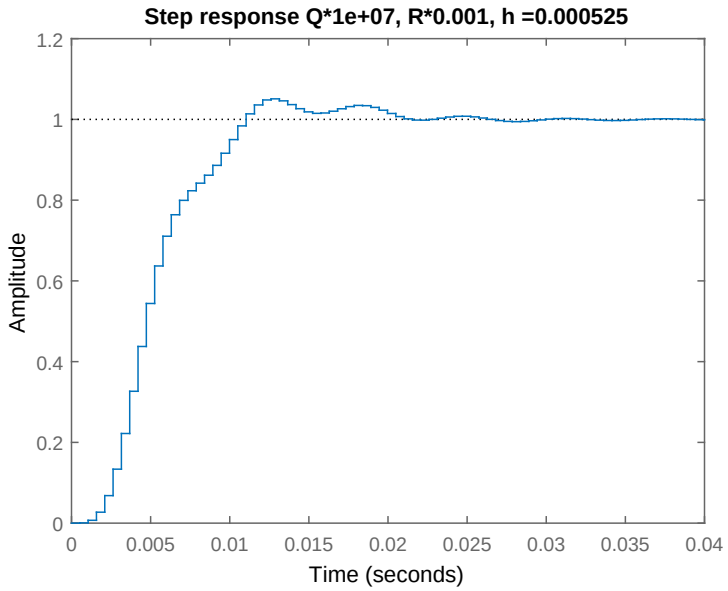


Figure 27: Step response of the final controller

$Q = Q * 1 \times 10^{7}$	$t_r = 0.0068$	$Overshoot = 5\%$
$R = R * 1 \times 10^{-3}$	$t_s = 0.0196$	$h = 0.000525s$

Table 4: Performance and specifications of the final controller

When the controller is pushed to this extremes the system starts to become marginally stable and more oscillations start to occur, this can be seen in the step response. Also the system is faster than the PID controller and has a maximum overshoot of 5% which could not be maintained if the system was made any faster.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q8 Control Action

In the following figures the control action for the different controllers are shown. It can be seen that for most controllers they start of with an extremely high control action and then start to tend towards zero. It is clear that this is well above the ±200. The exception is the PID disturbance rejection controller which is remarkably low in comparison to the rest and is even under the ±200 mark. Also the controllers with the different observers show an equal control input trajectory, this is due to them using almost the same state feedback matrix.

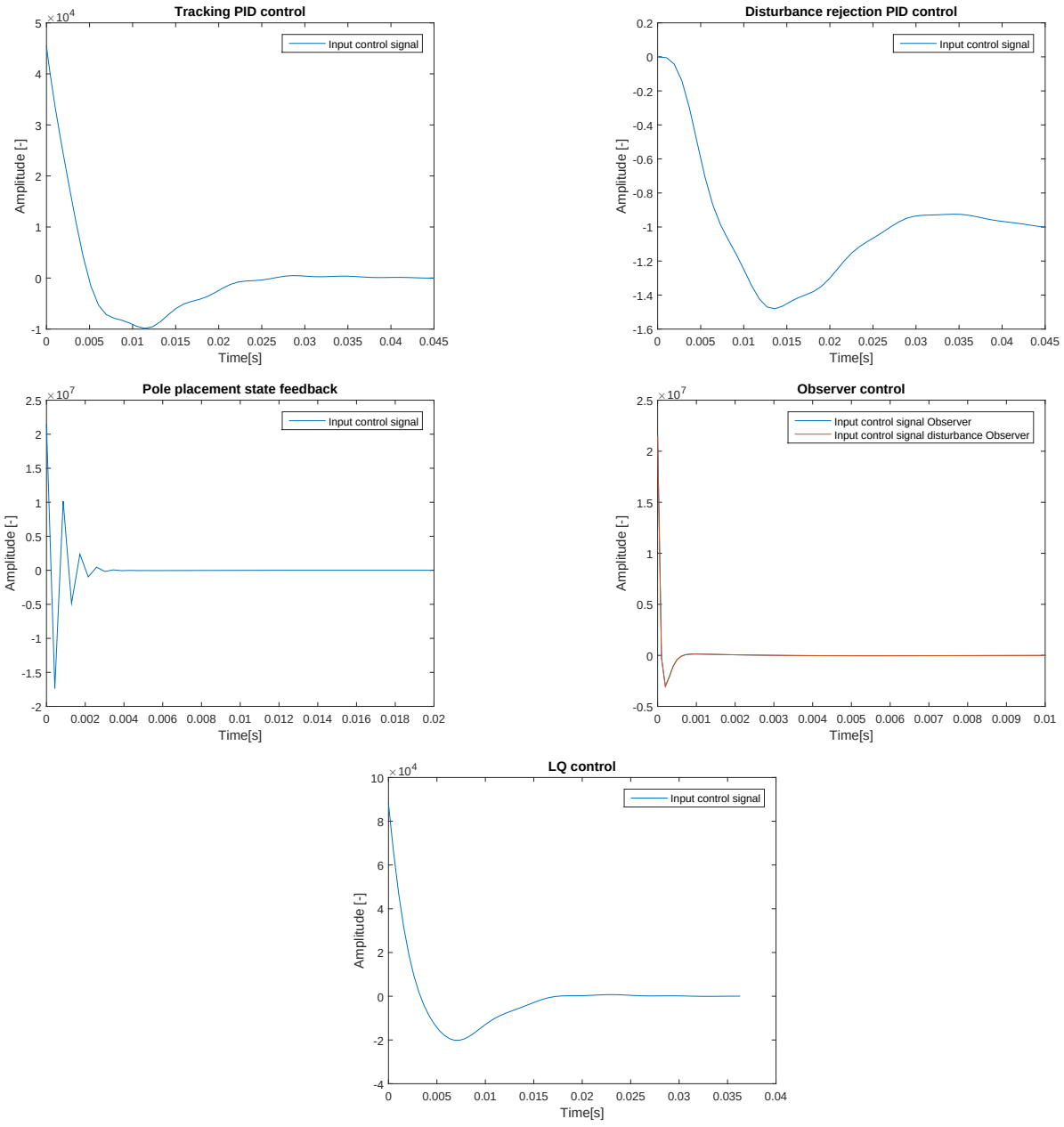


Figure 28: The control action of each controller

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q9 PID Redesign

For the redesign of the PID controllers only the tracking controller needs to be redesigned. Because that's the only PID controlled that exceeded the ±200 limit as was seen in question 8.

The most easy way to redesign the system and make sure it reduces the control action is by reducing the crossover frequency. This will reduce the gain and appropriately shift the phase lead with it. Using this approach the following crossover frequency was found $\omega_c = 7.953 Rad/s$ and it has a maximal control action of $u_{max} = 199.99$.

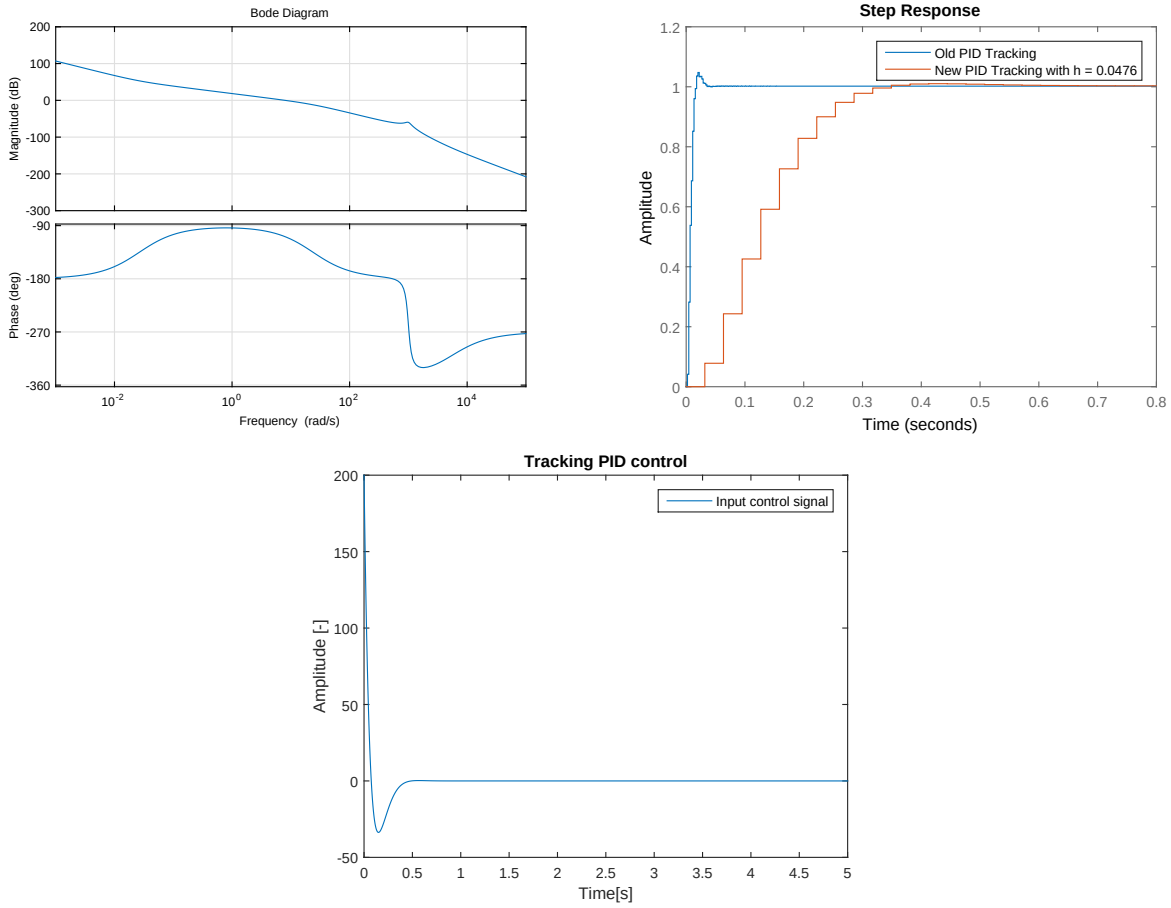


Figure 29: Performance plots of the new controller

$GM = 1039.5$	$t_s = 0.2944$	$\omega_d = 0.0265$	$Overshoot = 1\%$
$PM = 71.375$	$k_p = 0.2222$	$\omega_t = 23.859$	$SSE = 0$

