THE UNIVERSITY OF SYDNEY

AMME3500

Assignment 2

S.J. Tan

311225659

8 May 2016

List of Figures

Contents

1 Robot arm

- **Analyse a control system for a single link of a robot arm.**
- **Assume the robot arm is rigid and has a moment of inertia $J = 5$ kgm^2.**
- **Friction in the mechanism has been measured as $c = 0.7$ Nms.**
- **The arm operates in the horizontal plane only. Neglect gravity.**

The equation of motion of system is:

$$J\ddot{\theta} + c\dot{\theta} = T \tag{1}$$

where T is the total torque applied to the arm.

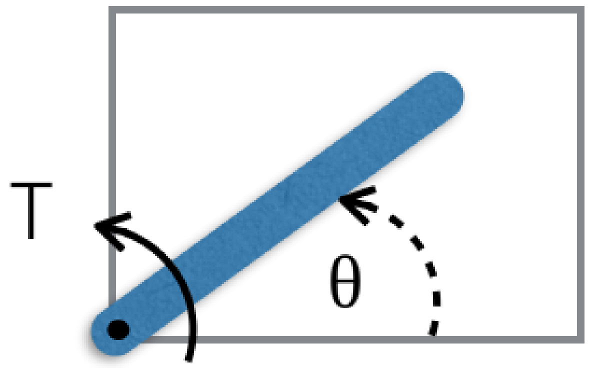


Figure 1: The robot arm operates in the horizontal plane with 1 DOF (top view).

(a) $\theta(s)/T(s)$

Take the Laplace transform of Eq. 1:

$$Js^2\theta(s) + cs\theta(s) = T(s) \tag{2}$$

Rearranging this gives the transfer function between the applied torque T and the arm angle θ:

$$\frac{\theta(s)}{T(s)} = \frac{1/J}{s^2 + cs/J} \tag{3}$$

(b) Proportional control & $\theta(s)/\theta_r(s)$

θ tracks a reference command θ_r:

$$u = K(\theta_r - \theta) \tag{4}$$

where K is the feedback gain. There is a disturbance torque w.

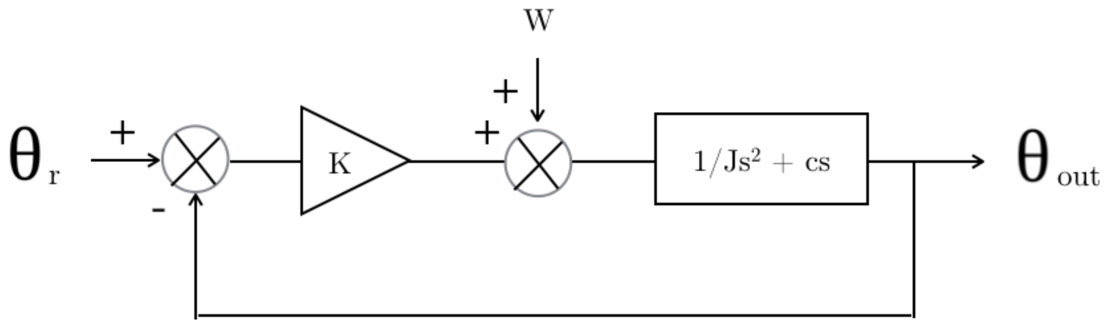


Figure 2: Block diagram including feedback gain K.

Fig. 2 is the block diagram of the resulting feedback system. The system is therefore represented by:

$$K(\theta_r(s) - \theta(s)) + W(s) = \theta(s)(Js^2 + cs) \tag{5}$$

Rearranging this expression yields the following:

$$\frac{\theta(s)}{\theta_r(s)} = \frac{K/J}{s^2 + cs/J + K/J} + \frac{W(s)/J}{(s^2 + cs/J + K/J)\theta_r(s)} \tag{6}$$

$$\frac{\theta(s)}{W(s)} = \frac{K\theta_r(s)/J}{(s^2 + cs/J + K/J)W(s)} + \frac{1/J}{s^2 + cs/J + K/J} \tag{7}$$

When there is zero disturbance torque ($w = 0$), $W(s) = 0$. Then:

$$\frac{\theta(s)}{\theta_r(s)} = \frac{K/J}{s^2 + cs/J + K/J} \tag{8}$$

(c) $\theta(s)/W(s)$

When $\theta_r = 0$, $\theta(s)/W(s)$ reduces to:

$$\frac{\theta(s)}{W(s)} = \frac{1/J}{s^2 + cs/J + K/J} \tag{9}$$

(d) % OS & T_s with $w = 0$

Where ζ is the damping ratio and ω_n is the natural frequency, the general 2nd-order form is given as:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{10}$$

Comparing this to the system equation Eq. 8 with this Eq. 10 yields:

$$\omega_n = \sqrt{K/J} \tag{11}$$

$$\zeta = \frac{c}{2J\sqrt{K/J}} \tag{12}$$

The percentage overshoot is expressed in [1] as:

$$\%\ \text{OS} = e^{-\zeta\pi/\sqrt{1-\zeta^2}} \tag{13}$$

Substituting ζ, the percentage overshoot can be described in terms of K:

$$\%\ \text{OS} = \exp\frac{-\pi c}{2J\sqrt{K/J}\sqrt{1-(c/(2J\sqrt{K/J})^2}} \tag{14}$$

The maximum K value for a designed overshoot is more easily resolved using:

$$\zeta = \frac{-\ln(\%\ \text{OS})}{\sqrt{\pi^2 + \ln^2(\%\ \text{OS})}} \tag{15}$$

With 5% overshoot, solving Eq. 15 gives $\zeta = 0.69$. Substituting this value (along with the given $J = 5$ kgm^2 and $c = 0.7$ Nms.) into Eq. 12 gives a K value of 0.05. Therefore, **K must not exceed 0.05 if the overshoot is to remain less than 5%**.

The inverse Laplace transform of the generic 2nd order transfer function multiplied by a step input (the step response) produces:

$$\theta(t) = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \cos(\omega_n \sqrt{1-\zeta^2} t - \tan^{-1}(\zeta/\sqrt{1-\zeta^2}) \tag{16}$$

If we set the time it takes for the decaying sinusoid term to reach 0.02, we can obtain the exact settling time. However, use this formula as it is the standard definition:

$$T_s = \frac{4}{\zeta\omega_n} \tag{17}$$

The numerator value of 4 is a middle value when ζ is allowed to vary from 0 to 0.9. Also, the equation assumes that $\cos(\omega_n \sqrt{1-\zeta^2} t - \tan^{-1}(\zeta/\sqrt{1-\zeta^2} = 1$ at the settling time.

Substituting Eqs. 11 & 12 into Eq. 17 produces the result that the $\sqrt{K/J}$ terms cancel. Therefore, **there is no effect of K on the settling time**. T_s is simplified to:

$$T_s = 8J/c \tag{18}$$

This returns a settling time of **57.14**s for this system's J and c.

(e) Step response

The step response for $K = 0.05$ is shown in Fig. 3. The peak value obtained was 1.05, which corresponds to a 5% overshoot. The response reached 2% of the steady-state value by 57.9s. This value is quite similar to that obtained analytically, considering that plot inspection in the MATLAB figure window was limited to increments of 0.7s and the fact that the analytically obtained settling time is also an estimate.

Therefore, **the zero-disturbance system does meet the 5% OS specification. It also exhibits a settling time similar to that obtained analytically**.

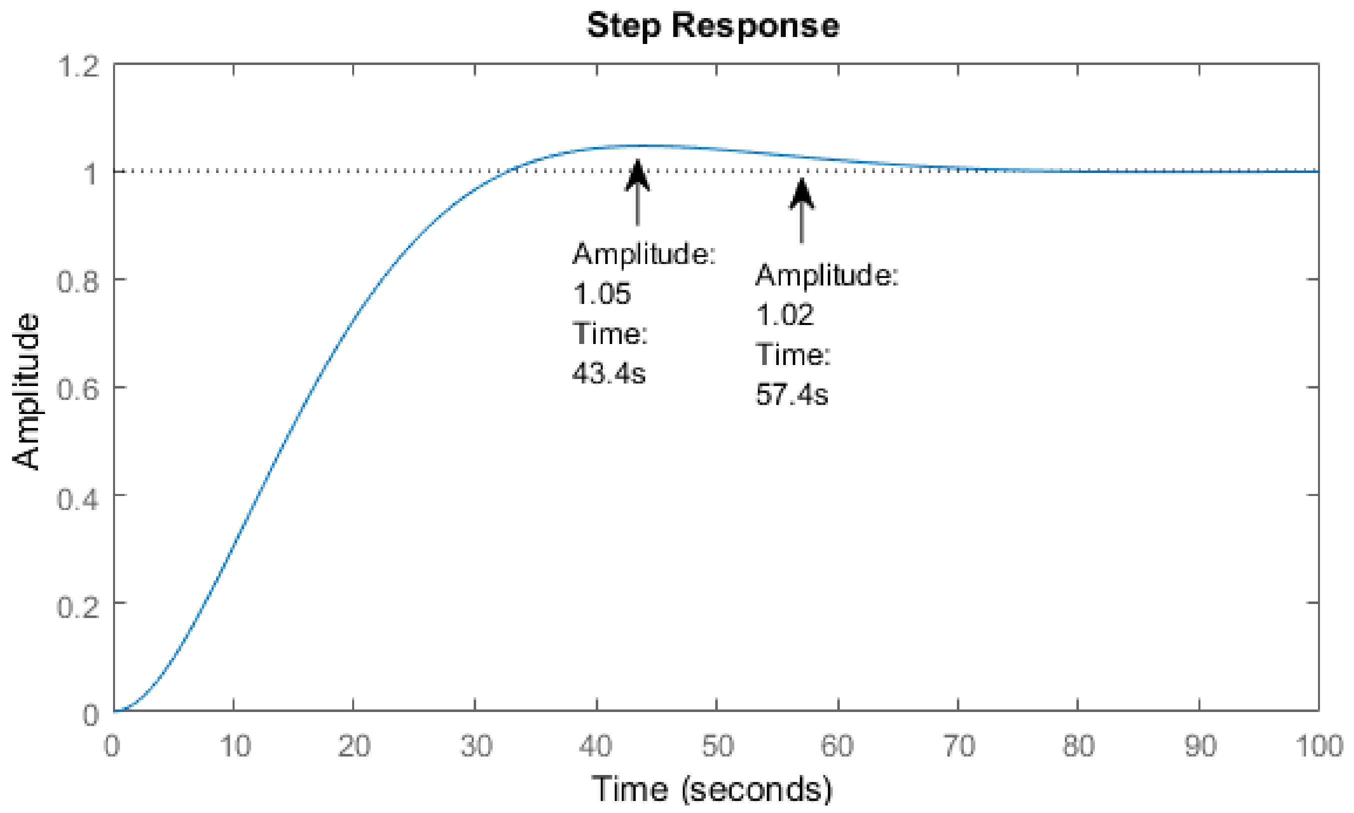


Figure 3: Open-loop step response.

(f) Steady-state error when $\theta_r = 0$ and $w(t) = 1$

Rearranging Eq. 9 gives

$$\theta(s) = \frac{W(s)}{Js^2 + cs + K} \tag{19}$$

When $w(t) = 1.0$, $W(s) = 1/s$. Applying the final value theorem shows that $\theta(t)$ tends to $1/K$ for a step input as $t \to \infty$ (Eq. 20). This result is consistent with theory for a system with pure gain. The error can not be zero if the output is to be finite and nonzero [1].

$$\lim_{t\to\infty} \theta(t) = \lim_{s\to 0} s\theta(s) = \lim_{s\to 0} \frac{s}{s(Js^2 + cs + K)} = 1/K \tag{20}$$

The steady state error is the reference value minus the steady state value:

$$e_{ss} = \theta_r(t) - \lim_{t\to\infty} \theta(t) = 0 - 1/K = -1/K \tag{21}$$

This evaluates to e_{ss} = **-20** when $K = 0.05$.

(g) Addition of derivative control

Derivative control has been added:

$$u = K(\theta_r - \theta) + K_d(\dot{\theta}_r - \dot{\theta}) \tag{22}$$

Fig. 4 displays the augmented block diagram. The system is now:

$$J\ddot{\theta} + c\dot{\theta} = K(\theta_r - \theta) + K_d(\dot{\theta}_r - \dot{\theta}) + w \tag{23}$$

$$\theta(s)[Js^2 + cs + K + K_d s] = \theta_r(s)[K + K_d s] + W(s) \tag{24}$$

The transfer functions θ_r to θ and w to θ are:

$$\frac{\theta(s)}{\theta_r(s)} = \frac{(K + K_d s)/J}{s^2 + (c + K_d)s/J + K/J} + \frac{W(s)/J}{\theta_r(s)(s^2 + (c + K_d)s/J + K/J)} \tag{25}$$

$$\frac{\theta(s)}{W(s)} = \frac{1/J}{s^2 + (c + K_d)s/J + K/J} + \frac{\theta_r(s)(K_d s + K)/J}{W(s)(s^2 + (c + K_d)s/J + K/J)} \tag{26}$$

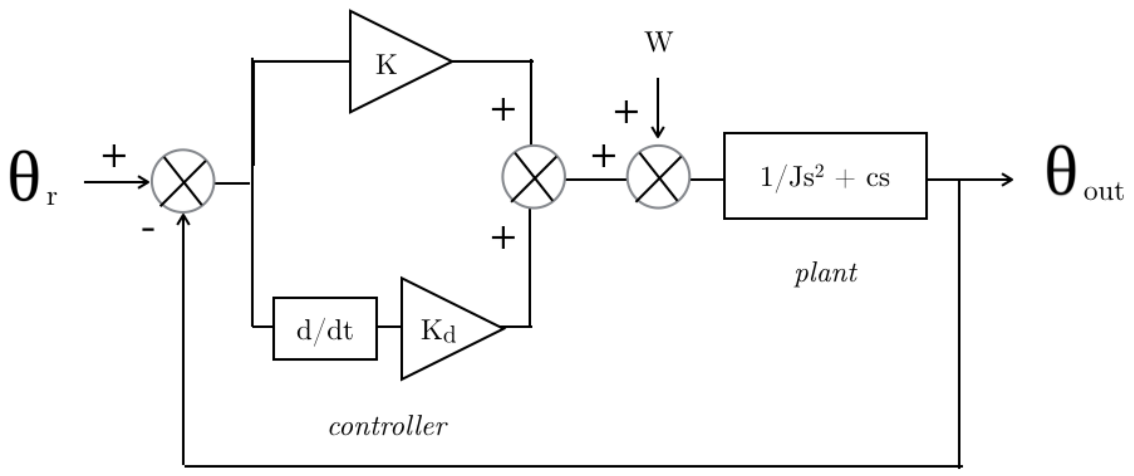


Figure 4: Block diagram for system with a PD controller.

(h) K and K_d for 5% OS and $T_s = 1$s

Take the system where disturbance w is zero. The settling time is now:

$$T_s = \frac{4}{\zeta\omega_n} = 4/\left[\frac{c + K_d}{2J\sqrt{K/J}} \times \sqrt{K/J}\right] = \frac{8J}{c + K_d} \tag{27}$$

Therefore, the settling time does not depend on K. For a settling time of 1s, substituting $T_s = 1$, $c = 0.7$ and $J = 5$ gives a K_d of **39.3**. A damping coefficient value was previously found that gives a % OS of 5%. Substituting $\zeta = 0.69$ and J and c values in Eq. 12 yields a K value of **168.0**. Whether these gains give exactly $T_s = 1$ and % OS=5 is explored in the next section.

(i) Step response

Plotting the system's step response shows that the overshoot (21%) exceeds the design specification (Fig. 5). The system technically settles within 2% of the steady-state value at 0.835s. Given that the equation used to set the settling time was itself an estimate, it could be said that the system achieved the 1s specification.

