Control of a 2 Degree of Freedom Robot Arm for Drawing Simple Shapes

By

William Sakyi, Ibrahim Arekat and Wishawin Lertnawapan

Department of Mechanical Engineering

UNIVERSITY OF BRISTOL

Abstract

A robotic device was commissioned to perform the task of drawing a circle ($r = 49$ *mm*), square ($s = 70$ *mm*) and triangle (110 *mm*) using a Raspberry Pi Pico as the microcontroller. The objective was to maximise the accuracy and speed of the drawings given the available electronic hardware. The robot is comprised of 2 limbs, actuated by motors fixed to each joint, and the free end of the second limb holds a pen. PID control was implemented for each motor due to its stability, versatility and simplicity. The control of the robot was improved through experimental gain tuning, implementing a velocity profile, mitigating integral windup and applying filters. The largest source of inaccuracy during the performance of the drawing task was the significant change in acceleration when the robot arm turns at corners. Thus, the velocity profile implementation was found to be the most impactful control method, with an 88.7% increase in error observed when it is removed. The final drawings of the circle and triangle were highly accurate with average absolute errors of 1.95 degrees and 1.82 degrees respectively. The low error achieved in the circle was through the use of a constant speed during its drawing, enabled by the lack or corners. For the triangle, the relatively long side lengths of 110 *mm* provided the robot dynamics sufficient time to stabilise. However, the square drawing had a comparatively higher average absolute error of 5.4 degrees. This appears to be due to the short time period available to stabilise after turning a corner before the next comes due to the relatively short 70 *mm* side length. Overall, the drawings were considered to be of good accuracy, as they all fall within the backlash envelope. The three shapes achieved similar drawing times, with an average of 4.43 *s*. For potential future performance improvements, more advanced control methods were simulated for the case of a 2-stage mass-spring linear actuator with a known plant to investigate their implementation, along with their advantages and challenges. For example, the multivariable control method, output feedback control (OFC), was found to be highly effective as, compared to PID control, it posed faster response times, reduced instability and completely eliminated steady state error. However, it must be emphasised that the plant should be well-defined to achieve effective results using the investigated methods.

December 3, 2024

1 Introduction

The aim of this project is to design, build and commission a robotic device to draw a circle, square and triangle. The circle is specified to have a radius of 49 *mm*. The square will have a side length of 70 *mm*. The triangle is equilateral and has a side length of 110 *mm*. The 2 main criteria that define the performance of the robot are the accuracy of the shapes drawn and the time taken to complete the drawings.

The fundamental design of the robot was a 2 degree of freedom robot that has 2 independently controlled limbs, with the end limb holding a pen, that move in the *x*-*y* plane. Overall, the performance of the robot was improved through cycles of operating the robot, identifying areas of improvement and applying changes to the control or physical structure accordingly. Figure 1 shows the kinematics of the robot.

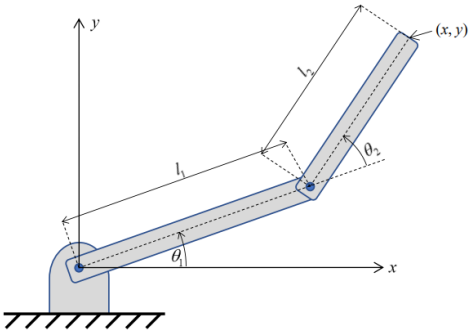


Figure 1: Fundamental kinematics of the 2 degree of freedom robot.

The forward kinematics are defined in Equation 1 as:

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2),\ y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \tag{1}$$

and the inverse kinematics in the *x*-*y* plane are defined as [1]:

$$cos\ \theta_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}, \tag{2}$$

$$sin\ \theta_2 = \sqrt{1 - cos^2 \theta_2}, \tag{3}$$

$$\theta_2 = tan^{-1}\left(\frac{sin\ \theta_2}{cos\ \theta_2}\right), \tag{4}$$

and let:

$$A = x, B = l_1 + l_2\ cos\ \theta_2, C = -l_2\ sin\theta_2, \tag{5}$$

then for $y > 0$:

$$w = \frac{C + \sqrt{C^2 + B^2 - A^2}}{A + B}, \tag{6}$$

$$\theta_1 = 2\ tan^{-1}\ w \tag{7}$$

2 Method

2.1 Electronics

The electronics enable the control and actuation of the robot. A Raspberry Pi Pico serves as the central microcontroller, running the control algorithms and interfacing with other components via a breadboard. An L298N motor driver acts as an intermediary, translating control signals into the appropriate directions and

power levels required to drive the motors. The DC geared motors are equipped with encoders that provide a resolution of 5.3 encoder counts per degree, enabling precise feedback for controlling motor speeds and positions. All components were connected according to the schematic provided in the lab notes [1].

2.2 Physical Structure

The initial structure built to support the robot arm, shown below in Figure 2, was based on the reference robot structure[1]. The first step completed to improve the structure was identifying the role of the individual components and how their current versions affected the robot performance in the drawing task. Thus, after multiple trials with the initial structure, areas of improvement were identified and modifications were made accordingly. The final structure is shown in Figure 3.

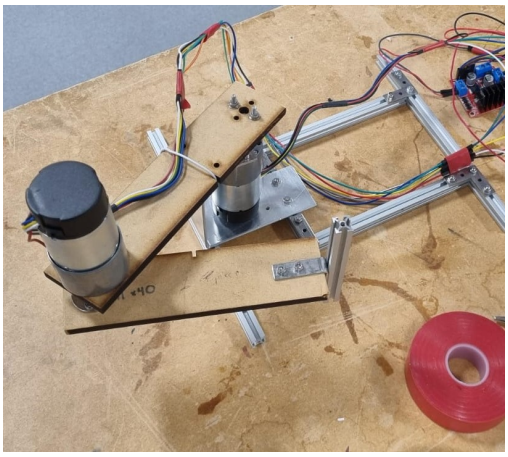


Figure 2: Initial physical structure based on the reference structure.

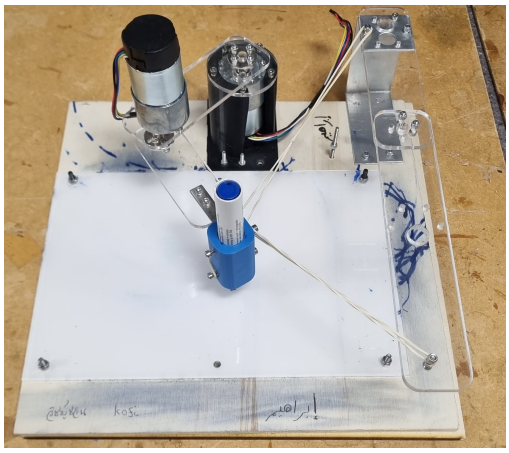


Figure 3: Improved physical structure with modified components.

