EGTRIB Journal
JOURNAL OF
THE EGYPTIAN SOCIETY OF TRIBOLOGY
VOLUME 22, No. 1, January 2025, pp. 81 - 121 ISSN 2090 - 5882
(Received September 28. 2024, Accepted in final form October 25. 2024) jest.journals.ekb.eg

INVERSE KINAMATICS ANALYSIS AND REAL TIME CONTROL OF 5-DOF ROBOT ARM USING PID AND FUZZY LOGIC CONTROLLERS WITH FRICTION MODELING

Abdelwahab Sabreen A.[1], Said Fatma El-Zahraa[2], Hassan Hassan A.[3] and Gouda Mohamed M.[4]

[1, 3] Production Technology Department, [2, 4] Electronics Department,
Faculty of Technology and Education, Helwan University, Cairo, Egypt.

ABSTRACT
Robots represent a vital base in modern and future industry. Hence, robot motion control is an important area for research. In this work, inverse kinematics analysis and real-time control of a 5-DOF robot arm are developed and simulated to reach specified positions with minimal error. Rules for analyzing inverse kinematics are developed for the robot arm using the geometric approach, along with analyzing DC motor models, and friction torque equations are introduced. Two control techniques were discussed: a PID controller is initially used, followed by the implementation of a fuzzy logic controller (FL). A FL controller is designed based on Mamadani pro-Max inference. Four defuzzification methods (BOA, MOM, SOM, and COG) are compared. Simulations were conducted using Matlab and Simulink. The methods BOA, MOM, and SOM are similar in giving optimum results. The results of both controllers were compared. FL outperforms PID, with lower rise time, settling time, steady-state error, and less overshoot. Real-time control has been performed. The system performance was good as the output tracks the reference input signal in a good way, and the FL gave better results also.

KEYWORDS
Fuzzy logic/PID controller, Robot arm, Forward and Inverse kinematics, Open loop dynamics, Friction torque.

INTRODUCTION
The robot arm is made up of a network of connecting links that may move rotational and/or translational movements. These links are connected together by joints; hence, kinematic chains are formed starting at the base and ending with the end effector; the base can be fixed or active. Likewise, the end-effector or gripper is in charge of holding and moving objects, [1]. Robot manipulators and their control approach must be modeled and studied before being used for precision operations. Robot kinematics may be forward or inverse kinematics. Algorithms for control are designed to move the robot arm in different directions and positions. Prior knowledge of the intended position or angle of each joint is necessary for this, and it can be obtained using inverse kinematics when the desired end location is known, and using

forward kinematics when the joint angle is known. It is more difficult to determine the required joint angles for a given end-effector location than only forward kinematics, [2].

Many studies have investigated manipulator robots as a developing field. Kinematic analysis of industrial and educational robot arms, including the PUMA 560, SCARA, and SG5-UT processors, have been studied in some publications, [3 - 5]. Other control problems have been studied by the researchers. As in Chakraborty, B. et al.'s study, [6], which examined a 6-DOF robot manipulator's trajectory tracking in a dynamic environment. The tracking of end effector of robot manipulators against dynamical uncertainties is discussed by Yilmaz B. Melih et al., [7]. An overview of the state-of-the-art developments in collaborative robotic manipulation from the perspectives of modeling, control, and optimization in multi-robot systems is investigated by Feng, Z et al., [8]. Robust hybrid impedance control that adapts to the changing environment for robot manipulators was the focus of Li, Jianfei et al., [9]. Adaptive manipulator control in changing environments was the subject studied by Chen J. et al., [10]. The researchers, [11], addressed the use of three controllers: a supervisory controller and fuzzy logic controller, and the PID controller as a reference baseline, where a 5DOF robot arm is modeled and controlled to achieve the desired position. The findings of the fuzzy logic controller and fuzzy supervisory controller are compared to those of the PID controller, they showed better performance.

Other studies in control technology have focused on PID and FL controllers, [5, 12 - 15]. A brief summary of some of the literature will be provided. Delibes, [16], used PID and FL control algorithms to control the position of a DC motor. Both controllers were implemented using the Labview application, and the application of these controllers produced a target position with a 0.4 percent overrun and an 80 msec settling time for FL, compared to a 4 percent overshoot and a 120 msec settling time for PID. Delibes focused on the effect of FL on the operation of a robot movement simulation controlled by a digital controller.

The authors, [17] described a fuzzy supervisory method for fine-tuning PID settings. By using this technique, the parameters supplied by Ziegler-Nichols (ZN) tuning performed better. The study proved that FL controller performed better in simulations. Path planning was done using FL controller and genetic algorithms by Bang V. K. et al., [18]. FL controller was used to optimize the control of an automated arm, showcasing its practicality and efficiency in enhancing mechanical arm motion. For each sampling, Yung Tao et al., [19] evaluated the effectiveness of the PID, fuzzy, and fuzzy PID techniques to ensure the consistency of the transponder path that was generated. A dependable and effective fuzzy PID performance was found through experiments.

For 2-DOF robot manipulators, a neural fuzzy controller (NFC) was used by Jaffa et al., [20] to regulate the position of an automated arm. Through hybrid learning, the neural network regulated the information input and output. The outcomes of the simulation demonstrated that NFC outperformed PID in controlling the automated arm path. Two distinct control strategies were used by Baghli Z.F. et al., [21], to operate a 2-DOF arm manipulator robot: smart adaptive fuzzy PID and single-output

control based on the traditional PID model. They discovered that, compared to traditional PID controllers, fuzzy PID was more stable and yielded superior outcomes. In their study, Nasr and Ayman, [22], used Matlab/Simulink to simulate a PID controller and control the system to a desired joint angle position. Their findings showed that small variations in the robot arm's initial joint angle positions led to varying desired joint angle positions, requiring modifications to the PID controller gains to avoid oscillation and overshoot caused by changes in parameter values.

Three different position control strategies were developed by Usman Kabir et al., [23], using a 3-DOF robot manipulator; PID, PD, and FLC controllers were installed in each link of the robot manipulator; performance comparisons based on transient and steady-state characteristics showed that all three controllers tracked the setpoint with a small steady-state error; PID and PD controllers performed better in terms of settling and rise times, whilst FLC had less overshoot. In [24], the design of a self-tuning particle swarm optimization (PSO) fuzzy PID positioning controller is described. PSO optimizes the quantization and scaling factors in the fuzzy PID algorithm to maximize the manipulator's robustness and accuracy. The findings demonstrate a notable improvement in the system's follower characteristics, tracking accuracy, and transient response speed.

Controlling a 4-DOF robot arm was the subject of a comparative study by Mohamed Fawzy et al., [25], they looked at the control of the arm robot using FLC and the 2-DOF PID controllers in great detail. Matlab/Simulink was used for the simulation. First, a desired model was constructed, and 2-DOF PID and FLC were devised for each joint of the arm robot. A reset mechanism was used to lessen overshoot and speed up response times. In comparison to the 2-DOF PID controller, the results showed that the FLC performed optimally, offering rapid response, a better rising time, and no overshoot.