The discrepancy between the calculations and the received response is due to the incorrect assumption that $K \approx K + K_d s$. Most simply, the designed K and K_d are

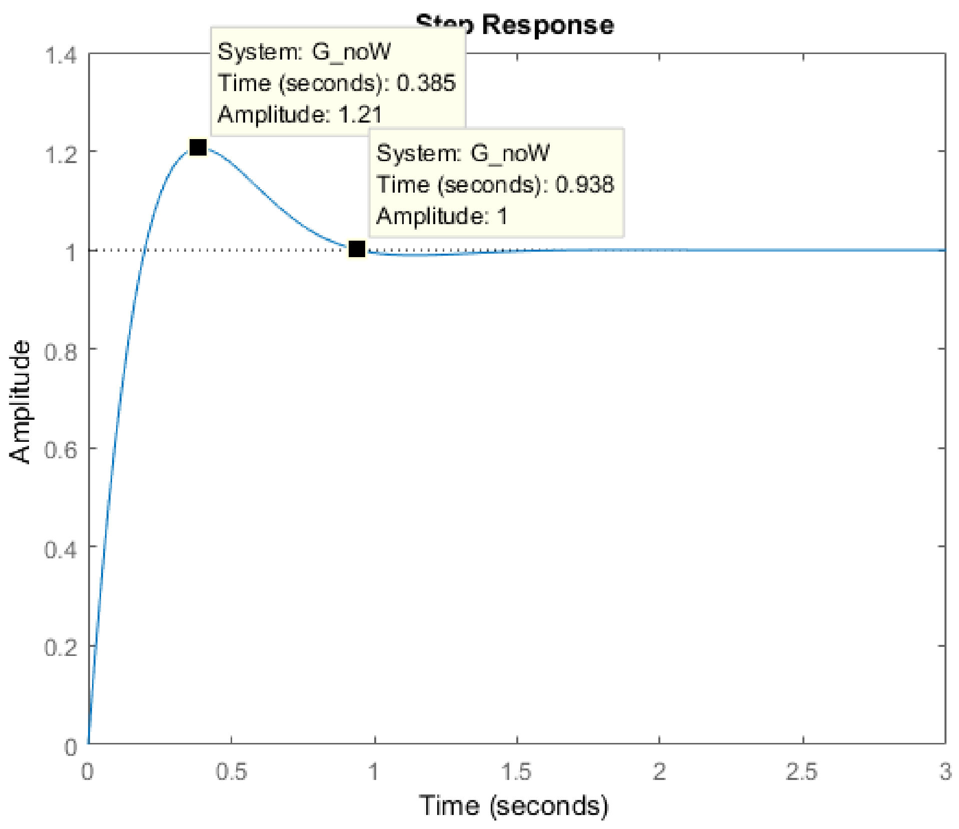


Figure 5: The system's step response for trialled gains $K = 168.0$ and $K_d = 39.3$.

for the following system:

$$KO_G = \frac{K}{Js^2 + (c + K_d)s + K} \tag{28}$$

When plotted, its step response meets the design specifications (Fig. 6). We can think of our PD system as:

$$\frac{\theta(s)}{\theta_r(s)} = KO_G(s) + K_d s O_G(s) \tag{29}$$

The additional $K_d s O_G(s)$ acts as a scaled derivative of the hypothetical system that would achieve the design specifications. Unfortunately, because K_d is also in the denominator, reducing the magnitude of K_d will also affect the nature of

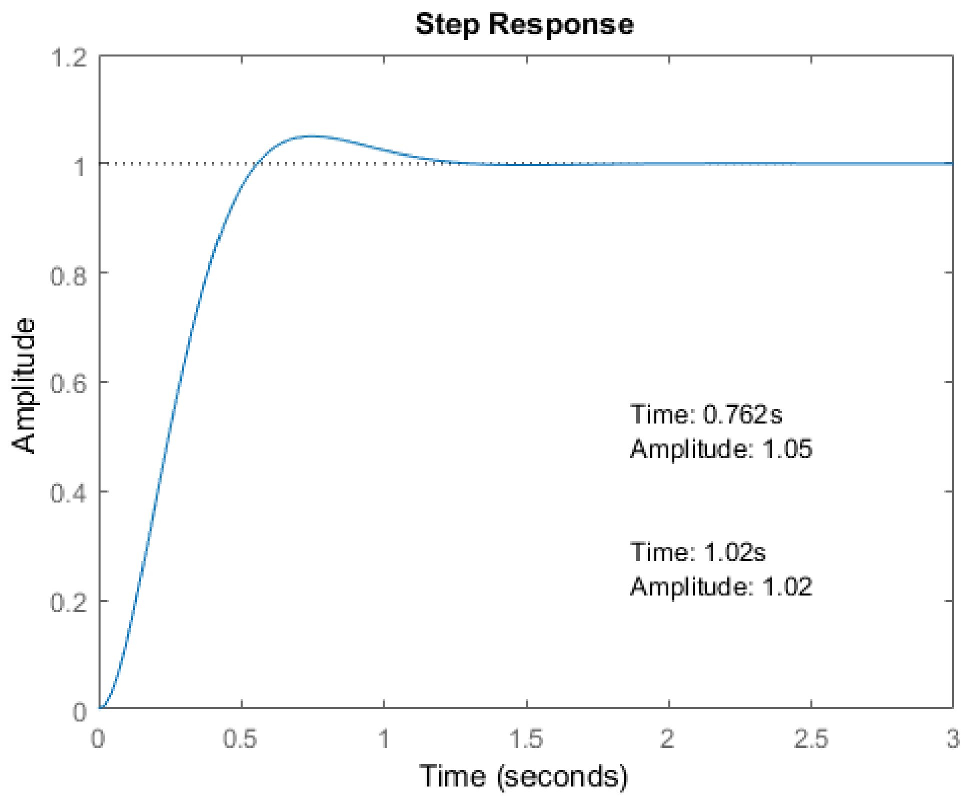


Figure 6: A system which meets the design specifications: $KO_G = \frac{K}{Js^2+(c+K_d)s+K}$.

the system response (overdamped, unstable, etc). One option is to derive anew formulae for settling time and % OS based on the form of the new system, though this could be quite cumbersome.

Currently, K_d is at such a magnitude that the zero is placed very close to the poles (Fig. 7). From the perspective of a pole-zero plot analysis, this is also an indicator that the additional zero has a substantial effect on the system's transient response [1].

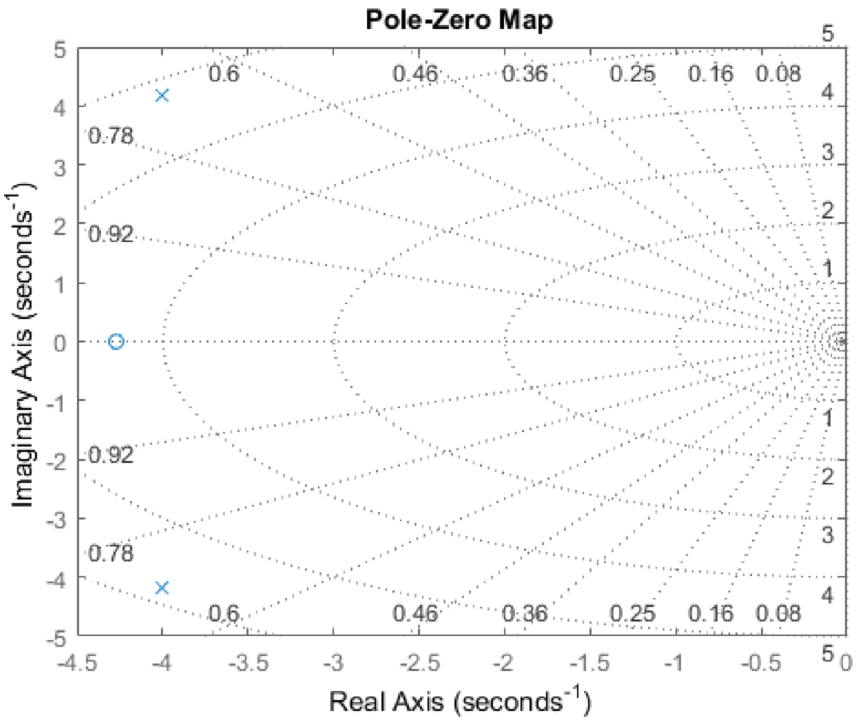


Figure 7: Pole-zero plot for the system with derivative control added.

(j) Steady-state error when $\theta_r = 0$ and $w(t) = 1$

Take Eq. 26. When $\theta_r = 0$:

$$e_{ss} = \theta_r(t) - \lim_{t\to\infty}\theta(t) = 0 - \lim_{s\to 0} s\theta(s) = 0 - \lim_{s\to 0}\frac{s}{s(Js^2 + (c+K_d)s + K)} = -1/K \qquad (30)$$

This evaluates to a steady-state error of **-0.006** using $K = 168.0$. To make the steady-state error exactly zero, **add an integrator**. This will have the following effect on evaluating the error:

$$e_{ss\ \mathrm{I}} = 0 - \lim_{s\to 0}\frac{s}{s(Js^2 + (c+K_d)s + K_i/s + K)} = 0 - \frac{1}{K_i/0 + K} = -1/\infty \qquad (31)$$

With an integrator, the closed loop transfer function adjusts to the above. As s $\to$ 0, the denominator tends to infinity, which then effectively nils the steady state

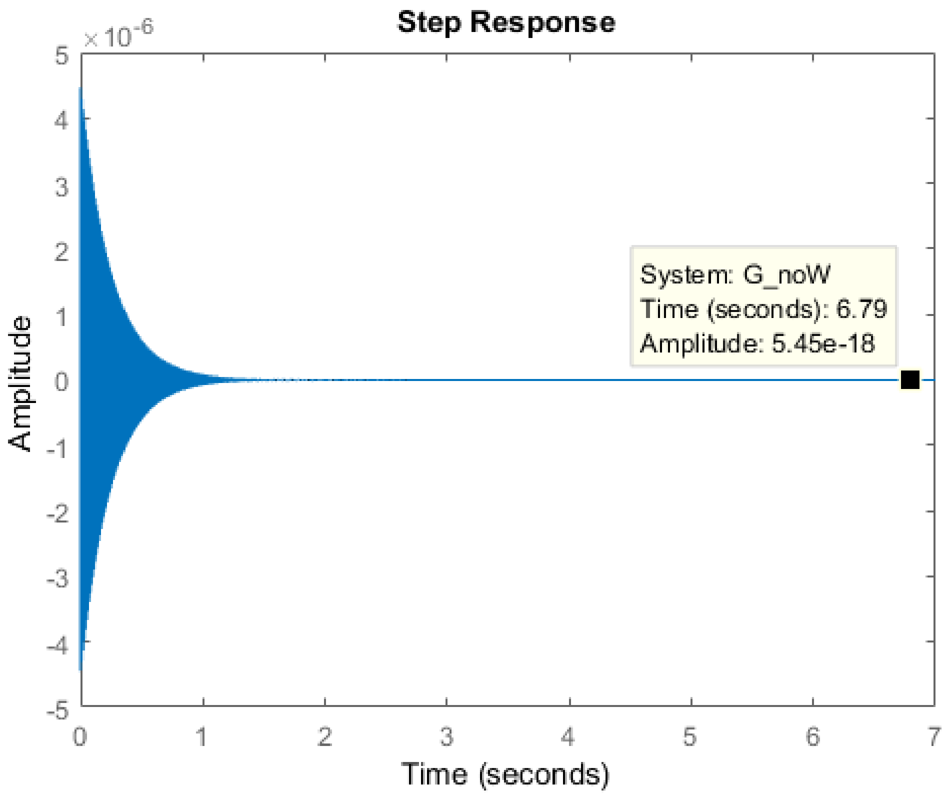


Figure 8: The step response of the original PD system with $K = 10^9$.

error. This is equivalent to derivations in Nise which inspect the effect of system order using the open loop transfer function [1].

An alternative is to **increase K**, which does not eliminate steady state error but diminishes it. It can be seen that the original expression Eq. 30 would also go to zero when $K = \infty$ which is impossible in practice. Fig. 8 shows the PD system with an incredibly high K of 10^9. While the transient response may be undesirable, the steady-state value does approach zero.

2 Position control for a bionic eye

- **Design a control system for the rotational position of a bionic eye about a single (Z) axis that halves settling time while minimising overshoot and steady-state error.**
- **Model the eye as a perfectly spherical inertial mass with damping constant of 3.05×10−4 Ns/m and stiffness of 1.65 × 10−2 N/m.**
- **Rotation is actuated through a motor with first-order dynamics and** $\tau = 0.01$**s.**
- **The motor drive is torque-limited to ± 0.5 Nm.**

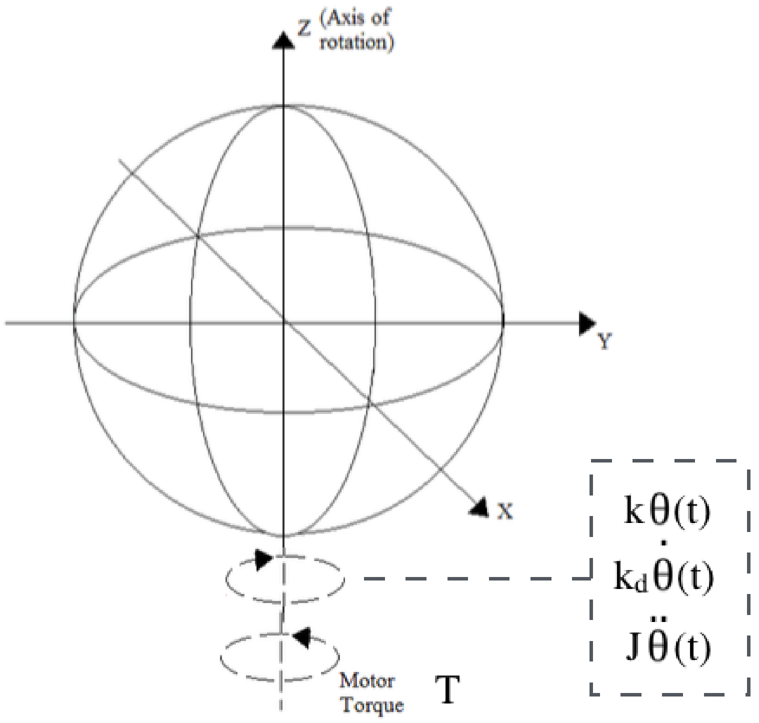


Figure 9: The bionic eye in 1 DOF. k_d = 3.05x10-4 Ns/m; k = 1.65x10-2 N/m; $J = 2mr^2/5$.

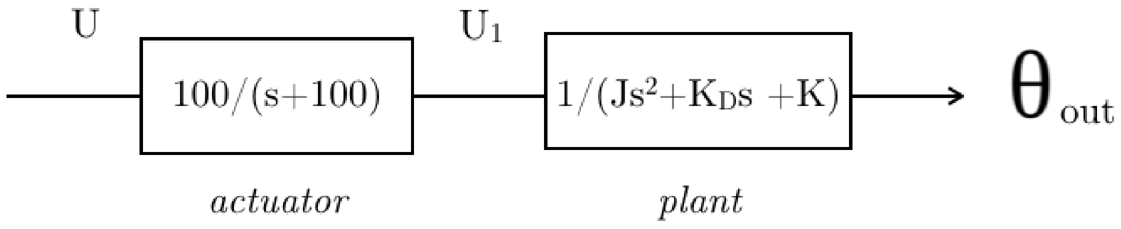


Figure 10: The open-loop block diagram for the bionic eye system.

(i) Problem modelling

A free body diagram is presented in Fig. 9. A block diagram is shown in Fig. 10. The plant is represented by:

$$J\ddot{\theta}(t) + K\theta(t) + K_d\dot{\theta}(t) = T(t) = u_1(t) \tag{32}$$

Rearranged in the frequency domain, this is also:

$$\frac{\theta(s)}{U_1(s)} = \frac{1/J}{s^2 + K_d s/J + K/J} \tag{33}$$

The actuator is represented by:

$$\frac{U_1(s)}{U(s)} = \frac{a}{s+a} = \frac{100}{s+100} \tag{34}$$

The open-loop transfer function is then:

$$\begin{aligned}\frac{\theta(s)}{U(s)} = \frac{\theta}{U_1(s)} \times \frac{U_1(s)}{U(s)} &= \frac{100}{(s+100)(Js^2+K_d s+K)} \\ &= \frac{100/J}{s^3+s^2(100+K_d/J)+s(100K_d/J+K/J)+100K/J}\end{aligned} \tag{35}$$

Along with the assumptions stated in the problem question, the model assumes zero disturbances. It is also assumed that the potential bionic eye is made to roughly the same size and mass as a human eye, so that $m = 7.5$g and $r = 12$mm [2]. It is also assumed that it is solid as opposed to hollow so $J = 2mr^2/5$.

(ii) T_s of OL system

The time it took for the OL system to reach 2% of the steady-state value was **0.066s** (Fig. 11). This follows the standard definition of settling time [1]. Now, a controller will be designed that aims to halve this. Note that the time it took to reach the actual steady-state value of 60.6 was 0.135s.

