COMP0128 Robotic Control Theory and Systems Coursework 2

Hou Nam Chiang (15055142)
hou.chiang.20@ucl.ac.uk

4th January, 2021

Question 1

(a) $u(t) = sin\theta(t)$ is a low level input / direct input because we can control the angle θ so it can model into an LTI system and so convert into a transfer function more easily.

(b) From the diagram, we have:

$$\begin{aligned} F_{net} &= mgsin\theta(t) - b\dot{d} \\ m\ddot{d} &= mgu(t) - b\dot{d} \\ m\mathcal{L}\{\ddot{d}\} &= mg\mathcal{L}\{u(t)\} - b\mathcal{L}\{\dot{d}\} \\ ms^2D &= mgU - bsD \\ \frac{D}{U} &= \frac{mg}{ms^2 + bs} \end{aligned}$$

(c) With a continuous P controller with gain K, the system becomes a closed loop, so:

$$\begin{aligned} \frac{KG(s)}{1 + KG(s)} &= \frac{K\frac{mg}{ms^2+bs}}{1 + K\frac{mg}{ms^2+bs}} \\ &= \frac{Kmg}{ms^2 + bs + Kmg} \end{aligned}$$

For the system to be both stable and without oscillations for a step response, the values of s must lie in the part of the root locus where the imaginary part of s is 0 and is in the left half plane of the complex plane. With reference to section 12.5 The Root Locus Method of the book Feedback Systems, we can first find the open loop poles by using the denominator of $G(s)$:

$$\begin{aligned} ms^2 + bs &= 0 \\ s &= \begin{cases} 0 \\ -\frac{b}{m} \end{cases} \end{aligned}$$

Since there are only two solutions above, there are only two branches for the root locus. Then we can further find the asymtotes of these two branches by locating its centroid and and angles of these asymtotes. Since there are $n = 2$ branches and $m = 0$ zeros in

$G(s)$, so $n - m = 2$.
Centroid:

$$\frac{1}{2}\left(-\frac{b}{m} + 0\right) = -\frac{b}{2m}$$

Angles, with $l = 0, 1$:

$$\frac{1}{2}\left(180^\circ + 360^\circ l\right) = \begin{cases} 90^\circ \\ 270^\circ \end{cases}$$

which means the root locus should look like a cross and therefore the centroid should be the breaking point, where the values beyond this break point will go into the imaginary plane, which means there will be oscillation. So we can find the value of K at $-\frac{b}{2m}$. Since the root locus is actually defined by $1 + KG(s) = 0$:

$$\begin{aligned} 1 + KG(s) &= 0 \\ K &= -\frac{1}{G(s)} \\ &= -\frac{ms^2 + bs}{mg} \\ &= -\frac{m\left(-\frac{b}{2m}\right)^2 - \frac{b^2}{2m}}{mg} \\ &= \frac{b^2}{4m^2 g} \\ K &= 5.74 \times 10^{-4} \end{aligned}$$

and for $s = 0, -\frac{b}{m}$:

$$\begin{aligned} K &= -\frac{1}{G(s)} \\ &= -\frac{ms^2 + bs}{mg} \\ &= 0 \end{aligned}$$

So the range of K is $0 \leq K \leq 5.74 \times 10^{-4}$.

(d) The root locus and the closed-loop step response plots are as shown and the numbers agree with (c):

