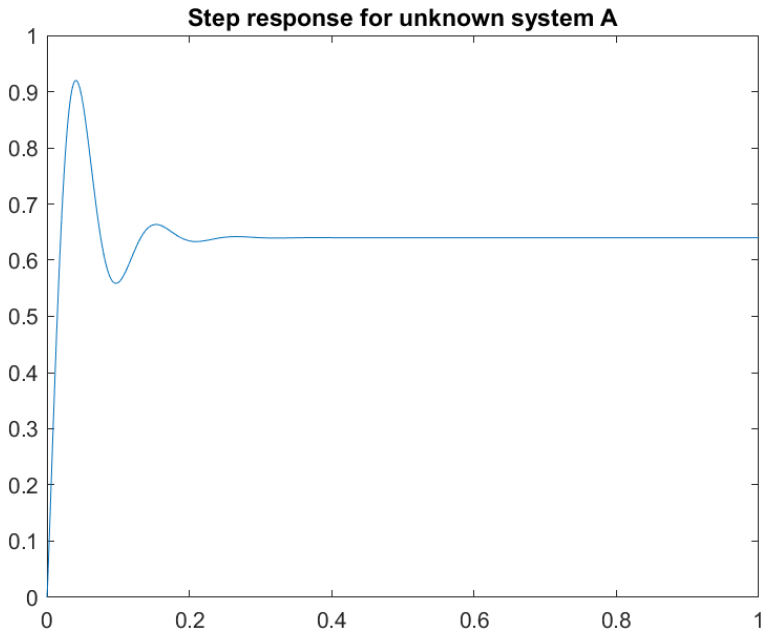
Take-Home Exam System and Control Theory 2023
Kobe Michiels (r0848543) – Transfer functie 9

Question 1 (System A)

Measure and visualize the step response and bode plot of your unknown system A.

To visualise the step response, I attach a step function to the input. I then plot the response of the Simulink model as a function of time.

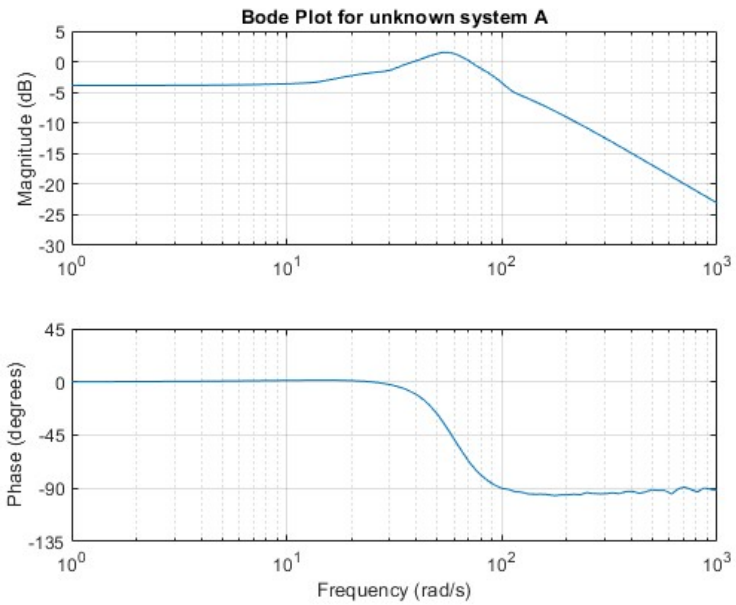
```
time_in = (0:1e-3:1)';
input = heaviside(time_in);
simOut = sim('transferfunction9A_harness.slx',time_in);
figure('Name', 'Generated step response for unknown system A');
plot(time_in,simOut.response)
title('Step response for unknown system A')
```