Base: The purposes of the base are to secure the position of the motor holder, minimise vibrations and prevent any unwanted movement from occurring whilst the robot is completing its task. Therefore, the ideal base is rigid and heavy enough to remain secure in its position and experience minimal vibrations and is completely level. The initial base was not heavy enough and would tip over due to the weight of the motors and arms, necessitating group members to hold it in place. The new base is a 12mm thick plywood board. Holes were power drilled in the appropriate locations, enabling other components to be securely fixed using screws and nuts. The board is flat and heavy enough to experience negligible levels of vibration and no movement during the operation of the robot.

Motor holder: The motor holder must secure the position of the motor and restrict its movement apart from shaft rotation. It must be rigid enough to not bend due to the weight of the motors and arms. The initial motor holder allowed the motor to be securely fastened using screws, however its upright section was insufficiently thick and would slightly bend due to the weight of the motors and arms. The degree of bending varied depending on the positions of the arms, meaning the pen height was not fixed and this caused inaccuracies in the drawings. The new motor holder is a 3D-printed vertical hollow cylinder. The top of the cylindrical wall has holes with threaded inserts allowing a cover to be attached to the top of the holder using screws. The cover has holes aligning with those of the motor, allowing the motor position to be secured. The height of the cylinder is similar to the height of the motor minus the shaft, allowing the motor to be placed at the base of the motor holder, whilst being attached to the cover. Thus, the weight of the motor is supported by the base, reducing vibrations and stress in the motor.

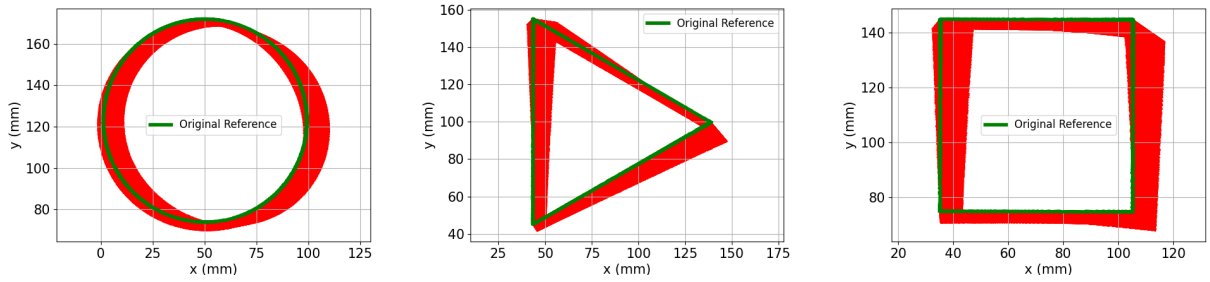
Arms: The design of the arms should allow for tight fastening to the motor shafts, as any loose joints may lead to increased vibrations. The material must be rigid and lightweight to prevent bending and minimise in-

ertia. Higher weight and inertia may reduce speed and make it more difficult for the robot to move smoothly, thus necessitating more challenging control to achieve faster, higher accuracy drawings. The initial wooden arms were lightweight, but their screws loosened through repeated operation of the robot over numerous trials. This was due to the degree of tightening of the screws being limited by the susceptibility of wood to localised compression caused by its relatively low hardness. The new arms are similar in geometry to the initial arms; however, acrylic was used as the higher hardness compared to wood allowed the screws to be screwed much tighter without damaging the material.

Pen holder: The pen holder must hold the pen securely and ensure it is perfectly perpendicular to the drawing surface. It must allow the pen to consistently be held at the appropriate height and contact the drawing surface with moderate pressure to avoid excessive friction whilst remaining in contact with the surface. The initial pen-holding mechanism was simply taping the pen to a short vertical beam screwed to the free end of the robot arm. The new pen holder consists of a 3D-printed hollow cylinder with multiple holes with threaded inserts perpendicular to the cylinder thickness. It is screwed to a bracket fixed to the second arm. Threaded holes are on opposite sides of the cylinder to allow screws to clamp the pen by squeezing it tightly. Thus, adjusting the height of the pen is possible by holding the pen in the holder at the desired height and tightening the screws.

Drawing surface: The position of the drawing surface must be completely fixed to prevent distortions in the drawn shapes. The surface must not cause excessive friction against the pen. Individual sheets of paper were used initially. The friction between the pen and the paper was acceptable, however group members had to hold the paper in place to ensure it remains unmoved as the robot carried out the task. A 6mm thick acrylic board was laser cut and fixed directly to the base of the structure using screws and cuts to be used as a whiteboard. This ensures a smooth, flat surface with no movement during the completion of the drawing task.

Backlash control: The backlash per motor due to the gearbox is 3.2 degrees. Figure 4 shows the envelope of error the shapes can fall into due to the backlash. This was computed by calculating the 4 different combinations of backlash of the two arms. This shows there is a large potential for uncertainty. For example, the maximum error possible for the top right corner of the square is 14.33 *mm*, 20% of its side length. Rubber bands were added to counter backlash and ensure the drawn shapes had the intended side lengths or diameter.



(a) Circle. (b) Triangle. (c) Square.

Figure 4: Figures Highlighting the backlash envelopes for each shape.

2.3 Control

Feedback control was used on the robot to draw the required shapes. This allows the controller to adjust its input based on the current error information that is being fed back to the controller. A reference input signal is fed into the controller and an error value is then calculated based on the reference signal and the current position signal. A reference signal was calculated for each motor and PID control was implemented for each.

To effectively control the robot a `Motor` class was created to be used. This class contained methods such as

'`void set_error(int ref)`' which sets the current error based on the current reference and current encoder count position.

Multiple control techniques were used to improve the accuracy of the shapes drawn. They included: PID control and system identification, velocity profile in the reference signal, nonlinear control of the integral gain to prevent wind up and input filtering to remove high frequency content for smoother control.

