Simulation of Engineering System 3 (ENG3036)

Assignment Report Template Part 2

Student Name: **Elsie Msalila** Student Number: **2778939M**

Assignment Topic: Position Control for a Robot Arm

2.0 Introduction for Part 2

After developing models and simulating the robot arm control system in part 1, it became clear that the system should be improved with further tuning. While the final forearm deflection angle did settle at the desired 55° there was a lot of oscillation within the gear and actuator.

Hence, in this part of the assignment the existing control system was tuned by changing the proportional gain value G_c. The effects of varying the value of G_c were then analysed to fine an optimal value. In addition to tuning the proportional control system, an integral term (K_I) was added to then change it into a proportional integral controller. Once this had been implemented, the value of K_i was altered and its effects on the system could be analysed.

Furthermore, the redesigned control system was then tested by varying a key parameter which had previously remained constant, in this case the deflection of the upper arm (θ_U). The varying values of θ_U were found by interpolating data points of the upper arm angle at specific times. This data was then implemented into MATLAB to show how the system responds.

Proper tuning of control systems is crucial as it optimizes the response of a system, ensuring stability and efficiency. Implementing dynamic parameters (e.g. varying θ_U over time) into the model reflects real-world scenarios where system dynamics change over time. This in turn enhances the adaptability of the model making it more useful in industry applications.

<u>Control System Design & Implementation</u>

2.1 G_c Control Gain Variation Results & Analysis

The first attempt to improve the control system was by altering the G_c value which had already been implemented withing the MATLAB script. The aim is to reach the desired deflection angle of 55° as quickly as possible without overshoot. Initially the value of G_c was increased from 3 to 4. This caused the forearm deflection to oscillate with increasing amplitude each period meaning the system is unstable (figure 1). Hence, the value of G_c was decreased.

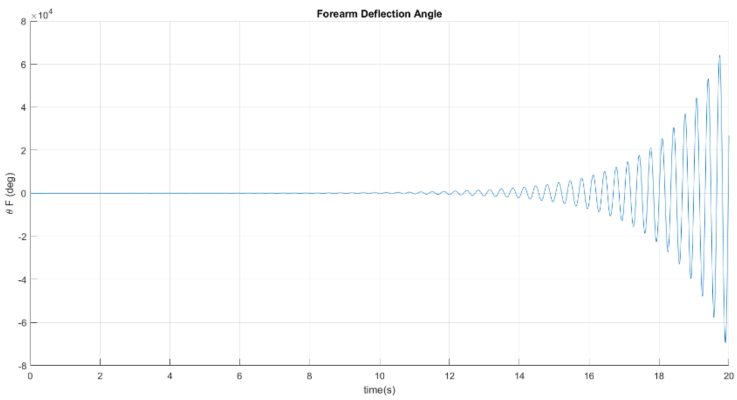

Figure 1 - Forearm deflection using a value of Gc = 4

After testing multiple values, it was found that values between 0.1 and one produced the most desirable results and so these values were plotted against each other to find a final optimal value (figure 2).

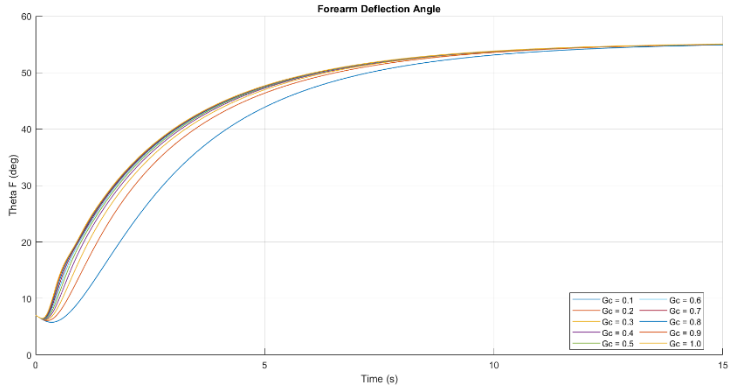

Figure 2 - Comparison of forearm deflection using values Gc = 0.1, 1

While figure 2 shows that larger values closer to 1 provide shorter settling times, figure 3 shows that values over 0.2 cause overshoot within the gear and actuator. Although a fast settling time is desirable, accuracy in movement is paramount in robotic applications. Hence, a value of $G_c = 0.2$ was chosen to give the shortest settling time whilst removing overshoot from the system.