To generate the bode plot, I put sines of different frequencies at the input. For each frequency, I calculate the magnitude and phase shift. I calculate the magnitude as the ratio of the output to the input. I calculate the phase from the time difference between the input and the output. I calculate this time difference by finding a local maximum of the input, and then finding the next local maximum of the output, and then subtracting the time instants when both maxima occur. With this method, I have to take into account that in case there is no phase difference, a phase difference of 360° is obtained because the next local maximum is then 1 period further away. When I have calculated magnitude and phase for all frequencies, I can plot them against frequency, on a logarithmic scale. I save the data in a MATLAB Data file so that I don't have to run this piece of code repeatedly as it does take quite some time.

```
% Initialisation of time vector
startTime = 0;
endTime = 50;
samplingTime = 0.0001;
time_in = (startTime:samplingTime:endTime)';

% Initialisation of frequency (in rad/s)
startFreq = 1;
endFreq = 1000;
numPoints = 100;
freq = logspace(log10(startFreq), log10(endFreq), numPoints);

% Initialisation of magnitude and phase vector to store generated data:
magnitude = zeros(1, numPoints);
phase = zeros(1, numPoints);

% Generation of Bode Plot data
for k = 1 : numPoints
    input = sin(freq(k)*time_in);
    simOut = sim("transferfunction9A_harness",time_in);
    output = simOut.response;

    % Magnitude
    magnitude(k) = max(abs(output)) / max(abs(input));

   % Phase
    % Local maximum of input after possible transition phenomena disappear:
    for i = (0.75*(length(time_in)-1)):length(time_in)
        if input(i-1) < input(i) && input(i+1) < input(i)
            input_index = i;
            input_time = input_index*samplingTime;
            break;
        end
    end

    % First next local maximum of output:
    for i = input_index:length(time_in)
        if output(i-1) < output(i) && output(i+1) < output(i)
            output_index = i;
            output_time = output_index*samplingTime;
            break;
        end
    end

    % Calculate phase with time delay:
    phase(k) = -(output_time-input_time)*freq(k)*180/pi;
    if phase(k) < -350
        phase(k) = phase(k) + 360;
    end
end
% Save data to MATLAB Data file:
save("transferfunction9A_BodePlotWorkspace.mat")
```

```
%% Generate Bode plot with MATLAB Data file data:
load("transferfunction9A_BodePlotWorkspace.mat")
figure('Name', 'Generated bode plot for unknown system A');
subplot(2,1,1);
semilogx(freq, 20*log10(magnitude));
title('Bode Plot for unknown system A')
ylabel('Magnitude (dB)');
xlim([1 1000]);
ylim([-30 5]);
yticks(-30:5:5)
grid on;
subplot(2,1,2);
semilogx(freq, phase);
xlabel('Frequency (rad/s)');
ylabel('Phase (degrees)');
xlim([1 1000]);
ylim([-135 45]);
yticks(-135:45:45)
grid on;
```



Analyze and interpret the figures. Discuss the stability (in detail), and the order.

As indicated in the assignment, system A is stable. Before the system reaches its steady state, there are some oscillations in the step response. The system is thus underdamped ($0<\zeta<1$) and will have two complexly added poles with a real part smaller than 0. Furthermore, one zero will also be present. I infer this from the phase response that goes from 0° to -90°, whereas in the case where there were only 2 poles and no zeros, it would drop to -180°. Hence we are dealing with a second-order system.

Question 2 (System A)

Reverse engineer the transfer function from your Bode Plot and/or step response.

In the magnitude response, there is first an increase in amplitude visible, which then changes to a decrease of -20dB/dec. We already know that the system is under-damped and thus has two complexly added poles. In the phase response, there is a transition from 0° to -90°. As cited earlier, I deduce that I am dealing with a second-order system with one zero and two poles. Hence, the transfer function will have the following simplified form:

$$T(s) = \frac{As + B}{s^2 + Cs + D}$$

I start by determining the constant D. For a second-order system, it is equal to the square of the natural eigenfrequency ω_n^2. The phase drops from 0° to -90° and reaches a value of -45° in ω=60. From this I deduce that the double pole is at this frequency and therefore $\omega_n = 60$, and that there will be a zero close to it here. Consequently, D will be equal to 3600.

Next, I determine graphically what the 2% settling time is. For this, I obtain a value of 0.175s and use this to calculate the damping factor.

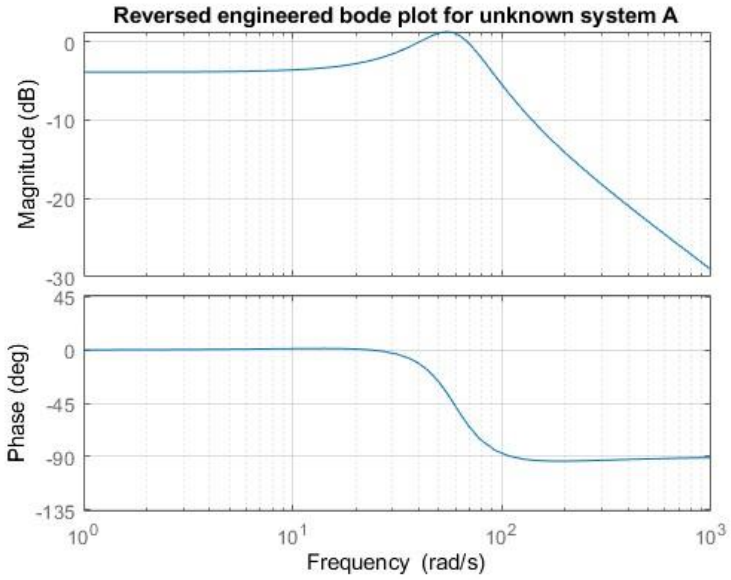
$$T_{s,2\%} \approx \frac{4}{\zeta \omega_n} \Leftrightarrow \zeta \approx \frac{4}{T_{s,2\%}\, \omega_n} = \frac{4}{0{,}175\,.\,60} = 0{,}38.$$