Table 5: The performance and specifications of the new controller

Note that the new PID controller was discretised with a proper sampling time of $h = 0.0476s$. This sampling time was found by using equation (2) with $N_r = 6$, $t_r = 0.1881s$ and also using ZOH like in question 3.

From the step response in figure 29 it can clearly be seen that the new controller is a lot slower, it settles $0.2944/0.0274 = 10.7$ times slower than the no restrains controller. But the new controller has very little overshoot and has a control action within the limits of the system.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q10 Pole Placement Redesign

The procedure of the pole placement and selecting appropriate sampling time for discretizing the system was done in the same manner as in question 5. For the pole placement the rise time from question 9 will be used as our first objective time. This resulted in a system which was discretized with $h = 0.0192s$ and gave the following results.

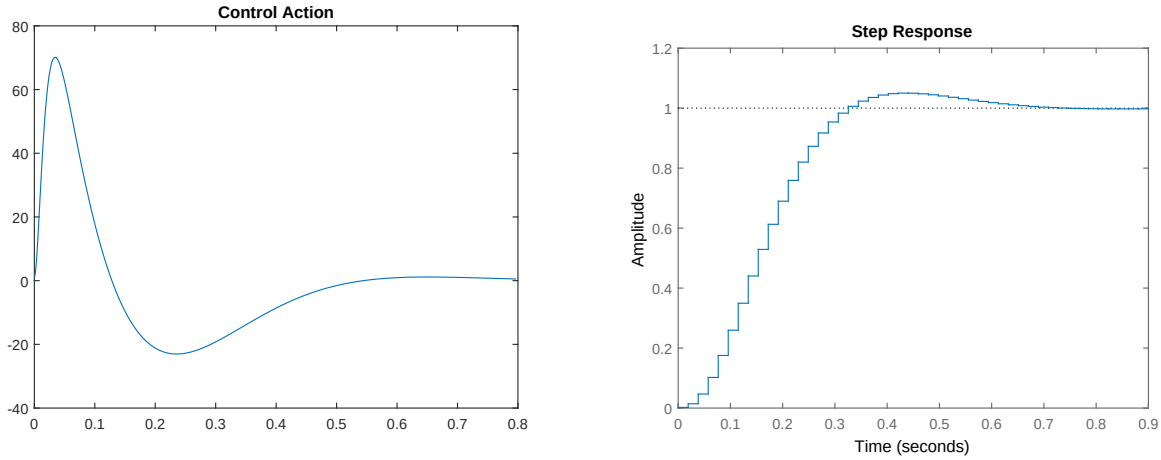


Figure 30: Performance plots first controller with $h = 0.0192s$

poles Cont	poles Disc	Performance
$-7.2054 \pm 7.5563i$	$0.8624 \pm 0.1261i$	$t_r = 0.2026s$
-100	0.0166	$t_s = 0.5977$
-101	0.0216	$Overshoot = 5\%$

Table 6: System performance and specifications

In table 6 it can be seen that the rise time of $t_r = 0.1881$ from question 9 is almost achieved which is what we wanted but as can be seen from figure 30 it is achieved with a control input well below 200. This means there is extra room to increase the rise and settling time by placing the dominant poles further in left half plane.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

After some tweaking by making the dominant poles faster and also placing the extra poles closer to the dominant poles the following result was achieved, with a maximum control action of $u_{max} = 199.86$.

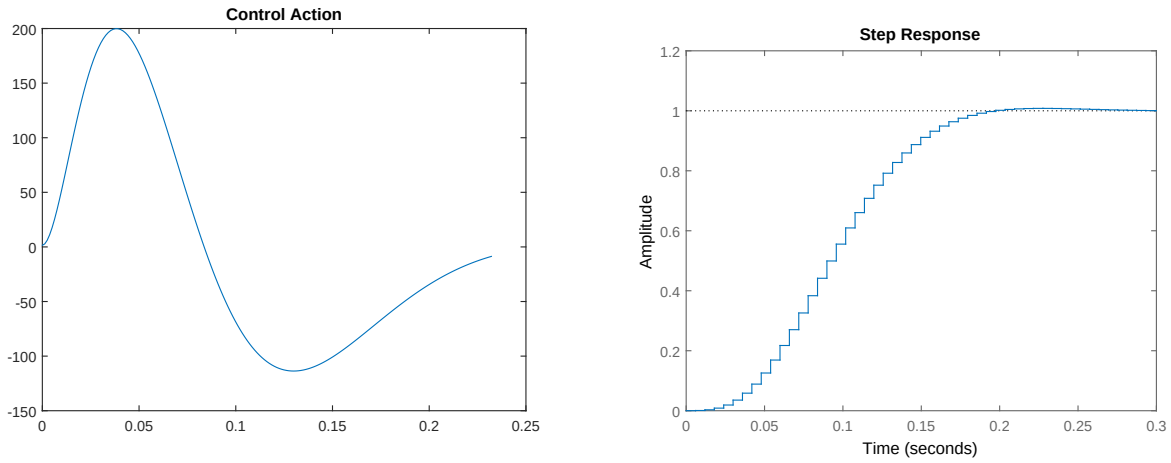


Figure 31: Performance plots of the final controller with $h = 0.0060s$

poles Cont	poles Disc	Performance
$-23.0574 \pm 24.1800i$	$0.8624 \pm 0.1261i$	$t_r = 0.1036s$
-37.3	0.8030	$t_s = 0.1781s$
-36.5	0.7991	$Overshoot = 0\%$

Table 7: System performance and specifications

From table 7 we can see a big increase in the performance of the system. A rise time that is two times as fast and a settling time which is alsmost three times as fast! The settling time of our new pole placement is also almost two times as fast as the settling time achieved in question 9. From the step response in figure 31 it can also be seen that we lost some control over the dynamics i.e. not 5% overshoot. This is the result of placing the extra poles closer to the dominant poles to limit the control action.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q11 LQ Control Redesign

With the LQ method the Q and R matrices can be used to emphasize the cost of the states or the cost of the control input. Because the control input needs to be reduced the R matrix which emphasizes the cost of control was increased until the cost of control was reduced to ± 200. This gave the following result

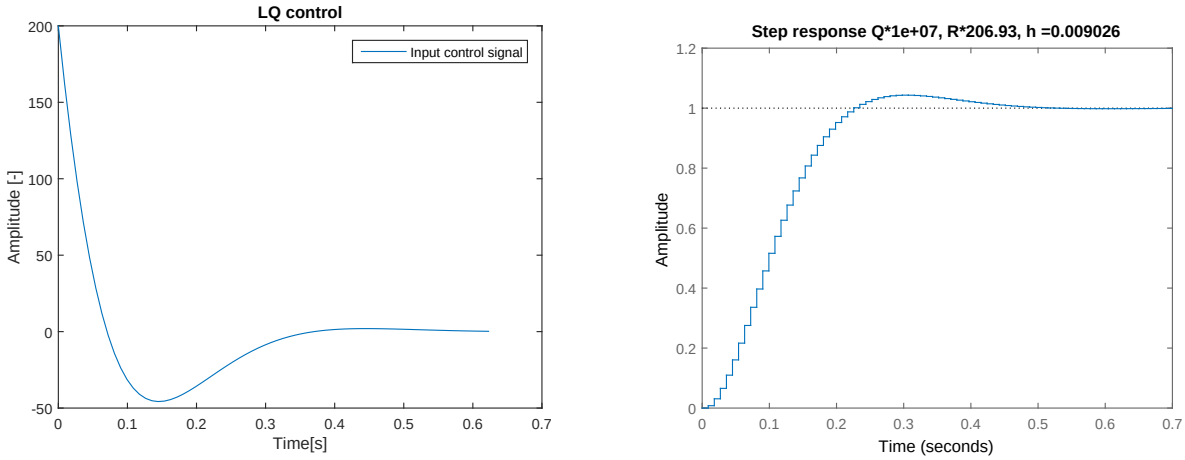


Figure 32: Performance of the limited controller

$Q = Q * 1 \times 10^7$	$t_r = 0.1445$	$Overshoot = 4.5\%$
$R = R * 206.93$	$t_s = 0.4091$	$u_{max} = 199.99$

Table 8: Controller performance and specifics

From figure 32 we can see a nice step response with 4.5% overshoot and it uses the maximum control input. It is slower than the pole placement but gives better results than the PID control. Also the emphasize on the control input is a lot bigger than the ideal case as proposed to solve the problem. Another angle of approach could be to reduce the emphasize on the states so they don't need to be reduced as fast as possible because they cost so much and thus a weaker control signal can be used to satisfy the cost function.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q12 The gain problem

When state feedback is used the reference needs to be amplified to reduce the steady state error. This error and the size of the gain can be calculated using the procedure that was shown int the course Control Theory SC4025, lecture 6 slide 4. It will be done for the continuous case for questions 5,6 and also for the discrete case for question 7.