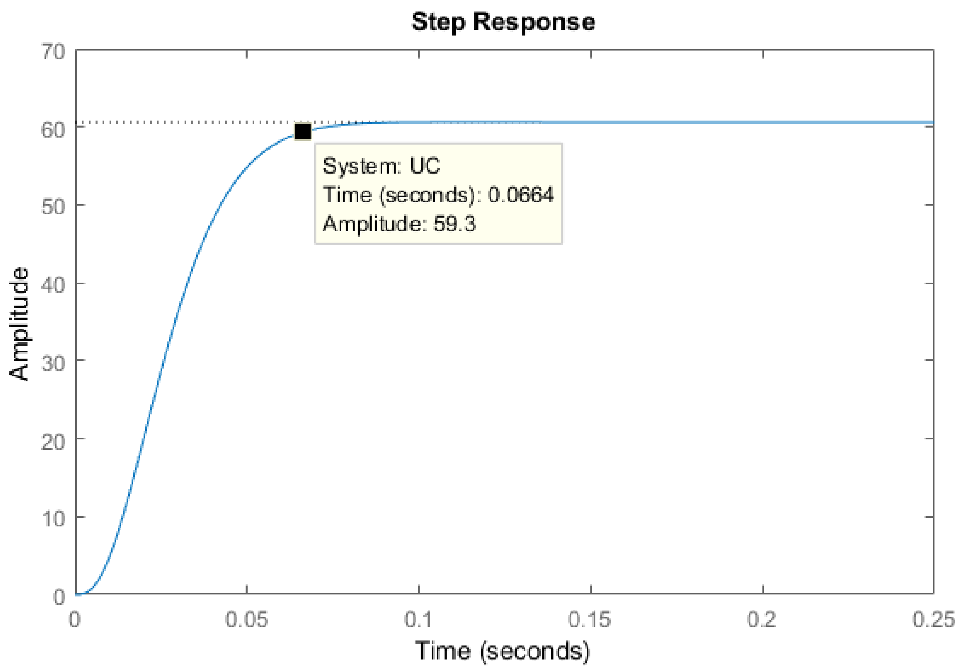


Figure 11: The step response of the uncompensated OL system, with the settling time shown.

(iii) Root Locus

A root locus has been plotted to aid controller design, first analytically then verified using `rlocus()` in MATLAB (Fig. 12). The analytical procedure is outlined.

1. Plot the poles of the OL transfer function. There are 3 poles and no zeros, so all 3 poles go to ∞. The poles exist at -647, -100 and -59.03.
2. Find where the asymptotes cross the real axis using:

$$\sigma_a = \frac{\sum \text{finite poles} - \sum \text{finite zeros}}{\#\text{finite poles} - \#\text{finite zeros}} = -92.17 \qquad (36)$$

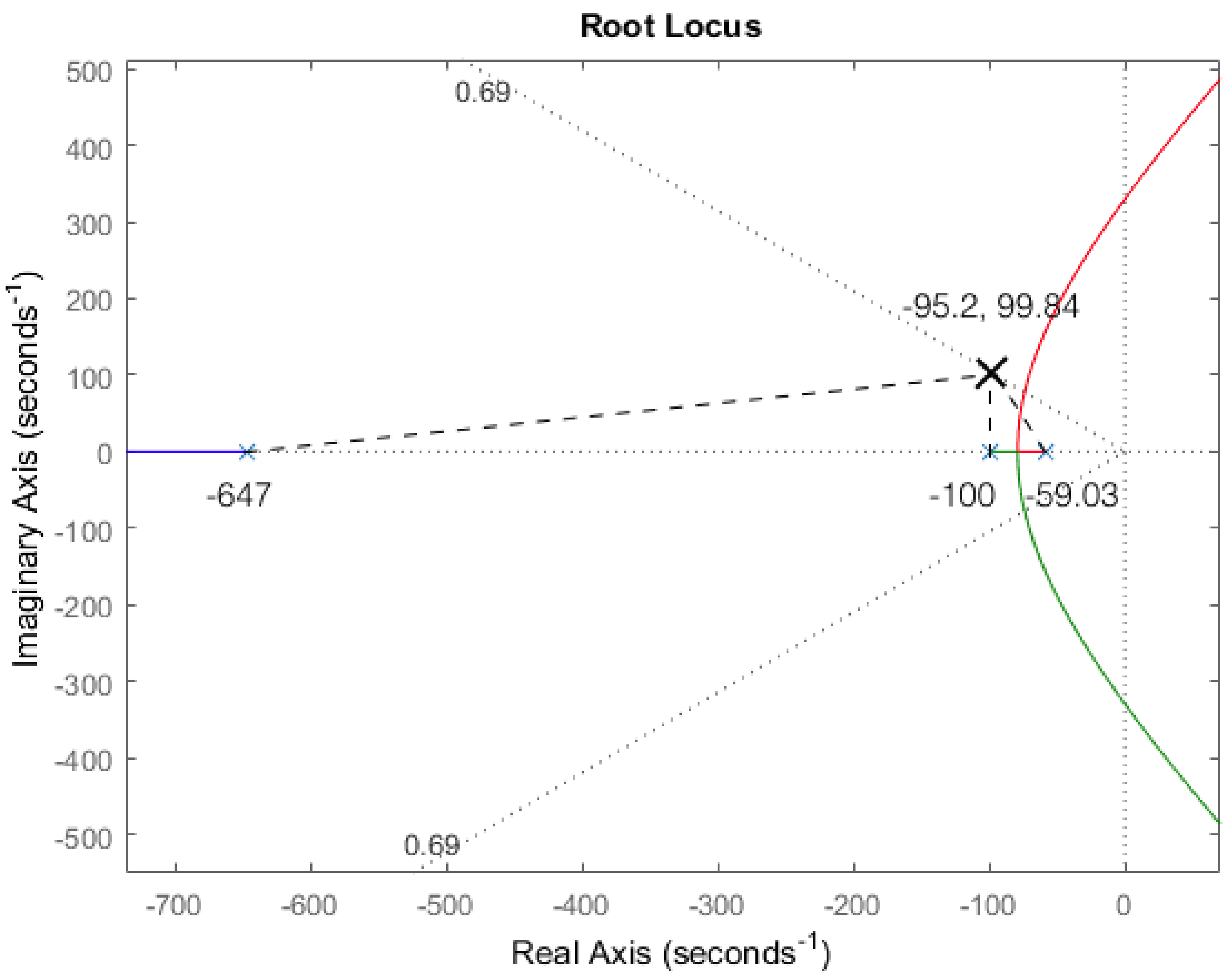


Figure 12: Root locus for the uncompensated system. The desired pole locations (mirrored about the real axis) are also shown).

3. Find the angle between the asymptotes and the real axis:

$$\theta_a = \frac{(2k+1)\pi}{3} = \pi/3, \pi, 5\pi/3 \text{ for } k = 0,1,2 \tag{37}$$

From inspecting the root locus, it is evident that 2nd order assumptions are justified. The third pole is considerably far from the other two poles.

(iv) Design with 5% OS

To begin with, design to halve settling time while maintaining a 5% overshoot. The procedure used is as follows, implemented in `Q2iv.m` which can be inspected in Appendix 1.

1. Find the desired dominant pole locations $-\sigma \pm \omega_d$ by evaluating:

$$\sigma = \pi/(0.5T_s) = 95.20 \tag{38}$$

$$\omega_d = \sigma \tan(\phi) = 99.84 \tag{39}$$

   where T_s was the OL settling time and $\phi = \cos^{-1}(\zeta)$.

2. Design the compensating zero so that the root locus goes through these points. First, find the sum of angles from the open-loop uncompensated poles to the desired dominant pole locations (Fig. 12). Subtracting 180° from this returns the necessary contribution of the compensator zero, 27.42°. The location of the new zero can now be calculated from geometry:

$$\frac{99.84}{\tan(27.42)} + 95.2 = (-)287.65 \tag{40}$$

3. Plot the root locus of the new derivative control-added system (Fig. 13):

$$\text{PD} = \frac{(s+287.65)}{(s+100)(Js^2 + K_d s + K_p)} \tag{41}$$

   The root locus now intersects -95.2 ± 99.8, with a gain K of 0.0119.

4. Plot the closed-loop step response to observe the improvement (Fig. 14), where the closed-loop system is `CL_PD=feedback(K*PD,1)`. The settling time is now 0.04s. This has reduced the settling time by 60%. Designing for 50% reduction but achieving 60% could be due to the fact that the settling time expression is only an estimate.

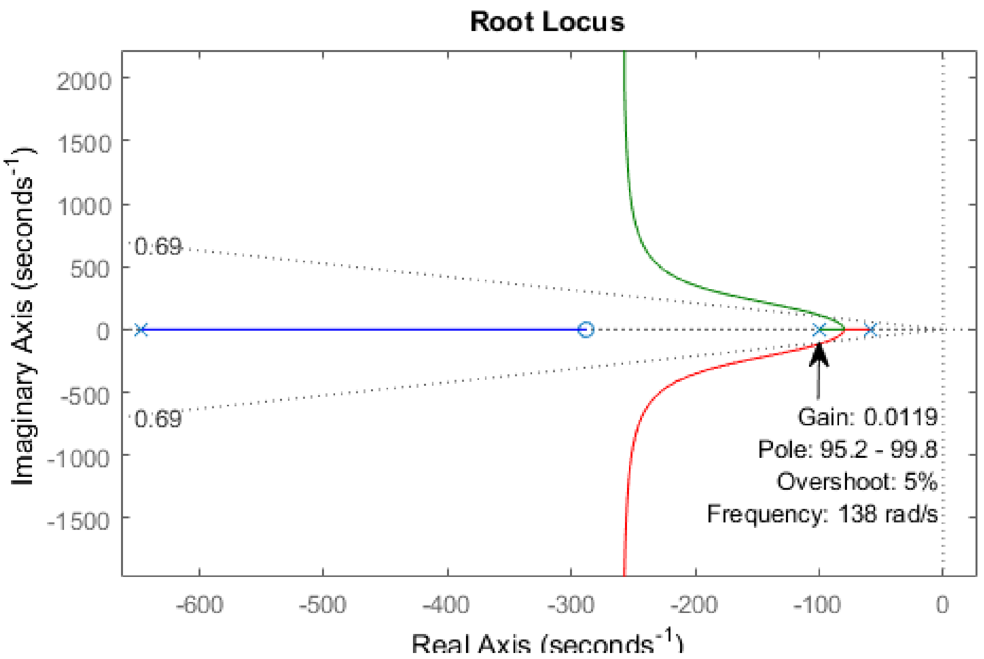


Figure 13: The root locus intersecting the desired pole locations with an added zero.

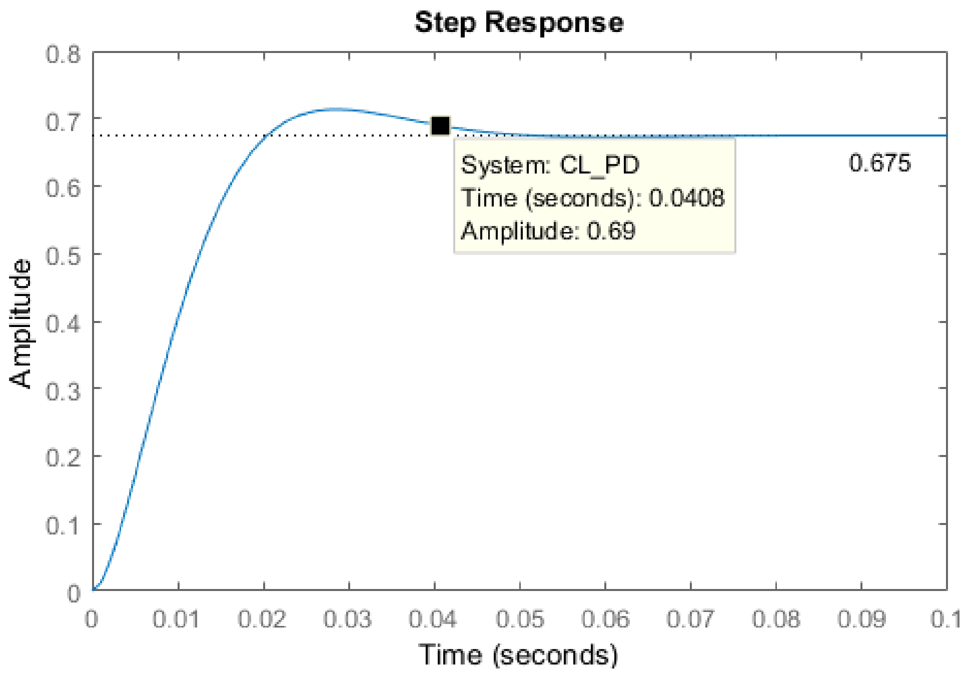


Figure 14: The settling time has reduced by 60% with PD control.

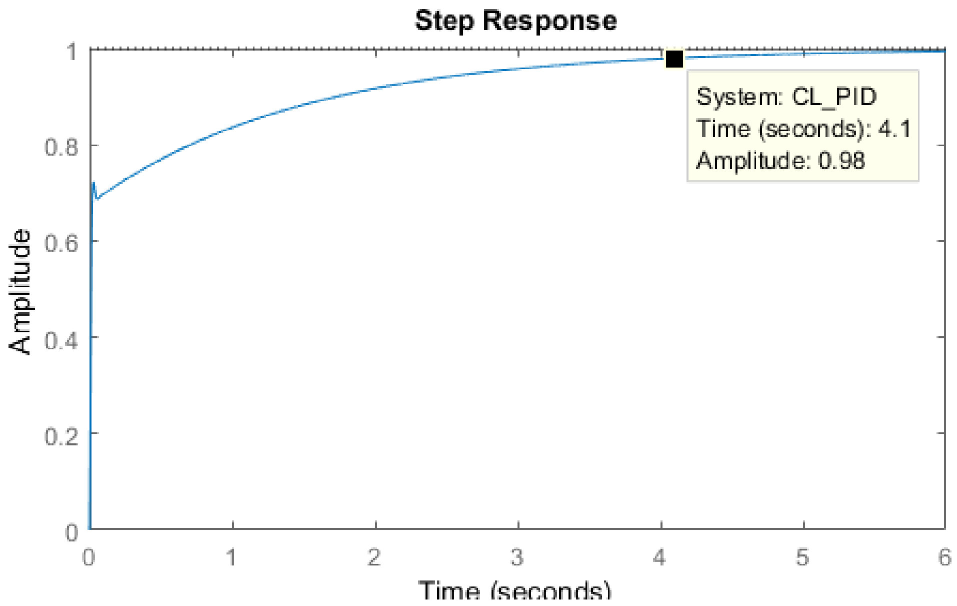


Figure 15: While there is zero steady-state error, the settling time has been significantly affected with the addition of integral control.

5. Continue by adding integral control by adding an 's' in the denominator and a zero very close to the imaginary axis to negate its change to the system:

$$\text{PID} = \frac{(s+1)}{s}\text{PD} \tag{42}$$

   Again, find the operating point where the PID root locus intersects the $\zeta = 0.69$ line. This returns K = 0.0120.

6. Plotting the step response `CL_PID` however, shows that the settling time has now been significantly affected. A settling time of 4.1s is not acceptable. However, the steady-state error is now zero.

At this point, there are several options: (1) to revisit the overshoot, or (2) adjust the position of the integrator zero [1]. Alternatively, one could begin by remedying the steady-state error first then improve the transient response. Another method is to find the most effective placement of added poles/zeros by brute force, testing a large range of possible placements and measuring the settling time and overshoot. Various techniques also exist in literature, such as the Ziegler Nichols method [3] [4]. The hypothesis is that there is likely to be a range of possible pole/zero options that will yield a result close to the optimum.

For the purposes of this assignment, the method used will be simply to vary the position of the integrator zero. A range of values were trialled and it was found that positioning the integrator zero at -40 yielded a step response that minimised settling time and overshoot (Fig. 17). The settling time is **0.025**s, which is 38% of the original open loop settling time. It was found that moving the zero from left to right about this range resulted in a lowering peak amplitude (Figs. 18 & 19). The middle value $I_{zero} = 40$ was chosen because graphically, placing the over- (and under-) shoot at equal distance from the steady-state line allowed it to settle within ±2% of the final value, largely reducing the settling time by its technical definition.

In further studies, reducing the initial percentage overshoot (to 1, 2, 3, 4%) before attending to the transient response should be trialled. In general, Nise proffers that simulation and iterative design is key to PID design [1].