A 5-DOF robot arm manipulator employing PID and fuzzy control was presented by Masoud Solouki et al., [26]. Although PID is a common controller for linear systems, fuzzy rules performed better than conventional techniques. Amin Rashidifar M. et al., [27], concentrated on enhancing the control of a 5-DOF robot arm; they implemented a PID controller and compared it with a fuzzy logic controller and fuzzy supervisory controller, FLC performed better than PID controller. In [28], A study is done on the performance of PID and fuzzy controllers for 6 DOF arm manipulator position control using different defuzzification strategies. FLC outperforms PID controllers in terms of overshot, however both FL and PID controllers are able to converge to the intended output. Dzulhizzam Bin Dulaidi, [29] controlled a 6-DOF robot arm using a FL controller and evaluated its performance against a PID controller; their investigation revealed that the FL controller responded more effectively than the PID controller in terms of overshoot, time response and steady-state error.

Separate NFCs for robot arm trajectory tracking were created by Jafar Tavoosi et al., [30], the NFC performed better in trajectory control than the PID controller, according to simulation data. A DC motor's direction was controlled by FLC and PID control algorithms, according to Jamal AbdAltayef et al., [31], both of the PID-FLC

controllers' architectural designs were derived from a Labview system; the FLC maintained the goal location more effectively than the PID controller, according to the results. A robotic arm was controlled by Yong-Lin Kuo et al., [32] using PID, fuzzy, and fuzzy PID systems, lower steady-state error was demonstrated by the fuzzy-PID controller.

1.2 Problem Statement

As robotic tasks become more intricate, there is a growing need for an intelligent, robust, computationally efficient, easy-to-install, and analyze controller to optimize and enhance the performance of industrial robot arms. Modeling, kinematic analysis, and real time control of 5-DOF robot arm are studied in this work using PID and FLC controllers to minimize the difference between the required and actual positions of robot joints or end-effector, meeting specific criteria such as reducing overshoot, minimizing rising time, eliminating steady-state error, and reducing the load on each joint motor to overcome system nonlinearity. Both controllers' performance will be evaluated by investigating the forward and inverse motion equations and dynamics of the robot arm including the friction effect, followed by the development, simulation, implementation, and assessment of the controllers of the robot arm system. Matlab and Simulink software packages will be used for dynamics and control simulation of the robot arm to build and test the practical model, and save costs and time.

This paper consists of eight sections and is outlined as follows: The first section is the introduction; the second is the proposed mechanical design and working principle of the robot arm. The 3rd section is the robot arm kinematic model development. The fourth section is the DC servo motor model development, and Simulink model development of the robot arm, mentioning friction equations. The fifth section is showing the inverse kinematics analysis and simulation of a 5-DOF robot arm using Matlab. And robot arm prototype and control circuit are presented in Section 6. Control system design, including real-time control, is mentioned in Section 7, and the conclusions are given in Section 8, followed by references.

1. PROPOSED MECHANICAL DESIGN AND WORKING PRINCIPLE OF ROBOT ARM

The mechanical design of the robot arm is a manipulator that mimics the functions of a human arm, [33 - 35]. The robot arm is composed of five links connected by rotational joints to enable movement. Each joint provides the robot with a certain degree of mobility, known as degrees of freedom. The manipulator is considered a part of a kinematic chain, [36]. The robot's base is connected to one end of the chain, while the other end is attached to a tool such as a hand, gripper, or end effector.

1.1 Mechanism Design in SolidWorks

SolidWorks is selected for creating the robot arm due to its capability for simultaneous design and visualization, along with its ability to assess collisions and interferences. Given that each link relies on the previous one, the design of the robot arm must commence at the base and end at the gripper. Thus, the base (Link 0) was the initial component designed, followed by Links 1, 2, 3, and so forth. The proposed material for the robot arm was wood sheet, with a thickness of 5 mm, considering a

load of 150 mg that the robot arm could support and manipulate. The arm features five rotating joints and a movable grip. Fig. 1 depicts a SolidWorks model of the robot arm.

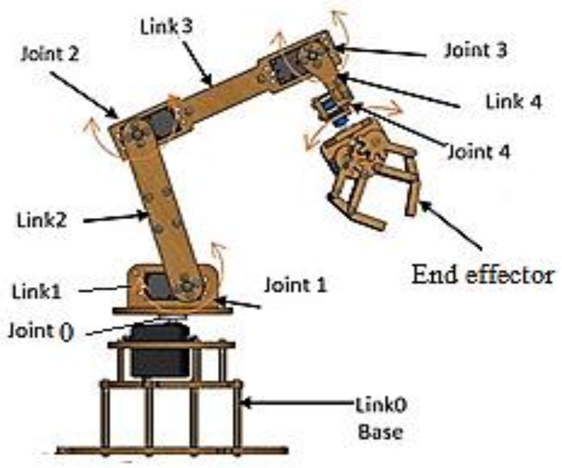


Fig. 1 Robot arm model in SolidWorks.

2. ROBOT ARM KINEMATIC MODEL DEVELOPMENT

Kinematics studies the motion of a body without considering its mass or the forces acting on it. In kinematics, position, velocity, acceleration, and all higher-order derivatives of position variables are studied, [37]. The robot arm is a serial-link manipulator that comprises a chain of mechanical links and joints. Each joint can move its outward neighboring link with respect to its inward neighbor. One end of the chain is the base, which is generally fixed, and the other end is free to move in space and holds the tool or end-effector. As the arm robot has multiple joints, the position of the end-effector will be a complex function of the state of each joint.

2.1 Forward Kinematics Model Development

Calculating the position and orientation of the end-effector in terms of the joint variables is known as forward kinematics (FK). To obtain the FK equations for the manipulator, the following steps must be done: Obtain the Denavit-Hertenberg (D-H) parameters, [37 - 39]: D-H parameters are the most common method for describing the manipulator FK. A coordinate frame is attached to each joint to determine the D-H parameters, as shown in Fig. 2, where the colors red and blue denote all things associated with links *i−1* and *i*, respectively. The sequence in which the elementary transforms are applied is shown by the numbers in circles. Where the four quantities(θ_i, d_i, a_i, α_i) are the parameters of link *i* and joint *i*. The parameters are given the following names: θ_i (link angle) is the angle between x_{i-1} to x_i measured about z_i, d_i (link offset) is the distance from x_{i-1} to x_i, measured along z_i, a_i (link length) is the distance from z_i to z_{i-1} measured along x_{i-1}, and α_i (link twist) is the angle between z_{i-1} to z_i measured about x_i. For a revolute joint θ_i is the joint variable and d_i is constant, while for a prismatic joint d_i is variable, and θ_i is constant. In many of the formulations that follow a generalized coordinate q_i is used.

The FK [40] shows the transition from one frame to the next, starting at the base and ending at the end-effector. Based on Fig. 2 using the D-H method, where each T_i

homogeneous transformation between the two adjacent frames is depicted as the product of four basic transformations, as in Equation (1), [15].