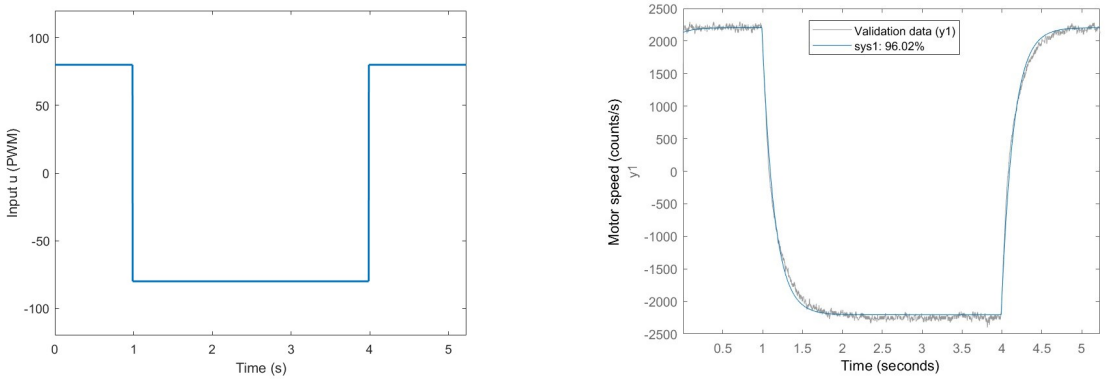
2.3.1 PID Control

PID was used due to its stability, versatility and relative simplicity. PID control consists of a proportional, integral and derivative components that are summed together to generate an input value. The input $u(t)$ is a pulse-width modulation (PWM) between -255 and 255. This corresponds to and is proportional to the input voltage of $\pm$ 12 V that is sent to the motor. Each motor is controlled by PID in each cycle of the Raspberry Pi and is therefore in discrete time. The cycle time of the Raspberry Pi was not a constant value, so to maintain consistent control a regulator was implemented in the software so that the time between the previous and next time steps was consistent. The discrete controller equation is defined as [1]:

$$u(t) = k_p e(t) + k_i \sum_{j=0}^{t} e(j)\Delta + k_d \frac{e(t) - e(t-1)}{\Delta}, \tag{8}$$

where k_p, k_i and k_d are the proportional,integral and derivative controller gains, $\Delta = 2000\ \mu s$ is the sampling interval, $e(t)$ is the error at time t.

Each of the three terms contribute to the input in different ways to ensure fast, smooth and accurate control. The proportional gain is multiplied directly by the current error to provide immediate correction. If the error is large the input value is increased to reduce it. However, proportional control alone is prone to steady state errors as a non-zero error is required to drive it so there will always be an error present. The integral error in Equation 8 accumulates over time, allowing the integrator to address persistent errors by increasing its value. This helps eliminate steady-state errors. However, the integrator introduces lag as it reacts to error accumulated over time. Finally, derivative control reacts to the rate of change of the error. If the error changes rapidly the derivative term will be large to reduce the error before it gets any larger acting as an artificial damper that smooths out the rapid changes and hence improves stability and reduces overshoot. The derivative term however, is more sensitive to noise which may falsely result in an erratic response.



(a) Input. (b) Output.

Figure 5: System Identification.

2.3.2 System Identification

To determine the appropriate gain values system identification was conducted. This was done by trying to identify the open loop transfer function. A step input of 80 counts, which equates to 3.8 V, that reversed in

polarity every 3 seconds was fed into the motor 1. The output motor velocity was derived from the recorded position data. The system identification toolbox in MATLAB was used to identify the first order transfer function shown in Equation 9:

$$G_{p_{dot}}(s) = \frac{\dot{\theta}(s)}{V(s)} = \frac{s\theta(s)}{V(s)} = \frac{\frac{\lambda}{T}}{s+\frac{1}{T}}\,, \tag{9}$$

where λ is the low frequency gain and T is the time constant. Figure 5 shows the input signal and the velocity response to this signal and the fitted system. From this, $\lambda = 27.6$ and $T = 0.15s$. This results in a plant transfer $G_p(s)$ as shown in Equation 10:

$$G_p(s) = \frac{G_{p_{dot}}}{s} = \frac{182.3}{s(s+6.6)} \tag{10}$$

Comparing the closed loop characteristic equation (CLCE) to the coincident roots $3^{(}rd)$-order CLCE, for a settling time $T_S = 0.35\ s$ results in gain values of:

$$k_p = 7.66;\ k_i = 55.07;\ k_d = 0.32 \tag{11}$$

2.3.3 Velocity Profile

The square and triangle contain sharp corners. If the drawing speed is constant, the acceleration at these corners will theoretically be infinite due to the instantaneous change in direction. This results in large amounts of jerk, reducing the accuracy of the control. To address this, a velocity profile was implemented for each side of the reference signal for the triangle and square, to enable smoother turning at corners. The profile accelerates from 0 to a maximum velocity, maintains the maximum velocity for a set period, and then decelerates back to 0. The maximum velocity and acceleration were defined as 120 mm/s and 200 mm/s^2 respectively.

2.3.4 Mitigating Integral Windup

When the error large enough (33 counts for proportional only control with the equation 11 k_p value) that the control signal is saturated (i.e. $|u| \geq 255$), the error integral will increase and wind up while having no effect on the input signal. This causes overshoots. To combat this once the input was saturated the sum of the error was set to 0 to "reset" the integrator. The logic used in the C code to implement this is shown below:

```
void set_error_sum() {
    if (abs(e * kp) + abs(kd * (e - e_prev)/delta_time_seconds) > 255) {
        // Sets integral to 0 when input is saturated
        e_sum = 0;
    } else {
        e_sum += e;
    }
}
```

2.3.5 Filtering

The input signal was filtered using a low-pass filter in software this is shown in this in Equation 12 [1]:

$$u_f(i) = \alpha \times u_f(i-1) + (1-\alpha) \times u(i)\,, \tag{12}$$

where $u_f(i)$ is the filtered version of u, and the constant $\alpha = 1 - \Delta \times \omega_f = 0.86$, where $\omega_f = 70\ rad/s$ and $\Delta = 2000\ \mu s$. This removes high-frequency content from the input signal. The effect of this is smoother control and less wear on the motors and reduction in heat.

