Applied System Identification to the P matrix for adaptive P controller implementation in robotic arm simulation using Matlab/Simulink

Manas Sangwotasil, Phichitphon Chotikunnan[*], Manutsawee Kiewongart,
Nuntachai Thongpance and Rawiphon Chotikunnan.

College of Biomedical Engineering, Rangsit University, Pathum Thani, Thailand
[*]Corresponding author, E-mail: Phichitphon.c@rsu.ac.th

Abstract

This research presents the programming process with M-Files from P matrix system equations to simulate a robotic arm and compare the simulation results via Simulink. In this process of design to cause the control system where the gain system has been adjusted between the time of step input changes in the batch process.

In the simulation, this research designed a two-controller, P controller, and adaptive P controller for comparison of performance in the control input, incidence function in step input profiles, and smooth function to compare the controller to test in different profiles.

In the simulation results, it was found that the form of written P matrix equations can be used for both normal and gain adjustments the same as Simulink. As for the simulation results, it was found that the system with an adaptive P controller can perform better than the P controller in terms of having less overshoot, although steady-state time is approximately the same.

Keywords: *MATLAB/Simulink; Robotic arms; Trajectory control*

1. Introduction

Today, robotic arms are increasingly playing a role in human daily life. It can be seen from humans using robots to help in factories or work in risky areas including being developed continuously and rapidly in hardware, software, and controllers.

MATLAB/Simulink is a program (Ren, 2017; Sheng, 2017; Gastli et al., 2019). with the ability to cover algorithm development, mathematical modeling, and simulation of the system. The research has presented a comparative method. It presents the advantages and disadvantages of the controller, for Example, PI controller (Gavran et al., 2017), PID, fuzzy logic controller (Dewi et al., 2020), PID - fuzzy logic (Oktarina et al., 2019), adaptive fuzzy control (Yang et al., 2018), neural networks (Dexu et al., 2019), Fuzzy-Sliding Mode Control (Liang et al., 2021) Pole placement and LQR control (Soufiani et al., 2020) to control the system.

In the design of all the above-mentioned control systems, it is important to model the system for the development of the algorithm. In estimating the simulation's output, before it can be used in a real-world system many times, the simulations in various research studies have limitations in the simulation for estimating the system's transfer function. In programming simulation of a batch process system, which cannot be written in a simulation program with normal functions.

Therefore, the research team designed the algorithm program in M-file, and the system P matrix equation was studied for use in system simulation and designed an adaptive P controller for the robot arm device by mimicking the movements of the human arm. By using the programs MATLAB and Simulink for control and simulation of robotic arms. The benefits of this research can be summarized in the following table for simulation work Table 1 shows the ability to use in various forms of system simulation.

Table 1 The comparison of the different simulations used to estimate the output of the system.

	Feedback control		**ILC batch process**		
Program(function)	**Fix gain**	**Adaptive gain**	**Fix gain in feedback control**	**An Adaptive gain in feedback control**	**Note**
M-file (system P matrix)	/	/	/	/	***Propose**
M-file (filter)	/	x	/	x	
Simulink	/	/	x	x	

"/ = Can do it" and "x = Cannot do it"

2. Objectives

1) To study the system simulation design through programming M-Files with a P matrix.

2) To compare the results of programming designed from the P matrix equation with the results from Simulink.

3) To compare the performance of the P controller and adaptive P controller in different profiles.

3. Materials and Methods

3.1 System identification modeling

To find the system equations, the robotic arm can consider the input signal and the output signal from the system which is in the time domain where the researcher simulates. The robot movement to move robot moves repeatedly to move on each axis. The system is defined as a discrete-time and the sampling time is 0.055 s, which results in estimating the transfer function of all motors using system identification in MATLAB and showing the parameters in Table 1.