T_i = Rot (z, θ_i) Trans (z, d_i) Trans (x, a_i) Rot (x, a_i) (1)

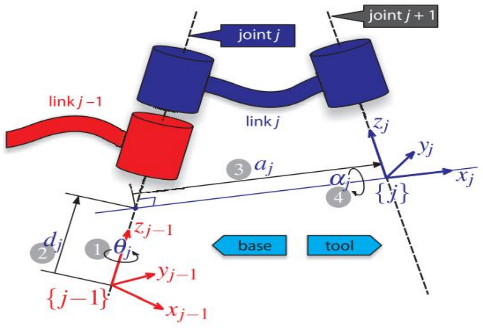


Fig. 2 Standard D-H parameters for a link.

Where the notation Rot (x, α_i) stands for rotation about the x_i axis by α_i , Trans (x, a_i) is the translation along the x_i axis by a distance a_i, Rot (z, θ_i) represents rotation by θ_i about the z_i axis, and Trans (z, d_i) represents translation along the z_i axis by a distance d_i. Hence, Equation (1) is rewritten. And the general transformation matrix is the standard D-H paramter matrix $^{i-1}T_i$ as in Equations (2) and (3).

$$^{i-1}T_i = \begin{bmatrix} \cos_{\theta i} & -\sin_{\theta i} & 0 & 0 \\ \sin_{\theta i} & \cos_{\theta i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos_{\alpha i} & -\sin_{\alpha i} & 0 \\ 0 & \sin_{\alpha i} & \cos_{\alpha i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

$$^{i-1}T_i = \begin{bmatrix} \cos_{\theta i} & -\cos_{\alpha i} * \sin_{\theta i} & \sin\alpha i * \sin\theta i & a_i * \cos_{\theta i} \\ \sin_{\theta i} & \cos_{\theta i} * \cos\alpha i & \sin_{\alpha i} & a_i * \sin_{\theta i} \\ 0 & \sin_{\alpha i} & \cos_{\alpha i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

The transformation matrix between each two successive frames can be formulated as follows: After each link's D-H coordinate system has been constructed, a homogeneous transformation matrix may be simply created using frame{i-1} and frame {i}. This transformation consists of five basic transformations as below. Getting the last matrix $^0T_5$ means that the position and orientation of the end-effector with respect to the base can be extracted. All the $^0T_1$, $^1T_2$, $^2T_3$, $^3T_4$, and $^4T_5$ will be found as in Equations from (4) to (8).

$$^0T_1 = \begin{bmatrix} C_{\theta 1} & 0 & S_{\theta 1} & 0 \\ S_{\theta 1} & 0 & -C_{\theta 1} & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4)$$

$$^1T_2 = \begin{bmatrix} C_{\theta 2} & -C_{(90)}S_{\theta 2} & 0 & L_2C_{\theta 2} \\ S_{\theta 2} & -C_{(90)}S_{\theta 2} & -S_{(90)}C_{\theta 2} & L_2C_{\theta 2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5)$$

$$ {}^{2}T_{3} = \begin{bmatrix} C_{\theta 3} & -S_{\theta 3} & 0 & L_3 C_{\theta 3} \\ S_{\theta 3} & C_{\theta 3} & 0 & L_3 S_{\theta 3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6) $$

$$ {}^{3}T_{4} = \begin{bmatrix} C_{\theta 4} & -S_{\theta 4} & L_4 S_{\theta 4} & 0 \\ S_{\theta 4} & 0 & -L_4 c_{\theta 4} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7) $$

$$ {}^{4}T_{5} = \begin{bmatrix} C_{\theta 5} & -S_{\theta 5} & 0 & 0 \\ S_{\theta 5} & C_{\theta 5} & 0 & 0 \\ 0 & 0 & 0 & L_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8) $$

Then ${}^{0}T_5$ could be formed by matrix multiplication of the individual link matrices. Starting by multiplying ${}^{4}T_5$ and ${}^{3}T_4$, ${}^{2}T_3$ which is multiplied by ${}^{1}T_2$ and so on until ${}^{0}T_5$ is obtained as in Equations (9) and (10), [4]:

$$ T_H = {}^{0}T_5 = {}^{0}T_1 . {}^{1}T_2 . {}^{2}T_3 . {}^{3}T_4 . {}^{4}T_5 \qquad (9) $$

And by multiplying the expanded matrices, the total transformation matrix of the robot is:

$$ {}^{B}T_W = {}^{0}T_5 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (10) $$

Where: $n_x = c12*c345$, $n_y = s12*c345$, $n_z = s345$, $o_x = s12$, $o_y = -c345$, $o_z = s12$, $a_x = -c12*s345$, $a_y = s12*s345$, $a_z = -c345$, $P_x = s12*d5 = c12*a4*c34 + c12*a3*c3$, $P_y = -c12*d5 + s12*a4*c34 + s12*a3*c3$, and $P_z = a3*s34 + a3*s3 + d1$.

Where Px, Py, and PZ are the global coordinates specifying the end effector's spatial position.

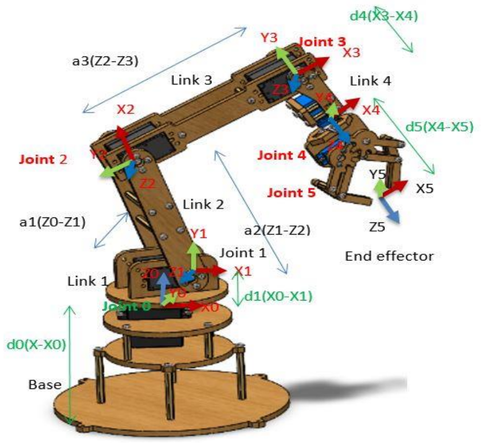
p_x : Position of the end-effector in x-direction = $a_i C_{\theta i}$

p_y : Position of the end-effector in y-direction= $a_i S_{\theta i}$

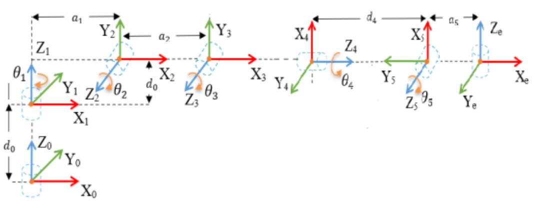
p_z: Position of the end-effector in z-direction

Using Matlab programming to multiply the individual matrices.

Where: c_n :cos(θ_n) , and s_n : sin(θ_n).



(a)



(b)

Fig. 3 (a) Links coordinates' diagram of the robot arm, (b) Link coordinate diagram of the robot arm aligned in the x-axis.

Using Equation (10), knowing the robot variables (θ_1, θ_2, θ_3, θ_4, θ_5) then $^0T_5$ will be identified, and the position and orientation of the robot wrist relative to the base frame will be known. Fig. 3(a) shows the links coordinates' diagram of the robot arm, Fig. 3(b) shows the five linkage arm that start to align to the x-axis. A1, A2, A3, and A4 are the lengths of the links, accordingly.

As shown in the Fig. 3, the first link advances by θ_1, the second link by θ_2, the third link by θ_3, the fourth link by θ_4, and the fifth link by θ_5, and the kinematic model with frame assignments based on D-H parameters is presented. Table 1 shows the kinetic parameters calculated using this model. While Table 2 shows the link lengths of the robot arm.