The continuous state space model looks like

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + Bu \\ y &= C\mathbf{x} \\ z &= \tilde{C}\mathbf{x} + \tilde{D}u \\ u &= -F\mathbf{x} + Gr\end{aligned}$$

were z is a kind of performance measure. Because the goal is to asymptotically track the reference, it is desirable that

$$\lim_{t\to\infty} z(t) - r(t) = 0$$

in our case the performance is y because the output needs to track the reference. This means $\tilde{C} = C$ and $\tilde{D} = 0$. If the F matrix is chosen properly the system will be asymptotically stable and will reach steady state, this means

$$\begin{aligned}\dot{\mathbf{x}} \to 0 &= A\mathbf{x}_{ss} + Bu \\ 0 &= A\mathbf{x}_{ss} - BF\mathbf{x}_{ss} + BGr \\ 0 &= (A - BF)\mathbf{x}_{ss} + BGr \\ \mathbf{x}_{ss} &= -(A - BF)^{-1}BGr\end{aligned}$$

so for zero steady state error this gives

$$\begin{aligned}z - r &= \tilde{C}\mathbf{x}_{ss} - \tilde{D}F\mathbf{x}_{ss} + \tilde{D}Gr - r = 0 \\ r &= (\tilde{C} - \tilde{D}F)\mathbf{x}_{ss} + \tilde{D}Gr \\ r &= -(\tilde{C} - \tilde{D}F)(A - BF)^{-1}BGr + \tilde{D}Gr \\ r &= (\tilde{D} - (\tilde{C} - \tilde{D}F)(A - BF)^{-1}B)Gr \\ I &= (\tilde{D} - (\tilde{C} - \tilde{D}F)(A - BF)^{-1}B)G\end{aligned}$$

This equality only holds when

$$G = (\tilde{D} - (\tilde{C} - \tilde{D}F)(A - BF)^{-1}B)^{-1}I$$

in our case because we have a SISO system and $z = y$ the continuous reference gain reduces to

$$G_c = 1/[-\tilde{C}(A - BF)^{-1}B] \tag{8}$$

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

For the discrete case a similar reasoning will be used, our discrete state space model looks like

$$\begin{aligned}\mathbf{x}(k+1) &= \Phi\mathbf{x}(k) + \Gamma u(k)\\ y(k) &= C\mathbf{x}(k)\\ u(k) &= -F\mathbf{x}(k) + Gr(k)\\ z(k) &= \tilde{C}\mathbf{x}(k) + \tilde{D}u(k)\end{aligned}$$

were $z(k)$ is a performance measure and $z(k) = y(k)$ and the goal is to get.

$$\lim_{k\to\infty} z(k) - r(k) = 0$$

When steady state is reached for a discrete system this give

$$\mathbf{x}(k+1) = \mathbf{x}(k)$$

which is equivalent to $\dot{\mathbf{x}} = 0$. When applying this to the discrete state space model the following result is found.

$$\begin{aligned}\mathbf{x}(k+1) = \mathbf{x}_{ss}(k) &= \Phi\mathbf{x}_{ss}(k) + \Gamma u(k)\\ \mathbf{x}_{ss}(k) &= \Phi\mathbf{x}_{ss}(k) - \Gamma F\mathbf{x}_{ss} + \Gamma Gr(r)\\ \mathbf{x}_{ss}(k) &= (\Phi - \Gamma F)\mathbf{x}_{ss}(k) + \Gamma Gr(k)\\ (I - (\Phi - \Gamma F))\mathbf{x}_{ss}(k) &= \Gamma Gr(k)\\ \mathbf{x}_{ss}(k) &= (I - (\Phi - \Gamma F))^{-1}\Gamma Gr(k)\end{aligned}$$

For zero steady state error this gives

$$\begin{aligned}z(k) - r(k) = \tilde{C}\mathbf{x}_{ss}(k) - \tilde{D}F\mathbf{x}_{ss}(k) + \tilde{D}Gr(k) - r(k) &= 0\\ r(k) &= (\tilde{C} - \tilde{D}F)\mathbf{x}_{ss}(k) + \tilde{D}Gr(k)\\ r(k) &= (\tilde{C} - \tilde{D}F)(I - (\Phi - \Gamma F))^{-1}\Gamma Gr(k) + \tilde{D}Gr(k)\\ r(k) &= (\tilde{D} + (\tilde{C} - \tilde{D}F)(I - (\Phi - \Gamma F))^{-1}\Gamma)Gr(k)\\ I &= (\tilde{D} + (\tilde{C} - \tilde{D}F)(I - (\Phi - \Gamma F))^{-1}\Gamma)G\end{aligned}$$

This equality only holds when

$$G = (\tilde{D} + (\tilde{C} - \tilde{D}F)(I - (\Phi - \Gamma F))^{-1}\Gamma)^{-1}I$$

in our case because we have a SISO system and $z(k) = y(k)$ the discreet reference gain reduces to

$$G_d = 1/(\tilde{C}(I - (\Phi - \Gamma F))^{-1}\Gamma) \tag{9}$$

In questions 5,6 and 7 it has been shown that these gains eliminate the steady state error for a reference input. In question 6 a procedure has been shown to completely reduce the influence of a step disturbance which won't be repeated here. But it is clear that with proper information about the type of disturbance and the use of the nice state space structure this hurdle can be overcome.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Q13 The time delay

Investigating the influence of a time delay is started by looking at time delay in the continuous time. In the Laplace domain a time delay is given by

$$delay = e^{-sT}$$

were T is the time delay. A system that would suffer from a time delay would get the system transferfunction multiplied by this time delay, for example

$$Delayed\ system\ F(s) = e^{-sT} * F(s)$$

Below is a bode plot of a 1 times, 0.5 times and 2 times a unit time delay so the influence on the system can be seen.

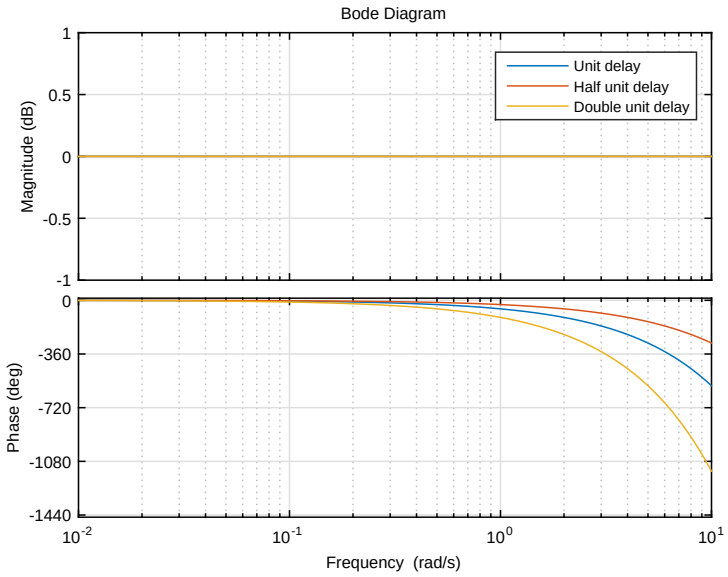


Figure 33: Bode plot of different time delays

As can be seen in figure 33 the time delay doesn't introduce a change in the magnitude of the signal but introduces a big phase lag. This can have terrible consequences for the stability of our system, if the phase lead is lost at the crossover frequency this could result in an unstable system. Also systems with a faster sampling time would have a shorter time delay, this would result in phase loss at higher frequencies.

PID controllers In this question the discrete transfer functions were used because it's easier to add an delay to those. They are computed as was described in the note from question 4. For the discrete time transfer functions only a z^{-1} operator should be added to achieve the time delay. This would change the controller in to

$$C(z) \rightarrow C(z)z^{-1}$$

for the tracking controller this gave the following performance of the step input.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

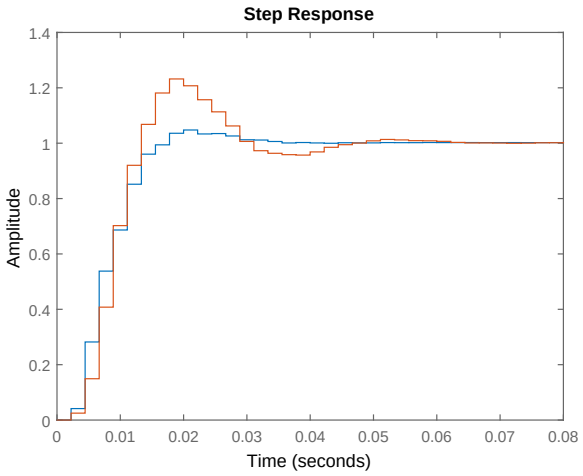


Figure 34: Step response PID tracker with time delay and old system

Clearly a deterioration in the performance can be seen, the system displays more overshoot and is more oscillatory. This can be explained by the drop of phase lead which reduces the damping in the system. This problem could be remedied by increasing the influence of the D action, this would require the gain to be reduced and would give a slower system response but $< 5\%$ overshoot.