2.3.6 Experimental Gain Tuning

The gains from the system identification were improved through real life testing by adjusting the gains and observing the resultant effects this had on the control. This was done systematically by adjusting each of the gain values one at a time and monitoring the improvements. It was found that the most suitable gain values were:

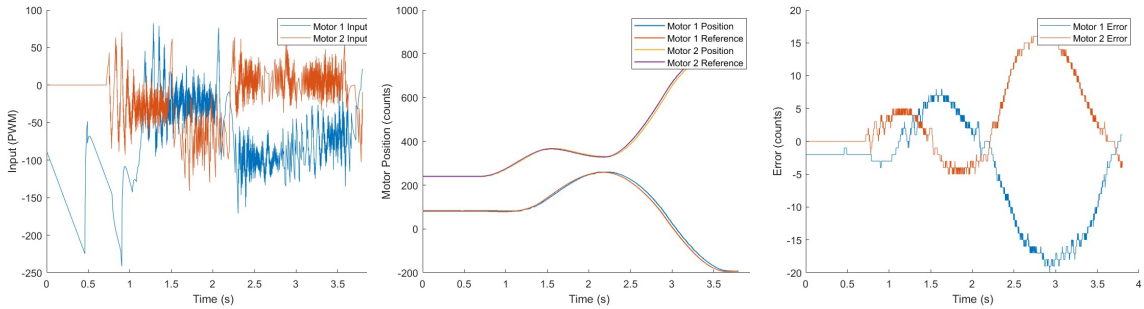
$$kp = 30.66; k_i = 150.10; k_d = 0.89 \tag{13}$$

3 Results

Figure 6 shows the most accurate shapes drawn. Figure 7 shows the plots of the input signal, reference vs position and the error for the triangle drawn in Figure 6b. Table 1 contains the key metrics defining the performance of the robot when drawing each shape. The physical structure of the robot for these shapes was the one shown in Figure 3. PID control was used on each respective arms with the experimental gain values defined in equation 13. An acceleration profile, filtering and non-linear control of the integrator were all used as well. The triangle had the lowest error whereas the square had the largest error. The all were completed with the same speed with the difference in time-taken due to the differing perimeters of each shape.

(a) Circle. (b) Triangle. (c) Square.

Figure 6: Images of the best shapes drawn by the robot

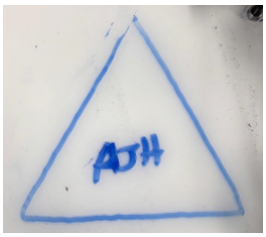


(a) Input. (b) Reference vs Position. (c) Error.

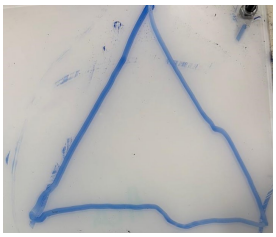
Figure 7: Plots of data collected from the microcontroller for triangle drawing.

Table 1: Key Performance Metrics for the most accurate shapes.

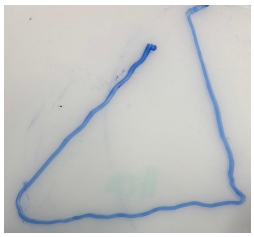
Shape	Absolute Mean Error (counts)	Time Taken (s)
Circle	10.31	4.00
Triangle	9.62	4.56
Square	16.06	4.74



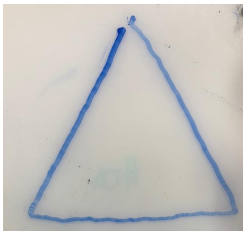
(a) Datum Triangle.



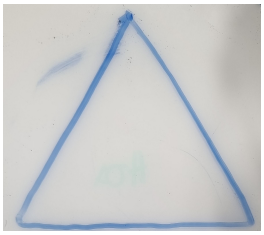
(b) System ID Gains.



(c) No Velocity Profile.



(d) No filter.



(e) No Backlash control.

Figure 8: Images of the different triangles with the respective control methods removed.

To quantify the effect of each control method in improving the accuracy of the system, the error value was recorded for the drawing of a triangle with all the methods included. Each method was then omitted from the control one at a time and the error was recorded again. Table 2 shows the average error when each method is omitted from the control. This shows that each method had a positive effect on the control. Adding a velocity profile had the most significant impact.

Table 2: The effects of each control method

Control Method	Motor 1 Average Error	Motor 2 Average Error	Percentage increase (%)
All control methods included (Datum)	8.12	9.12	0.0
System ID Gain Values	8.26	10.49	8.76
No Velocity Profile	14.35	18.19	88.74
No filter	9.17	11.58	18.51
No backlash control	8.14	11.11	11.03

4 Discussion

The final shapes the robot was able to draw had relatively high accuracy. The high accuracy of the circle is mainly due to the simplicity of the shape as it has no corners, thus the speed can remain constant throughout, minimising errors due to speed changes. The triangle also had a high accuracy, likely due to the longer sides providing the robot more time to stabilise whilst drawing. The changes in motor angles were consistent throughout each side length. The square was more inaccurate compared to the circle and triangle. This appears to be because of the 4 corners the robot has to navigate with short periods to stabilise due to the relatively short side length.

The speeds of the shape drawings are relatively slow. Speed of drawing completion was a performance criteria, however, during experimentation increasing the speed resulted in more erroneous shapes. Thus, it was decided to maintain a speed of $120mm/s$, due to the accuracy it enables.

All the shapes fall into the backlash envelope shown in Figure 4. This highlights the effectiveness of the various control methods and the quality of the physical structure. Including a velocity profile had the largest benefit given the amount of uncertainty the high accelerations at the corners added. Controlling the integrator

non-linearly also had a benefit, as the system is dynamic and the reference is continually changing at every time step and due to lag, this means that even at points when the controller is doing the most it can to reduce the error and match the reference, the integrator is always building up, leaving the input saturated constantly and thus the control becomes effectively bang-bang control. Experimental tuning of the gain values also improved accuracy by 8.76% showing that doing system identification using a single input value of 80 (Figure 5) is not representative and hints at the requirement of gain scheduling to have more effective gain values. An attempt was made to reduce backlash by applying a spring force, via a rubber band, to the pen holder to ensure the motors consistently remain engaged in one direction. This reduced backlash but did not suppress it completely.

5 Future Work

To improve the accuracy and speed of the robot various methods were considered. For example, higher quality motors that had a greater encoder resolution and less backlash. Multivariable and nonlinear control methods were also considered.

5.1 Gain scheduling

The system could benefit from gain scheduling, a nonlinear control technique. Gain scheduling adjusts the controller's gains based on a predefined variable, which, for the shape-drawing robot, would be the motor speed. System identification would be performed for different open-loop step input signals to determine the corresponding gain values. These gains would be stored in a lookup table, allowing optimal controllers to be designed for each speed. During the drawing task, the appropriate controller would be applied at each stage, enabling the control system to adapt to changes in the robot's dynamics. To further improve performance, the gains should be interpolated between the values assigned to the predefined stages, creating a continuous variation of gain values. This approach ensures smooth transitions and consistent control as the robot arm dynamics vary throughout the operation.