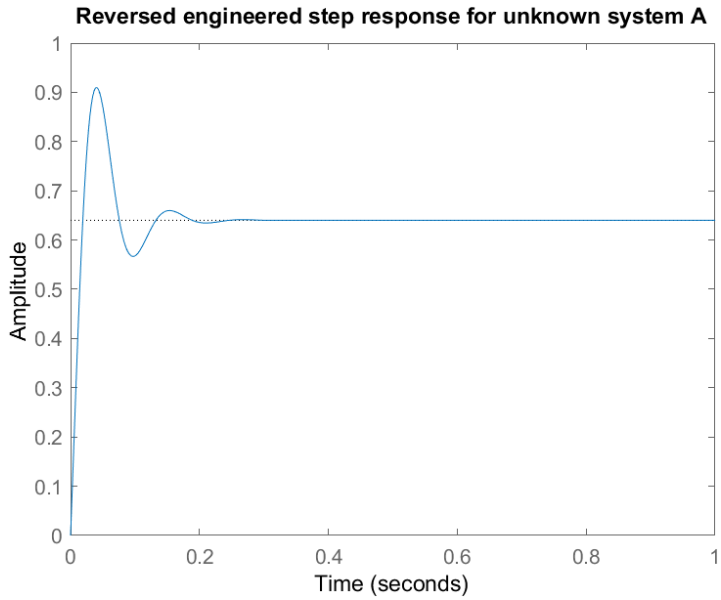
The denominator of a second-order transfer function has the following form: $s^2 + 2\zeta\omega_n s + {\omega_n}^2$. It follows that the term C from my simplified notation is equal to $C = 2\zeta\omega_n = 2.0{,}38.60 = 46$.

From the bode plot, I obtain a DC-gain of -3.87dB. As a result, I obtain $\frac{B}{D} = \frac{B}{3600} = 10^{-\frac{3{,}87}{20}} = 0{,}64 \Leftrightarrow B = 3600\,.0{,}64 = 2304$.

I determine the value for A from my zero. I know quite little about this, except that it will be close to the poles. Since there is no visible influence of the now on the phase response, other than a difference of 90°, I assume it will be behind the poles, and that the original rise in amplitude is partly caused by resonance. I try a value of 65 for the pole location and then obtain $A = \frac{2304}{65} = 35$. This gives me the following transfer function $T(s) = \frac{35s+2304}{s^2+46s+3600}$, with a bode plot and step response matching my generated plots. This makes me assume that the chosen position of my zero is quite accurate.

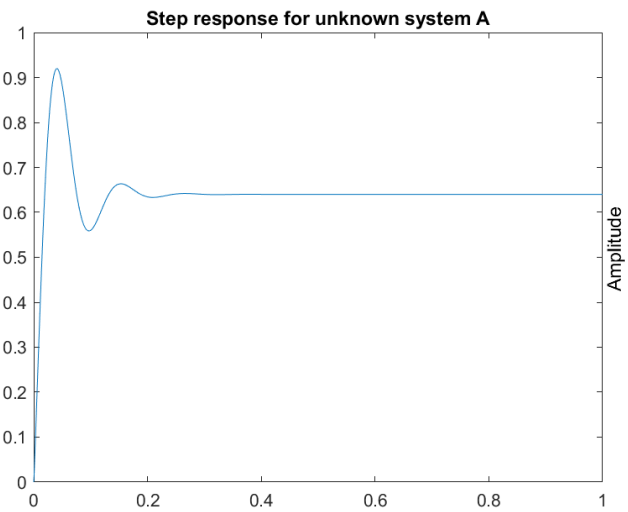
```
%Reverse engineered transfer function:
num_A = [35, 2304];
den_A = [1, 46, 3600];
TF_A = tf(num_A, den_A);

% Bode plot of reverse engineered transfer function:
figure('Name', 'Reversed engineered bode plot for unknown system A');
bode(TF_A)
title('Reversed engineered bode plot for unknown system A')
xlim([1 1000]);
grid on

% Step response of reverse engineered transfer function:
figure('Name', 'Reversed engineered step response for unknown system A');
step(TF_A)
title('Reversed engineered step response for unknown system A')
xlim([0 1])
```