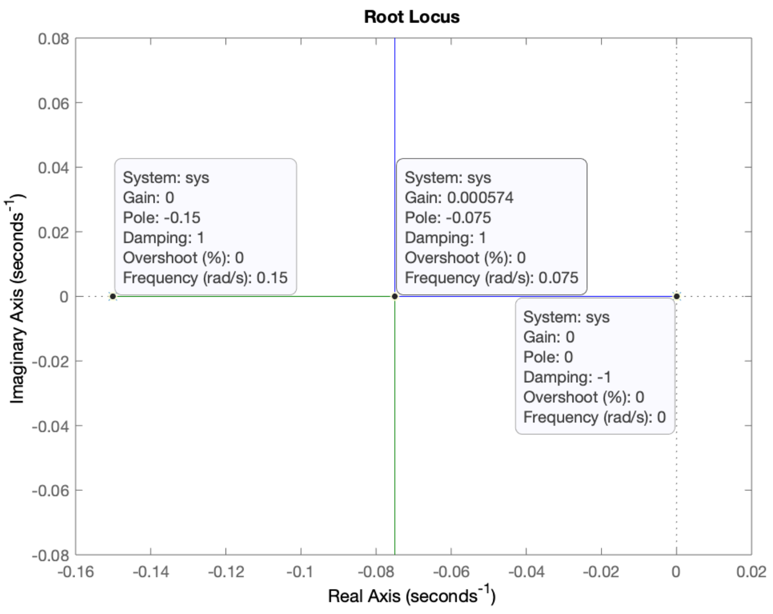


Figure 1: Root Locus

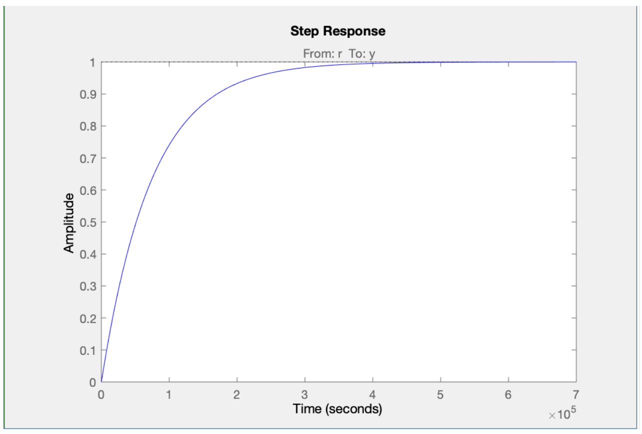


Figure 2: Step Response plot when gain K = 0

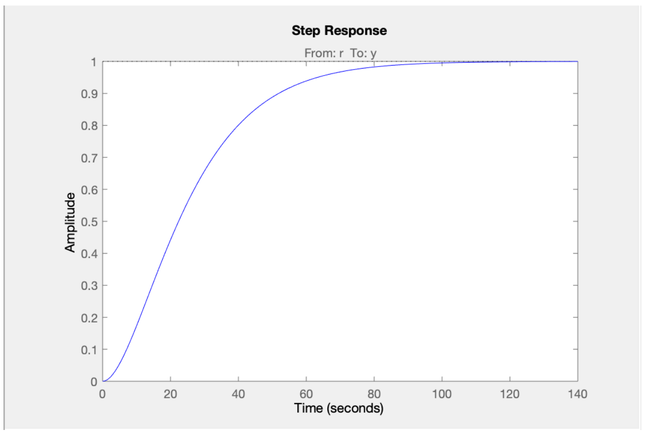


Figure 3: Step Response plot when gain K = 0.000574

(e) When $K = 0.05$: Its crossover frequency and phase margin is $0.692 rad/s$ and $12.2°$ respectively:

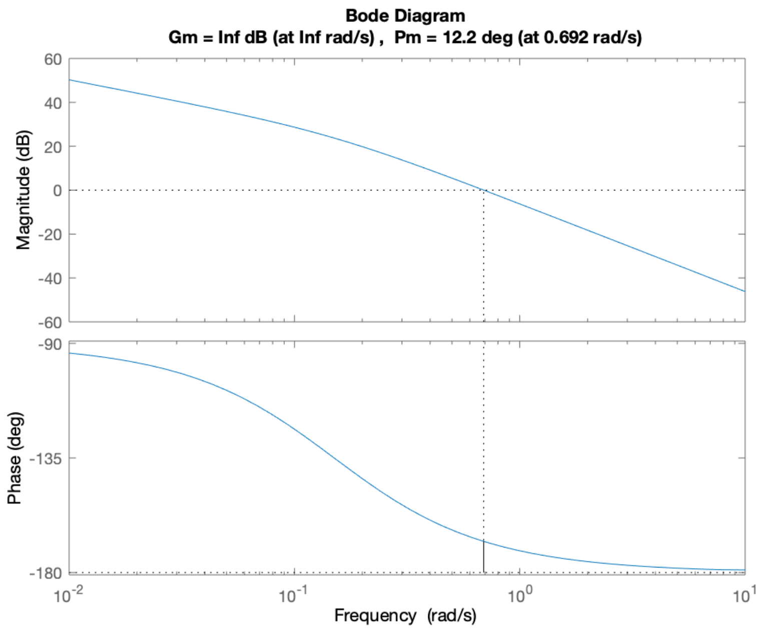


Figure 4: Bode plot showing crossover frequency and phase margin

From its closed-loop step response, the maximum overshoot is 1.71 at 4.488s and the approximate time it takes to converge to 1 is 95s:

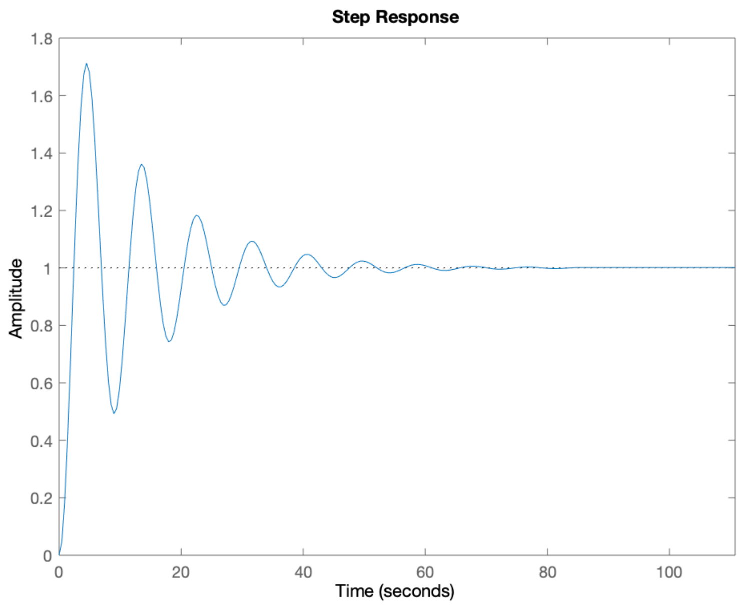


Figure 5: Closed-loop step response when K = 0.05

To reduce both the maximum overshoot and the convergence time of the closed-loop step response, a lead compensator is designed, knowing that the lead compensator is in the form:

$$K\frac{s+z}{s+p}$$

with $|z| < |p|$, and the average of z, p is the cross over frequency w_c, we have:

$$\frac{(w_c + a) + (w_c - a)}{2} = w_c$$

Let $a = 0.3$, we have:

$$z = 0.692 - 0.3 = 0.392$$
$$p = 0.692 + 0.3 = 0.992$$

Then, the lead gain of the compensator is the magnitude of the fraction:

$$\left|\frac{0.392 + 0.692i}{0.992 + 0.692i}\right| = 0.6575$$

Then, to maintain the position of the cross over frequency, we normalize the proportional gain (0.05) by the lead gain:

$$\frac{0.05}{0.6575} = 0.076$$

All the above gives the final lead compensator as shown:

$$0.076\frac{s+0.392}{s+0.992}$$

With the lead compensator, the root locus, step response and the bode plot are as shown, we can see that the maximum overshoot is reduced to around 1.4 and the convergence time is definitely less than 95s.