Table 1 The link parameters of robot arm manipulator (D-H parameters).

Joint	α_{i-1} (°)	a_{i-1}(mm)	d_{i-1}(mm)	θ_{i-1}(°)
1	α1 =90	a1	L1	0
2	α2 = 0	a2	0	θ_2
3	α3 = 0	a3	0	θ_3
4	α4 =-90	a4	0	θ_4
5	α5 = 0	0	d_5	θ_5
6	α6 = 0	0	0	Gripper

Table 2 The link lengths of robot arm 5-DOF

Link Joint	Waist	Shoulder	Elbow	wrist
Symbol	a1	a2	a3	a4
Link length (mm)	126.9	122	142	153

3.2 Inverse Kinematics Model Development

Inverse kinematics (IK) is concerned with the inverse problem of finding the joint variable in terms of the end-effector position and orientation. IK is used to determine the required joint angles of the robot arm to achieve the specified position and orientation of the end-effector. In this work, the geometric approach, [41] was used to solve the IK of the 5-DOF robot arm. Figure 4 shows the IK of a planar manipulator with 2 links and with 3 links.

$$X=a_1c_1+a_2c_{12} \quad (11)$$

$$Y=a_1s_1+a_2s_{12} \quad (12)$$

squaring and adding Equations (11) and (12) then :

$$x_2+y_2 = a_1^2+ a_2^2+ 2a_1a_2\ [c_1c_{12}+s_1s_{12}] = a_1^2 + a_2^2+2a_1a_2c_2 \quad (13)$$

$$C2 = \frac{x^2+y^2-(a_1^2+a_2^2)}{2a_1a_2} \quad (14)$$

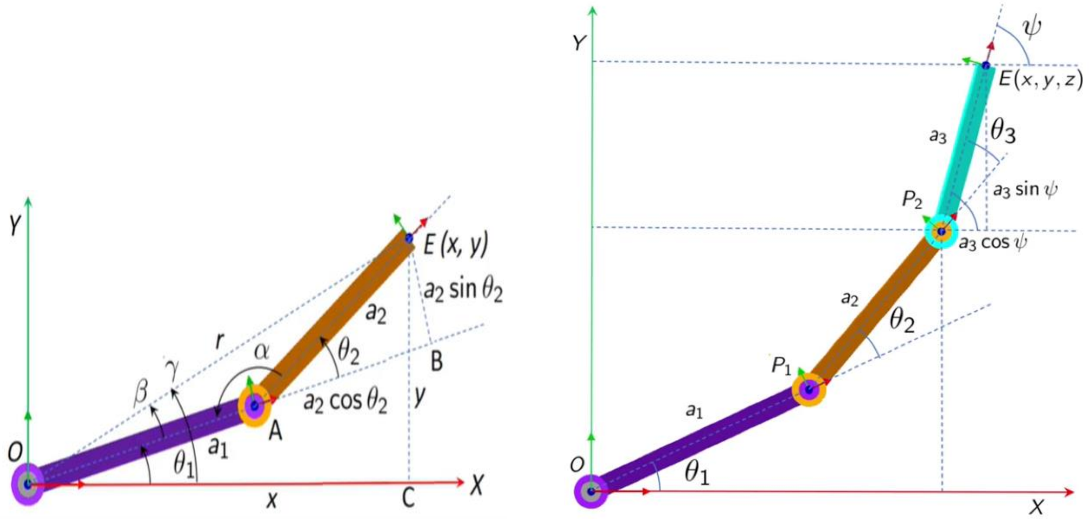
Finally

$\theta_2 = \cos^{-1} [\frac{x^2+y^2-(a1^2+a2^2)}{2a_1a_2}]$ (15)

For triangle OEC :$r^2 = x^2 + y^2$ (16)

Using parallellelogram low [42] as in Fig. 5:

$r^2 = a_1^2 + a_2^2 - 2a_1a_2 \cos \alpha$ (17)



(a) **(b)**

Fig. 4 Inverse kinematics of planar manipulator: (a) with 2 links, and (b) with 3 links.

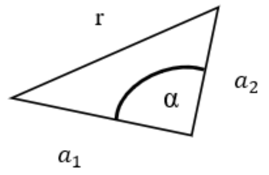


Fig. 5 Parallellelogram low explanation.

$\cos \alpha = \frac{a_1^2+a_2^2-r^2}{2a_1a_2}$ (18)

(or directly using cosine rule)

Since $\theta_2 = \Pi - \alpha$ (19)

Cos θ_2 **= - cos** α (20)

Cos $\theta_2 = \frac{x^2+y^2-(a_1^2+a_2^2)}{2a_1a_2}$ (21)

$\theta_2 = \cos^{-1} [\frac{x^2+y^2-(a1^2+a2^2)}{2a_1a_2}]$ (22)

For triangle OEB:

$\tan \beta = \frac{a_2 \sin \theta_2}{a_1+a_2 \cos \theta_2}$ (23)

For triangle OEC:

$\tan \Upsilon = \frac{y}{x}$ (24)

$B = \tan^{-1} \frac{a_2 \sin \theta_2}{a_1+a_2 \cos \theta_2}$ (25)

and$\Upsilon = \tan^{-1} \frac{y}{x}$ (26)

using $\theta = \Upsilon - \beta$, $\theta_1 = \tan^{-1}\frac{y}{x} - \tan^{-1}\frac{a_2 \sin \theta_2}{a_1 + a_2 \cos \theta_2}$ (27)
and:

$$\theta_2 = \cos^{-1}\left[\frac{x^2+y^2-(a1^2+a2^2)}{2a_1a_2}\right] \quad (28)$$

End effector pose is given by : E (x, y, ϕ) (29)

$$p_{2x} = x - a_3 \cos \phi \quad (30)$$

$$p_{2y} = y - a_3 \sin \phi \quad (31)$$

As $p_2(p_{2x}, p_{2y})$ is now known, and the remaining system is 2R – planar manipulator, then substituting ɵ1 and ɵ2 from Equations (27) and (28) in to Equation (32).

$$\theta_1 + \theta_2 + \theta_3 = \phi \quad (32)$$

$$\theta_3 = \phi - (\theta_1 + \theta_2) \quad (33)$$

thus θ_1 , θ_2 and θ_3 are all kown.

$$\theta_4 = \theta_{234} - \theta_2 - \theta_4 \quad (34)$$

$$\theta_5 = c_{234} \, . \, \theta_2 - 2 \operatorname{atan}(X_{5Y}, Y_{5X}) \quad (35)$$

Hence, by kowing $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ the joints' positions are known.

4. SIMULINK MODEL DEVELOPMENT OF ROBOT SYSTEM WITH FRICTION

4.1 Friction in 5-DOF Robot Arm

Friction has a major impact on how well a 5-DOF robot arm performs, how accurate the control is, and how much energy it consumes. Because friction forces are nonlinear and the robot arm is made up of numerous joints and connections, simulating friction in robotic systems can be challenging. The primary forms of friction that impact a robotic arm's functionality are: 1) Static friction (the resistance to motion that needs to be overcome). 2) Dynamic friction (the resistance to motion that arises after it has begun). 3) Viscous friction: proportionate to velocity, in which the moving part's velocity causes the frictional force to rise. 4) The Stribeck effect is a nonlinear friction phenomenon in which the frictional force decreases at small velocities and increases with speed as it increases.