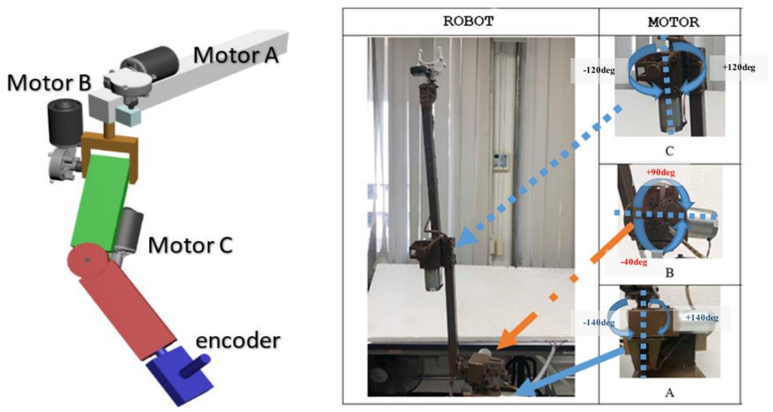


Figure 1 The structure of the robotics.

$$P(z) = \frac{\gamma_1 z}{z^2 + \beta_1 z + \beta_0} \tag{1}$$

where the coefficients are used to appear in Table 2.

Table 2 The parameters of the motor used in an open-loop system

	γ_1	β_1	β_0
Motor A	0.072800	-1.423520	0.423020
Motor B	0.065985	-1.416977	0.416566
Motor C	0.059974	-1.420447	0.420346

Equation (1) can be written in state-space and can be written in the form of a matrix. The system can display the time step response that occurs each time, the following equation can be obtained.

$$y_j(k) = \bar{A}x(0) + Pu_j(k) + v_d \tag{2}$$

$$y_j(k) = [y_j(1) \quad y_j(2) \quad \cdots \quad y_j(p)]^T \tag{3}$$

$$u_j(k) = [u_j(0) \quad u_j(1) \quad \cdots \quad u_j(p-1)]^T \tag{4}$$

$$P(k) = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \cdots & CB \end{bmatrix}; \bar{A}(k) = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^P \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} y_j(1) \\ \vdots \\ y_j(N) \end{bmatrix} = \underbrace{\begin{bmatrix} p_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ p_N & \cdots & p_1 \end{bmatrix}}_{P} \begin{bmatrix} u_j(0) \\ \vdots \\ u_j(N-1) \end{bmatrix} + \underbrace{\begin{bmatrix} q_1 \\ \vdots \\ q_N \end{bmatrix}}_{\mathbb{Q}} x_j(0) \tag{6}$$

where $p_i = CA^{i-1}B$ and $q_i = CA^i$, for $i \in [1,\ N]$. Let $x_j(0)$ be the presumed initial state variable of zero parameters to test for iteration. Considering the system P matrix containing p_i along the diagonal lines, the coefficients p_i are referred to as Markov parameters of the system.

which the output value in each axis is y(k), by setting the distance of error in motion is e(k), which can be calculated from Equation (2)

$$e(k) = y_d(k) - y(k) \tag{7}$$

And Equation (6) can be written in the form of the general equation as shown in Equation (8).

$$\begin{aligned} y_j(k+1) = {} & (CA^{N-1}B)u_j(k-N) + (CA^{N-2}B)u_j(k-(N-1)) + (CA^{N-3}B)u_j(k-(N-2)) \\ & + \cdots + \left(CA^{N-(N-2)}B\right)u_j(k-3) + \left(CA^{N-(N-2)}B\right)u_j(k-1) \\ & + (CA^{N-N}B)u_j(k) \end{aligned} \tag{8}$$

3.2 Design of robotic arm control system with discrete PID system

The design of the PID control system is a closed-loop control system or feedback control system consisting of a proportional-integral-derivative controller as shown in Figure 2. In this research, the controller in PID is used as a discrete-time P controller and using the technique of CHR tuning got K_p gain at value 15.5 and the control equation, to use the format discrete-time PID controller as in Equation (9).