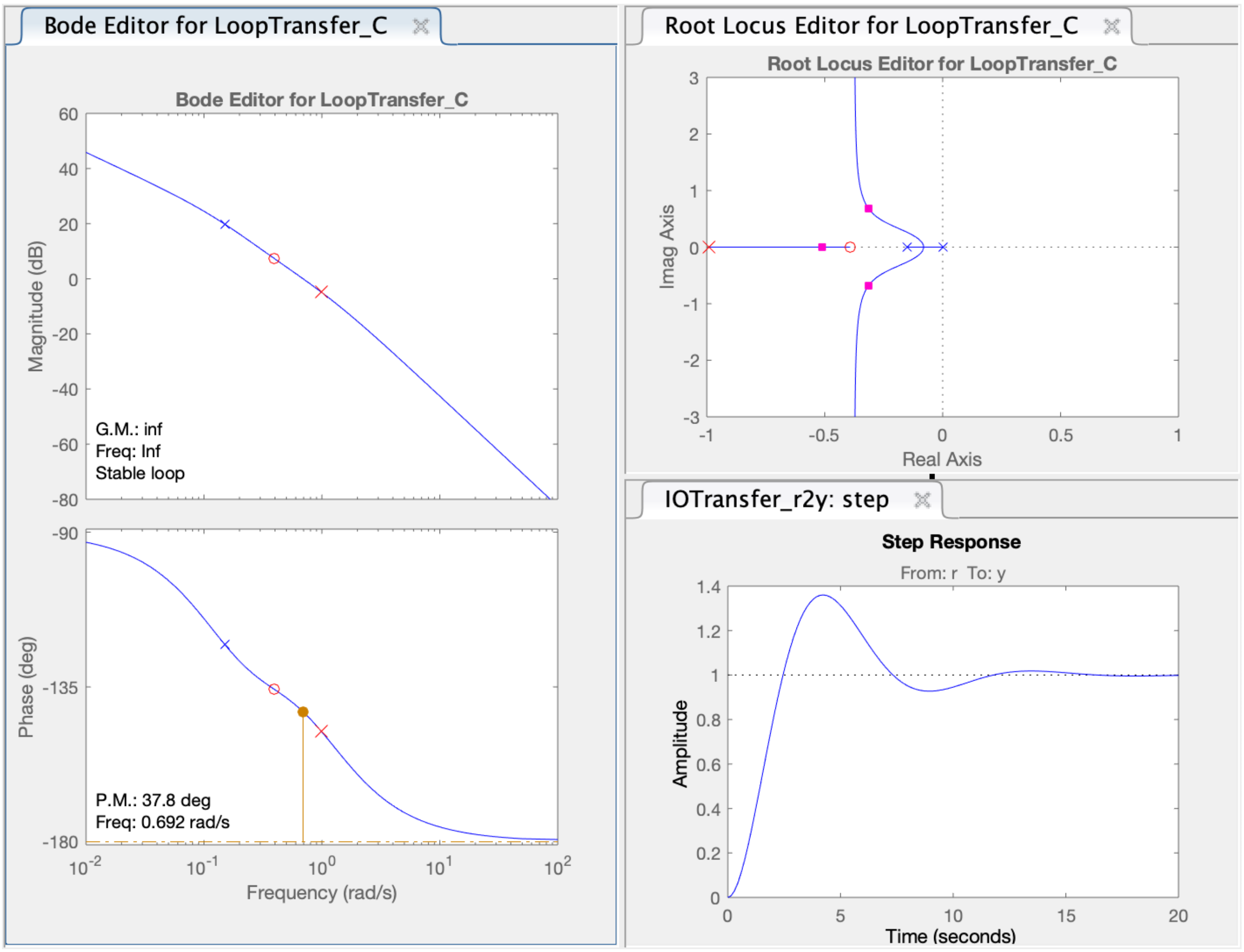


Figure 6: Lead compensated system

Question 2

(a) For a small angle approximation ($\theta \approx 0 \longrightarrow sin\theta \approx \theta$):

$$\begin{aligned} u(t) &= sin\theta(t) \\ u(t) &= \theta(t) \end{aligned}$$

the continuous open-loop transfer function of this new system is:

$$\begin{aligned} m\ddot{d} &= mg\theta(t) - b\dot{d} \\ m\mathcal{L}\{\ddot{d}\} &= mg\mathcal{L}\{\theta(t)\} - b\mathcal{L}\{\dot{d}\} \\ ms^2D &= mg\Theta - bsD \\ \frac{D}{\Theta} &= \frac{mg}{ms^2 + bs} \end{aligned}$$

Then for a control input $u(t)$ to the motor, the platform angle changes continuously according to:

$$\begin{aligned} \dot{\theta}(t) &= K_m(u(t) - \theta(t)) \\ \mathcal{L}\{\dot{\theta}(t)\} &= K_m(\mathcal{L}\{u(t)\} - \mathcal{L}\{\theta(t)\}) \\ s\dot{\Theta} &= K_m(U - \Theta) \\ \frac{\Theta}{U} &= \frac{K_m}{s + K_m} \end{aligned}$$