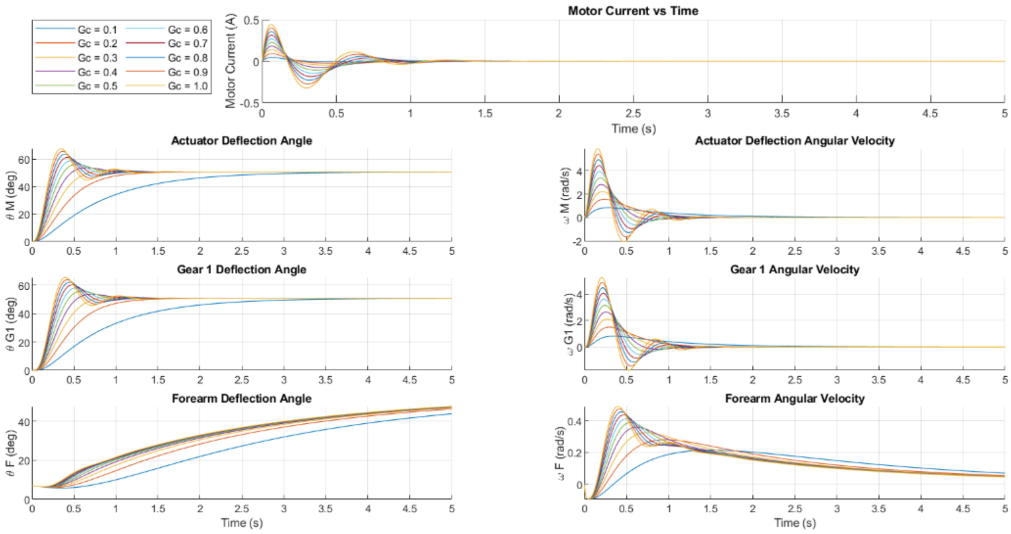

Figure 3 - System results using values Gc = 0.1, 1

The final response of the system using a value of $G_c = 0.2$ can be seen in figure 4.