5.2 Theoretical Multivariable study

The shape-drawing robot is a 2-degree-of-freedom system with coupled degrees of freedom, suggesting it could benefit from multivariable control. However, due to the computational limitations of the Raspberry Pi, multivariable control was not directly implemented in the robot. Instead, a theoretical study was conducted to explore the application of multivariable and nonlinear control strategies in the context of a simplified two-stage linear actuator. These advanced control strategies were compared to a traditional setup using PID controllers. The simulations were carried out in Python, utilizing discrete-time models for each controller architecture and computing the system response at each time step, with $dt = 0.01$.

5.2.1 System State Mapping

The system to formulate consists of two motors which control each stage of a mass-spring linear actuator arm, forming the basis of the input control signal. No direct damping components were involved, however, the effects of gear backlash and actuator arm friction were considered and used to demonstrate the controller performance in the presence of nonlinearity. The configuration of the system state space can be expressed as a function of motor torques u_1 and u_2 for motors 1 and 2 respectively. It is assumed that high precision sensors such as encoders are available; measuring the positional change for all given instants. Mathematically, a complete dynamical system and its control output can be expressed as

$$\begin{cases} \dot{x} = \mathbf{A}x + \mathbf{B}u \\ y = \mathbf{C}x \end{cases} \tag{14}$$

where $\mathbf{A}$ is the plant matrix, $\mathbf{B}$ is the control input matrix to be defined and $\mathbf{C}$ is the output matrix.

5.2.2 Dynamic Representation

To define A, the underlying equation of motion can be written as an extension to the conventional mass-spring damper system as shown in Equation in 15:

$$\mathbf{M}\ddot{x} + \mathbf{K}x = u_c + u_f + u_b, \tag{15}$$

where $\mathbf{M}$ is the mass matrix for the linear actuator, with its stiffness matrix as $\mathbf{K}$. Force components include the control torque u, with nonlinear components u_f and u_b. These nonlinearities were introduced to simulate the effects of feedback linearisation implementation, and were represented as a simple piecewise function using a dead zone parameter ε:

$$\begin{cases} u_b = u \text{ for } |u| > \varepsilon \\ u_b = 0 \text{ otherwise} \end{cases}, \tag{16}$$

where ε denotes a dead zone threshold value, influencing the actual value of the input u_b. Nonlinear Coulomb friction was also implemented, expressed using Equation 17:

$$u_f = \text{sign}(\dot{x}) \times \mu \tag{17}$$

where μ is the coefficient of friction of the actuator arm. Note the Coulomb formulation is a function of velocity, effectively being considered a damping term. For simplicity, the masses and stiffnesses of the linear actuators were assumed to be identical, using the following values: $m = 5kg$, $k = 2Nm^{-1}$, $\varepsilon = 0.02$, $\mu = 0.5$.

5.2.3 Reference Generation

The reference signal was generated using Python code for each segment of the linear actuator connected in series such that motor 1 drives the actuator by a distance x_1, until the arm reaches a critical length x_1^*. At this point, the control signal is switched to motor 2, positioned at the end of the first actuator, which acts until the second threshold length x_2^*. The total extension can be expressed as a linear combination of both lengths, resulting in a total displacement at the free end of $x_1 + x_2$.; with x_1^* defined as 1 m, and x_2^* as 1.5, as shown in Figure 9.

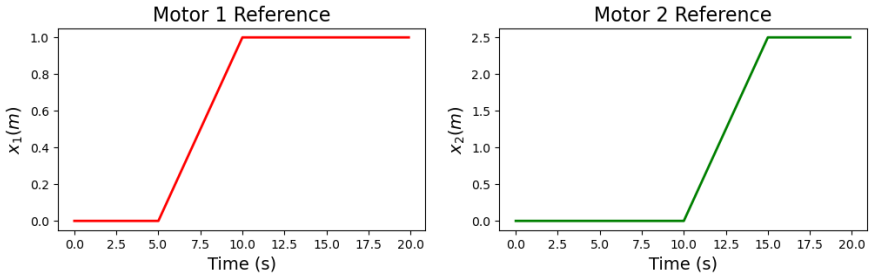


Figure 9: Motor reference signals

5.2.4 Feedback Linearisation

Feedback linearisation was implemented into both controller architectures. Recalling the RHS of Equation 15, the sources of nonlinearity come from the terms u_f and u_b. To perform linearisation, the feedback signal was directly manipulated and scaled using the general formulation shown in Equation 18[2]

$$u_c + (u_f + u_b) = \lambda v + f_\alpha(x) \tag{18}$$

such that the control signal is effectively decomposed into a linear component and nonlinear components where the new control input is defined by state vector v. For this linearity to hold, a candidate function using

the systems' original dynamics was applied. Following some calculations,

$$\begin{cases} f_\alpha(x) = mv - kx + u_f + u_b \\ \lambda = m \end{cases}, \quad \text{for } v = \ddot{x} \tag{19}$$

the new linearised control signal v is rewritten as

$$v = \frac{(kx + -u_f - u_b) + u_c}{m} \tag{20}$$

The effects of the nonlinearities and feedback linearisation are illustrated in Figure 10 using the plant dynamics and the prescribed reference signal. As shown, the plant dynamics in the presence of nonlinearities

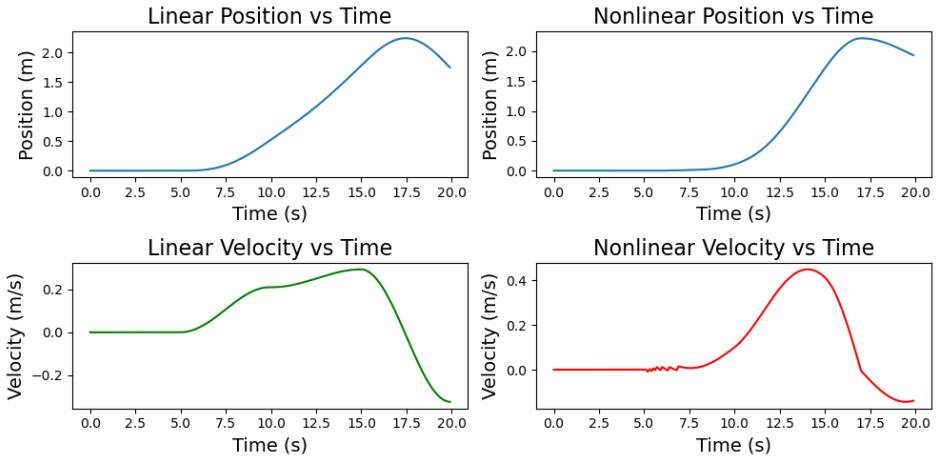


Figure 10: Nonlinearity effect on plant dynamics

creates delayed responses and apparent damping effects by the opposing friction, leading to poor tracking of the reference signal, and highlighting challenges for controllability.