Combining the transfer functions:

$$\begin{aligned} \frac{D}{\Theta}\frac{\Theta}{U} &= \frac{mg}{ms^2 + bs}\frac{K_m}{s + K_m} \\ \frac{D}{U} &= \frac{K_m mg}{s(s + K_m)(ms + b)} \end{aligned}$$

(b) Simulink is used to test the transfer function with P, PI and PD controller, the same gain ($K = 0.5$) is used. Results show that PD controller is the only one which can stabilize the sphere. We can see from the following locus that other than the PD controller, the other two always have locus in the right half plane. So P and PI controllers cannot stablize the sphere for all K.

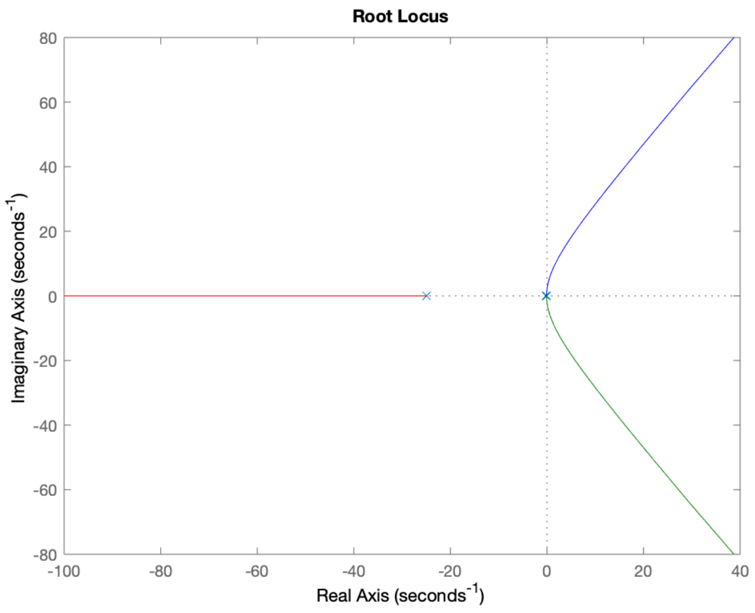


Figure 7: Step Responses using P controller

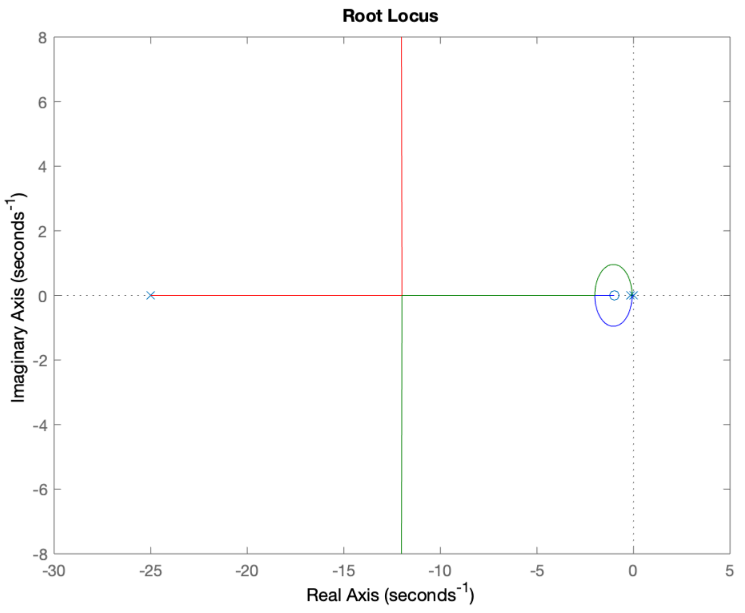


Figure 8: Step Responses using PD controller

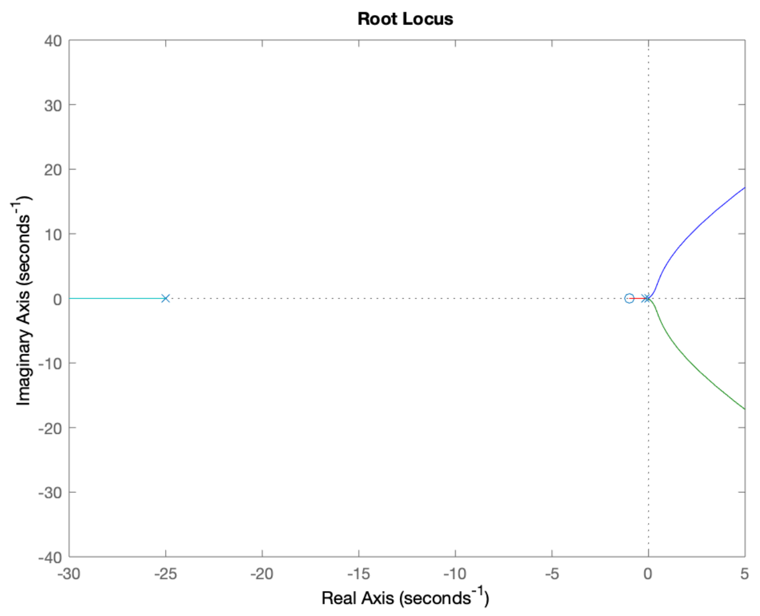


Figure 9: Step Responses using PI controller

(c) The gain for the PD conpensated system that can produce a fast response with a phase margin around 76° to 78° is 0.00064008 (77.1°).