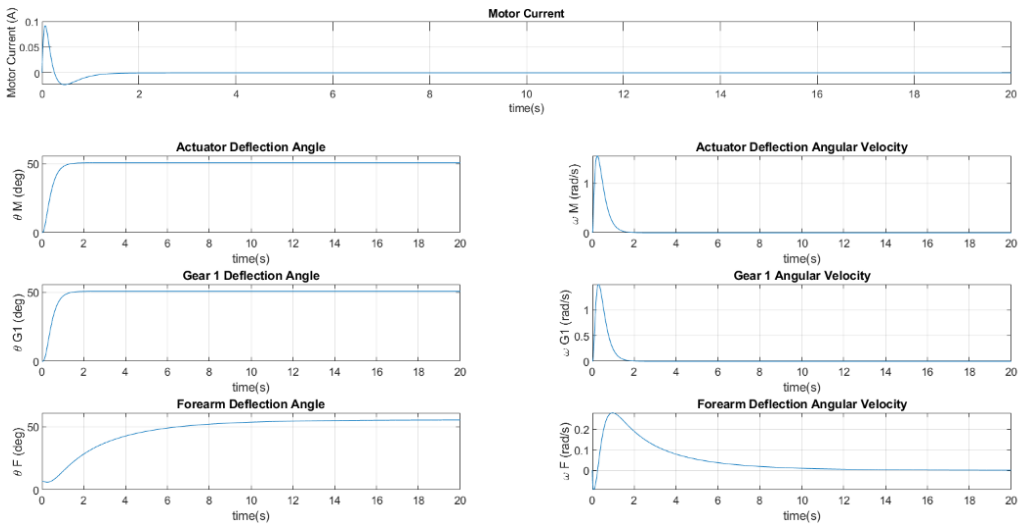

Figure 4 - State variables when implementing Gc = 0.2

2.2 Integration Implementation in Control System in MATLAB Code K_I Control Gain Variation Results & Analysis

After finding an optimal value for G_c, an integral term was introduced into the controller to try and further improve the system. The addition of this term creates a proportional-integral (PI) controller. Using equation 2 from the assignment brief, the MATLAB script was altered (figure 5). The integral term of the PI controller reacts to the accumulation of past errors to eliminate steady-state errors which lead to offset in the final settled value.

```
Delta_Theta = (Theta_Ref * Kr) - (x(2) * Ks);
Integral_DTheta = Integral_DTheta + (stepsize * Delta_Theta);
Ve = (Gc * Delta_Theta) + (Gc*Ki*Integral_DTheta);
Va = (Ve * Kg);
```

Figure 5 - Implementing PI controller into MATLAB code

Multiple values of K_I were tested while keeping G_c at 0.2. It was found that increasing the value of K_I reduces the settling time of the forearm deflection angle (figure 6).

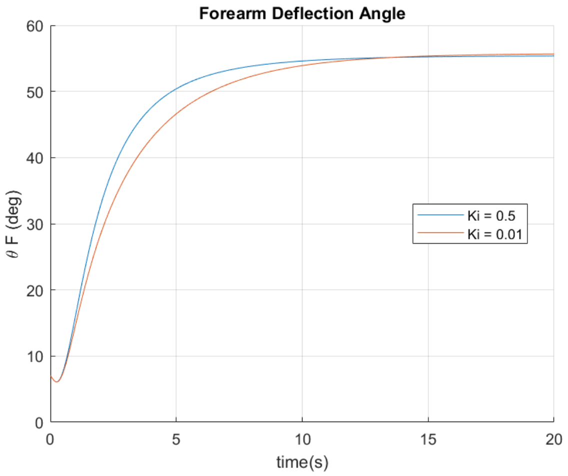

Figure 6 - Comparison of forearm deflection when using Ki values of 0.5 vs 0.01

However, this higher value of K_I introduces overshoot within the gear and the actuator (figure 7). Thus, it was judged that the PI controller had a negative impact on the system as although it reduces settling time, the accuracy of the system is compromised. Due to this a small value of K_I (0.01) was chosen.

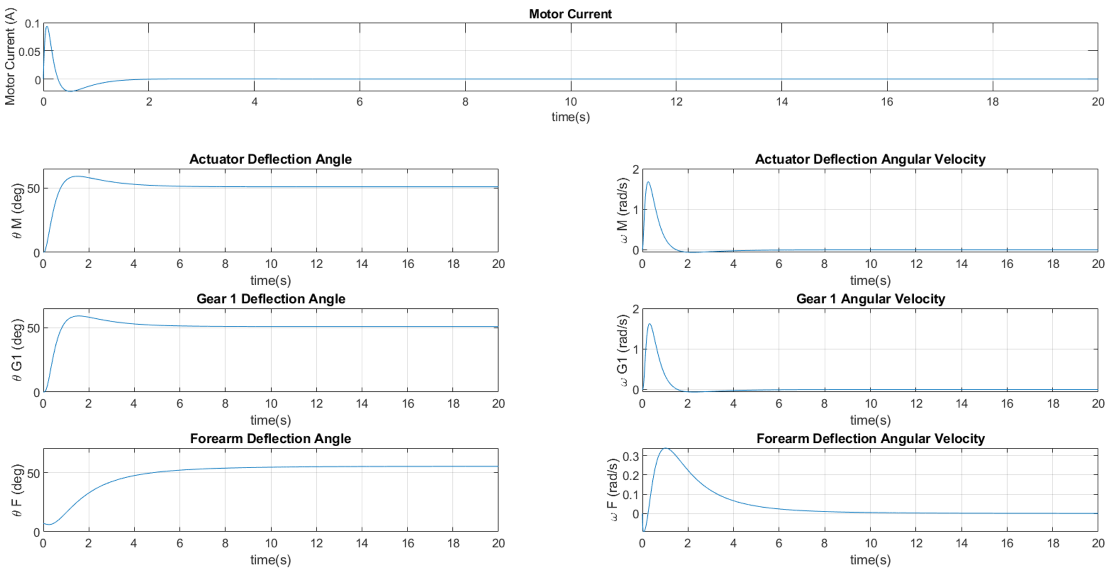

Figure 7 - State variables when implementing Ki = 0.5

To further improve the system, a proportional-integral-derivative (PID) controller could be implemented. Adding a derivative term predicts future errors which reduces the response time and overshoot in the system. This would allow for a shorted settling time without compromising on the accuracy of the movement of the forearm. These characteristics make PID controllers good for high speed, high accuracy applications.