The joints, which have moving components like motors, gears, and bearings, are where friction mostly arises. Complex friction forces act on the robotic arm's joints which enable movement and rotation. Static friction, viscous friction, and the Stribeck effect can all be used to quantitatively represent friction for each joint. Assume, for simplicity, that the robot joints are mainly rotational; however, in certain situations, linear friction may also be relevant. Next, a combination of these friction models can be used to express the total friction torque (τ_f) in a joint as in Equation (36):

$$\tau_f = \tau_c \sin(\dot{\theta}) + b\dot{\theta} + (\tau_s - \tau_c)\, e^{-\left(\frac{|\dot{\theta}|}{\dot{\theta}_s}\right)^2} \quad (36)$$

Where: τ_f is the total friction torque, τ_c is the Coulomb friction torque, *b* is the viscous friction coefficient, $\dot{\theta}$ is the angular velocity, τ_s is the stiction (static friction) torque, $\dot{\theta}_s$ is the Stribeck velocity.

Because friction introduces uncertainties and nonlinearities into the control system, it has an impact on control performance. Overcoming stiction, or static friction, at the start of motion is a crucial task in managing a robot arm with friction. This friction frequently results in jerky motion, particularly at low speeds. Furthermore,

energy is lost due to viscous friction, necessitating greater actuator torque to maintain target velocities. Compensation techniques are employed, such as model-based friction compensation, which entails determining friction characteristics and incorporating them into the control loop, to enhance control performance. Robot arm movement can be less affected by friction by using an adaptive controller or a feedforward controller, [43 - 45].

When determining the necessary motor torque to produce the desired motion, friction torque at the joints must be taken into consideration. The load that the motor must overcome is increased by the overall friction torque, which opposes motion. As a result, in addition to compensating for the other dynamic forces operating on the system, the total motor torque must also account for the total friction torque. The torque required to generate the intended motion at a joint, taking into account friction, inertia, gravitational forces, and Coriolis forces, is known as the total motor torque, or τm. Equation (37) represents the relationship between friction torque and motor torque:

$$\tau_m = \tau_{dynamic} + \tau_f \tag{37}$$

Where τ_{dynamic} is the torque needed to counteract dynamic influences including centrifugal, inertial, and Coriolis forces in addition to gravity, and τ_f is the total friction torque that opposes motion. τ_m is the total torque that the motor needs to deliver.

Friction raises energy consumption in addition to affecting control precision. In order to accomplish the necessary motion, the motors must produce more torque due to increased friction in the joints. The friction-related power loss, P_f, can be written as in Equation (38):

$$P_f = \tau_f \dot{\theta} \tag{38}$$

This formula demonstrates that power loss rises as joint velocity and friction torque increase. The robotic arm's efficiency can be increased by minimizing energy losses and utilizing advanced control techniques, lubricating the joint to reduce friction, or improving the mechanical design [43 - 45].

4.2 DC Servo Motor Model Development

Direct current (DC) motor is a common actuator in industrial applications such as industrial and educational robots, [37, 46]. The position, speed, and torque control of a DC servo motor is an essential issue when building a robot arm. The DC servo motor mathematical model, [47 - 48] will be brifly presented in the following. Studying the electrical features of DC motor: motor coils generate torque in the armature when a voltage is applied. The torque created by the motor, τ_m, is proportional to the armature current, i_a, and can be found as in Equation (39),

$$\tau_m(t) = K_t\, i_a(t)$$

where K_t is the torque constant.

The armature windings' rotation in the fixed magnetic field generates the back electromotive force, or EMF voltage $v_b(t)$, whose polarity acts against the current that generates motion. A linear relation is provided by relating the EMF to the angular speed of the motor shaft $\omega(t)$ as in Equation (40),

$$v_b(t) = K_b\, \omega(t) \tag{40}$$

where K_b is the EMF constant.

The electrical equivalent of the armature circuit can be described as an inductance L_a in series with a resistance R_a, and R_a is in series with an induced EMF voltage that opposes the voltage source V_{in}. These differential equations form the mathematical model that describes the electric characteristics of the armature controlled DC motor. By summing the voltages across the R-L circuit, Kirchoff's law is applied around the electrical loop to yield, as shown in Equations (41):

$$\begin{gathered} \sum V = V_{in} - V_R - V_L - EMF = 0 \\ V_{in}(t) = R_a i_a + L_a \frac{di_a}{dt} + K_b \frac{d\theta(t)}{dt} \\ V_{in}(s) = R_a I(s) + L_a s I(s) + K_b s \theta(s) \\ I(s)(R_a + L_a s) = V_{in}(s) - K_b s \theta(s) \end{gathered} \tag{41}$$

In DC motor mechanical properties; the torque generated by the motor results in an angular velocity, denoted as $\omega(t) = d\theta(t)/dt$, based on the damping friction, b_m, and the inertia J_m of the motor and load. It is possible to develop a mathematical model of the DC motor system's mechanical properties in the form of differential equations by performing the energy balancing; as the torques' sum must equal zero, as shown in Equations (42):

$$\begin{gathered} \sum T = J_m * \alpha = J_m * d^2\theta/dt^2 \\ T_e - T\alpha - T\omega - T_{EMF} = 0 \end{gathered} \tag{42}$$

The following values can be substituted in an open loop DC motor system without a load applied, so that the change in τ_m is zero: $T_e = K_t i_a(t)$, $T\alpha = J_m * d^2\theta/dt^2$, and $T\omega = b_m * d\theta/dt$. Then, the Laplace transform is taken and Equation (42) is reaaranged, giving Equation (43):

$$\begin{gathered} K_t i_a(t) - T_{load} - J_m * d^2\theta/dt^2 - b_m * d\theta/dt = 0, \\ K_t * I(s) = (J_m * s + b_m) s \theta(s) \end{gathered} \tag{43}$$

To derive the transfer functions for DC motor open loop systems; first get $I(s)$ on the right side from Equation (41) to get Equation (44). Then, this value of I(s) is used to replace the DC mechanical characteristics in Equation (43), as follows in Equation (44):

$$I(s) = \frac{1}{La\, s + Ra} [V_{in}(s) - K_b \omega(s)] \tag{44}$$

Equation (45) represents the DC motor electric component transfer function that links armature current and voltage:

$$\frac{I(s)}{[V_{in}(s) - K_b \omega(s)]} = \frac{1}{L_a s + R_a} \tag{45}$$

Equation (46) represents the mechanical component transfer function of a DC motor in terms of input rotor speed and output torque:

$$\frac{\omega(s)}{[K_t I_a(s) - T_L \omega(s)]} = \frac{1}{J_m s + b_m} \tag{46}$$

In case, there is no load attached, $T_{load} = 0$, then:

$$\frac{\omega(s)}{[K_t I_a(s)]} = \frac{1}{J_m s + b_m} \tag{47}$$

Currently, replacing Equation (43) in Equation (42) yields:

$$K_t \left[\frac{1}{La\, s + Ra}\right] [V_{in}(s) - K_b \omega(s)] = J_m s^2 \theta(s) + b_m s \theta(s)$$
(48)

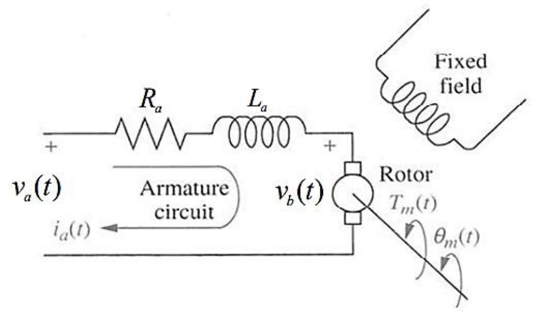
Rearranging Equation (48) to get the DC motor open loop transfer function with no load applied, which is provided by coupling the motor shaft output angle, $\theta_m(s)$, to the input voltage, $V_{in}(s)$ as follows.