The total system is then:

$$PID = \frac{(s+40)(s+287.65)}{s(s+100)(Js^2+K_d s+K_p)} \tag{43}$$

which can be rewritten as:

$$PID = \frac{k_p + k_i/s + k_d}{s^3 + s^2(100 + K_d/J) + s(K/J + 100K_d/J) + 100K/J} \tag{44}$$

Controller gains $k_p = 3.90$, $k_d = 0.0119$, $k_i = 136.92$ were found comparing Eq. 43 to the general:

$$G_{\text{controller}} = \frac{K_d(s^2 + K_p s/K_d + K_i/K_d)}{s} \tag{45}$$

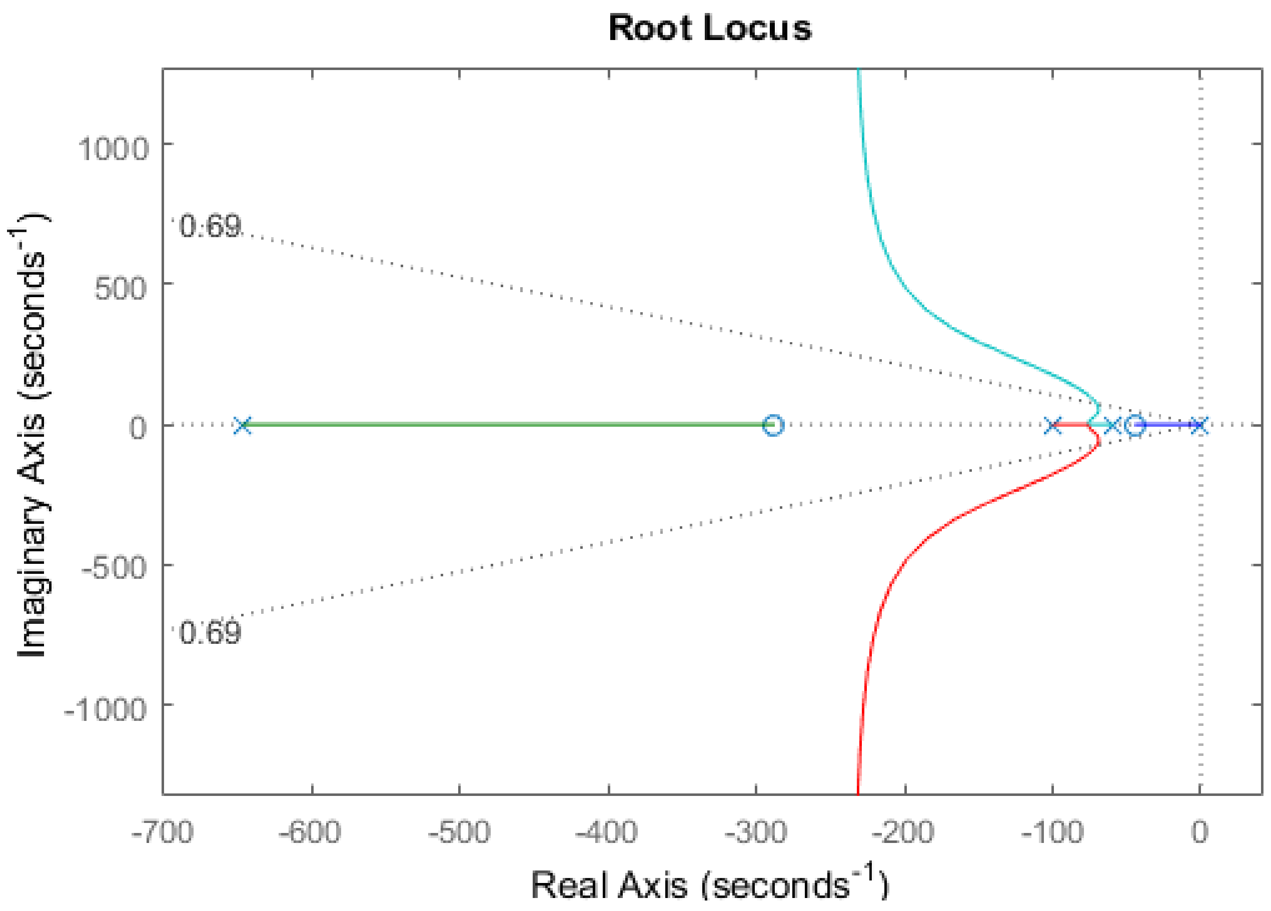


Figure 16: The root locus for the proposed PID controller achieving $T_s = 0.025$s.

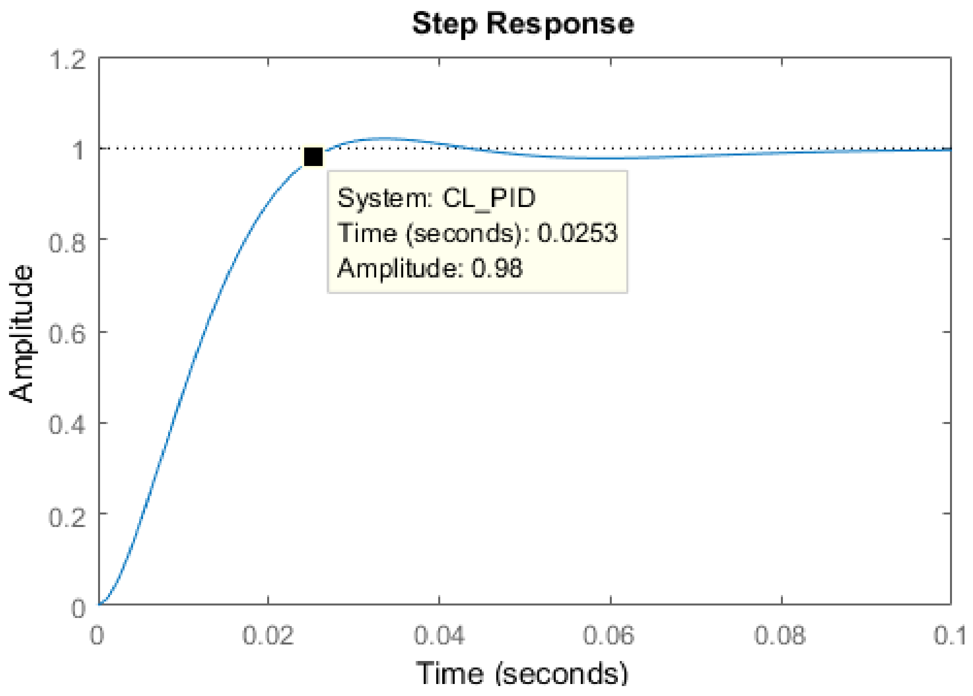


Figure 17: The PID-controlled step response where the integrator zero is adjusted to $I_{zero} = 40$. $T_s = 0.025$s.

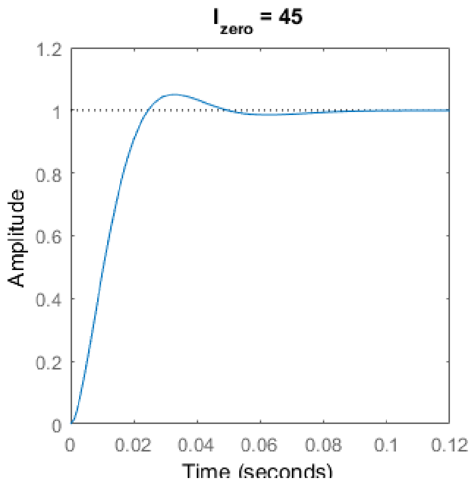


Figure 18: PID step response where $I_{zero} = 45$.

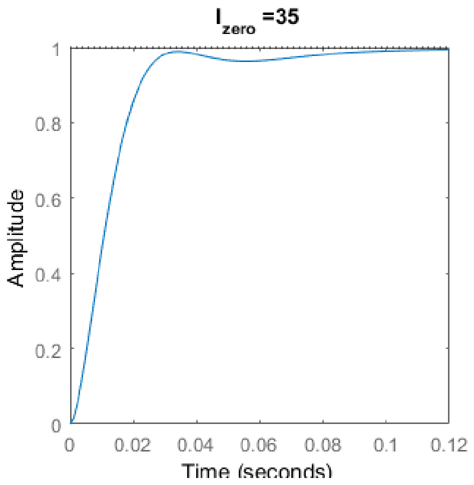


Figure 19: PID step response where $I_{zero} = 35$.

(v) Torque limitation

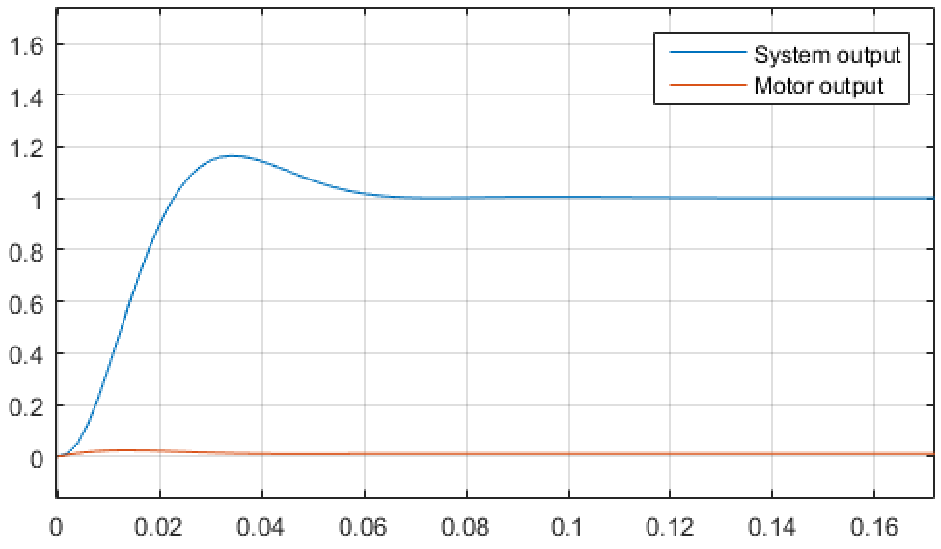


Figure 20: The motor output U_1 (plant input) is shown with the system output θ. As configured, the addition of a saturator did not make a difference.

The torque limitation was explored in Simulink. The system with the designed PID control was arranged, and a saturator was added to simulate the torque limit of the motor (Fig. 21). However, it was found that for this system, the torque-limited and non-torque-limited responses were exactly the same. The motor output even without the saturator was within ± 5 Nm (Fig. 20). Therefore, the torque limitation had no effect on the system.

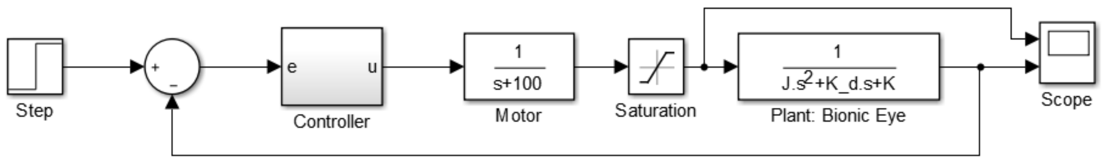


Figure 21: The Simulink model with a saturator.

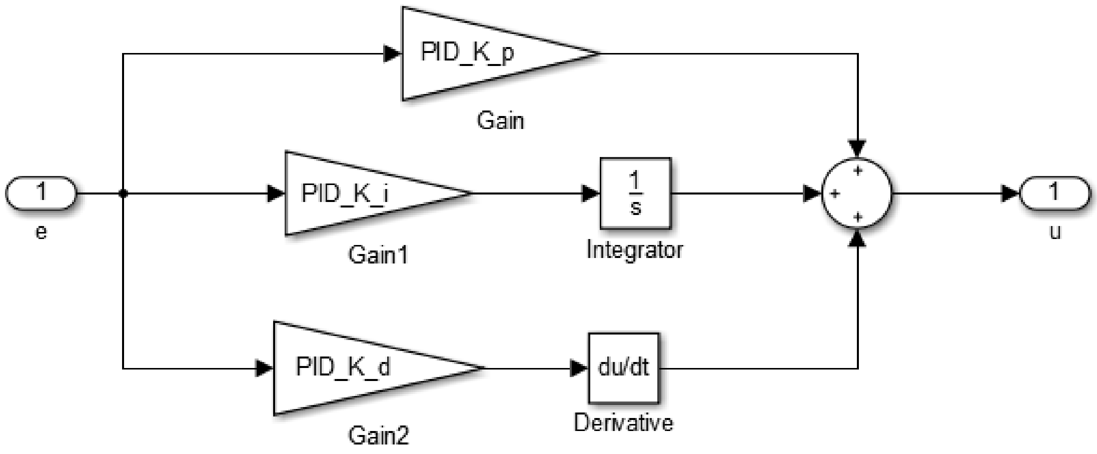


Figure 22: The controller subsystem.

3 Laboratory

This question explores and builds upon results in the first laboratory.

(a) Simulation vs. hardware

The hardware vs. simulation open-loop step response is shown in Fig. 25. The step input for both was 2V, beginning at $t = 1$s. The hardware response was obtained using the QUBE-Servo. A transfer function was written based on the hardware response. This transfer function was then used to create the simulated response.

Observing the hardware response, the final output value was about 46.1 rad/s. Therefore, the implied gain K is 23.05 when the input step is 2V. The system took 1.12s to reach 63% of its final value with a 1s step delay, so $\tau = 0.12$. Given this, the transfer function of the system is approximated as:

$$G(s) = \frac{23.05}{0.12s + 1} \tag{46}$$

Model parameters K and τ were verified via simulation. The simulated response is extremely similar to the experimental response in terms of shape and settling value. However, it excludes the disturbances.

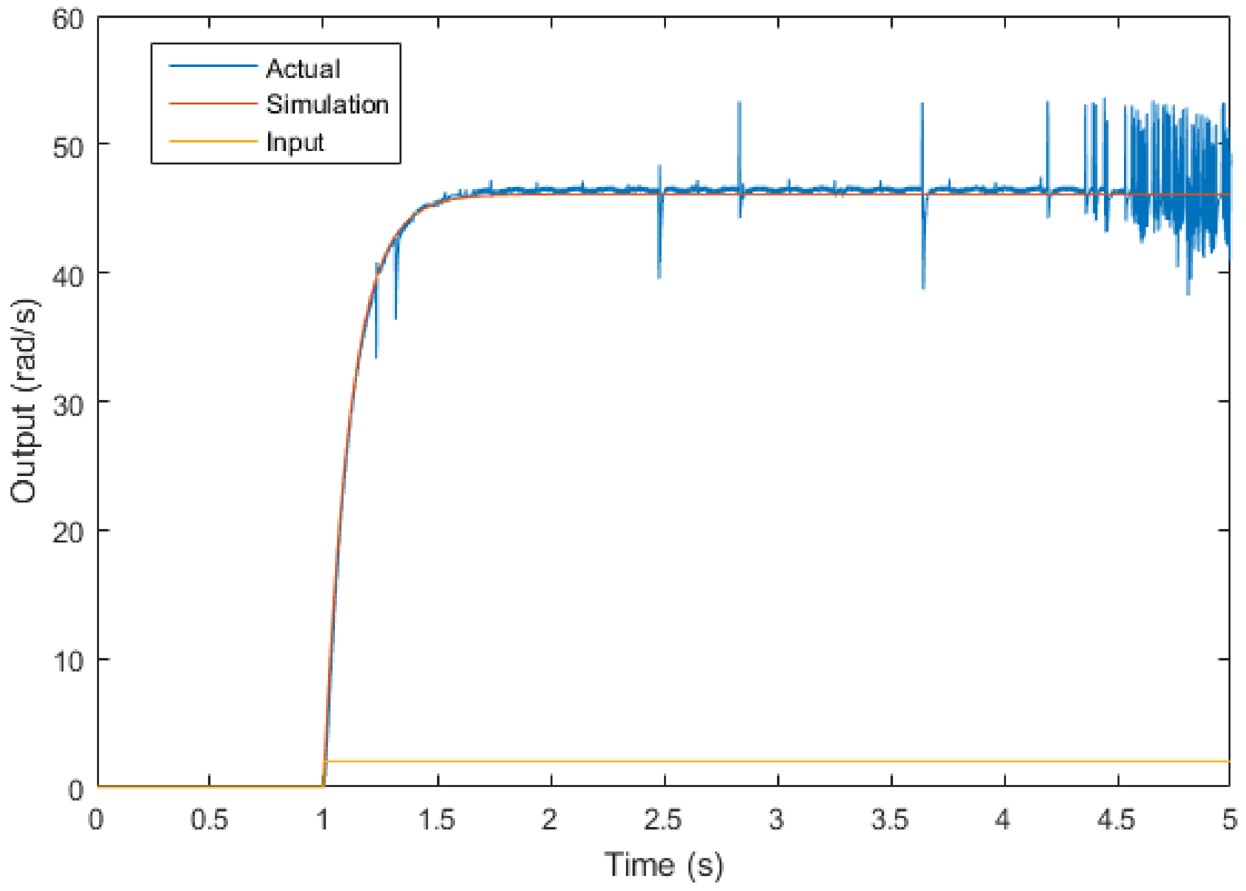


Figure 23: The simulated system of $G(s) = 23.05/(0.12s+1)$ vs. the hardware open-loop step response. The input step amplitude is 2V, where the step starts at $t = 1$s.