<u>Interpolation</u>

2.3 Interpolation Algorithm Calculation

To take the motion of the upper arm into consideration, the system was altered to include a varying value of θU. This allows us to examine the forearm control system in a dynamic environment. The values of θ_U interpolated from the given data points in Appendix B of the assignment brief. This was done using Newton's divided difference (figure 8).

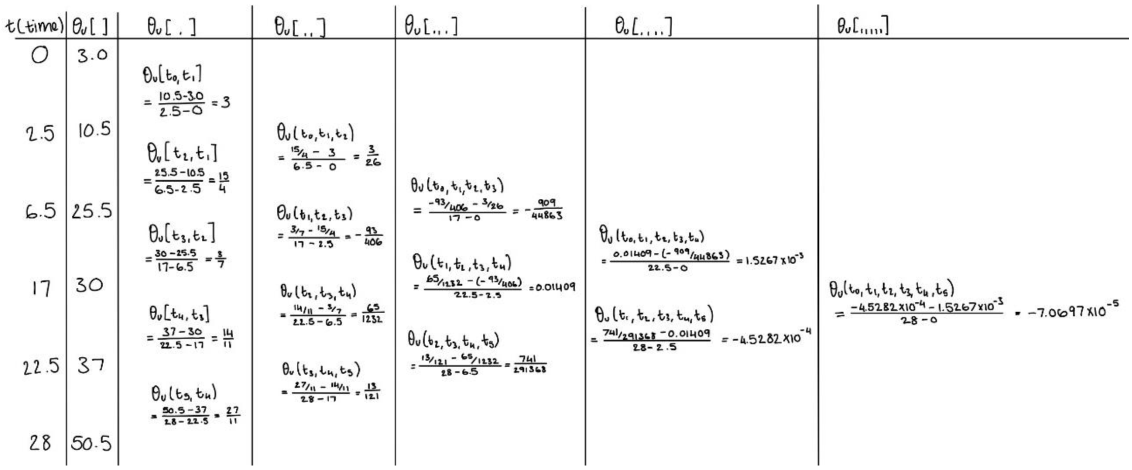

Figure 8 – Newton's divided difference table

The final values of the divided differences can be seen in table 1.

t (Time)	θ_U[]	θ_U[,]	θ_U[, ,]	θ_U[, , ,]	θ_U[, , , ,]	θ_U[, , , , ,]
0	3					
		3				
2.5	10.5		0.115			
		3.75		-0.0203		
6.5	25.5		-0.229		0.00153	
		0.429		0.0141		-0.0000707
17.0	30		0.0528		-0.000453	
		1.273		0.00254		
22.5	37		0.107			
		2.455				
28.0	50.5					

Table 1 - Final divided difference table

Once the divided differences had been calculated, they were put into the nested form to give the final polynomial (figure 9). This expression can be used to approximate θ_U at the times between the data points.

$$P(t) = 3 + 3(t-0) + 0.115(t-0)(t-2.5) - 0.0203(t-0)(t-2.5)(t-6.5) + 0.00153(t-0)(t-2.5)(t-6.5)(t-17) - 0.0000707(t-0)(t-2.5)(t-6.5)(t-17)(t-22.5)$$