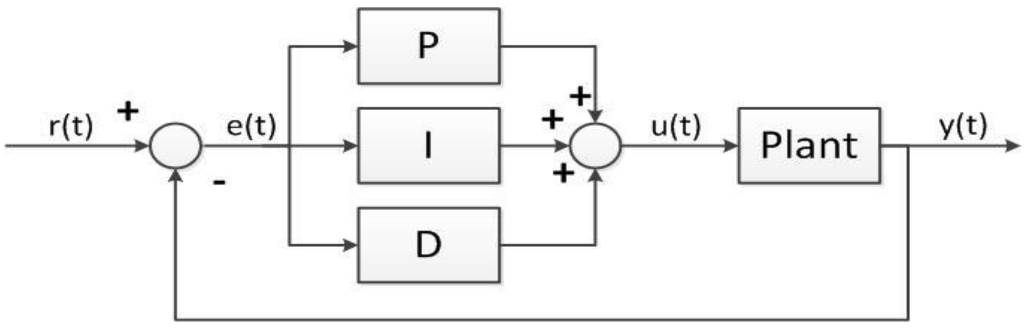


Figure 2 Block diagram of PID control

$$u(k) = K_p e(k) + K_i \sum_{0}^{k} e(k) + K_d \frac{e(k) - e(k-1)}{\Delta t} \tag{9}$$

3.3 Adaptive P controller

An expert system is a computer program (software) that uses artificial intelligence (AI) to reproduce the judgment of a human with expert knowledge in a particular field and is designed to solve problems by if-then rules. Components of an expert system have a knowledge base, inference engine, and user interface. In this research, the expert system was designed to be used to adjust the adaptive gain value of the P controller.

The pseudocode of the expert system for adaptive gain is shown in Figure 3, by $f(e)$ is an input of the algorithm, and the output gain is an output of the algorithm.

```
IF (($f(e)$ >=0)&&( $f(e)$<5))              THEN
        output_gain=2
END
IF (($f(e)$>=5)&&( $f(e)$<10))              THEN
        output_gain =2.5
END

IF (($f(e)$>=10)&&( $f(e)$<15))             THEN
        output_gain =5
END

IF (($f(e)$>=15)&&( $f(e)$<40))             THEN
        output_gain =10
END

IF (($f(e)$>=15)&&( $f(e)$<40))             THEN
        output_gain =15.5
END
```

Figure 3 Pseudo code of expert system for adaptive gain, the P controller

3.4 Programming in MATLAB/Simulink

The program used in the simulation of this research has shown an example of a system simulation programming and ordering the data. this section presents two examples, M-File and Simulink.

Example: Writing an M-File

The system P matrix from Equation 1 to Equation 5 was constructed by using a command line. Program:

```
num_R = num_system_motor_A;  % numerator coefficients of the transfer function
den_R = den_system_motor_A ;  % denominator coefficients of the transfer function
tf_d = tf (num_R,den_R,0.055)
[A_Model_R,B_Model_R,C_Model_R,D_Model_R]=tf2ss(num_R,den_R)
A = A_Model_R;   B = B_Model_R;   C = C_Model_R;   D = D_Model_R;
P = zeros(length(setpoint_R),length(setpoint_R));
for i=1:length(setpoint_R)
    P(i,1) = C*(A^(i-1))*B;
end
s_k=1;
for i=2:length(setpoint_R)

   for j=1+s_k:length(setpoint_R)
    P(j,i) = P(j-1,i-1);
   end
   s_k= s_k + 1;
end
```

Figure 4 Program the system P matrix

For a program with step input in feedback control, an adaptive P controller is used by using the command line.