5.2.5 Independent PID Control

PID control was employed to demonstrate SISO control for each motor, which required the plant matrix A to be decoupled as two independent subsystems, with state spaces

$$x_i = (x_i \; \dot{x}_i)^T, u_i = u_i \tag{21}$$

for each motor i. This leads to the expanded matrix representation in Equation 22

$$\begin{bmatrix} \dot{x}_i \\ \ddot{x}_i \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k_{eff}+k}{m} & -\frac{u_{eff}+c_{eff}}{m} \end{bmatrix} \begin{bmatrix} x_i \\ \dot{x}_i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u_i \tag{22}$$

where k_{eff} and u_{eff} represent the equivalent effects of nonlinearity and c_{eff} the effective damping relationship caused by Coulomb friction. The control output matrix is then $\mathbf{C} = [1 \;\; 0]$. Calculating the controllability matrix for each of the discrete-time invariant systems was computed through rank computation linear combinations of the input and plant matrices $\mathscr{C}_i = [\mathbf{B}_i \;\; \mathbf{A}_i\mathbf{B}_i]$ showed full rank, ensuring the system states are all explicitly reachable by available inputs. Both systems were also confirmed for observability for proper performance. - i.e. $\mathscr{O} = \begin{bmatrix} \mathbf{C}_i & \mathbf{C}_i\mathbf{A}_i \end{bmatrix}^T$ [3].

When incorporating feedback linearisation, the PID controller from Equation 8 can be augmented such that the virtual control input cancels nonlinearities.

$$v_i = (mv - kx + u_f + u_b) + \lambda(k_p e_i + k_i dt \sum_t e_{t,i} + k_d \dot{e}_i) \tag{23}$$

5.2.6 Output Feedback Control

An Output Feedback Controller (OFC) [3]was designed and simulated, using the same reference signal generated in the previous section. As the control is multi-variable, the state space of vector x must augmented to represent the state of both motors using inputs u_1 and u_2 in a single state, which allows the system to embed coupling behaviours between x_1 and x_2. This leads to the larger matrix representation shown below in Equation 24

$$\begin{bmatrix} \dot{x}_1 \\ \ddot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_{eff}+k}{m} & -\frac{u_{eff}+c_{eff}}{m} & \alpha & \beta \\ 0 & 0 & 0 & 1 \\ \gamma & \delta & -\frac{k_{eff}+k}{m} & -\frac{u_{eff}+c_{eff}}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{24}$$

. The expanded states allow the coupling to be represented in the plant dynamics unlike in PID control, where $\alpha, \gamma = -\frac{k}{m}$and $\beta, \delta = -\frac{c_{eff}}{m}$ represent displacement and velocity coupling terms respectively [4]. The output matrix equation is thus

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{bmatrix} \tag{25}$$

. As before, both the controllability and observability conditions were validated. The observability condition is especially critical for the underlying SFC and LO components, as both control input and estimated states are directly computed using the state vector x. For the OFC architecture, control dynamics are governed by the control law, where the closed loop form [3] is augmented with estimated states

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}-\mathbf{BK} & \mathbf{BK} \\ -\mathbf{LC} & \mathbf{A}-\mathbf{LC}-\mathbf{BK} \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} \mathbf{BK}_r \\ \mathbf{BK}_r \end{bmatrix} r \tag{26}$$

where the estimated states are $\hat{x}$, and K, L are feedback and observer gains. By incorporating feedback linearisation components into both architectures, the OFC output equation 26 rearranges to

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} -\lambda\mathbf{BK} & \lambda\mathbf{BK} \\ -\mathbf{LC} & \mathbf{A}-\mathbf{LC}+\lambda\mathbf{BK} \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} \mathbf{BK}_r \\ \lambda\mathbf{BK}_r \end{bmatrix} r \tag{27}$$

The size of the original state vector has been augmented, which to illustrate, is a matrix $x = (x_1\ \dot{x}_1\ x_2\ \dot{x}_2\ \ \hat{x}_1\ \hat{\dot{x}}_1\ \hat{x}_2\ \hat{\dot{x}}_2)^T$

5.2.7 Gain Tuning

For PID gains, the Ziegler-Nichols method [5] was used to fine-tune errors. An optimisation loop was created, varying k_p until an oscillation was detected at k_u with period T_u. Other gains were then calculated using the tuning formulas $k_p = 0.6k_u$,$k_I = 2\frac{k_p}{T_u}$ and $k_d = \frac{k_p T_u}{8}$ leading to gain values of;

$$\begin{cases} k_{p,1} = 0.06 \ \ k_{I,1} = 11.34 \ \ k_{d,1} = 7.936e-0.5 \\ k_{p,2} = 0.17 \ \ k_{I,2} = 1.79 \ \ k_{d,2} = 0.00403 \end{cases}$$

Integral wind-up prevention was implemented due to the large error accumulation during phase transition, which led to large oscillations for motor 2 early on causing saturation. A clipping logic was integrated into the PID calculation to ensure the signal would be limited past a saturation threshold of 15.