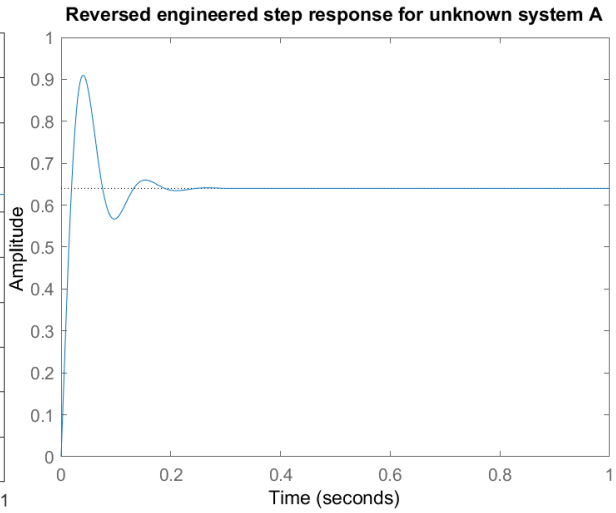


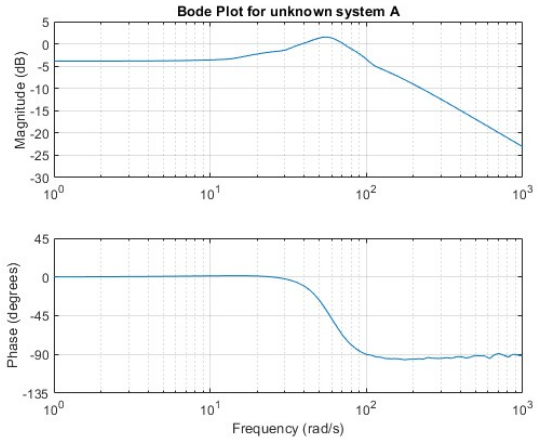


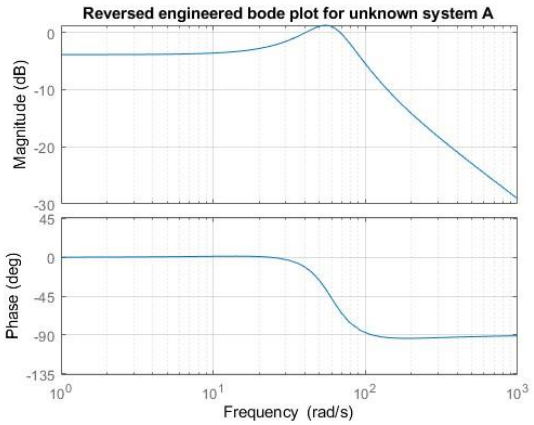
Analyse and discuss the mismatch between your transfer function and the protected transfer function.

The step response of my transfer function and that of the transfer function of the protected system match reasonably well. The only difference is that the peaks of the oscillations of the protected system are slightly higher than those of my transfer function. The explanation I find for this is that I used the 2% settling time to determine my damping ratio. The formula I used for this is this one for a second-order under-damped system. However, my transfer function also includes a zero, which will amplify the amplitude of the oscillations. This will also cause the settling time to deviate from the effective value. Furthermore, I derived this settling time graphically from my step response, which can also lead to a slight deviation. This deviation may explain the difference in the damping of the oscillations. Furthermore, I have also estimated the position of the zero. This too will affect my final bottom plot and step response. Comparing the bode plots, I see that they have a similar shape. The peak of the magnitude plot of the protected system is slightly higher than that of my transfer function. Also, the protected transfer feature will decrease less rapidly in magnitude. So, there will generally be greater amplification at those frequencies, which I also explain by the lower damping factor.