Program :

```
%%%%%%%%%%%%%%%%%% Loop fb control %%%%%%%%%%%%%%%%%%
 control_input_R = setpoint_R * 0 ;
 Y (1,1) = 0;
output_R (1,1) = 0;
Error_R(1,1) =0 ;
for i=2:(length(setpoint_R))
    Error_R(i,1)  = (setpoint_R (i,1) - output_R (i-1,1)) ;
%%%%%%%%%%%%%%%%%% Expersystem control %%%%%%%%%%%%%%%%%%
        U_input_fuzzy_kp  = abs( Error_R(i,1) );
       a_r = U_input_fuzzy_kp;
        if((a_r>=0)&&(a_r<5))
           b_r=2;
        elseif((a_r>=5)&&(a_r<10))
           b_r=2.5;
        elseif((a_r>=10)&&(a_r<15))
           b_r=5;
        elseif((a_r>=15)&&(a_r<40))
           b_r=10;
        else
            b_r=15.5;
        end
       output_expert_Kp  =  b_r;
%%%%%%%%%%%%%%%%%% end control %%%%%%%%%%%%%%%%%%
    control_input_R(i,1) =  Error_R(i,1) *output_expert_Kp ;
    system_R_step = P (i,:) ;
    output_R (i,1) =   system_R_step * control_input_R  ;
end
%%%%%%%%%%%%%%%%%% system output %%%%%%%%%%%%%%
for i=2:length(setpoint_R)
    Y(i,1) = output_R(i-1,1) ;
end
%%%%%%%%%%%%%%%%%% Error of system %%%%%%%%%%%%%%
ck_error_R = setpoint_R - Y;
Error_RR = zeros(length(setpoint_R),1);
for i=1:length(setpoint_R)
   if i < length(setpoint_R)
      Error_RR(i) =   ck_error_R(i+1);
   else
      Error_RR(i) =   setpoint_R(i)- Y(i);
   end
end
RMSE_R = sqrt(mean((Error_RR).^2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 5 Program the feedback control using an adaptive P controller

To program the feedback control using the P controller (Fix gain) by filter function, a command line is used.

Program :