Selection of **L** and **K** was done to ensure stability. These were done separately as noted by the separation principle, through Python eigenvalue placement libraries such that poles of **L** remained in the LHS of the

complex plane at positions (-2,-3.5) to balance stability and actuation. and $(\mathbf{A}-\mathbf{BK})$. Poles of $\mathbf{K}$ were placed in the range (-6,-10.5) for a 3x times faster response in the CL system [3].

$$\mathbf{K} = \begin{bmatrix} 345.70 & 83.66 & 57.25 & 6.68 \\ 57.25 & 6.68 & 310.05 & 79.34 \end{bmatrix}, \mathbf{L} = \begin{bmatrix} 5.73 & 0.19 \\ 6.61 & -0.14 \\ 0.12 & 4.87 \\ -0.40 & 4.39 \end{bmatrix}$$

For OFC, steady state error was able to be mitigated by solving for the reference gain matrix K_r at a steady state $\lim_{x\to\infty}\hat{x} = 0$, using Equation 28:

$$\mathbf{K}_r = -\frac{1}{\mathbf{C}(\mathbf{A}-\mathbf{BK})^{-1}\mathbf{B}} = \begin{bmatrix} 349.70 & 59.25 \\ 59.25 & 314.05 \end{bmatrix} \tag{28}$$

5.2.8 Theoretical Study Discussion

From the graphical response of OFC control compared to PID control, as shown in Figures 11 and 12 respectively, the plots show clear performance improvements for accurate reference tracking when using a multivariable architecture, and is especially apparent in the second stage of extension. In PID control, a distinct delayed oscillatory response is often observed, along with a persistent steady-state error. This steady-state error was addressed by calculating $\mathbf{K}_r$ within the OFC architecture, which helps mitigate the issue, providing an additional advantage to the system. The lack of coordination between independent controls likely led to major challenges in transition, where the inputs being passed led to instability and often oscillatory responses from inability to consider interactions of coupled states. The gains calculated from the Ziegler-Nichols method were only sufficient for the first phase of the PID controller, leading to evident instability in the responses shown in Figure 11by overshoots and unstable oscillations.