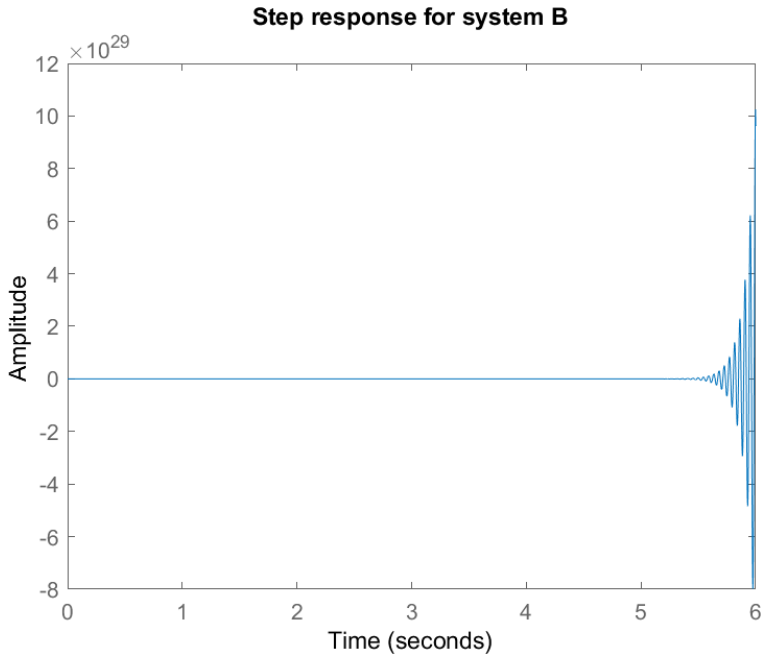


Question 3 (System B)

Discuss the (in)stability of the system. Make a root locus plot and find gain and phase margin

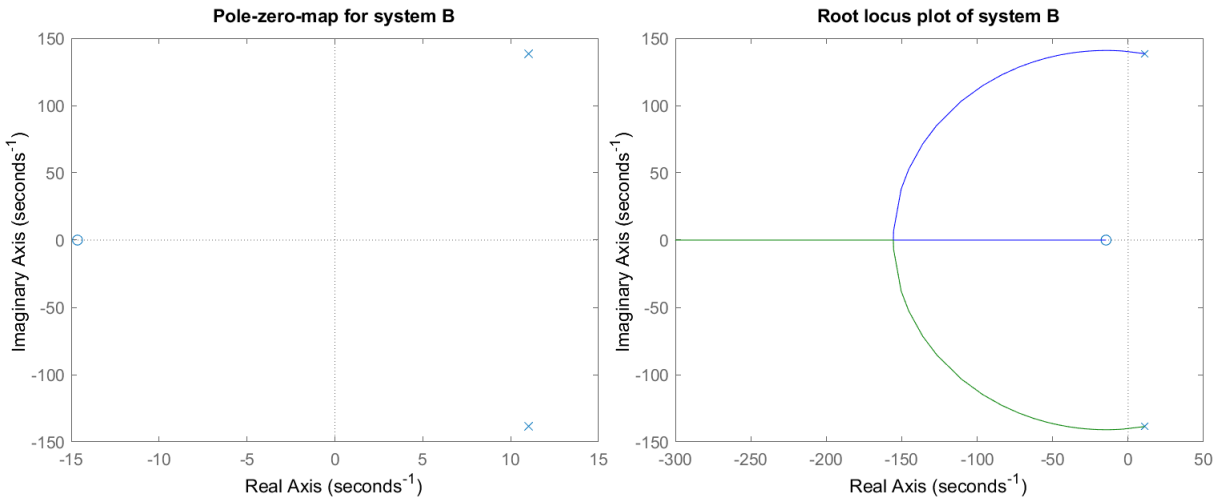
My system B has the following transfer function: $T(s) = \frac{3114s+45598}{s^2-22s+19321}$. The negative value at the s-factor in the denominator indicates a negative damping factor. This is already a first indication that the system will be unstable. When I draw the step response of this system, one can clearly see that this is the case.

```
num_B = [3114, 45598];
den_B = [1, -22, 19321];
TF_B= tf(num_B, den_B);

% Step response:
figure('Name', 'Step response for system B');
step(TF_B)
title('Step response for system B')
```