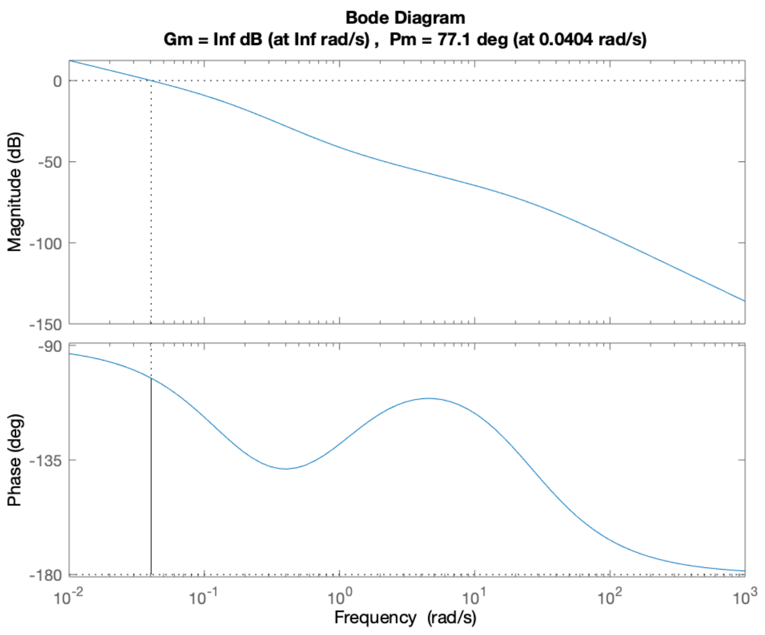


Figure 10: Bode plot of PD compensated system with K = 0.00064008

The discretized PD controller is therefore:

$$\frac{0.01344z - 0.01216}{z + 1}$$

But this gives a very slow response (need around 80 seconds). Therefore a new gain is found. The gain for the PD compensated system that can produce a fast response with a phase margin around 67.8° is 0.70172.