Below is the disturbance rejection PID controller.

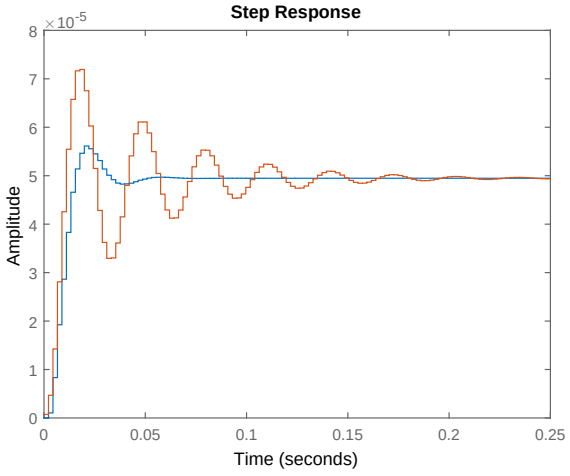


Figure 35: Step response PID disturbance rejector with time delay and old system

This controller has the same problem as the tracking controller. Because the pole and zero of the controller were pushed together as close as possible there isn't much phase lead. By reducing this phase lead even more the system starts to become unstable and oscillations enter the system. This could be solved by separating the pole and zero more this would increase the phase lead and thus reduce the oscillations but it would also increase the steady state error. Which would lead to a worse rejection of the step disturbance which could be undesirable.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Pole Placement For the pole placement controller a simulink model was made so the delay could be placed at the control input, this was not possible with state space structure of Matlab. The controller with the pole placement had an extremely fast response and a time delay could be detrimental for the reference tracking.

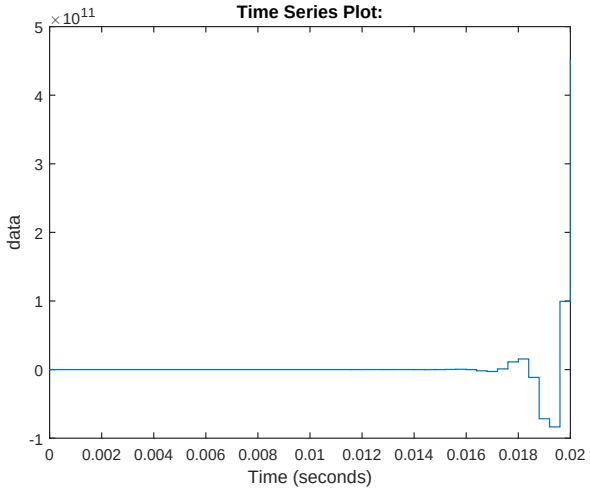


Figure 36: The pole placement controller with time delay

It can clearly be seen that the system has become unstable, the old system isn't even visible in the figure. The reason it has become unstable is because the extremely fast response of the system causes the discrete poles to be pushed to the edge of the unit circle. Adding 'instability' by introducing phase lag pushed this system over the edge and made it unstable. The system can easily be stabilized by choosing poles which are closer to the origin of the unit circle, this would allow for some phase loss.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Observer with disturbance Only the controller with the disturbance rejection is shown here because the unstability of the pole placement controller will automatically make the controllers with an observer unstable as well because they use the same pole placement. And they both have the same remedy. It might be thought that the observer with disturbance estimation is different, the following figure shows this is not the case.

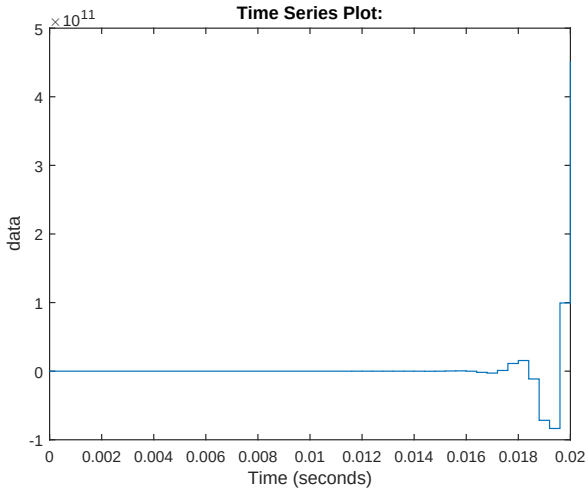


Figure 37: Observer with disturbance rejection and time delay

LQ control For the LQ control the same simulink model was used as for the pole placement controller. The LQ control with a single time delay shows the following step response.

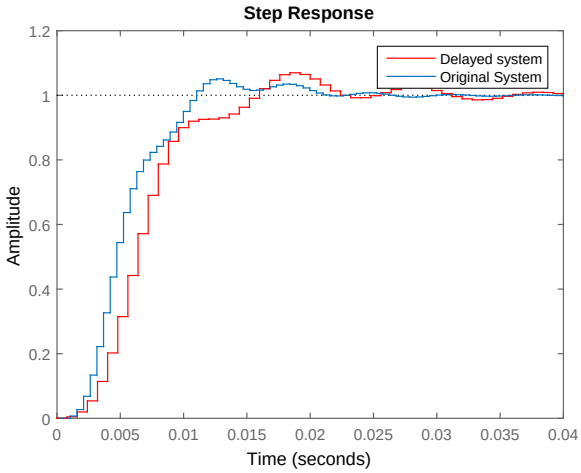


Figure 38: The LQ controller with time delay

Clearly the delayed system shows some performance deterioration but very little. We know from continuous time that the LQ controller has amazing stability properties. This is showcased here in discrete time with a super fast respond even though there is a time delay. The LQ controller has no problems, so there is nothing that needs to be redesigned.

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Overall Conclusions

The overall conclusion of the hard disk heads control is that is can be made to react very fast, even with the limitations on the control action. If a choice had to be made I would choose the pole placement because it yields the best results and has the most flexibility. But LQ control is a close second because of the ease of the controller design and the amazing stability properties. PID control would come in last because it almost feels like an art to tune your controller perfectly and it's hard to get equivalent results to the pole placement or LQ control. Lastly the observers need to be mentioned, their power is clearly demonstrated in question 6 when you estimate an input disturbance which is compensated for over time. Observers are perfect if there is information about the nature of the disturbances available, if this is not the case it can be tricky.

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Matlab Code

```
%% The Quest System info
%% By Martijn Kerklaan 4095723

% Define Constants

Omega = 1000;    % [Rad/s]
Zeta = 0.1;      % [-]

% Define Transfer function
Num = [0 0 0 2*Zeta*Omega Omega^2];
Den = [1 2*Zeta*Omega Omega^2 0 0];
G = tf(Num,Den)

% Bode plot options
opts = bodeoptions;
opts.Grid = 'on';

save('Plant','G','Num','Den','opts')

%% The Quest Question 1
%% By Martijn Kerklaan 4095723

clc
clear all
close all

% Define Constants

Omega = 1000;    % [Rad/s]
Zeta = 0.1;      % [-]

% Define Transfer function
Num = [0 0 0 2*Zeta*Omega Omega^2];
Den = [1 2*Zeta*Omega Omega^2 0 0];
G = tf(Num,Den)

% Bode plot options
opts = bodeoptions;
opts.Grid = 'on';

% View the system
figure
title('Bode plot of the plant')
bode(G,logspace(1,5,1000),opts)

%% Start controlling the system
```


Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
% Select fc
Omega_c = 100;

% Calculate Diverential control
Omega_d = Omega_c/3; % Choosing the upper and lower D value
Omega_t = 3*Omega_c;


C_d =tf([1/Omega_d 1],[1/Omega_t 1]);

% Build controller
C = C_d;

% Check controller
figure
bode(C)

% Check system
figure
bode(G*C,logspace(1,5,1000),opts)
title('System controlled by controller 1.0')

%% Adjusting gain
% Get parameters
[MAG,PHASE,W] = bode(G*C,logspace(1,5,1000));
MAG = squeeze(MAG); % [-]
PHASE = squeeze(PHASE); % [Deg]
W = squeeze(W); % [Rad/s]

% Find new K value to get desired omega_c

kt = interp1(W,MAG,Omega_c);
kp = (1/kt);
% Adjust controller to go to our desired Omega_c
C = kp*C;

% Try again
figure
bode(G*C,opts)
title('System controlled by controller 1.1')

%% Checking the step response

SysCL = CL_sys(G,C);

figure
step(SysCL,0.2)
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
[y,t] = step(SysCL,0.2);
si = stepinfo(y,t)

%% System round 2
% Calculate Diverential control
Omega_d = Omega_c/300; % Choosing the upper and lower D value
Omega_t = 3*Omega_c;

C_d =tf([1/Omega_d 1],[1/Omega_t 1]);

% Build controller
C = C_d;