(b) Open vs. closed loop

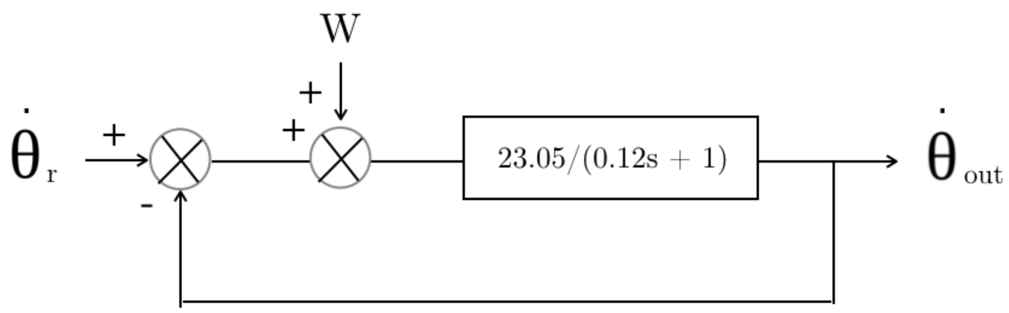


Figure 24: Block diagram for the system showing a closed loop.

The block diagram for the system showing the closed loop is given in Fig. 24. The open and closed step responses involving disturbances are shown for the total time of 20s in Fig. 25.

In general, it is evident that closing the loop results in finer, shorter deviations, that is, it corrects disturbances much more quickly. In an open-loop system, the plant input is simply $\dot{\theta}_r$ whereas for a closed-loop system it is the error: $\dot{\theta}_r - \dot{\theta}$. This explains the fast attenuation of error as well as the different steady-state values.

Unfortunately, a mistake was made in the laboratory where `u_in` values were overwritten for the closed loop test. Nevertheless, it would have been expected that while the open-loop input was a constant 2V, the closed-loop input would have deviated through time working against the direction of deviations in response ($E = R - Y$).

Figs. 26 and 27 take a closer look at the transient and steady-state response. The transient response shows a much faster response in the closed loop system, albeit with an overshoot. The closed loop system is still first order, therefore the explanation of overshoot could be sourced to hardware configurations. The steady-state response shows more clearly that the closed loop system is better at retaining the steady-state value. It is desirable in most cases for the output to remain as close as possible to the input at all times, if the input is a reference value.

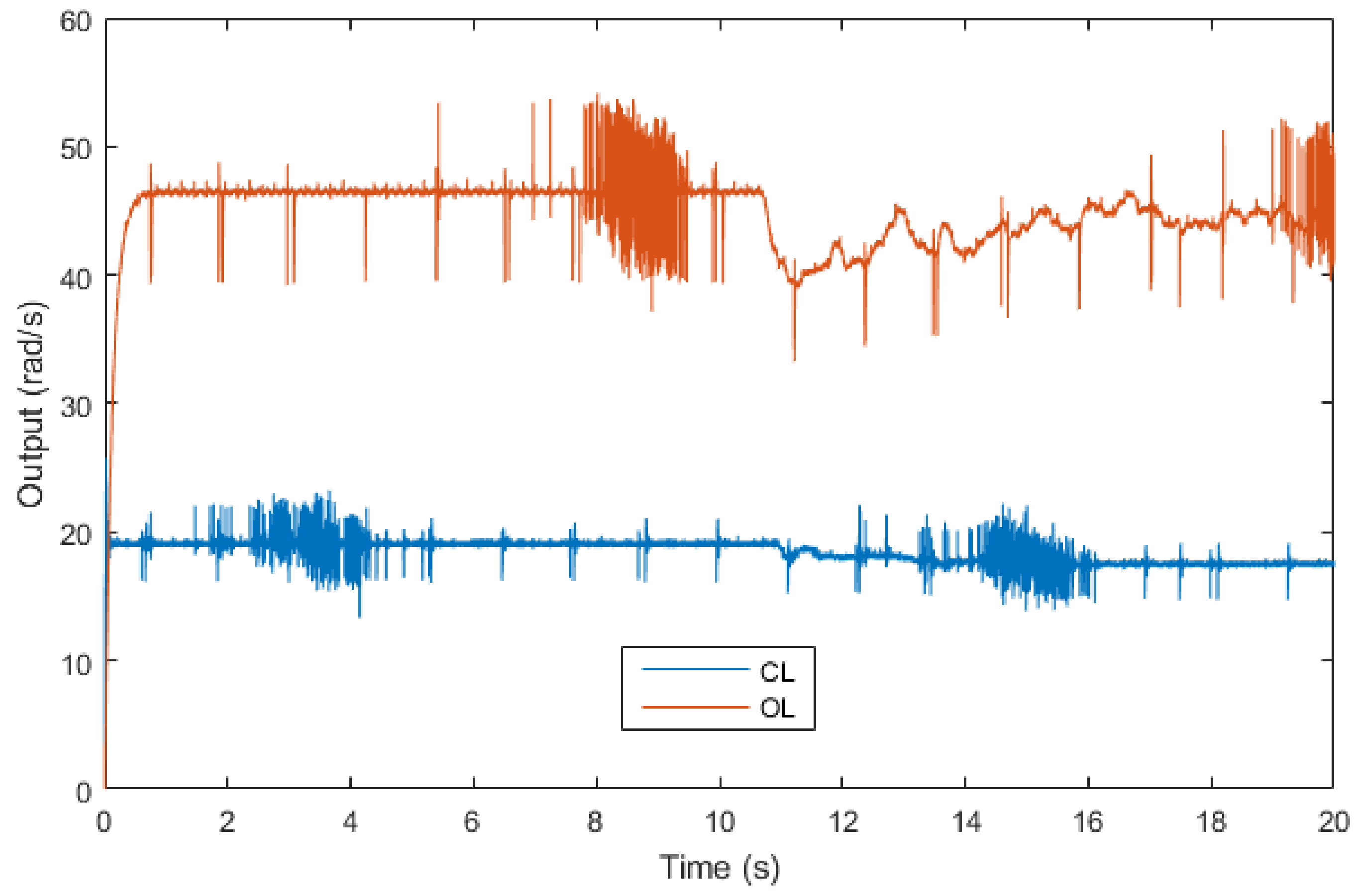


Figure 25: The open (OL) vs. closed (CL) loop responses where random disturbances were introduced.

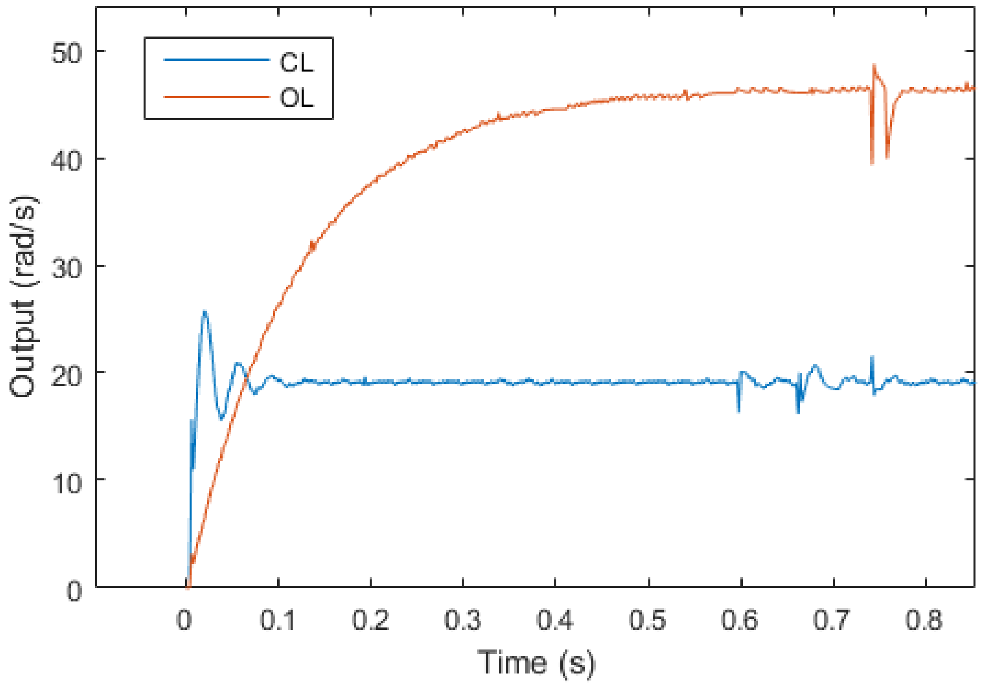


Figure 26: A comparison of the transient responses.

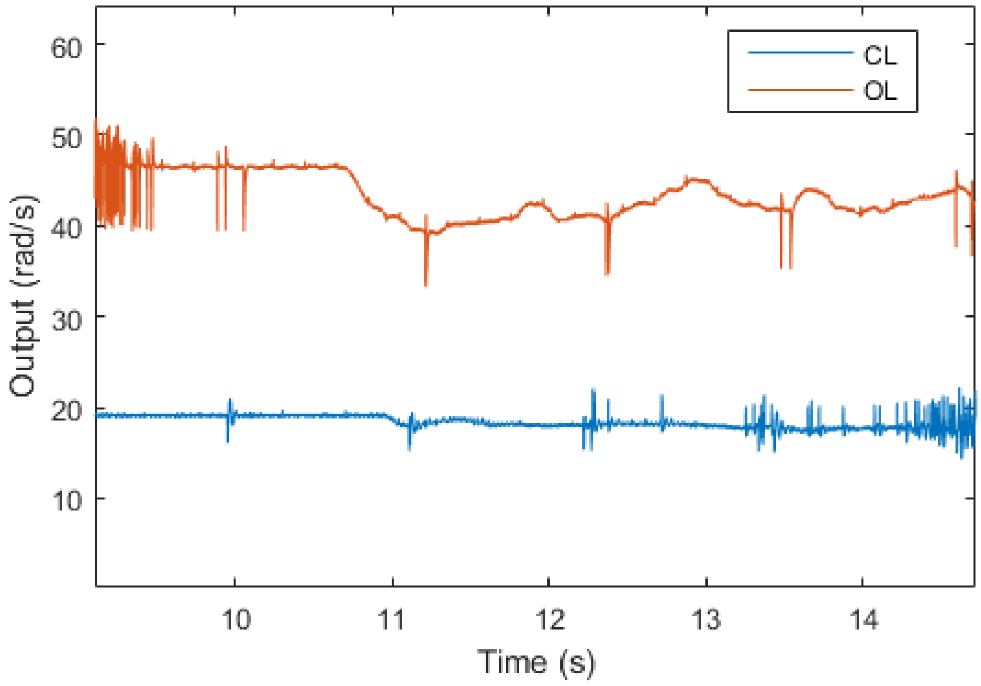


Figure 27: A comparison of a section in time of the steady-state responses.

(c) Toy robot

- **Use the model in the lab to drive a toy robot.**
- **Each wheel will need to be controlled.**
- **Assume the robot's mass is negligible and that there is high gear ratio between each motor and wheel.**
- **The wheel radius r is 0.05m and axle length L is 0.1m.**
- **Follow a curved road.**
- **Convert a curved road on a 2D plane into reference commands for v_r and v_l.**
- **The constant desired forward velocity v is 1m/s.**
- **Simulate the system and report on how well the robot followed the desired path. Discuss accuracy.**

The kinematic equations for the robot are:

$$\dot{x} = v\cos(\phi) = \frac{r}{2}(v_r + v_l)\cos(\phi) \tag{47}$$

$$\dot{y} = v\sin(\phi) = \frac{r}{2}(v_r + v_l)\sin(\phi) \tag{48}$$

$$\dot{\phi} = \omega = \frac{r}{l}(v_r - v_l) \tag{49}$$

where v is the forward velocity of the robot, v_L, v_R are the velocities of the left and right wheels and ω is the angular velocity of either the left or right wheel.

(i) Problem modelling

Two diagrams are shown to demonstrate the problem model (Figs. 28 & 29). Model inputs and outputs are shown in Fig. 30. A constant velocity input is assumed. Path inputs are path radius about a centre of rotation. There should be no issues to this approach. It is scaleable and modular so extended path designs can utilise this input model with additional (e.g. linear traverse) code. The output of interest is the actual velocity, though other data (such as the ω of each wheel) can also be retrieved from the Simulink/MATLAB model.

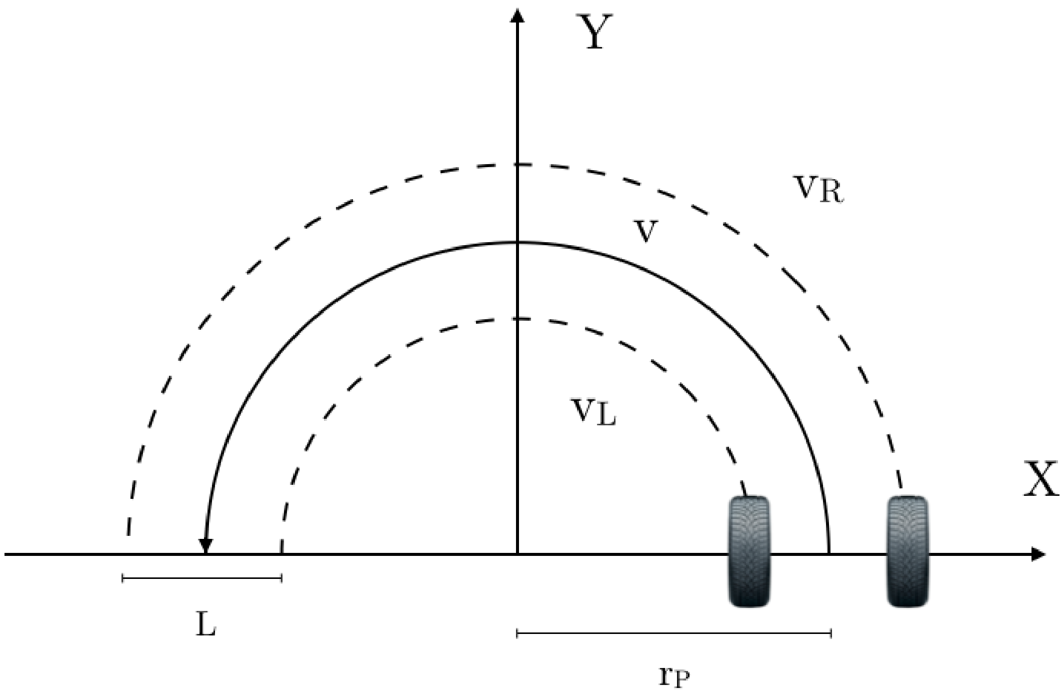


Figure 28: XY diagram.

The following equations were used pre-simulation to transform the input into reference values for the ω of each wheel. The robot's angular velocity about the instantaneous centre of rotation (IC) (rad/s) is:

$$\omega = v/r_P \tag{50}$$

The linear velocity of the wheel closest to the IC (e.g. *'v_L'* if turning counter-clockwise) [m/s]:

$$v_L = \omega(r_P - (L/2)) \tag{51}$$

The linear velocity of wheel furthest from the IC [m/s]:

$$v_R = w * (r_P + (L/2)) \tag{52}$$

The angular velocity of wheel closest to IC [rad/s]:

$$w_L = v_L/r \tag{53}$$

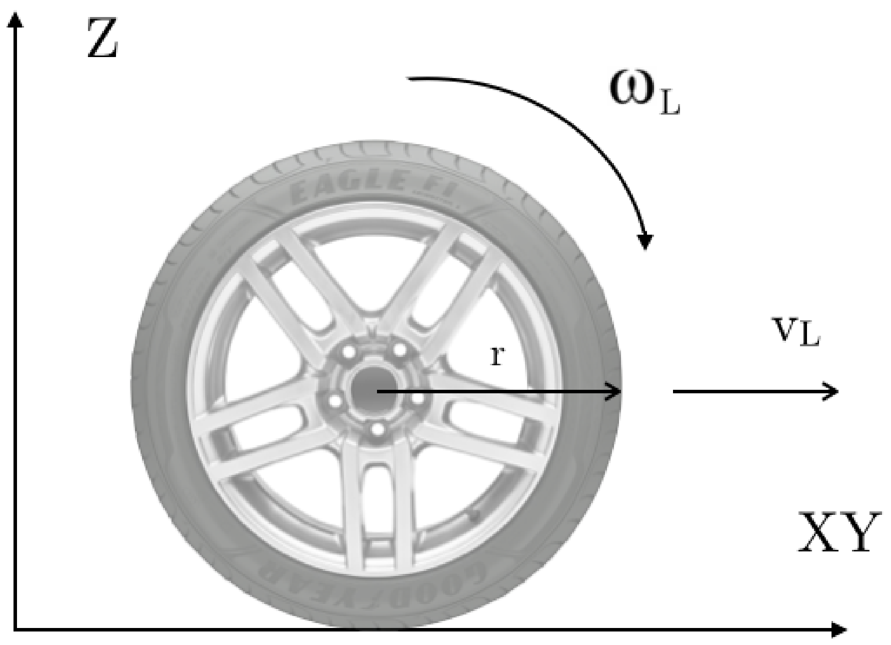


Figure 29: Z-XY diagram. For right wheel, subscript variables with ’R’.