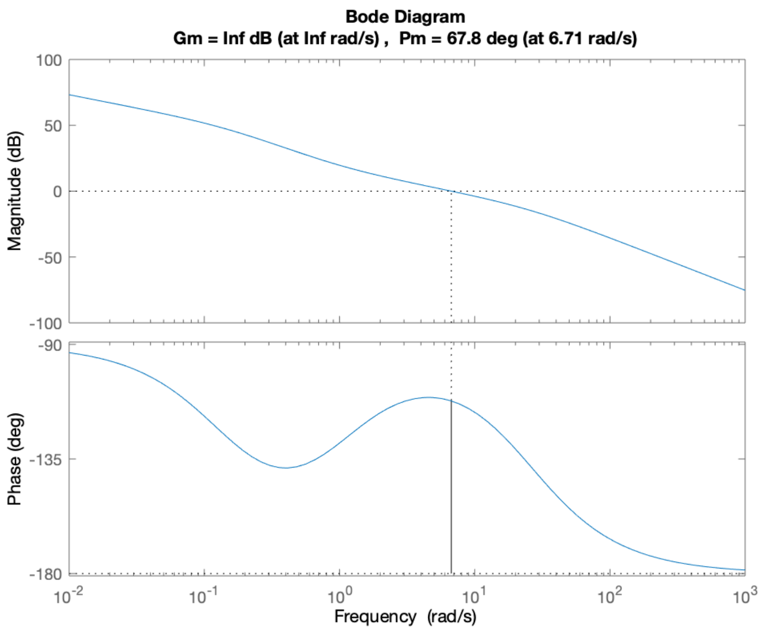


Figure 11: Bode plot of PD compensated system with K = 0.70172

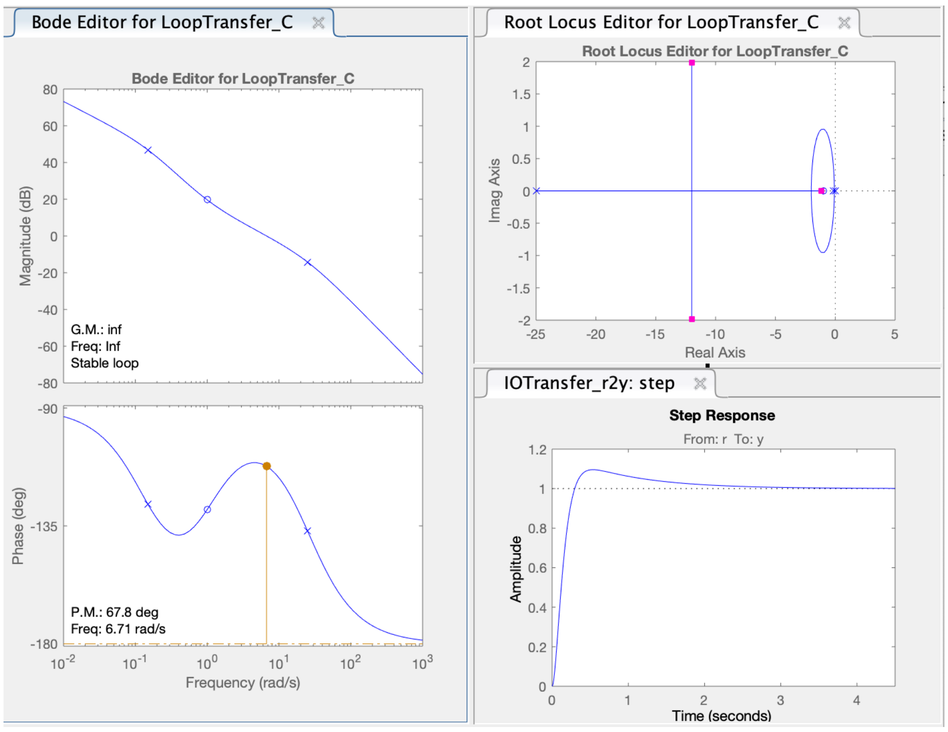


Figure 12: Step response of PD compensated system with K = 0.70172

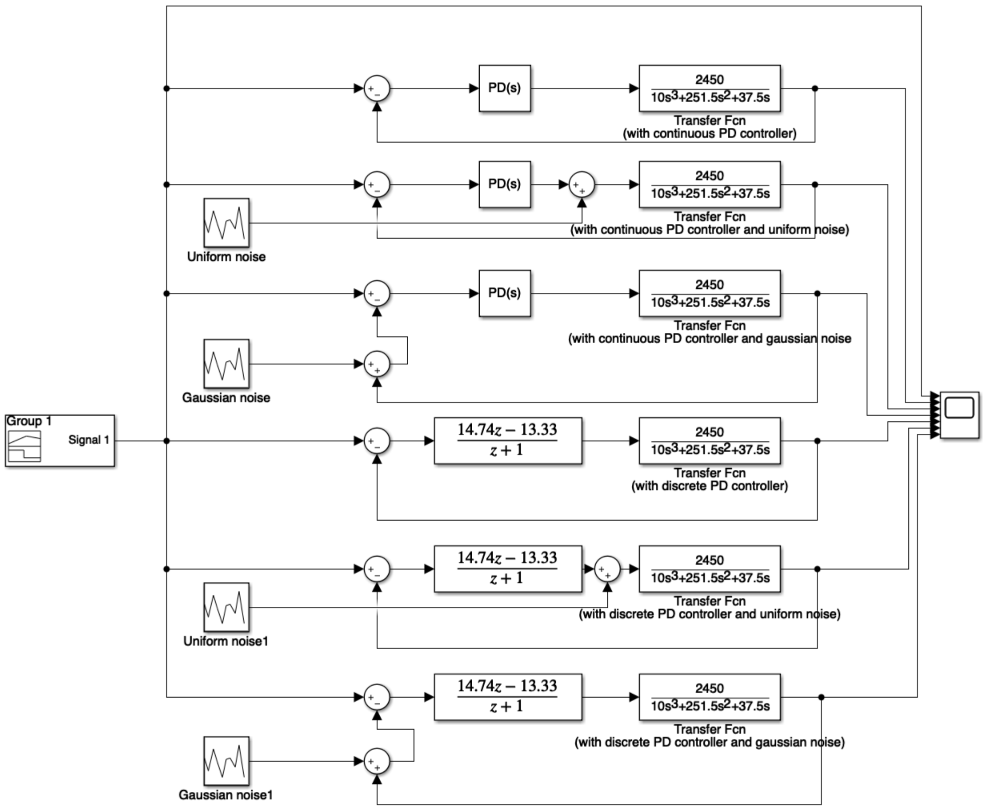
The discretized PD controller is therefore:

$$\frac{14.74z - 13.33}{z + 1}$$

The block diagram and responses of following $r_d(t)$:

$$r_d(t) = \begin{cases} 0.3 & 0 \leq t < 4 \\ 1.1 - 0.2t & 4 \leq t < 8 \\ -0.5 & 8 \leq t < \infty \end{cases}$$

using the continuous and discretized PD controller with and without the uniform and gaussian noises are as follow:



PD(s)
2450 / 10s3+251.5s2+37.5s
Transfer Fcn (with continuous PD controller)
Uniform noise
Transfer Fcn (with continuous PD controller and uniform noise)
Gaussian noise
Transfer Fcn (with continuous PD controller and gaussian noise
Group 1
Signal 1
14.74z − 13.33 / z + 1
Transfer Fcn (with discrete PD controller)
Uniform noise1
Transfer Fcn (with discrete PD controller and uniform noise)
Gaussian noise1
Transfer Fcn (with discrete PD controller and gaussian noise)