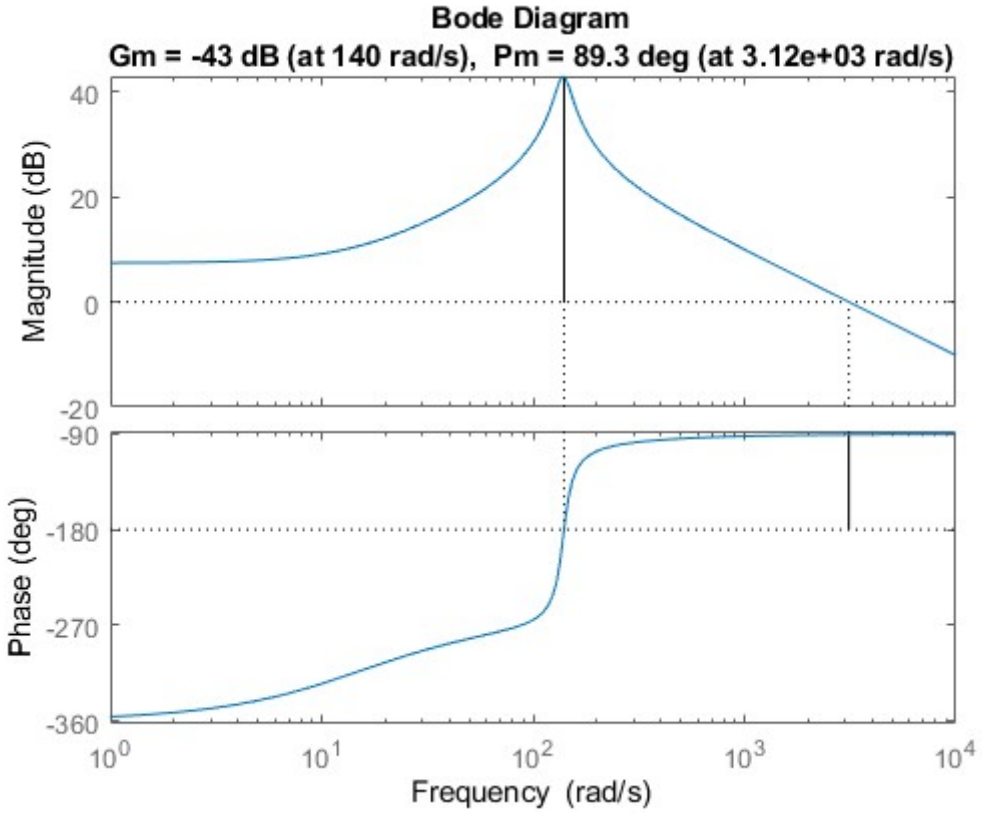
To further investigate the stability of the system, I first examine the poles of this system. I do this using a pole-zero-map and the root locus plot. Both plots show that the poles have a positive real part. This indicates that the system is unstable.

```
% Pole-zero-map:
figure('Name', 'Pole-zero-map for system B');
pzmap(TF_B)
title('Pole-zero-map for system B')
% Root locus plot:
figure('Name', 'Root locus plot of system B');
rlocus(TF_B)
title('Root locus plot of system B')
```

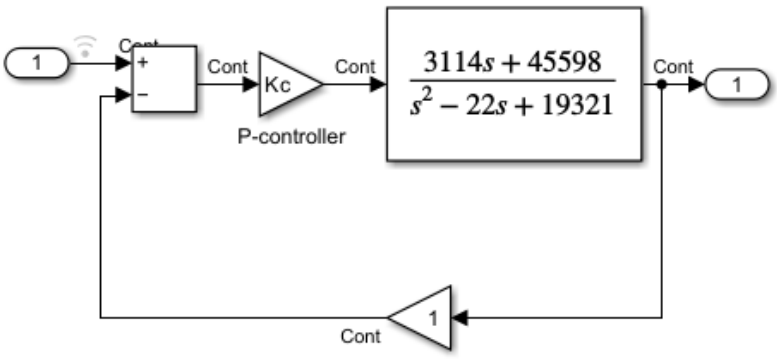


The system has a phase margin of -43dB. This negative phase margin confirms a fourth time that the system will be unstable.

```
%Gain and phase margin:
figure('Name', 'Gain and phase margin of system B');
margin(TF_B)
```



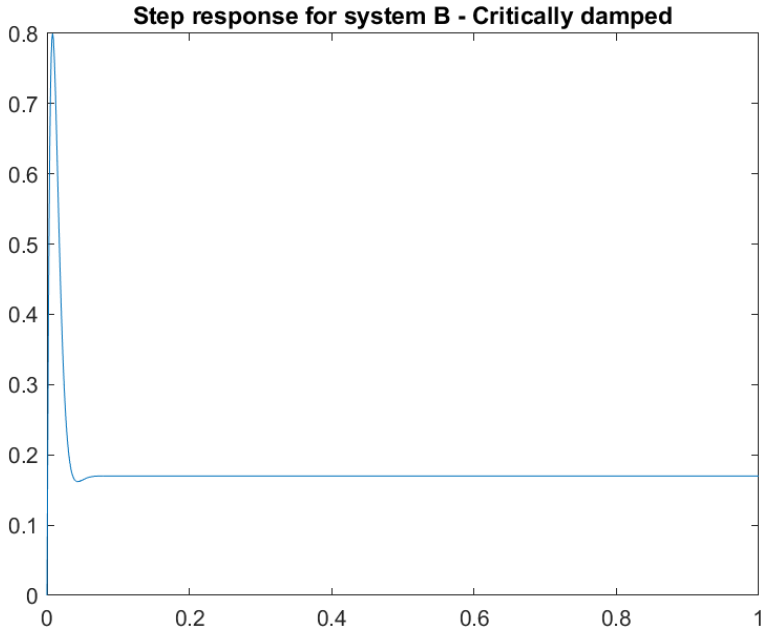
Build a P controller around the unprotected transfer function (B) to stabilize the system.