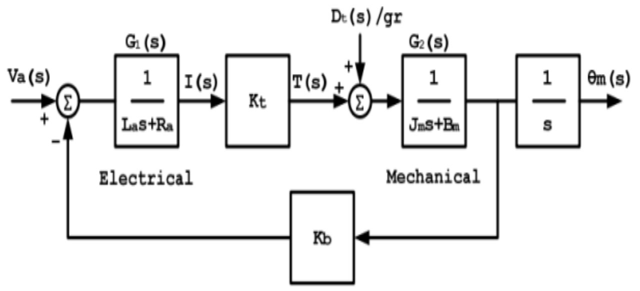
$$G_{angle}(s) = \frac{\theta(s)}{V_{in}(s)} = \frac{K_t}{[L_a J_m s^3 + (R_a J_m + b_m L_a) s^2 + (R_a b_m + K_t K_b) s]} \quad (49)$$

The open loop transfer function of a DC motor, which links the motor shaft output angular velocity, $\omega(s)$, to the input voltage, $V_{in}(s)$, is as follows:

$$G_{speed}(s) = \frac{\omega(s)}{V_{in}(s)} = \frac{K_t}{[L_a J_m s^2 + (R_a J_m + b_m L_a) s + (R_a b_m + K_t K_b)]} \quad (50)$$

Table 3 shows the DC motor parameters' values chosen for motor simulation. The schematic diagram in Fig. 6(a) is modeled as a block diagram in Fig. 6(b). This block diagram represents an open loop system, and the motor has built-in feedback EMF, which tends to reduce the current flow.





(a) (b)

Fig. 6 (a) Schematic diagram of DC motor system, (b) Block diagram for DC motor system.

The benefit of using a block diagram is that it provides a clear image of the transfer function relation between each block of the system. Therefore, based on the block diagram in Fig. 6(b). Equation (51) depicts the total transfer function of a DC servo motor system:

$$\frac{\theta_m(S)}{vt(s)} = \frac{0.884}{3.315e^{-08} S^3 + 0.0001856\ S^2 + 1.016\ S} \quad (51)$$

Where: v_t is the voltage input, θ_m is the angle output, and S is the Laplace variable.

Using Simulink, the model of the motor may be created. This model includes all the parameters derived previously. Fig. 7 shows the Simulink model of a DC motor.

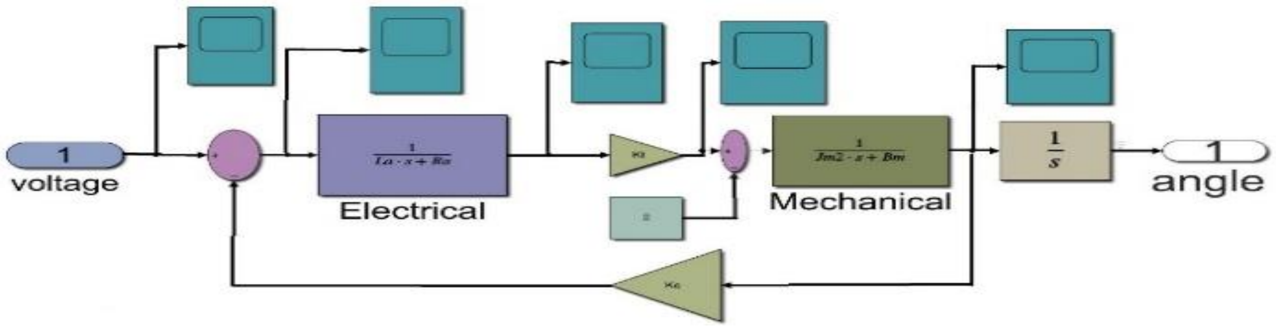
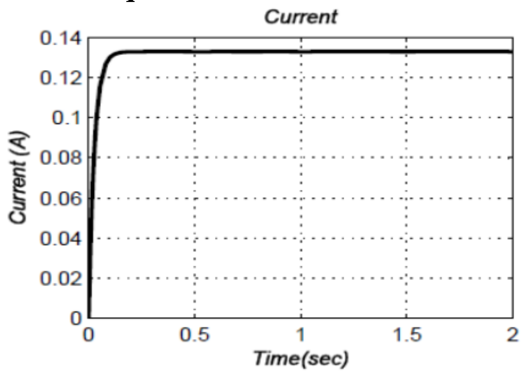


Fig. 7 DC motor subsystem using Simulink.

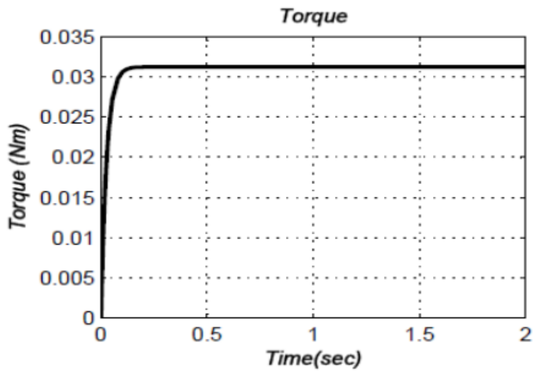
Table 3 The values and DC motor parameters selected for the motor simulation.

Parameters	Values	Unit
Moment of inertia (b_m)	0.0243	kg.m2
Friction coefficient (J_m)	0.000026852	N.ms
Back EMF constant (K_b)	1.058	v/ms-1
Torque constant (K_t)	0.884	Nm/A
Electric resistance (R_a)	3.33	Ohm
Electric inductance (L_a)	0.000000015	H

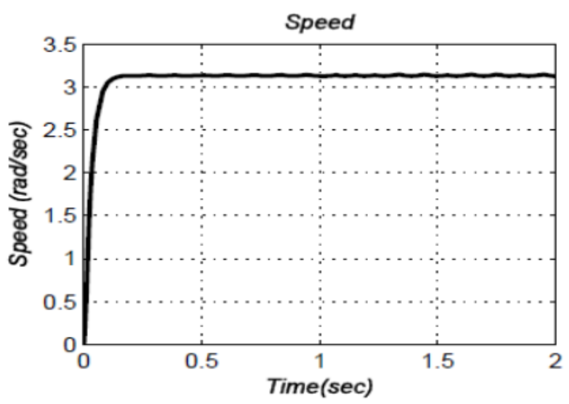
Simulation results using Simulink are shown in Figs. 8, 9 for the DC motor model without and with load disturbance. Simulation results demonstrated that, the motor was running at no-load conditions at startup, and still running to reach the steady state value as shown in Fig. 8. When a mechanical load is applied suddenly to the shaft as shown in Fig. 9, a small no-load current does not produce enough torque to carry the load; thus, the motor starts to slow down. This causes counter EMF to diminish resulting in a higher current and a correspondingly higher torque. When the torque developed by the motor is exactly equal to the torque imposed by the mechanical load, then the speed will remain constant. However, for the motor to move the robot arm to a proper angular position corresponding to the input, this can be achieved by using the control technique as a PID controller.