```
%%%%%%%%%%%%%%%%%% P control (fix gain)  %%%%%%%%%%%%%%%%%%
Kp_R = 15.5 ;
sys = series(Kp_R,tf_d)
sys_d = feedback(sys,1,-1)
[ fb_num_R , fb_den_R , ts] = tfdata( sys_d ,'v' )
[output_sys_dis,time_sys] = filter(fb_num_R ,fb_den_R ,setpoint_R) ;
for i=2:length(setpoint_R)
   Y_fix(i,1) = output_sys_dis(i-1,1) ;
end
%%%%%%%%%%%%%%%%%%
```

Figure 6 Program the feedback control using the P controller (Fix gain)

Example: Writing a Simulink

In this example, present the program in Simulink to simulate the system. Figure 7 shows the block control for testing the transfer function and Figure 8 shows the design expert system from Figure 4.

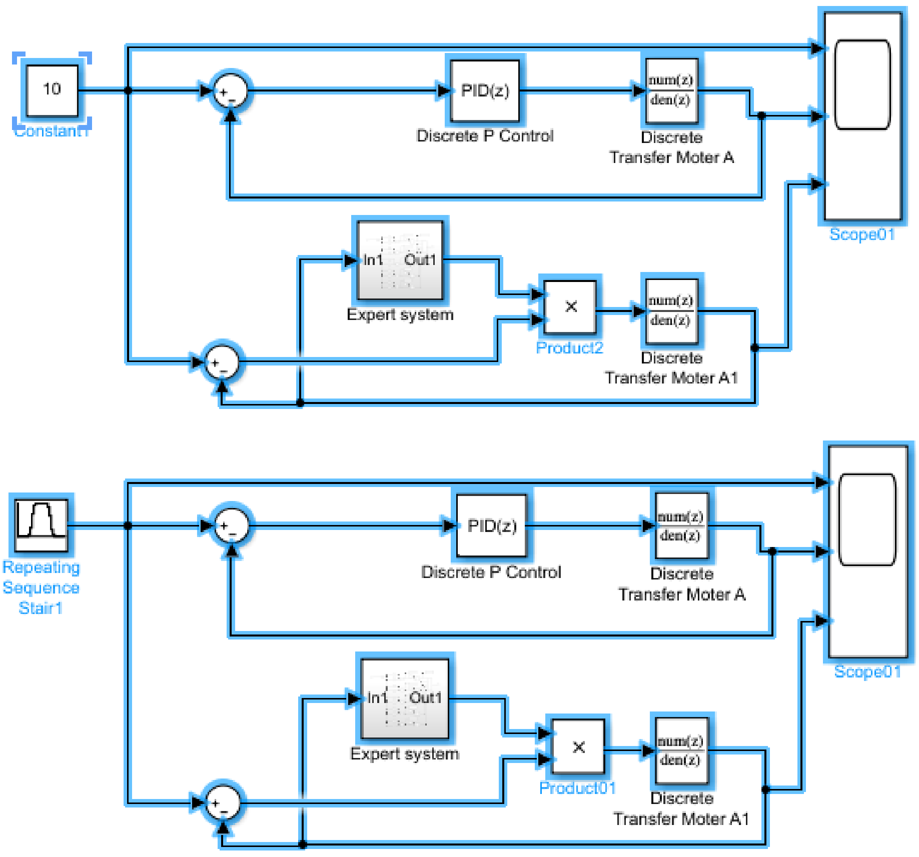


Figure 7 Block diagram in Simulink with feedback control

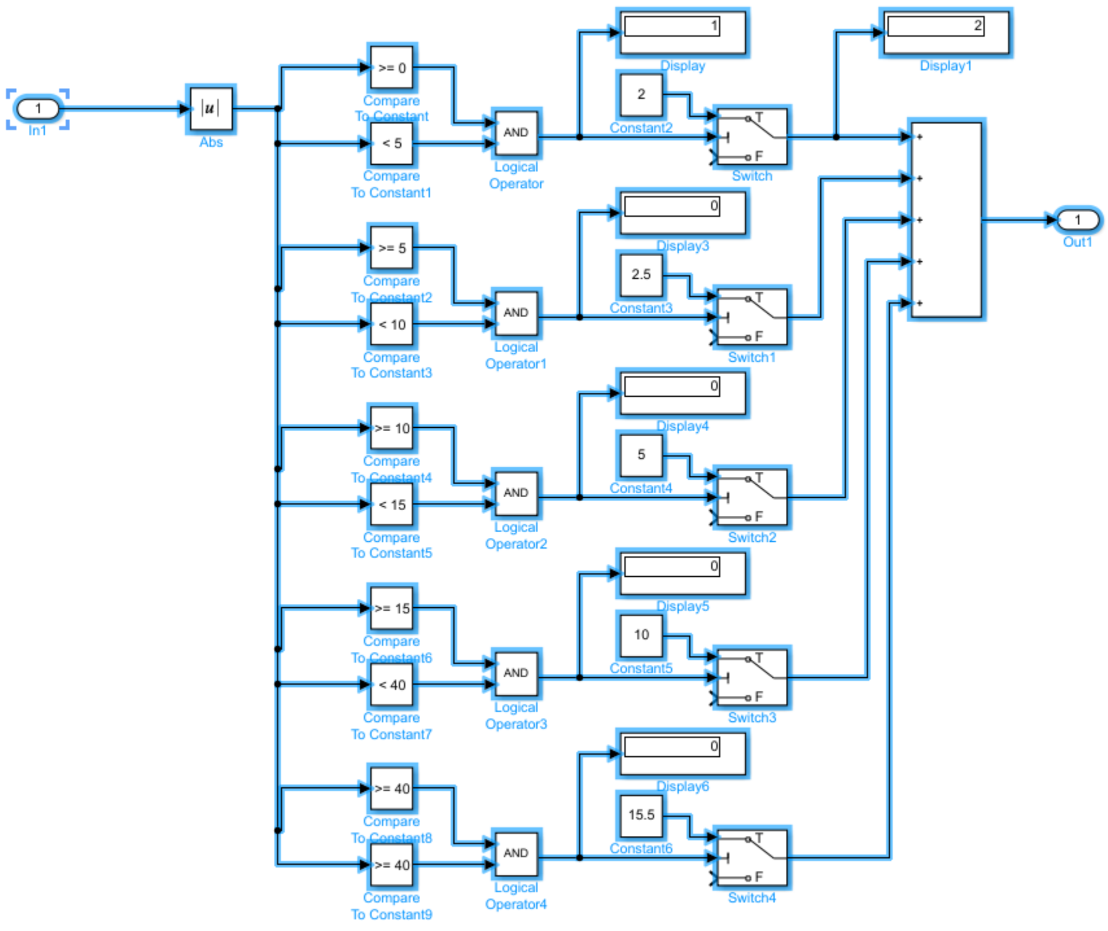


Figure 8 Diagram of expert system block

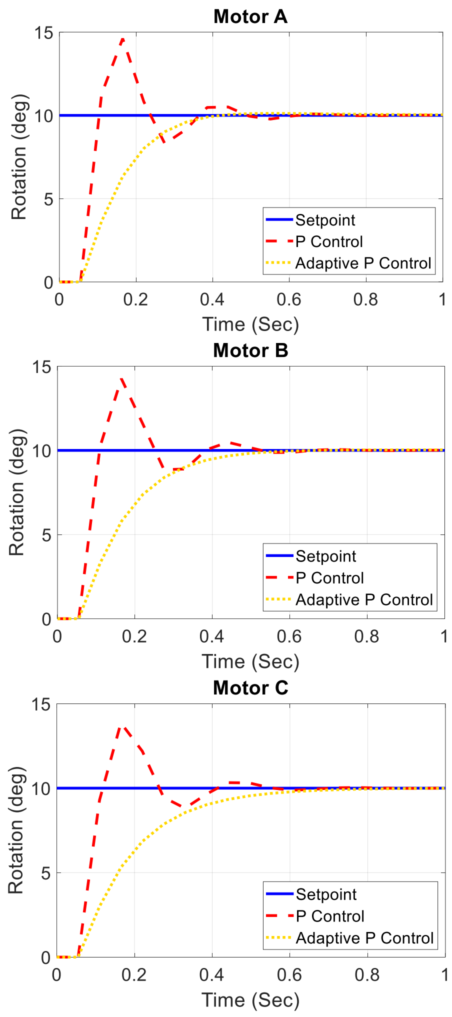
4. Results and Discussion

In the simulation result, for the controller, design two controllers with P controller and adaptive P controller. by feedback control using control input signal in the form of step input and smooth function. The simulation with programming through M-file using system P matrix and filter function to analyze the system response and use the Simulink program to check the answers of the output of simulation systems.

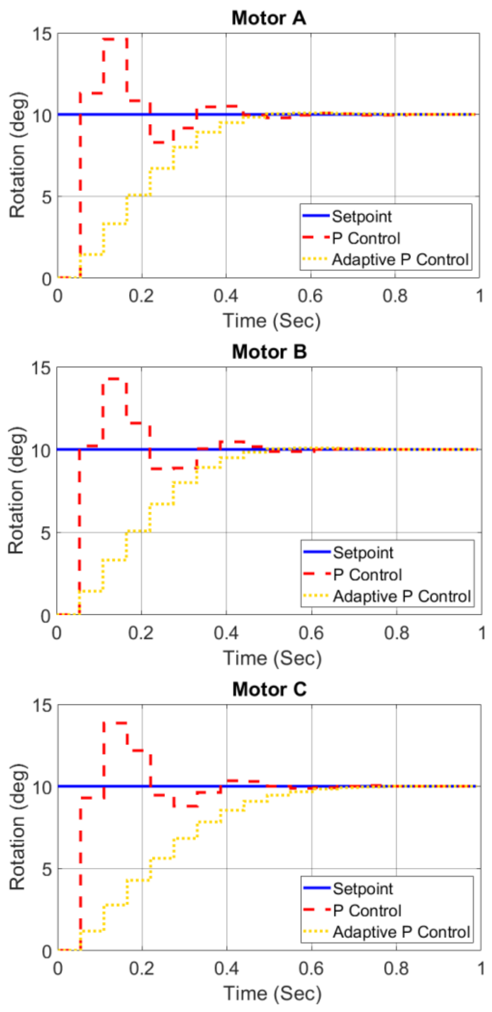
4.1 Control input in Step input

From the system simulation, it is found that the control input signal type is step input. The system simulation results showed that using the P controller alone in A, B, and C motor systems, the overshoot averaged about 40% and steady-state error averaged about 0.6 sec.