Tune the P controller so the system is critically damped.

```
Kc = 0.0867;
t = (0:1e-3:1)';
input = heaviside(t);
out = sim("transferfunction9B.slx",t);
figure('Name', 'Step response for system B - Critically damped');
plot(t,out.y)
title('Step response for system B - Critically damped')
```

The transfer function of the closed loop is equal to $T_{\mathrm{cl},r\to z}(s) = \frac{3114\,K_C\,s+45598K_C}{s^2+(3114K_C-22)\,s+(45598K_C+19321)}$. When dealing with critical damping, we know that the damping factor ζ is equal to 1. From the formula for a second-order transfer function, it follows that $2\omega_n = 3114K_C - 22$ and $\omega^2{}_n = 45598K_C + 19321$. If we transform both functions to ${\omega_n}^2$ and set them equal, we can calculate the K_C value leading to critical damping. This will be equal to 0.0867 and gives the step response below. The same value can be obtained by setting the discriminant of the polynomial in the denominator equal to 0.

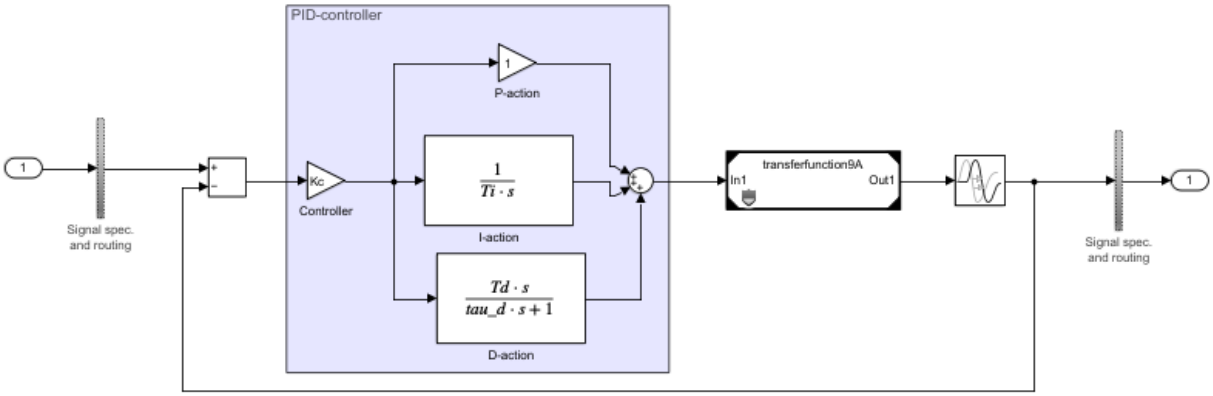


This step response moves quickly to the equilibrium position, but there is a large overshoot. To solve this overshoot and the steady-state error, a PID controller is needed.

Question 4 (System A)

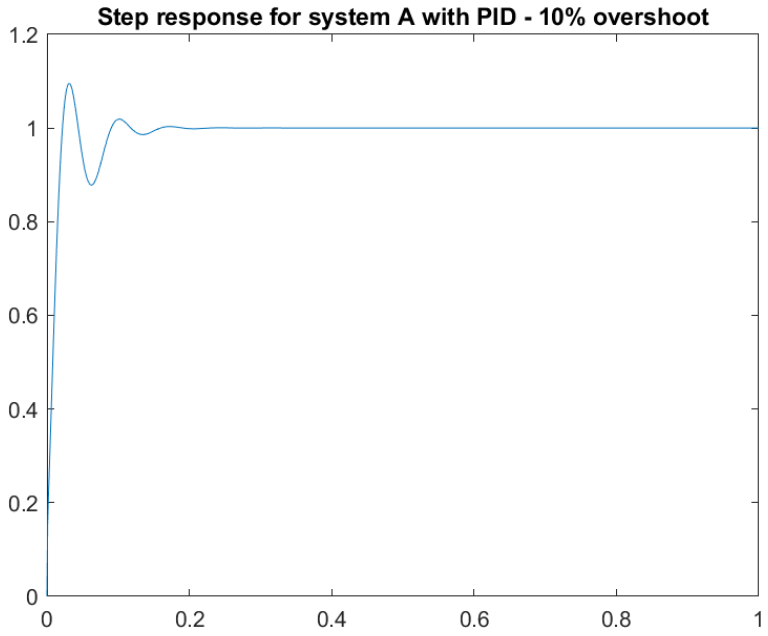
Build a control loop around the protected transfer function.

For this, I create a new Simulink model, so that the step response and bode plot derived from the previous tasks cannot be affected by the parameters from the control loop. I do already add the time delay from question 5 but set it equal to 0.