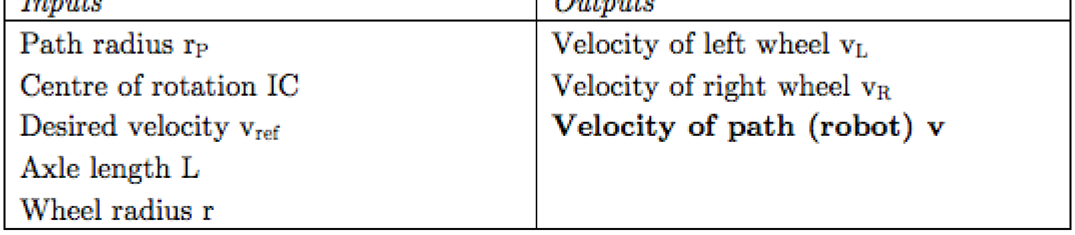
The angular velocity of wheel furthest from IC [rad/s]:

$$w_R = v_R/r \tag{54}$$

The linear velocity of the robot (path velocity) is:

$$v = (V_L + V_R)/2 \tag{55}$$

This set of equations is equivalent to the expressions given in the problem question (Eqs. 47-49). Note again that ω here is not the wheel velocity as in the



Inputs	*Outputs*
Path radius r_P	Velocity of left wheel v_L
Centre of rotation IC	Velocity of right wheel v_R
Desired velocity v_{ref}	**Velocity of path (robot) v**
Axle length L	
Wheel radius r	

Figure 30: Model inputs and outputs.

problem statement but the angular velocity of the circular path about its IC (in XY).

This set of equations, derived from geometrical constraints, is consistent with other derivations for similar problems including other differential-drive robots following a circular path [5] [6].

(ii) Simulink

The Simulink set-up is shown in Fig. 31. All configuration settings were made equal to those of the lab, including the solver (ODE1 Euler), by creating this model in the same file. Really, any amount of data could have been exported, but because only the velocity was of interest, v_L, v_R, and v were exported to the workspace via logging as a 'Dataset' in Scope settings. For convenience, their reference values were also fed into the Scope, though these are independent (inputs) of the simulation.

Reference values for ω_L and ω_R are created in `load_inputs.m`, where the necessary inputs are specified (Appendix 1). The Simulink model has been set up so that `load_inputs.m` is run every time the .slx is opened, placing .m variables on the workspace. This is done within Simulink: Model Properties > Model Properties. Hence, the Simulink model has been written in terms of named variables, and changes to their values can only be made in the .m file.

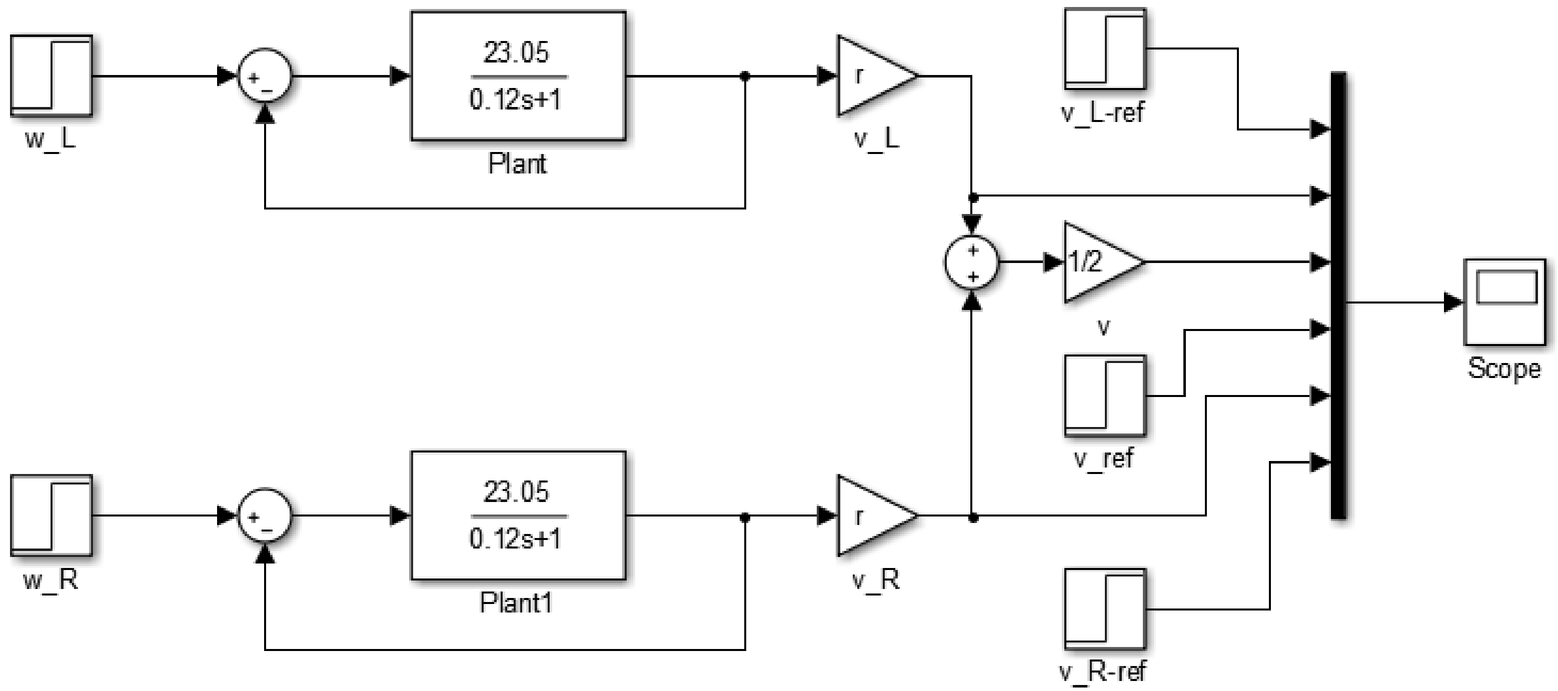


Figure 31: Model setup in Simulink. Data export is done within Scope logging settings.

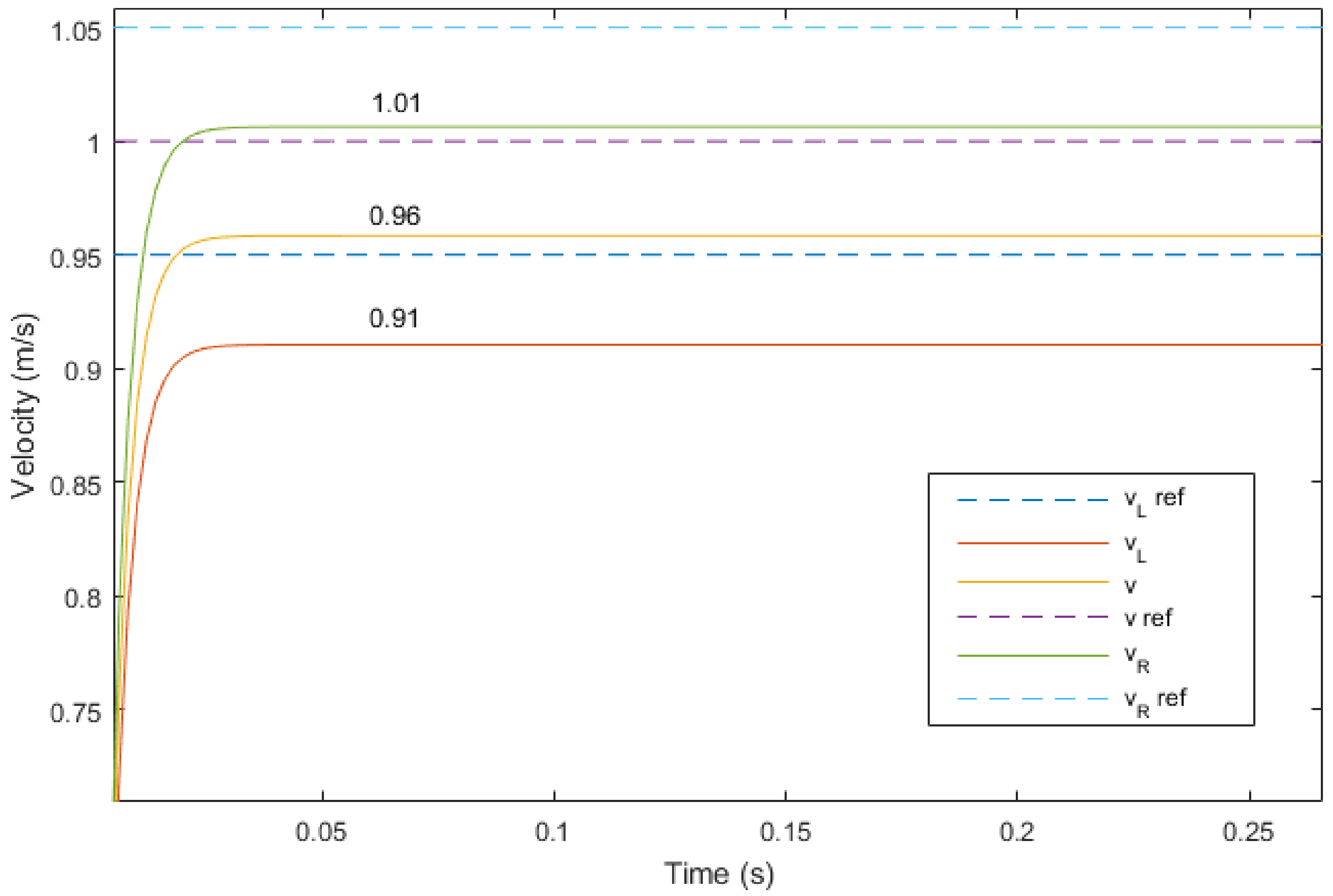


Figure 32: Reference vs. obtained response for left, right, and path velocities using a step input for the angular velocity of each wheel.

(iii) Results

Figure 32 shows the results of the simulation using the motor of the first laboratory, represented as a transfer function. Altogether, the resultant angular velocities of each wheel were less than their reference values, resulting in a lesser linear velocity for each wheel and resultant path velocity. This is attributed to the steady-state error phenomenon that always exists whenever there is proportional gain (and no integrator in the forward direction) [1].

As seen in Fig. 32, the resultant velocities of the left and right wheel are still in proportion of each other. That is, the system has acted to lower their velocities but in the same proportion. Therefore, the path of the robot has been unaffected. The path is only limited in that the robot traverses a lesser part of it given a certain time (Fig. 33). In other words, the geometric $v = (v_L + v_R)/2$ forming a circle holds, however, the final velocity is 0.96 m/s as opposed to 1 m/s.

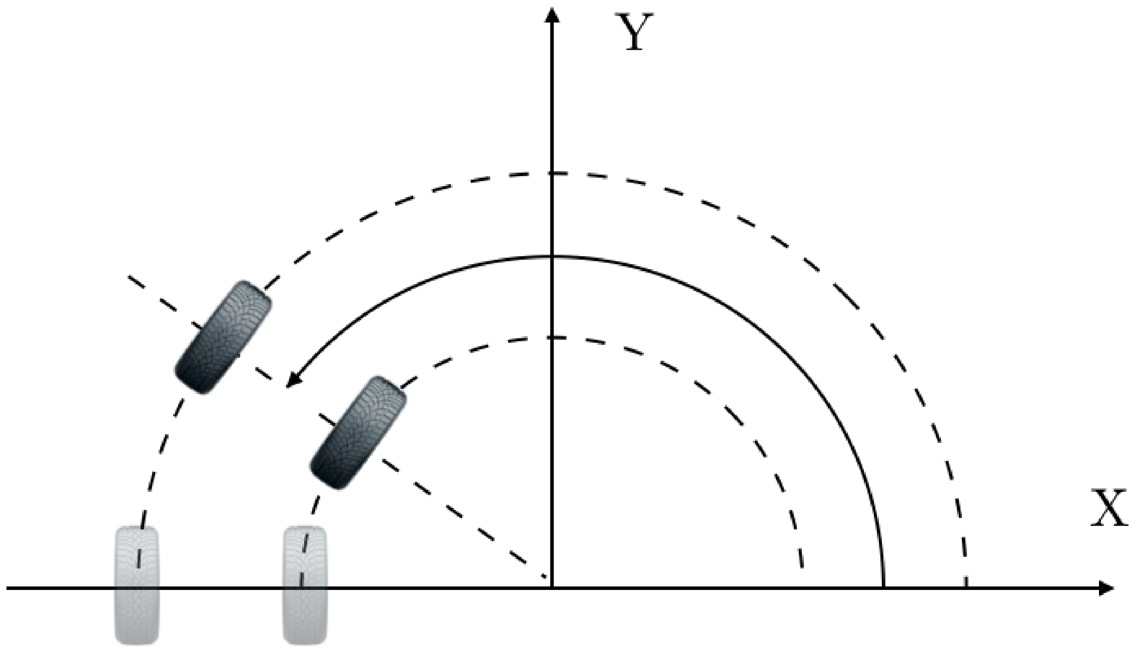


Figure 33: The path is the same, but the velocity is diminished due to steady-state error, so the robot travels less of a distance in the same time.

The situation might be very different if even the slightest disturbance is introduced, if actual (as opposed to simulated) plants were used, or if this simulation included axle steering dynamics. If the velocities of each wheel are not in proportion, the robot will change direction. Here, the system dynamics associated with

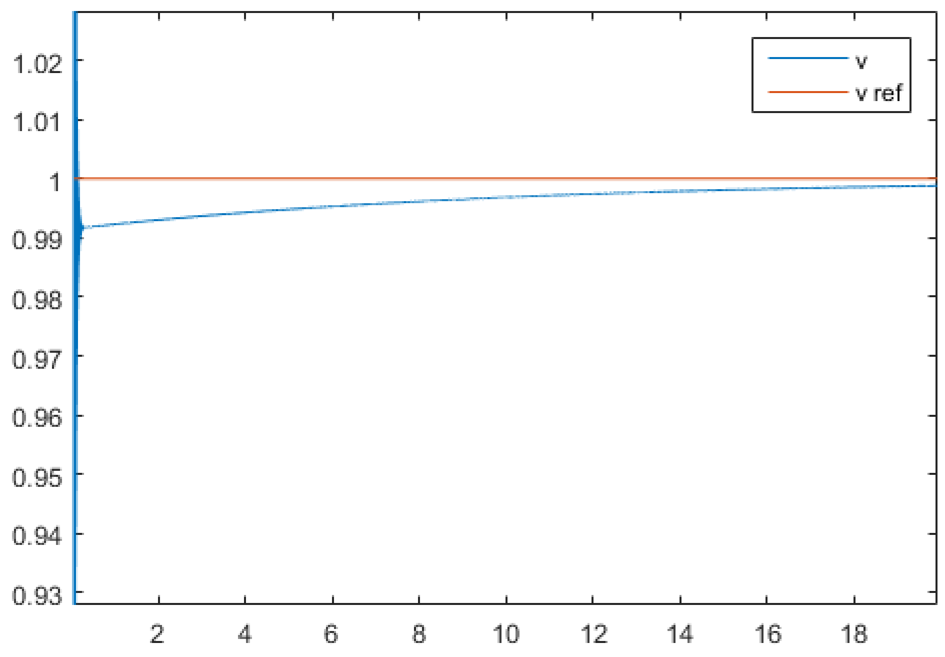


Figure 34: Velocity response with added integral control for each motor ($K_i = 5$).

the steering mechanism has not been analysed. If it is imperfect, inevitably the path direction will also stray.

As a point of discussion, if the desired path was not circular as selected here but instead a winding curve, it is hypothesised that the diminished velocities will in fact have an effect, because of the changing instantaneous centres of rotation. In this problem, the centre of rotation and its rotational radius are fixed.