Figure 9 - Interpolated polynomial derived from Newton's Divided Differences

2.4 Interpolation MATLAB Code, Results & Analysis

The polynomial was implemented into MATLAB as shown in figure 10.

```
% Interpolation data
t = [0, 2.5, 6.5, 17, 22.5, 28]; % Time data points
Theta_U_values = deg2rad([3, 10.5, 25.5, 30, 37, 50.5]); % Convert to radians
interpol = [3, 3, 0.115, -0.0203, 0.00153, -0.0000707]; % Coefficients

% Dynamic Simulation Loop
for time = 0:stepsize:EndTime

    % Interpolated Theta_U using nested polynomial form
    Theta_U = interpol(1) + ...
              interpol(2) * (time - t(1)) + ...
              interpol(3) * (time - t(1)) * (time - t(2)) + ...
              interpol(4) * (time - t(1)) * (time - t(2)) * (time - t(3)) + ...
              interpol(5) * (time - t(1)) * (time - t(2)) * (time - t(3)) * (time - t(4)) + ...
              interpol(6) * (time - t(1)) * (time - t(2)) * (time - t(3)) * (time - t(4)) * (time - t(5));
```

Figure 10 - Implementing polynomial in MATLAB code

Plotting the varying value of θ_U against time shows the movement of the upper arm (figure 11). The response of the system to the varying value of θ_U can be seen in figure 12.

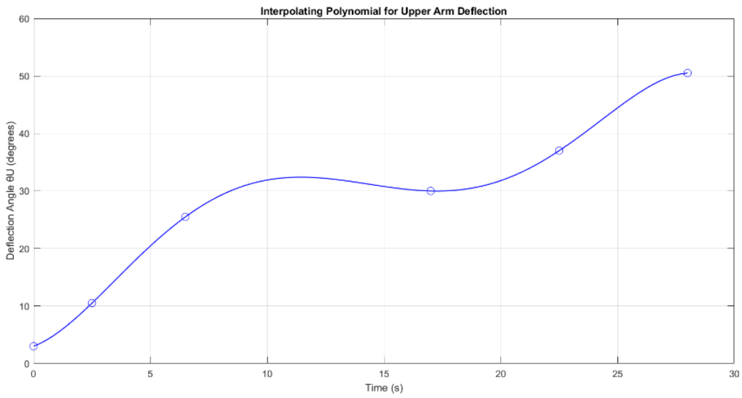

Figure 11 - Interpolated ϑ_U against time

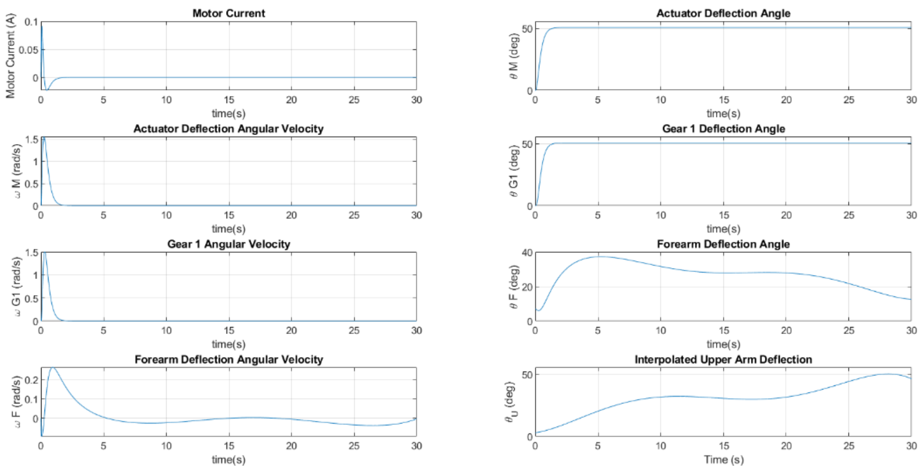

Figure 12 - State variables when implementing a variable value of ϑ_U

2.5 Conclusions

In conclusion, the control system was significantly improved by tuning G_c with the optimal value of 0.2 balancing a reduced settling time with stability. The addition of an integral term to the control system allowed for faster settling times for the forearm deflection however, as the value of K_I is increased, overshoot is introduced to the system and thus it was kept small at 0.01. The control system could be further improved by implementing a PID controller for faster responses whilst reducing overshoot. The implementation of a varying value of θ_U improves the realism of the system allowing it to adapt to dynamic environments. Implementing other dynamic parameters could further enhance the model.