The adaptive P controller has no overshoot and the average steady-state error is about 0.5 sec. From the overall performance of both controllers, the adaptive P controller can perform better than the P controller. It is shown in Figure 9.



(a) Output from M-file (triangle hold)

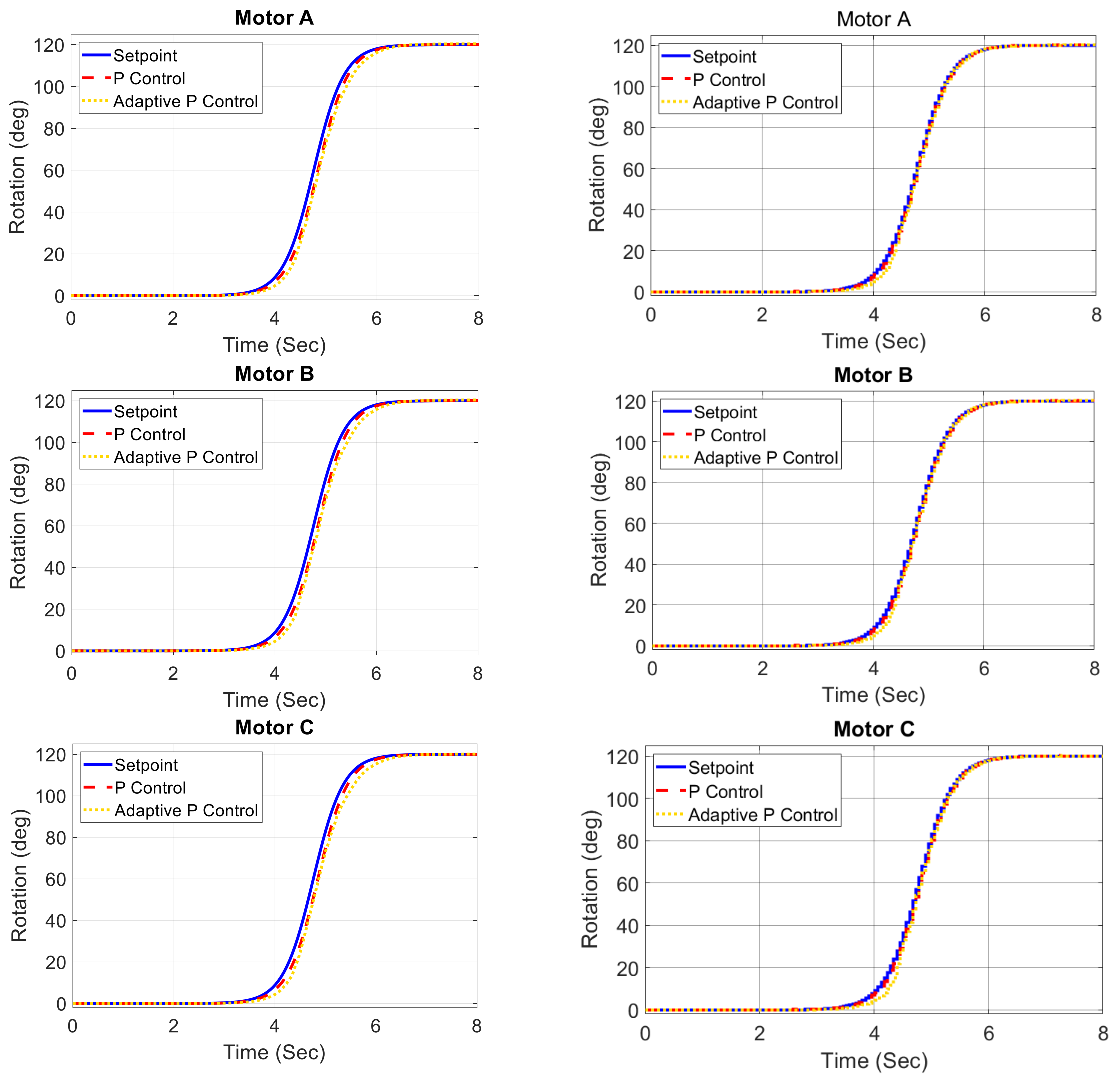


(b) Output from Simulink (zero-order hold)

Figure 9 System response to a step input of all motors

4.2 Control input in smooth function

From the simulation of the system, it was found that in the form of a smooth function control input signal. The system simulation results of the P controller and the adaptive P controller in the A, B, and C motor systems did not detect any overshoot values and the steady-state error value is the same. Although the adaptive P controller has a slower transition period than the P controller. In the overall performance of the two controllers, it was found that the P controller performed slightly better in terms of speed of movement. It is shown in Figure 10.



(a) Output from M-file (triangle hold) (b) Output from Simulink (zero-order hold)

Figure 10 System response in the smooth function of all motors

5. Conclusion

In this research, the design issues are presented using the system equations in the form of a system P matrix that can be processed in time step input, which also compares the results of the system equation design. Compare the simulation results via Simulink to prove the simulation results have the same system response.