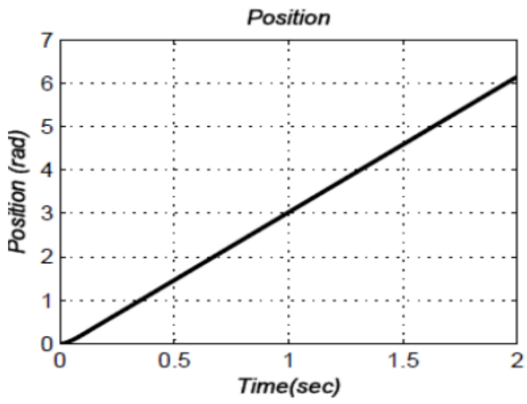
(a)



(b)

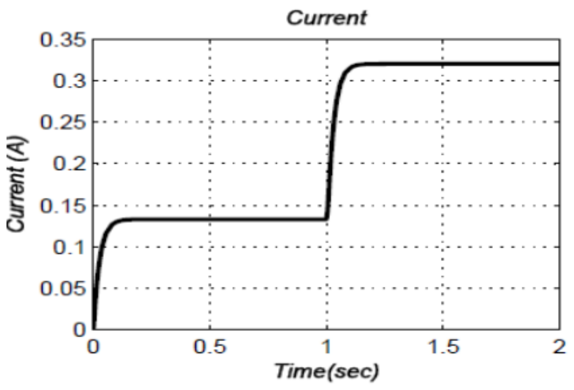


(c)

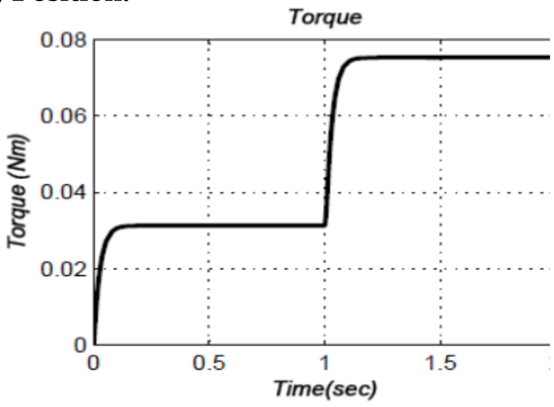


(d)

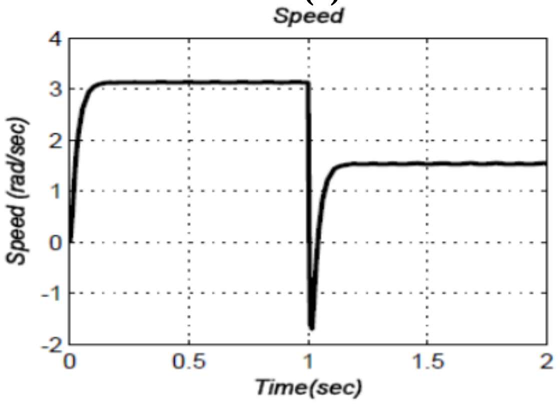
Fig. 8 DC motor open loop step response without load:(a) current, (b) Torque, (c) Speed, and (d) Position.



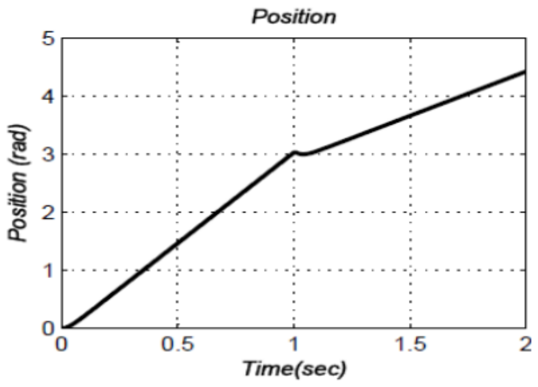
(a)



(b)



(c)



(d)

Fig. 9 DC motor model simulation with load disturbance :(a) Current, (b) Torque, (c) Speed, and (d) Position.

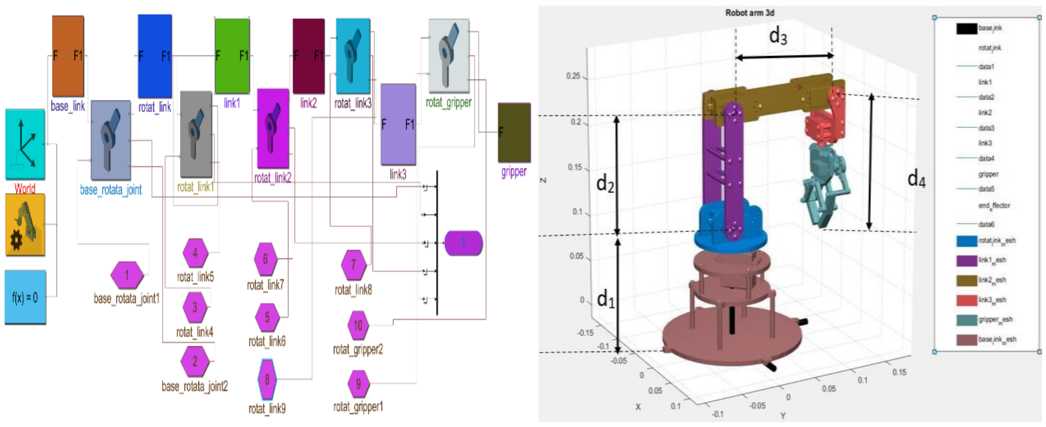
4.3 Simulink Model Development of Robot Arm

All software used in this work depends on the Matlab/Simulink program. Matlab, is an acronym for Matrix Laboratory, Matlab is a multifunctional computer program designed to facilitate engineering and scientific computations. It employs the Matlab programming language and offers a vast array of pre-built functions to streamline and simplify technical programming tasks. Matlab has many advantages, such as ease of use, platform independence, predefined functions are available, device-independent plotting, and a graphical user interface. However, Matlab is an

interpreted language. The main disadvantage of interpreted languages is execution speed. Simulink is an add-on product to Matlab, provides an interactive, graphical environment for modeling, simulating, and analyzing dynamic systems. It makes it possible to quickly and easily create virtual prototypes in order to investigate design concepts at any level of detail. Simulink offers a graphical user interface (GUI) for creating block diagram models. It comes with an extensive library of pre-made blocks that may be dropped to create graphical models of systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or hybrid of the two. Simulink is integrated with Matlab, and data can be easily shared between the two programs, [49].