In this study, accuracy (i.e. not the transient response) can be improved by the addition of an integral controller. This will mitigate steady-state error following the same argument presented in **Section 1i**. To demonstrate, integral control was added to this simulation. While its gain has not been tuned, Fig. 34 shows that it gradually acts to bring the velocities (and thus position per instant in time) closer to the reference value.

Finally, an element of error is added if we are trying to control position via velocity. In the future, a suggestion is to trial a system directly based on position rather than velocity.

(iv) Discussion

In this study, a maximum wheel rotation rate of 5 revolutions/second will not restrict the system in any sense other than that the mean velocity of the robot will be restricted to 1.55 m/s. This is equivalent to placing a saturator on the left and right wheel velocity outputs. For a counter-clockwise turning vehicle, the right wheel will be the limiter while the opposite is true for a clockwise turning vehicle.

It is imagined that if the chosen path were not a circle (one constant centre of rotation and radius), that the wheel velocity restriction would affect the feasibility of proposed paths. Within any path made up of various curve (part-circle) segments, the maximum curve radius must fit so the outward wheel can traverse it at 1.6m/s. This is analogous to the suggestion of ’straighter’ paths.

4 Quarter-car model

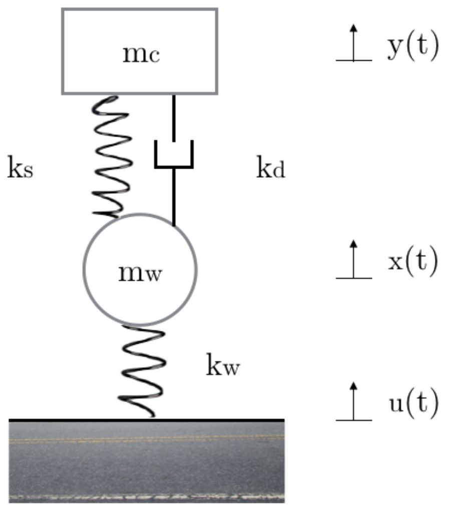


Figure 35: The quarter-car model.

(a) Steady-state expression

In the previous assignment, the following gains were specified for the quarter-car model (Fig. 35): $k_s = 20000$, $k_d = 5000$, $k_w = 180000$. The system was stable, meaning that the natural unforced response gradually dies out (Fig. 36). The steady-state response therefore, is solely caused by the forced input [7]. This applies for a sinusoidal input $u(t) = \sin(\omega t)$ so that:

$$y(t) = M\sin(\omega t + \phi) \tag{56}$$

$$M = |G(j\omega)| = |G(s)|_{s=j\omega} = \sqrt{Re(G(j\omega))^2 + Im(G(j\omega))^2} \tag{57}$$

$$\phi = \angle G(j\omega) = |\angle G(j\omega)|_{s=j\omega} = tan^{-1}\left[\frac{Im(G(j\omega))}{Re(G(j\omega))}\right] \tag{58}$$

Evaluating $G(j\omega)$ yields:

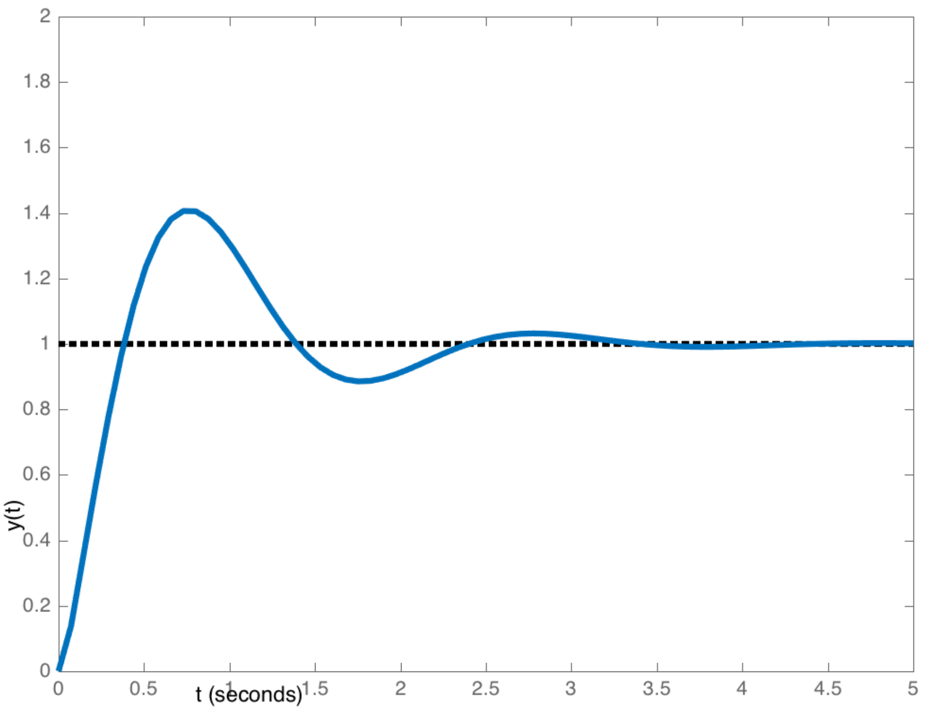


Figure 36: Step response for $k_s = 20000$, $k_d = 5000$, $k_w = 180000$.

$$G(j\omega) = \frac{[k_s k_w] + j[\omega k_d k_w]}{\left[\omega^4 m_c m_w - \omega^2(k_s m_c + k_w m_c + k_s m_w) + k_s k_w\right] + j\left[-\omega^3(k_d m_c + k_d m_w) + k_d k_w\right]}$$

$$= \frac{N_R + jN_I}{D_R + jD_I} = \frac{[N_R D_R + N_I D_I] + j[N_I D_R - N_R D_I]}{D_R^2 + D_I^2} \tag{59}$$

Solving for M:

$$M = \frac{\sqrt{(N_R D_R + N_I D_I)^2 + (N_I D_R - N_R D_I)^2}}{D_R^2 + D_I^2} \tag{60}$$

Solving for ϕ:

$$\phi = \tan^{-1}\left[\frac{N_I D_R - N_R D_I}{N_R D_R + N_I D_I}\right] \tag{61}$$

Therefore, for the sinusoidal input:

$$\lim_{t\to\infty} y(t) = \frac{\sqrt{(N_R D_R + N_I D_I)^2 + (N_I D_R - N_R D_I)^2}}{D_R^2 + D_I^2} \sin\left[(\omega t + \tan^{-1}\left[\frac{N_I D_R - N_R D_I}{N_R D_R + N_I D_I}\right]\right] \tag{62}$$

(b) Bode plot

MATLAB was used to plot the gain and phase of the transfer function for $s = j\omega$ using the above equations (Figs. 37 & 38).

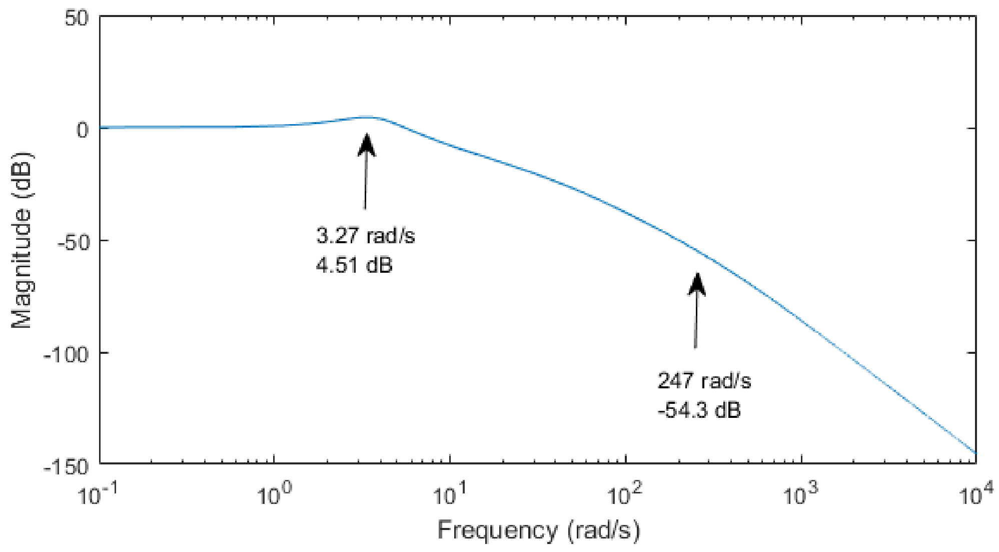


Figure 37: The magnitude generated from Q4b.m. The resonant frequency and corresponding gain is shown. The frequency and gain is also given for $v = 10\text{m/s}$ and $r(x)$ given later.

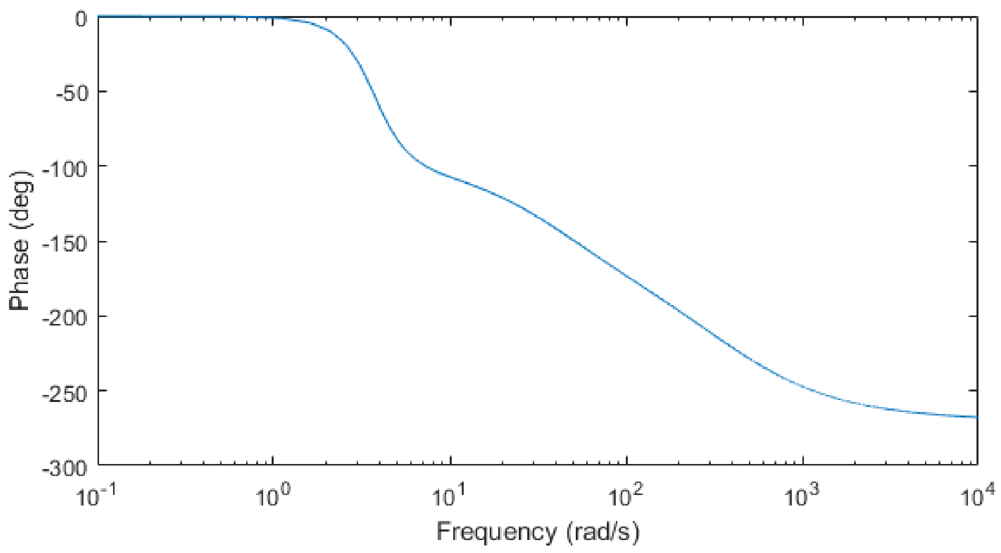


Figure 38: The phase generated from Q4b.m.

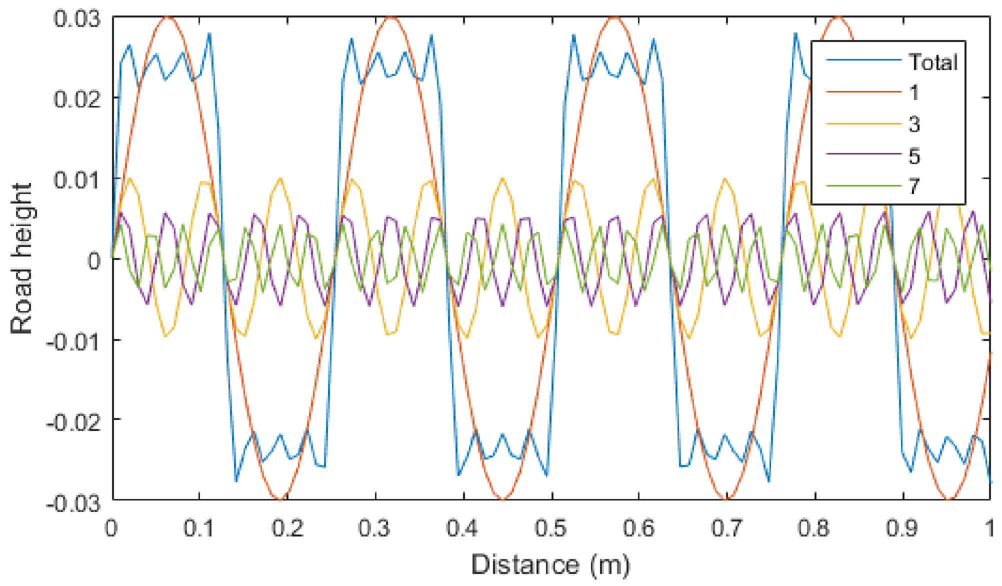


Figure 39: The road profile shown with its constituents ($d = 0.127$). The numbers used as a label are the coefficients within the sin term of the constituent.

(c) Simulation

The height of the road can be modelled as a Fourier series:

$$r(x) = 0.03\left[\sin(\pi x/d) + \frac{1}{3}\sin(3\pi x/d) + \frac{1}{5}\sin(5\pi x/d) + \frac{1}{7}\sin(7\pi x/d)\right] \tag{63}$$

where x is in metres and $d = 0.127$.

Fig. 39 displays the road's profile with its constituents. It is easy to see that the profile is simply a linear sum of its parts. Varying d quickly shows that its effect is to scale the frequency, without changing amplitude, phase, or geometry of the response (Fig. 41).

With this in mind, let us first inspect the geometric contribution of each term. This is best seen when d is adjusted to 0.5 so that one cycle fits within 1m (or 1s for a traversing velocity of 1m/s). Within a cycle, there is always 1x cycle of the '1' term, 3x of the '3' term, 5x of the '5' term, and 7x of the '7' term. Because the patterns always fit within half and full cycles, the geometry is repeated consistently over time. The fact that one cycle (2π rads) fits in 1m when $d = 0.5$ is due to the π/d term within each sin term. It follows that two cycles will fit with

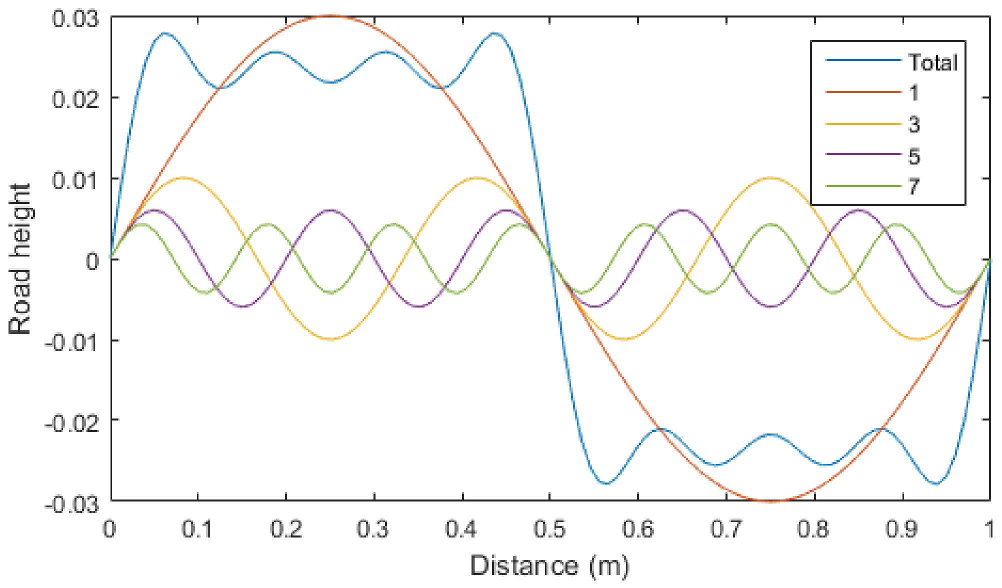


Figure 40: $d = 0.5$

$d = 0.25$.

The effect of each sin() term is scaled to its amplitude, so 1 has the greatest effect and 7 has the least. This is due to the coefficients in front of each sin() term.

As shown in Fig. 41, changing d scales the frequency of the profile and its constituents. However, the amplitude remains the same. The amplitude is consistently 0.03, the global coefficient. Last, there is no phase change varying d; there are no terms that would alter phase.

If the car is driven along the road at 10m/s, the input must be $u(t) = r(10t)$, because $x = vt$ where v is velocity. Analytically:

$$\begin{aligned} y(t) = 0.03[M(\omega_1)(\sin(\omega_1 t + \phi(\omega_1)) + 1/3M(\omega_3)(\sin(\omega_3 t + \phi(\omega_3)) \\ + 1/5M(\omega_5)(\sin(\omega_5 t + \phi(\omega_5)) + 1/7M(\omega_7)(\sin(\omega_7 t + \phi(\omega_7))] \end{aligned} \quad (64)$$

where $\omega_1 = 10\pi/d$, $\omega_3 = 30\pi/d$, $\omega_5 = 50\pi/d$, and $\omega_7 = 70\pi/d$. This system has been simulated in MATLAB, its response for $0 < t < 1$ is shown in Fig. 42. The car height evidently oscillates rapidly within a range of $6 \times 10^{-}5$m.