Figure 13: Responses of the system

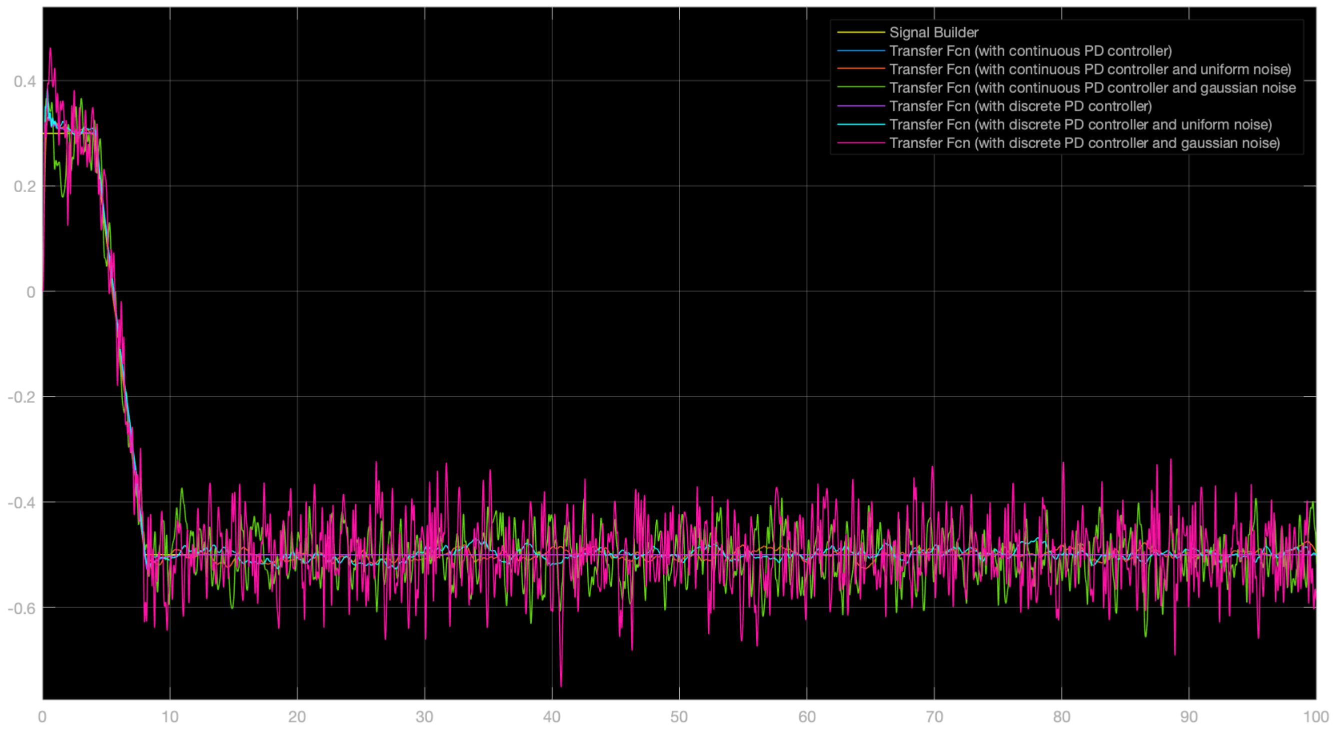


Figure 14: Responses of the system

We can see that the responses have similar behaviour.

(d) Discrete time domain function of the discrete PD controller:

$$\begin{aligned}
\frac{U(z)}{E(z)} &= \frac{14.74z - 13.33}{z+1} \\
(z+1)U(z) &= (14.74z - 13.33)E(z) \\
(1+z^{-1})U(z) &= (14.74 - 13.33z^{-1})E(z) \\
u[k] + u[k-1] &= 14.74e[k] - 13.33e[k-1] \\
u[k] &= 14.74e[k] - 13.33e[k-1] - u[k-1]
\end{aligned}$$

(e) In this part we lifted the assumption of $sin\theta \approx \theta$, and together with the controller developed in part 2d) we now try to create a simulation that follows the trajectory r_d with uniform input disturbance before the nonlinear function and and gaussian sensor noise in the feedback loop:

$$r_d(t) = \begin{cases} 0.15 & 0 \leq t < 8 \\ 0.95 - 0.1t & 8 \leq t < 16 \\ -0.65 & 16 \leq t < \infty \end{cases}$$

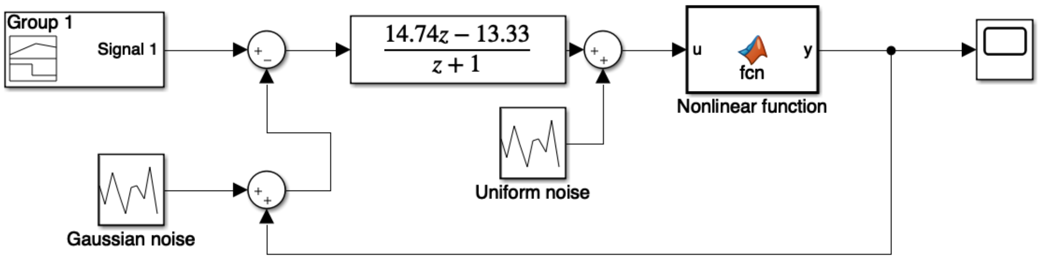


Figure 15: Block diagram of the simulation

where Signal 1 in Figure 14 is $r_d(t)$.

The main script which implemented this control system is in main.mlx. This script is mainly divided into 3 parts:

1. Initiialize parameters
2. Control system loop
3. Plot $r_d(t)$ and the response of the system

The code for this is as shown, $\theta, d, \dot{d}$ are updated using the sampling time Δt, and since the discretized PD controller requires $e[k-1]$ and $u[k-1]$, we also need to initialize them as 0 and update them every loop:

```
% Initialize:
u_k_1   = 0;
e_k_1   = 0;
d_prev  = 0;
delta_t = 0.01;
theta   = 0;
d_k     = 0;
```

```
d_dot    = 0;
d_mem    = [];
r_mem    = [];

% Control system:
for t = 1:delta_t:50
    r_d           = trajectory(t);
    e_k           = r_d - d_prev;
    u_k           = discrete_PD_controller(e_k, e_k_1, u_k_1) +
       input_disturbance();
    [d_ddot, theta_dot] = nonlinear_fn(u_k, d_dot, theta);

    d_dot  = d_dot + d_ddot*delta_t;
    d_k    = d_k + d_dot*delta_t;
    theta  = theta + theta_dot*delta_t;
    theta = min(max(theta, -pi/2),pi/2);

    % Update:
    e_k_1  = e_k;
    u_k_1  = u_k;
    d_prev = d_k + sensor_noise();

    % Save:
    r_mem  = [r_mem r_d];
    d_mem  = [d_mem d_k];
end

% Plot graphs:
plot(1:delta_t:50, r_mem)
hold on
plot(1:delta_t:50, d_mem)
legend('r_d(t)', 'Response')
xlabel('time(s)')
```