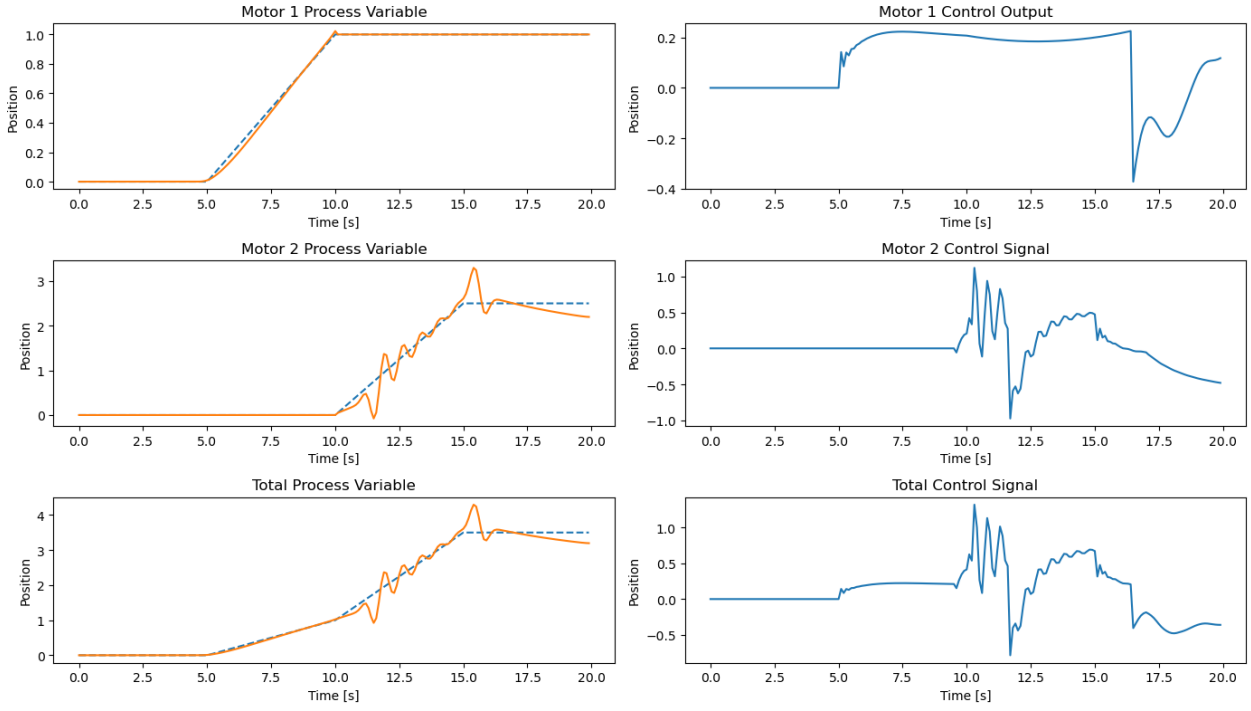


Figure 11: PID Response

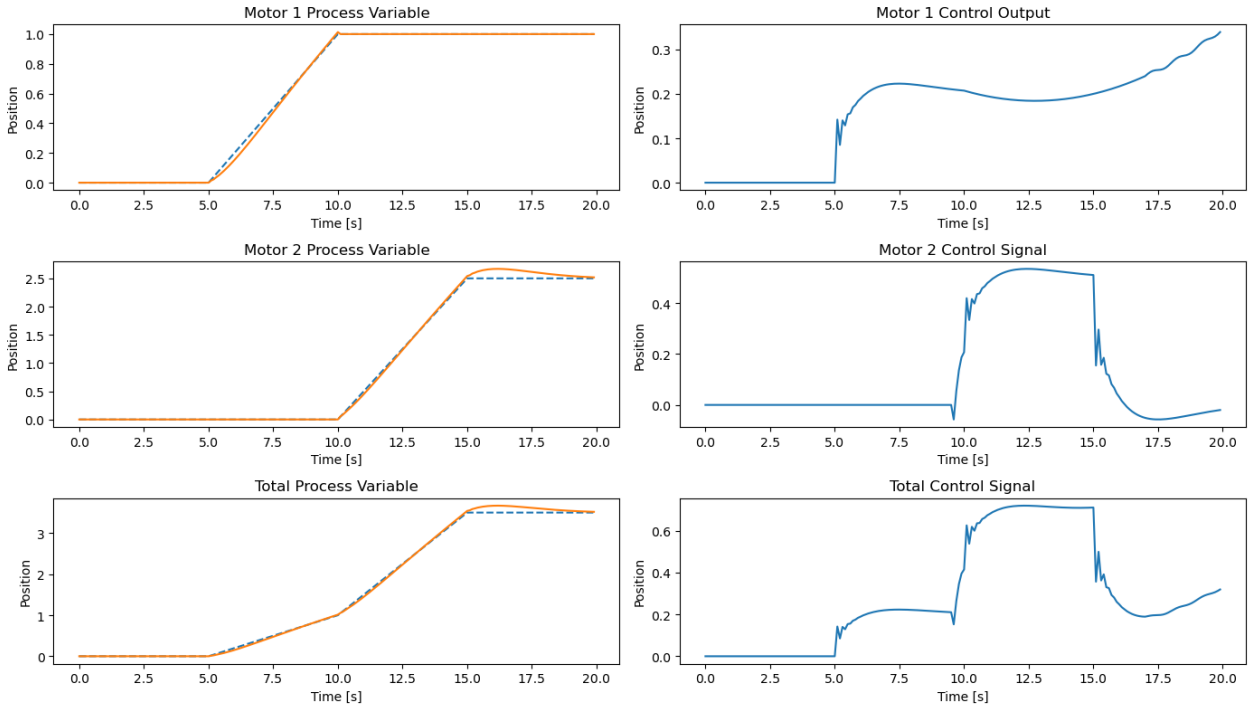


Figure 12: OFC Response

The effects of nonlinearities in shown Figure 13 demonstrate that, without it, the controller exhibited significant challenges in reference tracking, producing responses that were heavily saturated and often times oscillatory. The control signals often exceeded actuator limits, resulting in erratic behaviour and poor trajectory tracking. The presence of discontinuities in the backlash and friction formulations likely create abrupt shifts in the system's response, making it difficult for controllers to smoothly track the desired trajectory. These effects cause sudden spikes or dips in control effort, as the PID attempt to overcompensate in dead zones.

However, in practice the advantages are not as clear. In this study, the plant was inherently simple in design with major assumptions made, especially in the formulations of coupling interactions. In practice, plant $\mathbf{A}$ is less likely to be as well defined to account for these dynamics, making analytical calculations for $\mathbf{L}$ and $\mathbf{K}$ impossible and necessitating the use of system identification methods.

Experimentally, another limitation was the use of using discrete time models is the lack of insight into real-time response. As OFC and multivariable controllers quickly become complex, computational overhead may lead to delayed responses.

6 Conclusion

The designed robot was able to successfully complete drawings of the defined circle, triangle and square with a reasonably high accuracy with all shapes falling inside the backlash envelope. The final design applied independent PID controllers at each rotational link of a robot, with additional mechanisms to counteract sources of instability. On average, each shape took 4.43 seconds to complete. Throughout the design process, both hardware and control considerations were applied to improve robust control while mitigating the effects of backlash. Additionally, several control methods, including system identification and filtering, were tested, with the most successful modification being the addition of a velocity profile to the reference signal array. A further study conducted into the effects of multivariable control simulated on a known plant using OFC against two independent PID controllers showed improved performance through eliminating steady state

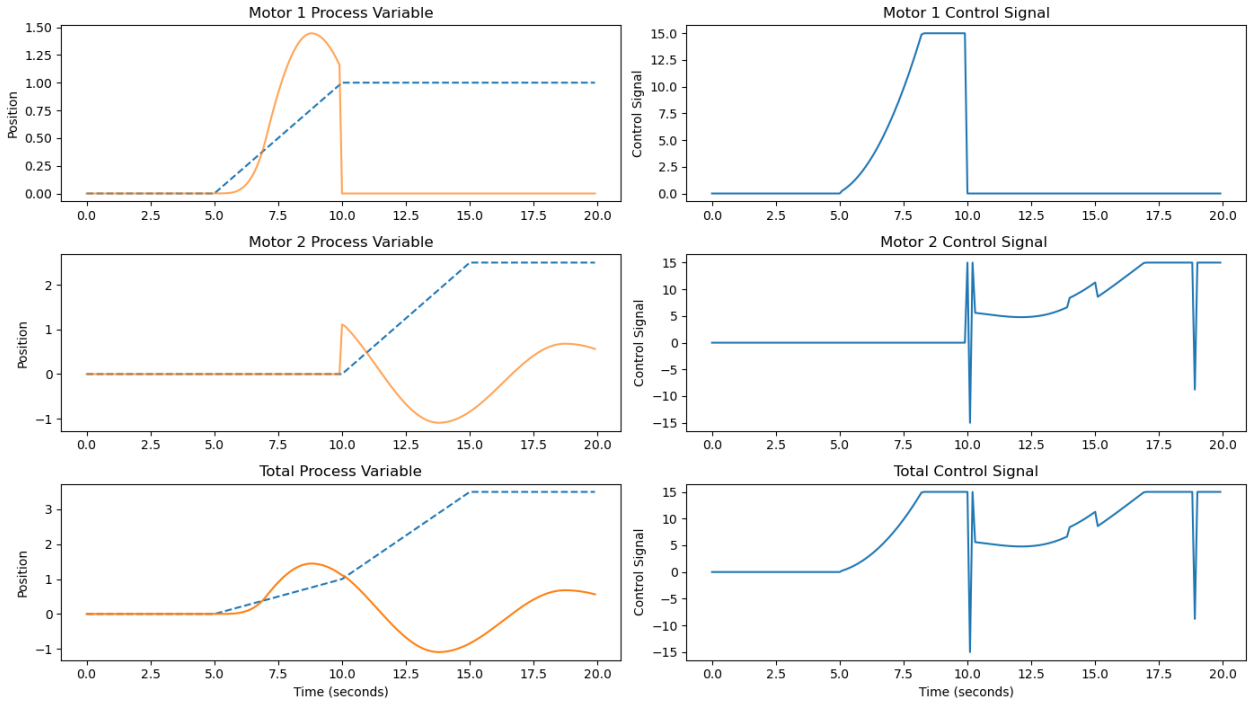


Figure 13: PID controller without linearisation

errors, and significantly improved reference tracking especially at transitional stages.

References

[1] A. Harrison, MENGM0067 Multivariable and Nonlinear Control Lab Notes, *No Journal* 2024, 2024.

[2] MENGM0067 Multivariable and Nonlinear Control Non-linear Notes Notes, *No Journal* 2024, 2024.

[3] A. Harrison, MENGM0067 Multivariable and Nonlinear Control Multivariable Notes 2024, 2024.

[4] K. Ogata, *Modern Control Engineering*, 5th. Upper Saddle River, NJ: Prentice Hall, 2010, pp. 1–800, ISBN: 978-0-13-615673-8. available from: https://www.amazon.com/Modern-Control-Engineering-Katsuhiko-Ogata/dp/0136156738 [Accessed 12/02/2024].

[5] G. Ellis, Chapter 6 - four types of controllers, in *Control System Design Guide (Fourth Edition)*, G. Ellis, Ed., Boston: Butterworth-Heinemann, Jan. 1, 2012, pp. 97–119, ISBN: 978-0-12-385920-4. DOI: 10.1016/B978-0-12-385920-4.00006-0. available from: https://www.sciencedirect.com/science/article/pii/B9780123859204000060 [Accessed 12/02/2024].