From the gain plot (Fig. 37), a resonant frequency was found to exist at $\omega = 3.27$ rad/s (where the gain value was a maximum of 4.51dB). The first sin() term

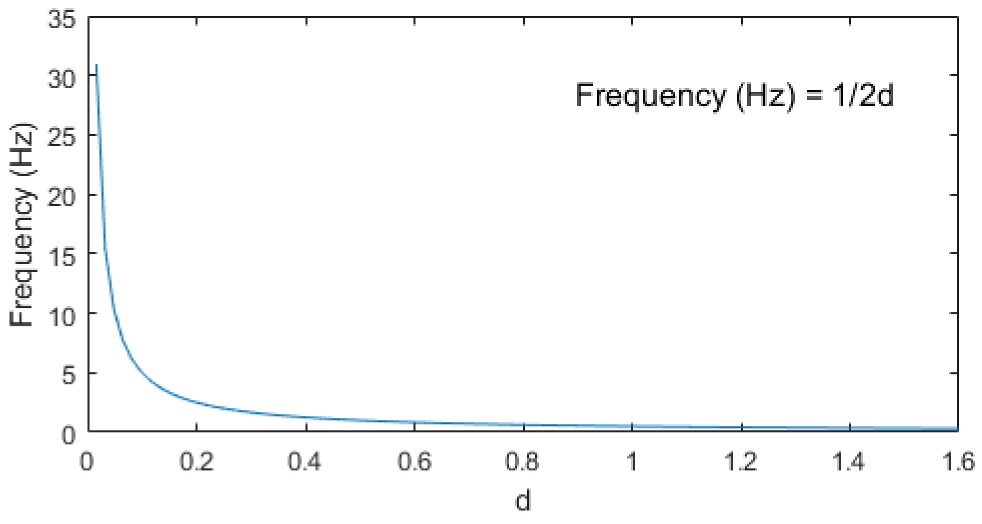


Figure 41: The frequency in cycles per metre (or second for 1m/s) over d.

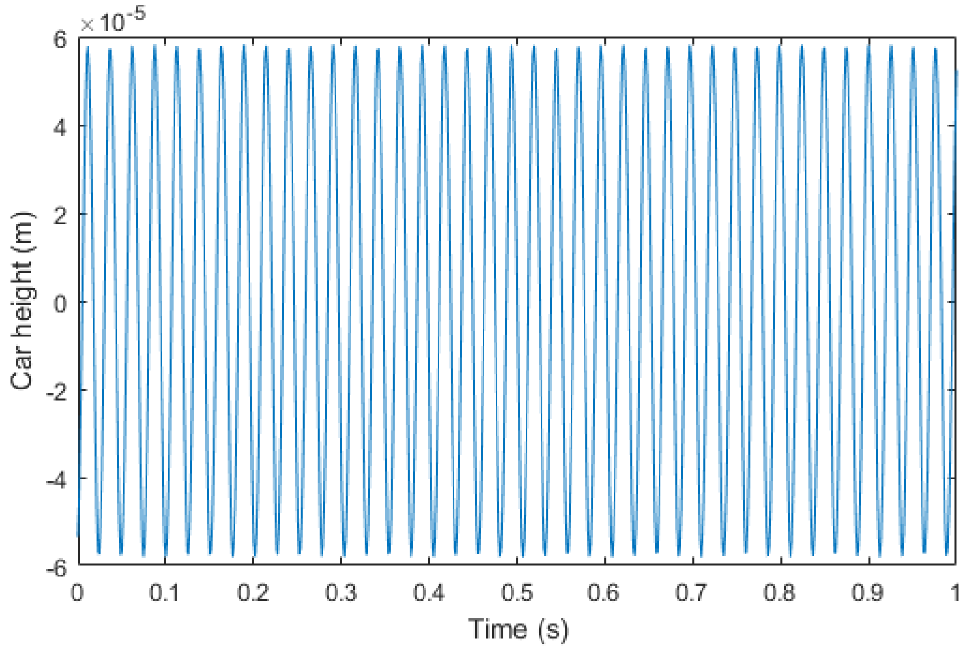


Figure 42: The simulated response.

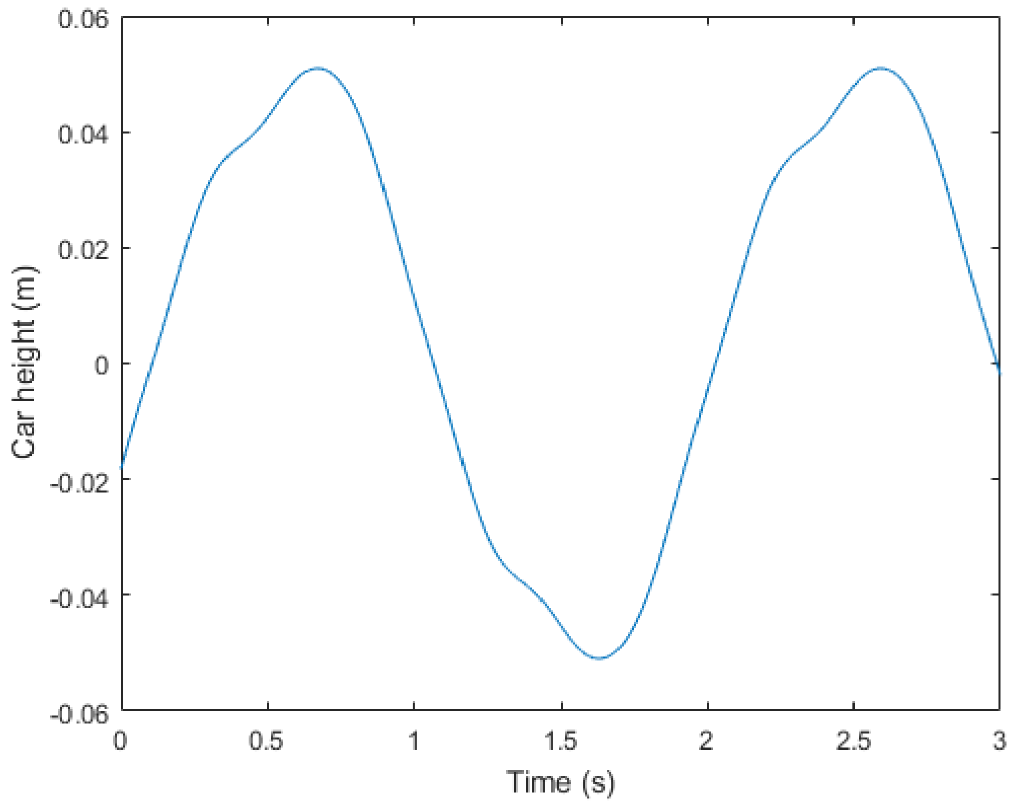


Figure 43: Car height when $v = 0.1322$ ($\omega = 3.27$), the first mode of resonance.

has the greatest contribution to the Fourier series input. Equating the frequency component to 3.27 rad/s gives a velocity of 0.13m/s:

$$v = 3.27/(\pi/0.127) = 0.1322 \text{ m/s} \tag{65}$$

Trialling this and a range of other values (including the other modes of resonant frequency) proved that this velocity value indeed reflected the (first) mode of resonant frequency. It returned the highest amplitude in car height oscillations, giving the most uncomfortable ride. It also has a very long period. As a point of reference, the frequency associated with $v = 10$m/s was 247.4 rad/s, also shown in Fig. 37. Its corresponding gain is -54.3dB, much lower than that of the resonant peak.

References

[1] N. Nise, *Control Systems Engineering*. Wiley, 2011.

[2] E. Britannica, “Human eye.” `http://www.britannica.com/science/human-eye`, 2016. Accessed: 1 May 2016.

[3] K. Åström and T. Hägglund, “Revisiting the ziegler–nichols step response method for {PID} control,” *Journal of Process Control*, vol. 14, no. 6, pp. 635 – 650, 2004.

[4] J. Lumkes, *Control Strategies for Dynamic Systems: Design and Implementation*. Dekker Mechanical Engineering, Taylor & Francis, 2001.

[5] J. Rajput, F. Memon, and M. A. Unar, “Simulation of a differentially-driven vehicle robot in simulinkÂ®,” in *2005 Pakistan Section Multitopic Conference*, pp. 1–6, Dec 2005.

[6] L. Huang, “Speed control of differentially driven wheeled mobile robots—model[U+2010]based adaptive approach,” *Journal of Robotic Systems*, vol. 22, pp. 323–332, 6 2005.

[7] I. Manchester, “Frequency response and bode plots,” 2016.

5 Appendices

(a) MATLAB code

All code also available at `https://github.com/sojung21/amme3500`.

(b) Compliance Statement

(c) Disabilities Adjustment

```
% Q1e. Step response of P system

J = 5;  % [kgm^2]
c = 0.7;    % [Nms]
K = 0.05;

s = tf('s');

G = K/(J*s^2 + c*s + K); % No disturbance

% rlocus(G);
% figure
% stepplot(feedback(G,1));
% Q1i. Step response for PD system

clc; clear;

J = 5;
c = 0.7;
zeta = 0.6901;  % for <5% OS
K_d = 39.3;
K = 167.9832;
omega = sqrt(K/J);
T_s = -log(0.02*sqrt(1-zeta^2)) / (zeta*omega); %
T_s_simple = 4/(zeta*omega);  %

s = tf('s');
G_noW = (K + K_d*s)/(J*s^2 + (c+K_d)*s + K);
stepplot(G_noW, 3);
figure;
pzmap(G_noW);
grid on
```

```
clear; clc;

r = 0.012;  % [m]
m = 0.0075;   % [kg]
J = 2*m*r^2/5;
K_d = 3.05*10^-4;   % [Ns/m]
K_p = 1.65*10^-2; % [N/m]

OS = 5; % [%]
zeta = -log(OS/100)/sqrt(pi^2 + log(OS/100)^2);
phi = acosd(zeta); % [deg]

s = tf('s');

%% UNCOMPENSATED SYSTEM

UC = 100/((s+100)*(J*s^2 + K_d*s + K_p));

% grid on
% stepplot(UC, 0.25);
% figure;
rlocus(UC);
% sgrid(zeta,0)

P1 = 59.03;
P2 = 100;
P3 = 647;
K = 0.0161;
T_s = 0.066;

%% COMPENSATED SYSTEM

%% Reduce T_s by half
sigma = pi/(0.5*T_s);
omega_d = sigma*tand(phi);

theta_P1 = 90 + atand((sigma-P1)/(omega_d)); % Desired dominant pole upper left of pole
theta_P2 = 90 + atand((sigma-P2)/(omega_d)); % Desired dominant pole upper left of pole
theta_P3 = atand(omega_d/(P3-sigma)); % Desired dominant pole upper right of pole

%% Compensating D zero
theta_Z =  theta_P1 + theta_P2 + theta_P3 - 180;
Z = (omega_d/tand(theta_Z)) + sigma;

PD = (s+Z)/((s+100)*(J*s^2 + K_d*s + K_p));
% rlocus(PD);
% sgrid(zeta,0);
K = 0.0119;
CL_PD = feedback(K*PD,1);
% stepplot(CL_PD,0.1);

%% Compensating I zero
I=40;
PID = (s+I)*(s+Z)/(s*(s+100)*(J*s^2 + K_d*s + K_p));
% rlocus(PID);
sgrid(zeta,0);
% sigma = 95.2;
% omega = 99.8;
K = 0.0119;
% figure;
CL_PID = feedback(K*PID,1);
% stepplot(CL_PID);
```

```
%% Calculate gains

PID_K_d = K;
PID_K_p = (Z+I)*PID_K_d;
PID_K_i = (Z*I)*PID_K_d;
```

```
% Q3 Auto-loaded variables for Simulink model

%% ASSUMPTIONS

% Two wheeled-robot
% Negligible mass of robot
% High gear ratio between motor and wheel
% Constant velocity input

%% INPUTS

IC = [0, 0];    % Instantaneous center as an [X, Y] coordinate [m]
r_P = 1;        % Path radius [m]

r = 0.05;       % Wheel radius [m]
L = 0.1;        % Distance between contact points of L & R wheel [m]

v_P = 1;        % Robot (path) velocity [m/s]

%% V_L & V_R

w = v_P/r_P;    % Robot's angular velocity about IC [rad/s]

v_L = w*(r_P-(L/2)); % Linear velocity of wheel closest to IC [m/s]
v_R = w*(r_P+(L/2)); % Linear velocity of wheel furthest from IC [m/s]

w_L = v_L/r;    % Angular velocity of wheel closest to IC [rad/s]
w_R = v_R/r;    % Angular velocity of wheel furthest from IC [rad/s]
```

```
% Q4b. Bode Plot

close all;

omega = 0.1:0.5:10000;

kd = 5000;    % [N/m]
ks = 20000;     % [N/m]
kw = 180000;      % [Ns/m]

mc = 1400;      % [kg]
mw = 12;        % [kg]

G = tf([kw*kd, kw*ks], ...
    [mc*mw, ...
    mw*kd + mc*kd, ...
    mw*ks + mc*kw + mc*ks, ...
    kd*kw, ...
    ks*kw]);

bode(G);
figure;

N_R = ks*kw;
N_I = (kd*kw)*omega;
D_R = mc*mw*(omega.^4) - (ks*mc+kw*mc+ks*mw)*(omega.^2) + ks*kw;
D_I = kd*kw*(omega.^1) - (kd*mc+kd*mw)*(omega.^3);

M = sqrt((N_R.*D_R + N_I.*D_I).^2 + (-N_R.*D_I+N_I.*D_R).^2)./(D_R.^2 + D_I.^2);
phi = atan2((N_I.*D_R-N_R.*D_I),(N_R.*D_R + N_I.*D_I));
phi = 180/pi*unwrap(phi);

semilogx(omega, 20*log10(M));
figure;
semilogx(omega, phi);
```

```
% Q4c. Modelling the road height

clc; clear all; close all;

x = linspace(0,1);
d = 0.127;

r = 0.03 * (...
    sin(pi*x/d)...
    + (1/3)*sin(3*pi*x/d)...
    + (1/5)*sin(5*pi*x/d)...
    + (1/7)*sin(7*pi*x/d)...
    );

a = 0.03 * sin(pi*x/d);
b = 0.03 * (1/3)*sin(3*pi*x/d);
c = 0.03 * (1/5)*sin(5*pi*x/d);
d = 0.03 * (1/7)*sin(7*pi*x/d);

plot(x,r,x,a,x,b,x,c,x,d);
legend('Total','1', '3', '5', '7');
xlabel('Distance (m)');
ylabel('Road height');
```

```
% Q4c. Simulating the model response

d = 0.127;
t = linspace(0,3,2500);   % [s]
v = 0.1322;

y = 0.03*(...
    superimpose_this(v*pi/d, t) +...
    1/3*superimpose_this(v*3*pi/d, t) +...
    1/5*superimpose_this(v*5*pi/d, t) +...
    1/7*superimpose_this(v*7*pi/d, t));

plot(t,y);
xlabel('Time (s)');
ylabel('Car height (m)');
```

```
% Q4c. Function to calculate y(t) contributions as a function of omega

function [this] = superimpose_this(omega_expression, time_vector)

omega = omega_expression;
t = time_vector;

kd = 5000;    % [N/m]
ks = 20000;     % [N/m]
kw = 180000;      % [Ns/m]

mc = 1400;      % [kg]
mw = 12;        % [kg]

N_R = ks*kw;
N_I = (kd*kw)*omega;
D_R = mc*mw*(omega.^4) - (ks*mc+kw*mc+ks*mw)*(omega.^2) + ks*kw;
D_I = kd*kw*(omega.^1) - (kd*mc+kd*mw)*(omega.^3);

M = sqrt((N_R.*D_R + N_I.*D_I).^2 + (-N_R.*D_I+N_I.*D_R).^2)./(D_R.^2 + D_I.^2);
phi = atan2((N_I.*D_R-N_R.*D_I),(N_R.*D_R + N_I.*D_I));
phi = 180/pi*unwrap(phi);

this = M*sin(omega*t + phi);
```

The University of Sydney

STUDENT PLAGIARISM: COURSE WORK - POLICY AND PROCEDURE

COMPLIANCE STATEMENT

INDIVIDUAL / COLLABORATIVE WORK

I/We certify that:

(1) I/We have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*;

(2) I/We understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to the University commencing proceedings against me/us for potential student misconduct under Chapter 8 of the *University of Sydney By-Law 1999* (as amended);

(3) this Work is substantially my/our own, and to the extent that any part of this Work is not my/our own I/we have indicated that it is not my/our own by Acknowledging the Source of that part or those parts of the Work.

Name(s): SOO JENG TAN

Signature(s): [illegible]

Date:

6 APRIL 2016