It also compares the operation between the P controller and the adaptive P controller in response to the control input signal in the form of step input and smooth function. It was found that the adaptive P controller processor has performed better than the P controller in the output of the system because the proper gain adjustment is made to the faulty signal entering the system, which makes it possible to adapt to control input signals with a more frequency range.

6. Acknowledgements

In this research, I would like to thank the Research Institute. Rangsit University granted research funding to the research team to create a prototype robotic arm, this time

7. References

Dewi, T., Nurmaini, S., Risma, P., Oktarina, Y., & Roriz, M. (2020). Inverse kinematic analysis of 4 DOF pick and place arm robot manipulator using fuzzy logic controller. International Journal of Electrical & Computer Engineering (2088-8708), 10(2).

Dexu, B., Weiwei, K., & Yunlong, Q. (2019). A task-space tracking control approach for duct cleaning robot based on fuzzy wavelet neural network. Journal of Dynamic Systems, Measurement, and Control, 141(11).

Gastli, A., Kiranyaz, S., Hamila, R., & Ellabban, O. (2019, November). Matlab/Simulink Modeling and Simulation of Electric Appliances Based on their Actual Current Waveforms. In 2019 2nd International Conference on Smart Grid and Renewable Energy (SGRE) (pp. 1-7). IEEE.