Tune the PI(D) controller to have less than 10% overshoot. The D-action may or may not be necessary.

```
time_in = (0:1e-3:1)';
input = heaviside(time_in);
Kc = 1.5;
Ti = 0.01;
Td = 0.003;
tau_d = Td/(10*Kc);
time_delay = 1e-12;
simOut = sim('transferfunction9A_harness_with_control_loop.slx',time_in);
figure('Name', 'Step response for system A with PID - 10% overshoot');
plot(time_in,simOut.response)
title('Step response for system A with PID - 10% overshoot')
```



Discuss your tuning method, parameters and results. Discuss whether you implemented a D-action, and why (not)?

During the lecture, we saw the Ziegler-Nichols tuning method. This uses the K-value at edge stability. Since the system is guaranteed to be stable and this method also often leads to large overshoots, I chose not to use this method. I calculated the parameters from my PID controller via trial-and-error. I first set the value for T_I as 10^9 and that of T_D as 10^{-9}, so that the I-action and D-action have no influence. I then increased the value of K_C in small increments, as we did in practice session 11, but I immediately noticed a large overshoot. As a result, I introduced the I-action and started systematically decreasing the value of T_I until, at the combination K_C = 1.5 and T_I = 0.01, I no longer had steady-state and only a slight overshoot. To reduce the oscillations and consequently the overshoot, I also introduced a D-action. Again, via trial-and-error, I arrived at the above step response for a value for T_D of 0.003, which has an overshoot of less than 10%.

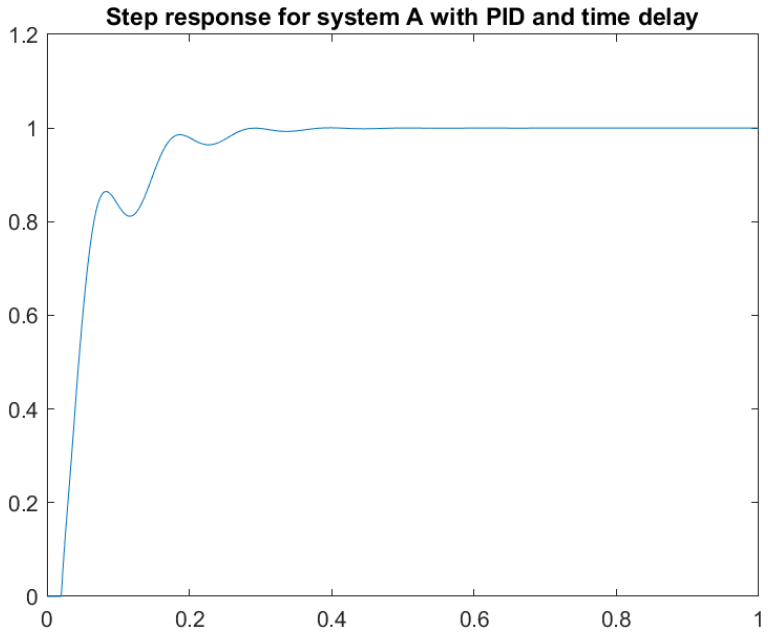
Question 5 (System A)

In Simulink, add a time delay in series with system A. If the delay becomes too large, this destabilizes the system (discuss!).

A time delay t_d in the time domain, creates an additional factor $e^{-t_d s}$ in the Laplace domain. Thus, the time delay will cause the phase to decrease exponentially and infinitely, and this effect will increase with increasing frequency. The maximum time delay that can be added is a time delay calculated from the phase margin with its associated frequency ($t_d = \frac{PM.\pi}{180°.\omega_{PM}}$). When this time delay is present, the system is edge stable. A larger time delay will lead to instability.

Tune the PI(D) controller again so the output is desirable even with this delay.

```
time_in = (0:1e-3:1)';
input = heaviside(time_in);
Kc = 0.25;
Ti = 0.01;
Td = 0.01;
tau_d = Td/(10*Kc);
time_delay = 0.02;
simOut = sim('transferfunction9A_harness_with_control_loop.slx',time_in);
figure('Name', 'Step response for system A with PID and time delay');
plot(time_in,simOut.response)
title('Step response for system A with PID and time delay')
```



Calculate or find the maximum delay for which the system can be made stable.

The maximum time lag for which the system can be stable is the time lag for which the phase difference equals the phase margin. In this case, this is a time lag of 0.0299s.

```
[Gm,Pm,Wcg,Wcp] = margin(tf([35, 2304], [1, 46, 3600]));
max_time_delay = Pm*pi/(180*Wcp)
```