The discretized PD controller is implemented directly in the function discrete_PD_controller() as shown:

```
function u_k = discrete_PD_controller(e_k, e_k_1, u_k_1)
    u_k = 14.74*e_k - 13.33*e_k_1 - u_k_1;
end
```

After the PD controller, the uniform noise bounded between -0.05 and 0.05 rad is added to the signal before going through the nonlinear function:

```
function input_dist = input_disturbance()
    lower_bound = -0.05;
    upper_bound = 0.05;
    input_dist = lower_bound + (upper_bound - lower_bound)*rand(1);
end
```

Since we are now are not assuming $sin\theta \approx \theta$ anymore, we use the following equations directly:

$$m\ddot{d} = mgsin\theta(t) - b\dot{d}$$
$$\dot{\theta}(t) = K_m(u(t) - \theta(t))$$

This is implemented in the function nonlinear_fn() as follows:

```matlab
function [d_ddot, theta_dot] = nonlinear_fn(u, d_dot, theta)
    % Constants:
    m   = 10;
    g   = 9.8;
    b   = 1.5;
    K_m = 25;

    theta_dot = K_m*(u - theta);
    d_ddot    = g*sin(theta) - (b/m)*d_dot;
end
```

The non linear function gives the output d and it is feeded back to the system with the gaussian sensor noise with variance 0.005:

```matlab
function sensor_noi = sensor_noise()
    sensor_noi = normrnd(0, sqrt(0.005));
end
```

With the above functions, the following result is generated by running main.mlx. Firstly, the response without any distrubance and noise is as follows:

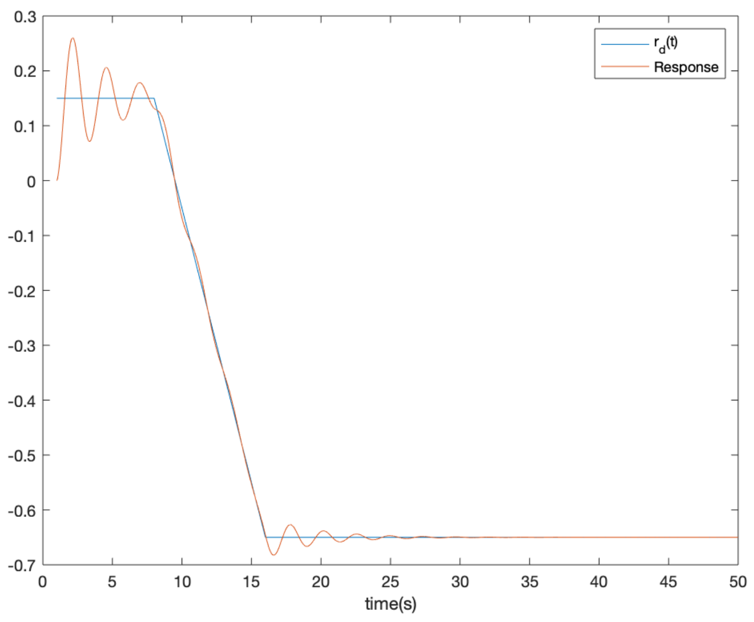


Figure 16: $r_d(t)$ and the response of the simulation without disturbance and noise

Then below is the response with both input disturbance and sensor noise:

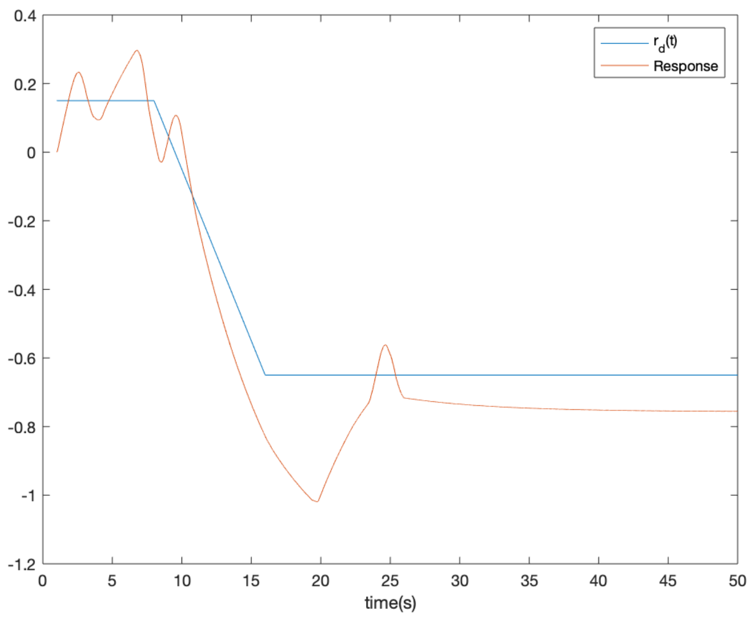


Figure 17: $r_d(t)$ and the response of the simulation with disturbance and noise

We can see that the sensor noise messed up the response to some degree.