% Adjusting gain
[Gm,Pm,Wgm,Wpm] = margin(C_d*G); % Get parameters

% Adjust controller to go to our desired Omega_c
kp = 1\Gm;
C = kp*C_d;

SysCL = CL_sys(G,C);

% Still missing that 3dB
Rest = db2mag(3);
kp = kp*1/Rest;

C = kp*C_d

% Tweaking
Value = 0.255 % Found after tweaking
kp = kp*Value;
C = kp*C_d;

figure
step(CL_sys(G,C));
[y,t] = step(CL_sys(G,C));
stepinfo(y,t)

% Try again
figure
bode(G*C,opts)
title('System controlled by controller 1.2')
[Gm,Pm,~,~] = margin(C*G); % Get parameters
mag2db(Gm)
Pm
clear Gm Pm

save('Q1','C','C_d','Omega_t','Omega_d','Omega_c','kp')
```


Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
%% The Quest Question 2
%% Made by Martijn Kerklaan 4095723
clc
clear all
close all

load('Plant')
%%
Omega_c = 200;

% Calculate Diverential control
Omega_d = Omega_c/1.9; % Choosing the upper and lower D value
Omega_t = 1.9*Omega_c;

C_d =tf([1/Omega_d 1],[1/Omega_t 1]);
C = C_d;
% Get parameters
[MAG,PHASE,W] = bode(G*C,logspace(-3,5,1000));
MAG = squeeze(MAG); % [-]
PHASE = squeeze(PHASE); % [Deg]
W = squeeze(W); % [Rad/s]

% Find new K value to get desired omega_c

kt = interp1(W,MAG,Omega_c);
kp = (1/kt);
% Adjust controller to go to our desired Omega_c
C = kp*C;

figure
bode(G*C,opts)

[Gm,Pm,~,~] = margin(G*C)

figure
step(G/(1+C*G))
[y,t] = step(G/(1+C*G));
stepinfo(y,t)

save('Q2','C','C_d','Omega_t','Omega_d','Omega_c','kp')

%% The Quest Question 3
%% Made by Martijn Kerklaan 4095723

clc
clear all
close all
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
%% Q3 State Space + Discretize system
load('Plant')
load('Q1');

[y,t] = step(CL_sys(G,C));
tmp = stepinfo(y,t);
% Plant
[Ag,Bg,Cg,Dg] = tf2ss(Num,Den);

% Controller
[NumC,DenC] = tfdata(CL_sys(G,C));
[Ac,Bc,Cc,Dc] = tf2ss(NumC{1},DenC{1});

% Discretize plant
Nr = 4;
h = tmp.RiseTime/Nr;
PlantC = ss(Ag,Bg,Cg,Dg);
PlantD = c2d(PlantC,h,'zoh');

save('Q3','PlantD','PlantC');

%% The Quest Question 4
%% Made by Martijn Kerklaan 4095723

clc
clear
close all

%%  Q4 Comapre Step responses

% Load
load('Plant');
load('Q1');

% Determine sampling time
[y,t] = step(CL_sys(G,C));
tmp = stepinfo(y,t);
Nr = 4;
h = tmp.RiseTime/Nr;

% Discretize closed loop system
[NumC,DenC] = tfdata(CL_sys(G,C));
[Acl,Bcl,Ccl,Dcl] = tf2ss(NumC{1},DenC{1});
CL_TrackC = ss(Acl,Bcl,Ccl,Dcl);
Cl_TrackD = c2d(CL_TrackC,h,'zoh');

figure
grid on
```


Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
step(CL_sys(G,C),0.045); hold on;
step(Cl_TrackD,0.045);
legend({'Continous' 'Discretized h = 0.0022'})

%% Do it for question 2
tmp = load('Q2');
C = tmp.C;
[y,t] = step(G/(1+C*G),0.045);
tmp = stepinfo(y,t);

Nr = 4;
h = tmp.RiseTime/Nr

[NumS, DenS] = tfdata(G/(1+C*G));
[Acl,Bcl,Ccl,Dcl] = tf2ss(NumS{1},DenS{1});

CL_DistC = ss(Acl,Bcl,Ccl,Dcl);
CL_DistD = c2d(CL_DistC,h,'zoh');

figure
step(G/(1+C*G),0.045); hold on
step(CL_DistD,0.045)
legend({'Continous' 'Discretized h = 0.0022'})

save('Q4','Cl_TrackD','CL_DistD')

%% The Quest Question 5
%% Made by Martijn Kerklaan 4095723

clc
clear
close all

load('Plant');
load('Q1');

% Get that data
[Sys_StateFBD1, Sys_StateFBC1, ~, ~, ~] = PolePlace(1);
[Sys_StateFBD2, Sys_StateFBC2, ~, ~, ~] = PolePlace(2);
[Sys_StateFBD4, Sys_StateFBC4, ~, ~, ~] = PolePlace(3);
[Sys_StateFBD8, Sys_StateFBC8, K, Ggain, PolesK] = PolePlace(4); % Only save the be

% Plot figure and show eigenvalues
figure
step(CL_sys(G,C)); hold on;
step(Sys_StateFBD1);
step(Sys_StateFBD2);
step(Sys_StateFBD4);
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
step(Sys_StateFBD8);
legend({'System Q1' 'Object Set' 'Object Set/2' 'Object Set/4' 'Object Set/8'})

% Check eigenvalues
eig(Sys_StateFBC1.a)
eig(Sys_StateFBC2.a)
eig(Sys_StateFBC4.a)
eig(Sys_StateFBC8.a)

eig(Sys_StateFBD1.a)
eig(Sys_StateFBD2.a)
eig(Sys_StateFBD4.a)
eig(Sys_StateFBD8.a)

% Check step response ifo
[y1,t1] =  step(Sys_StateFBD1);
[y2,t2] =  step(Sys_StateFBD2);
[y3,t3] =  step(Sys_StateFBD4);
[y4,t4] =  step(Sys_StateFBD8);

stepinfo(y1,t1)
stepinfo(y2,t2)
stepinfo(y3,t3)
stepinfo(y4,t4)

% Save info
save('Q5','K', 'PolesK','Ggain', 'Sys_StateFBD1','Sys_StateFBD2','Sys_StateFBD4','S

%% The Quest Question 6
%% Made by Martijn Kerklaan 4095723

clc
clear
close all

%% Q6

load('Plant')
load('Q3');
load('Q5');

h = Sys_StateFBD8.Ts;

A = PlantC.a;
B = PlantC.b;
C = PlantC.c;
D = PlantC.d;
```


Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
% Place poles observer
PolesL = PolesK*2;
L = place(transpose(A),transpose(C),PolesL)';

% Build combined ss model
At = [A -B*K;
     L*C A-(L*C)-(B*K)];

Bt = [B*Ggain;
     B*Ggain];

Ct = [C zeros(size(C));
     zeros(size(C)) C];

SysStateObsC = ss(At,Bt,Ct,0);

% Discretize model
h = h/2; % Reduce from w*h= 0.2 to w*h = 0.1
SysStateObsD = c2d(SysStateObsC,h,'tustin');

% Create block wave for observer and non observer
t1 = 0:h:0.04; % Sampling at f = 1/h;
u1 = square(2*pi*50*t1);

t2 = 0:Sys_StateFBD8.Ts:0.04; % Sampling at f = 1/h;
u2 = square(2*pi*50*t2);

% Plot models
x0 = [0 50 0 0]; % give initial condition

% Calculate errors and plotting
[yobs,tobs, xobs] = lsim(SysStateObsD,u1,t1,[x0 0 0 0 0]);
[ystate,tstate, xstate] = lsim(Sys_StateFBD8,u2,t2, x0);

figure
plot(tobs,xobs(:,2)-xobs(:,6))
xlabel('time [s]')
ylabel('Error [-]')
title('Estimation error of true state x_2')

%% No observer

% Define new state matrices for constant load dist
Ae = [A B; zeros(1,length(C)) 0];
Be = [B; 0];
Ce = [C 0];
De = 0;
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
% performance matrices
C_tilde = C;
D_tilde = D;
Ce_tilde = Ce;
De_tilde = De;

% New Feedback gain
Ke = [K 1];

% Design new opserver gain
Le = place(Ae',Ce',[PolesK*1.1 -10])';

% Combine
At = [Ae -Be*Ke;
      Le*Ce Ae-Be*Ke-Le*Ce];

Bt = [Be*Ggain;
     Be*Ggain];

Ct =  [Ce zeros(size(Ce));
     zeros(size(Ce)) Ce];

SysObsLoadC = ss(At,Bt,Ct,0);
SysObsLoadD = c2d(SysObsLoadC,h,'tustin');

Sys_StateC = ss(Ae-Be*Ke,Ggain*Be,Ce,0);
Sys_StateD = c2d(Sys_StateC,Sys_StateFBD8.Ts,'tustin');

% Constant load disturbance so set initial condition to dist
x0 = [0 0 0 0 -200000];
o0 = [0 0 0 0 0];

t1 = 0:h:1; % Sampling at f = 1/h;
u1 = square(2*pi*50*t1);

t2 = 0:Sys_StateFBD8.Ts:1; % Sampling at f = 1/h;
u2 = square(2*pi*50*t2);

[yobs, tobs, xobs] = lsim(SysObsLoadD,u1,t1,[x0 o0]);
[ystate,tstate, xstate] = lsim(Sys_StateD,u2,t2, x0);

figure
plot(tobs,yobs(:,1),tstate,ystate(:,1)); hold on
plot(t1,u1,'Color',[0.5 0.5 0.5])
ylim([-2 2])
xlabel('time [s]')
ylabel('Amplitide [-]')
legend({'System with Observer' 'System with state feedback' 'Reference'})
```


Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
title('Simulation Results')

figure
plot(tobs,xobs(:,5)-xobs(:,10))
xlabel('time [s]')
ylabel('Error [-]')
title('Estimation error of true state w')

% Save them values
save('Q6','SysObsLoadC', 'SysObsLoadD', 'SysStateObsC', 'SysStateObsD','K','Ke','G

%% The Quest Question 7
%% Made by Martijn Kerklaan 4095723

clc
clear
close all

%% Question 7

load('Q3')
A = PlantC.a;
B = PlantC.b;
C = PlantC.c;
D = PlantC.d;

mq = 1e7;
mr = 1e-3;
Q = C'*C*mq;
R = 1*mr;

% Continous LQ control
[Kc,S,e] = lqr(PlantC,Q,R,0);
GgainC = 1/((D-(C-(D*Kc)))*inv(A-(B*Kc))*B);
SysC = ss(A-B*Kc,GgainC*B,C,0);

% Discretize with appropriate sampling time
[y,t] = step(SysC);
tmp = stepinfo(y,t);
Nr = 16;
h = tmp.RiseTime/Nr

PlantD = c2d(PlantC,h,'zoh');
% Disc equiv
A = PlantD.a;
B = PlantD.b;
C = PlantD.c;
D = PlantD.d;
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
Q = Q;
% Q = eye(4)*1e15;
R = R;

% dlqr
[Kd,S,e] = dlqr(A,B,Q,R,0);
GgainD = 1/((C-D*Kd)*inv(eye(4)-(A-B*Kd))*B+D);
SysD = ss(A-B*Kd,GgainD*B,C,0,PlantD.Ts);

% Plot figure
str = sprintf('Step response Q*%g, R*%g, h =%f', mq, mr, h)
figure
step(SysD)
title(str)

[y,t] = step(SysD);
stepinfo(y,t)

save('Q7','SysD','Kd','GgainD')

%% The Quest Question 8
%% Made by Martijn Kerklaan 4095723

clc
clear
close all

% For Question 1
Q1 = load('Q1');
tf = 0.045;
[NumC, DenC] = tfdata(Q1.C);
SimOut = sim('ModelQ1');

figure
plot(u)
xlabel('Time[s]')
ylabel('Amplitude [-]')
legend({'Input control signal'})
title('Tracking PID control')

% For Question 2
clear
Q2 = load('Q2');

tf = 0.045;
[NumC, DenC] = tfdata(Q2.C);
SimOut = sim('ModelQ2');
```


Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
figure
plot(u)
xlabel('Time[s]')
ylabel('Amplitude [-]')
legend({'Input control signal'})
title('Disturbance rejection PID control')

% For Question 5
clear
load('Q5')

[yQ5,tQ5,xQ5] = step(Sys_StateFBD8);

figure
plot(tQ5,-K*xQ5'+Ggain*1)
xlabel('Time[s]')
ylabel('Amplitude [-]')
title('Pole placement state feedback')
legend({'Input control signal'})


% For question 6
clear
load('Q6')

[yQ6S,tQ6S,xQ6S] = step(SysStateObsC,0.01);
[yQ6O,tQ6O,xQ6O] = step(SysObsLoadC,0.01);

figure
plot(tQ6S,-K*xQ6S(:,1:4)'+Ggain*1); hold on
plot(tQ6O,-Ke*xQ6O(:,1:5)'+Ggain*1); hold on
xlabel('Time[s]')
ylabel('Amplitude [-]')
title('Observer control')
legend({'Input control signal Observer' 'Input control signal disturbance Observer'

% For Question 7
clear
load('Q7')

[yQ7,tQ7,xQ7] = step(SysD);

figure
plot(tQ7,-Kd*xQ7'+GgainD*1)
xlabel('Time[s]')
ylabel('Amplitude [-]')
title('LQ control')
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
legend({'Input control signal'})

%% The Quest Question 9
%% Made by Martijn Kerklaan 4095723

close all
clear

load('Plant')
XXX = load('Q4')

% Found omega_c
Omega_c = 7.953;
%% Build lead lag controller
% Calculate Diverential control
Omega_d = Omega_c/300; % Choosing the upper and lower D value
Omega_t = 3*Omega_c;

C_d =tf([1/Omega_d 1],[1/Omega_t 1]);

% Build controller
C = C_d;

%% Adjusting gain
% Get parameters
[MAG,PHASE,W] = bode(G*C_d,logspace(-4,3,1000));
MAG = squeeze(MAG); % [-]
PHASE = squeeze(PHASE); % [Deg]
W = squeeze(W); % [Rad/s]

kt = interp1(W,MAG,Omega_c);
kp = (1/kt);

% Adjust controller to go to our desired Omega_c
C = kp*C;

figure
bode(C*G,opts)

%% Find control action
tfinal = 5;
[NumC, DenC] = tfdata(C);
SimOut = sim('ModelQ1');

figure
plot(u)
xlabel('Time[s]')
ylabel('Amplitude [-]')
```


Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
legend({'Input control signal'})
title('Tracking PID control')

max(u)

%% Discretize system
% Determine sampling time
[y,t] = step(CL_sys(G,C));
tmp = stepinfo(y,t);
Nr = 6;
h = tmp.RiseTime/Nr

% Discretize closed loop system
[NumC,DenC] = tfdata(CL_sys(G,C));
[Acl,Bcl,Ccl,Dcl] = tf2ss(NumC{1},DenC{1});
CL_TrackC = ss(Acl,Bcl,Ccl,Dcl);
Cl_TrackD = c2d(CL_TrackC,h,'zoh');

%% Plot stepresponze
figure
grid on
step(XXX.Cl_TrackD); hold on;
step(Cl_TrackD);
legend({'Old PID Tracking' 'New PID Tracking with h = 0.0476'})

%% Show info
[y,t] = step(Cl_TrackD);
stepinfo(y,t)
[Gm,Pm,~,~] = margin(C*G)

save('Q9','C','Omega_c','C_d','Omega_d','Omega_t')

%% The Quest Question 10
%% Made by Martijn Kerklaan 4095723

clear
close all

load('Plant')
load('Q3')
load('Q5')
load('Q9')

[y,t,x] = step(Sys_StateFBD8);
figure
plot(t,x(:,1), t,x(:,2), t,x(:,3), t,x(:,4))
title('Old Control Action')
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
% Load starting objectives ans values
[y,t] = step(CL_sys(G,C));
X = stepinfo(y,t);
ts = X.SettlingTime/3.2;
% ts = X.SettlingTime/1; % Initial Guess
Os = 5;

clear X

% Calculate pole locations
zeta = sqrt((log(Os/100))^2/(pi^2+(log(Os/100))^2));
Omega_n = (-1*log(0.01))/ts*zeta;

p1 = -1*zeta*Omega_n + j*Omega_n*sqrt(1-zeta^2);
p2 = -1*zeta*Omega_n - j*Omega_n*sqrt(1-zeta^2);

% Build the stater feedback matrix
PolesK = [-37.3 p1 -36.5 p2];
% PolesK = [-100 p1 -101 p2]; % Initial guess
K = place(PlantC.a,PlantC.b,PolesK);

% Calculate reference gain
Ggain = 1/((PlantC.d-(PlantC.c-(PlantC.d*K)))*inv(PlantC.a-(PlantC.b*K))*PlantC.b);

% Create a state space model with gain on the reference and state feedback
Sys_StateFBC = ss(PlantC.a-PlantC.b*K,Ggain*PlantC.b,PlantC.c,0);

% Disc that shit
h2 = 0.2/Omega_n;
Sys_StateFBD = c2d(Sys_StateFBC,h2,'tustin');