Gavran, M., Fruk, M., & Vujisić, G. (2017, May). PI controller for DC motor speed realized with Arduino and Simulink. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1557-1561). IEEE.

Liang, D., Sun, N., Wu, Y., Liu, G., & Fang, Y. (2021). Fuzzy-Sliding Mode Control for Humanoid Arm Robots Actuated by Pneumatic Artificial Muscles with Unidirectional Inputs, Saturations, and Dead Zones. IEEE Transactions on Industrial Informatics.

Oktarina, Y., Septiarini, F., Dewi, T., Risma, P., & Nawawi, M. (2019). Fuzzy-PID controller design of 4 DOF industrial arm robot manipulator. *Computer Engineering and Applications Journal*, *8*(2), 123-136.

Ren, G. (2017). MATLAB/Simulink-Based Simulation and Experimental Validation of a Novel Energy Storage System to a New Type of Linear Engine for Alternative Energy Vehicle Applications. *IEEE Transactions on Power Electronics*, *33*(10), 8683-8694.

Sheng, J. (2018, June). Teaching devices and controls for computer engineering and systems students using Arduino and MATLAB/Simulink. In *2018 IEEE 14th International Conference on Control and Automation (ICCA)* (pp. 318-323). IEEE.

Soufiani, B. N., & Adlı, M. A. (2020). Pole placement and LQR control of slosh-free liquid transportation with dual-arm cooperative robot. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *35*(4), 2255-2267.

Yang, C., Jiang, Y., Na, J., Li, Z., Cheng, L., & Su, C. Y. (2018). Finite-time convergence adaptive fuzzy control for dual-arm robot with unknown kinematics and dynamics. IEEE Transactions on Fuzzy Systems, 27(3), 574-588.