% Plot control action
[yn,tn,xn] = step(Sys_StateFBC);

figure
plot(tn,-K*xn'+Ggain*1)
title('Control Action')

max(-K*xn'+Ggain*1)

figure
step(Sys_StateFBD)
[y,t] = step(Sys_StateFBD);
stepinfo(y,t)
eig(Sys_StateFBD.a)

save('Q10','Sys_StateFBC','Sys_StateFBD','K','Ggain','PolesK')
```


Dit document is gratis te downloaden op StudeerSnel.nl

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
%% The Quest Question 11
%% Made by Martijn Kerklaan 4095723

clear
close all

load('Q3')
A = PlantC.a;
B = PlantC.b;
C = PlantC.c;
D = PlantC.d;

% Determinie weighing matrices
mq = 1e7;
mr = 2.0693e2;
Q = C'*C*mq;
R = 1*mr;

% Continous LQ control
[Kc,S,e] = lqr(PlantC,Q,R,0);
GgainC = 1/((D-(C-(D*Kc)))*inv(A-(B*Kc))*B);
SysC = ss(A-B*Kc,GgainC*B,C,0);

% Discretize with appropriate sampling time
[y,t] = step(SysC);
tmp = stepinfo(y,t);
Nr = 16;
h = tmp.RiseTime/Nr;

PlantD = c2d(PlantC,h,'zoh');
% Disc equiv
A = PlantD.a;
B = PlantD.b;
C = PlantD.c;
D = PlantD.d;

Q = Q;
% Q = eye(4)*1e15;
R = R;

% dlqr
[Kd,S,e] = dlqr(A,B,Q,R,0);
GgainD = 1/((C-D*Kd)*inv(eye(4)-(A-B*Kd))*B+D);
SysD = ss(A-B*Kd,GgainD*B,C,0,PlantD.Ts);

str = sprintf('Step response Q*%g, R*%g, h =%f', mq, mr, h)

figure
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
step(SysD)
title(str)
[y,t] = step(SysD);
stepinfo(y,t)

% Plot and calculate control control action
[yQ7,tQ7,xQ7] = step(SysD);
u = -Kd*xQ7'+GgainD*1;
figure
plot(tQ7,u)
xlabel('Time[s]')
ylabel('Amplitude [-]')
title('LQ control')
legend({'Input control signal'})

u_max = max(u)

%% The Quest Question 13
%% Made by Martijn Kerklaan 4095723

clc
clear
close all

load('Plant')

T = 1;
s = tf([1 0],[0 1])
delay1 = exp(-s*T);
delay05 = exp(-s*T*0.5);
delay2 = exp(-s*T*2);

figure
bode(delay1,logspace(-2,1,100),opts); hold on
bode(delay05,logspace(-2,1,100),opts);
bode(delay2,logspace(-2,1,100),opts);
legend({'Unit delay' 'Half unit delay' 'Double unit delay'})

%% Add delay to PID Track
clear
load('Plant')
load('Q1')
load('Q4')

h = Cl_TrackD.Ts
z = tf('z',h)

Cd = c2d(C,h,'tustin')
```


Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
Pd = c2d(G,h,'tustin')

figure
step(Cl_TrackD); hold on
step(CL_sys(Pd,Cd*z^(-1)))

%% Add delay to PID dist
clear
load('Plant')
load('Q2')
load('Q4')

h = CL_DistD.Ts
z = tf('z',h)

Cd = c2d(C,h,'tustin')
Pd = c2d(G,h,'tustin')

figure
step(CL_DistD); hold on
step(Pd/(1+Pd*Cd*z^(-1)))
%% Add delay to pole placement
clear
load('Plant')
load('Q3')
[Sys_StateFBD8, Sys_StateFBC8, K, Ggain, PolesK] = PolePlace(4);

poles = eig(Sys_StateFBD8.a);
h = Sys_StateFBD8.Ts

% Disc that shit
Sys = c2d(PlantC,h,'tustin');

% Find discrete K matrix
K = place(Sys.a,Sys.b,poles);

A = Sys.A;
B = Sys.B;
C = Sys.C;

GgainD = 1/(C*inv(eye(4)-(A-B*K))*B);

tfinal = 0.02;
SimOut = sim('DiscretePoleSS');

figure
plot(y);
```


Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
%% Observer dist Delay
clear
load('Plant')
load('Q3')
load('Q5');
load('Q6');

h = SysObsLoadD.Ts;

A = SysObsLoadD.a(1:5,1:5);
BK = -SysObsLoadD.a(1:5,6:10);
LC = SysObsLoadD.a(6:10,1:5);
ABKLC = SysObsLoadD.a(6:10,6:10);

BG1 = SysObsLoadD.b(1:5,1);
BG2 = SysObsLoadD.b(6:10,1);
Steps = round(0.02/h); % Number of steps
t = linspace(0,h*Steps,Steps+1); % Time vector

r = ones(Steps);
r(1) = 0;

x(:,1) = [0 0 0 0 0];
x(:,2) = [0 0 0 0 1];

xhat(:,1) = [0 0 0 0 0];
xhat(:,2) = [0 0 0 0 0];

for k = 2:Steps
    x(:,k+1) = A*x(:,k) - BK*xhat(:,k-1) + BG1*r(k-1);
    xhat(:,k+1) = LC*x(:,k) + ABKLC*xhat(:,k) + BG2*r(k);
end

y1 = SysObsLoadD.c(1,1:5)*x;
y2 = SysObsLoadD.c(1,1:5)*xhat;
figure
stairs(t,y1)
xlabel('Time [s]')
ylabel('Amplitude [-]')
title('Step response')
legend('Observer with Load disturbance')

%% Add delay LQ
clear
load('Plant')
load('Q3')
load('Q7')
```


Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

```
h = SysD.Ts;

SysNew = c2d(PlantC,h,'tustin');

A = SysNew.a;
B = SysNew.b;
C = SysNew.c;

K = Kd;
GgainD = GgainD;
tfinal = 0.04;

SimOut = sim('DiscretePoleSS');

figure
plot(y,'-r'); hold on
step(SysD,0.04)
legend({'Delayed system' 'Original System'})
```

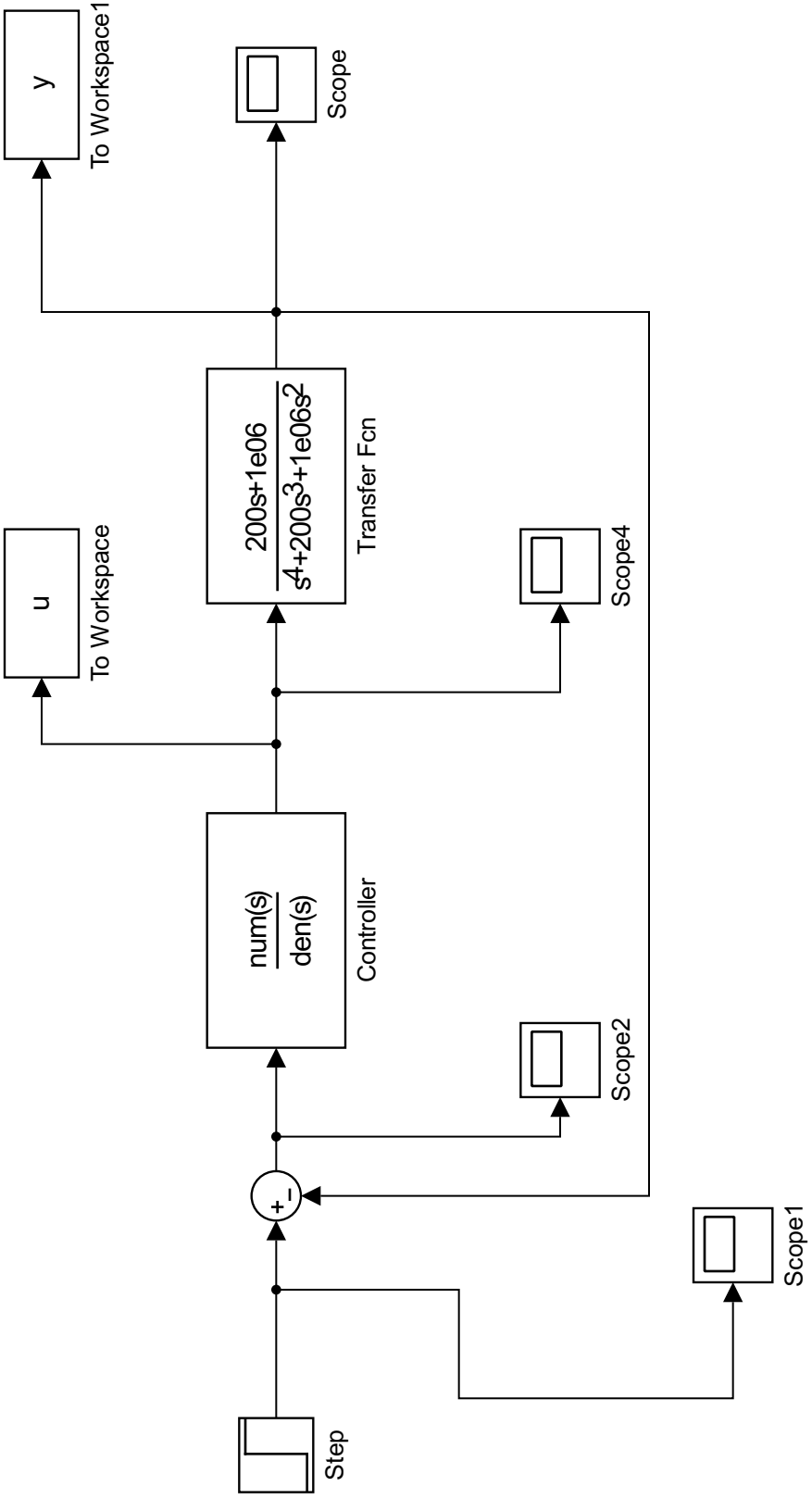

Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Matlab Functions

```
function [H] = CL_sys(System, Controller)
%Computes the closed loop transfer function H = CG/(1+CG) from the system
%and the controller

H = (Controller*System)/(1+Controller*System);
end

function [Sys_StateFBD, Sys_StateFBC, K, Ggain, PolesK] = PolePlace(Divider)
%UNTITLED8 Summary of this function goes here
%   Detailed explanation goes here
load('Plant')
load('Q1')
load('Q3')

% Load starting objectives ans values
[y,t] = step(CL_sys(G,C));
X = stepinfo(y,t);
ts = X.SettlingTime/Divider;
Os = 5;

clear X

% Calculate pole locations
zeta = sqrt((log(Os/100))^2/(pi^2+(log(Os/100))^2));
Omega_n = (-1*log(0.01))/ts*zeta;

p1 = -1*zeta*Omega_n + j*Omega_n*sqrt(1-zeta^2);
p2 = -1*zeta*Omega_n - j*Omega_n*sqrt(1-zeta^2);

% Build the stater feedback matrix
PolesK = [-10000 p1 -10001 p2];
K = place(PlantC.a,PlantC.b,PolesK);

% Calculate reference gain
Ggain = 1/((PlantC.d-(PlantC.c-(PlantC.d*K)))*inv(PlantC.a-(PlantC.b*K))*PlantC.b);

% Create a state space model with gain on the reference and state feedback
Sys_StateFBC = ss(PlantC.a-PlantC.b*K,Ggain*PlantC.b,PlantC.c,0);

% Disc that shit
h2 = 0.2/Omega_n;
Sys_StateFBD = c2d(Sys_StateFBC,h2,'tustin');

end
```

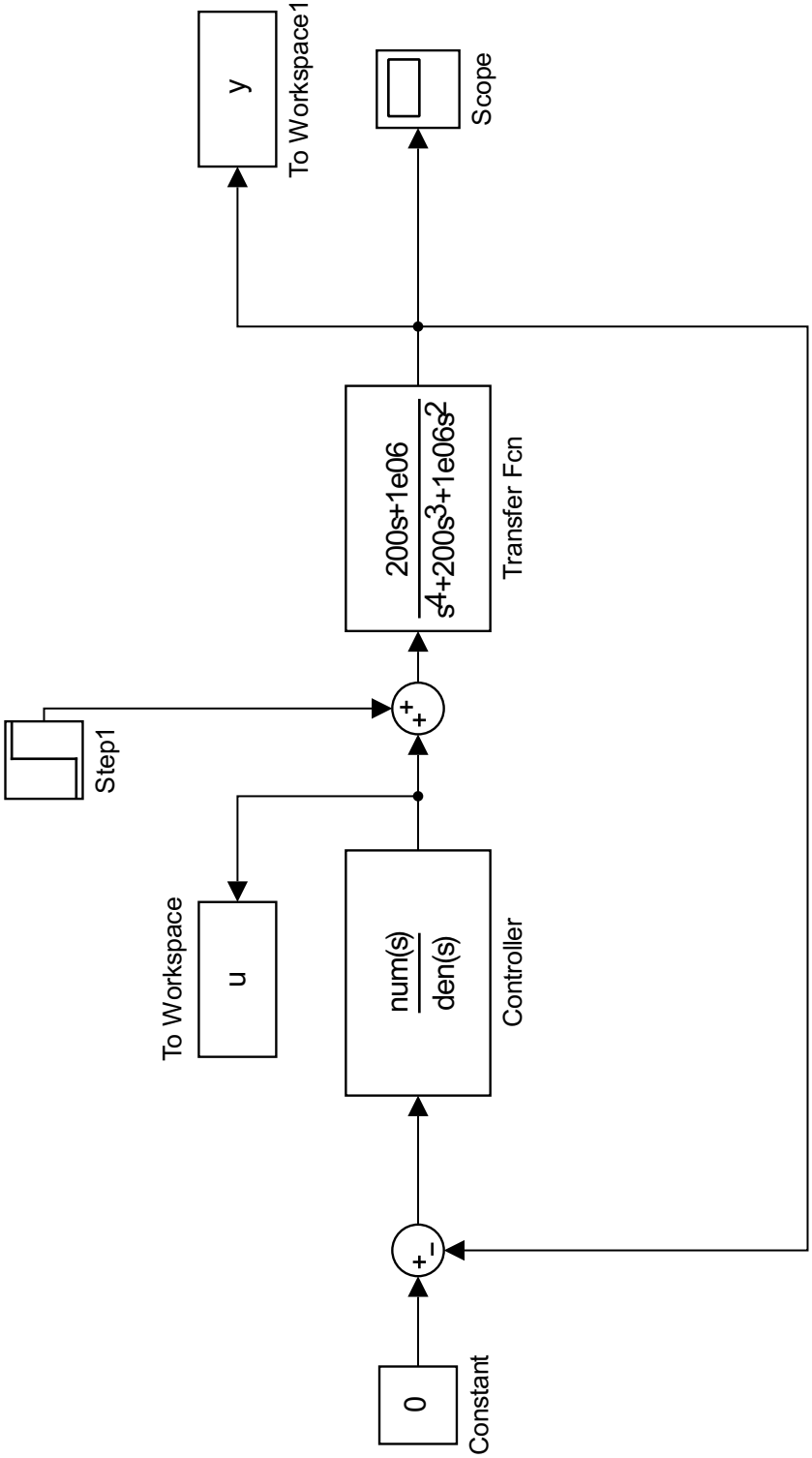

Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Figure 39: Simulink Model of PID Track used in Q8



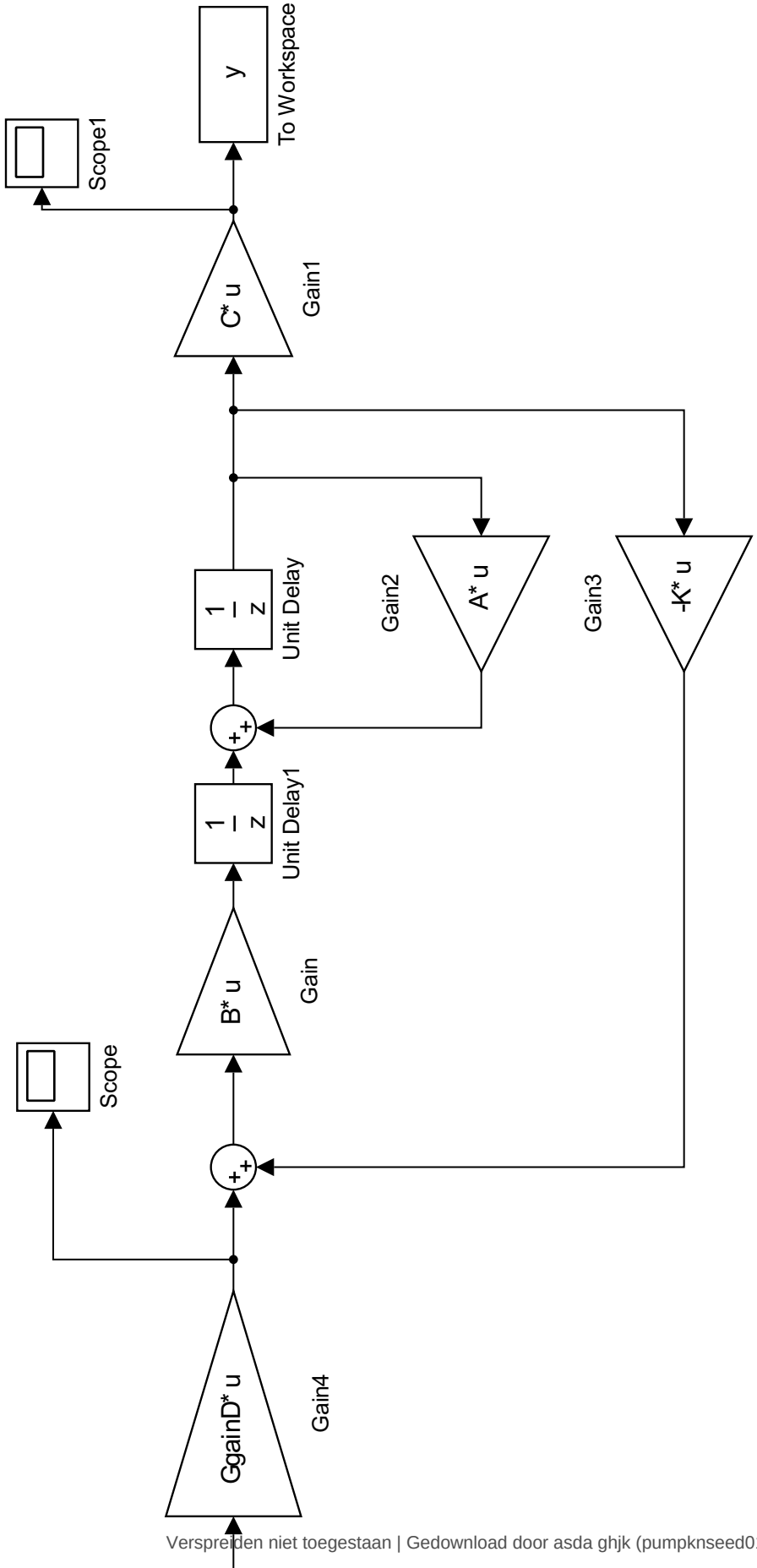
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Figure 40: Simulink Model of PID reject used in Q8



Dit document is gratis te downloaden op StudeerSnel.nl
Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)

Figure 41: Simulink Model state feedback used in Q8



Verspreiden niet toegestaan | Gedownload door asda ghjk (pumpknseed0132@gmail.com)