The robot arm model developed in SolidWorkswas is transferred into Matlab using the option "Export- simscape multi-body first generation" where the file is a CAD assembly file exported to Matlab as an XML file format based on the Sim-Mechanics library. Simulation is performed to verify the design dynamics in the form of a Simulink model containing the blocks as shown in Fig. 10(a). The blocks imported into Simulink are rearranged according to engineering laws, physical laws, and the required assembly. The input block, system block, and output block are the three primary block types in the Simulink model. The input block shows a simulation of the reference input signals (position of joints) designed to obtain the robot arm's temporal response (gripper position) before adding the controls. System bocks represent robot arm links and joints and the final gripper. The output block represents the gripper position. Fig. 10(b) shows a 3D visualization of the system blocks.

FK and IK are the two kinematics solutions for robotic arms. FK stands for conversion from configuration to cartesian space. In contrast, IK stands for transformation from cartesian space to joint space. Robots FK are simulated in this work by calculating the end-effector coordinates knowing their joint angles using the D-H parameters presented in Table 1. Further, a 3D conception of the system blocks is shown in Fig. 10(b).



(a) (b)

Fig. 10 Robot arm model in Simulink, (a) Model blocks, (b) 3D visualization of the system blocks.

5. INVERSE KINEMATICS ANALYSIS AND SIMULATION OF A 5-DOF ROBOT ARM USING MATLAB

Matalb Graphical User Interface (GUI) was used in this work to perform many functions, as shown in Fig. 11. It illustrated the inputs to the program, which were the position of the end-effector, while the outputs were the joint angles (joints' position). Also, a 3D simulation of the robot arm movement was presented in GUI as an output. The positions (px, py, and pz) of the goal were set relative to the base of the robotic arm as inputs. To determine the joint angles of the robot arm, the Equations (11) to (35) were used. The position of each joint of the robot arm was determined using the Equations from (2) to (10). 3D simulation of the movement of the robot arm was applied using the Robotics toolbox in Matlab, depending on the D-H parameters.

To test the efficiency of the geometric approach in the IK analysis, several cases were studied. The inputs to the GUI program, shown in Fig. 11, were the position of the goal (px, py, and pz), and the outputs were the joint angles of the robotic arm. The 3D representation of the robotic arm depending on the D-H parameters is shown in Fig. 12. The 3D representation illustrates four cases. The extracted angles of joints were applied in Equation (10) to determine the position of the end-effector. The algorithm of the geometric approach was developed for the IK analysis of the robotic arm, and the results of this method were compared to results extracted using the algebraic solution. The errors between them were obtained and listed in Table 4 to indicate the efficiency of the program used. A large match was found, so the geometric approach method was applied.

Table 4 indicated a little small error occurred in the position of the end-effector of the robotic arm that confirms the effectiveness of the program used, where the maximum error for the end-effector's x, y, and z coordinates were, respectively, 0.0239%, 0.1085%, and 0.0251%.

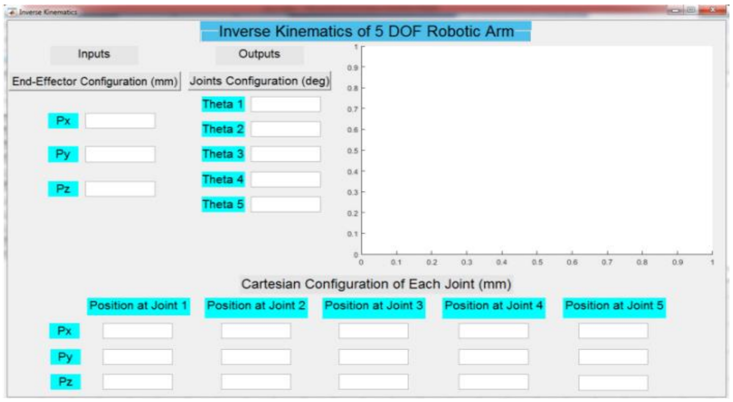
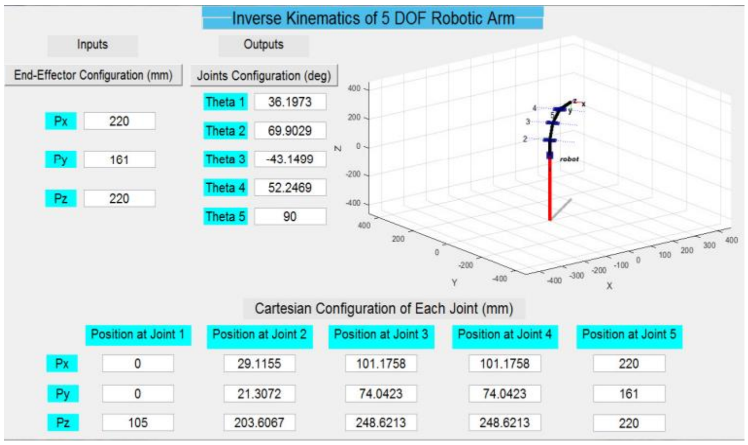


Fig. 11 The window of the GUI program.



Inverse Kinematics of 5 DOF Robotic Arm

Inputs — End-Effector Configuration (mm)

Px	220
Py	161
Pz	220

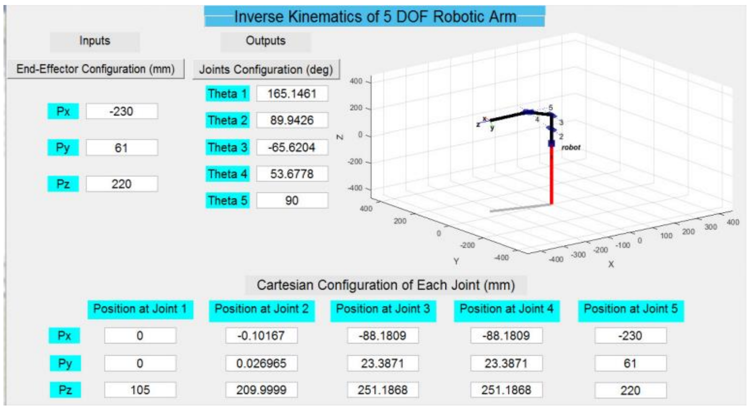
Outputs — Joints Configuration (deg)

Theta 1	36.1973
Theta 2	69.9029
Theta 3	-43.1499
Theta 4	52.2469
Theta 5	90

Cartesian Configuration of Each Joint (mm)

	Position at Joint 1	Position at Joint 2	Position at Joint 3	Position at Joint 4	Position at Joint 5
Px	0	29.1155	101.1758	101.1758	220
Py	0	21.3072	74.0423	74.0423	161
Pz	105	203.6067	248.6213	248.6213	220

Case (1)



Inverse Kinematics of 5 DOF Robotic Arm

Inputs — End-Effector Configuration (mm)

Px	-230
Py	61
Pz	220

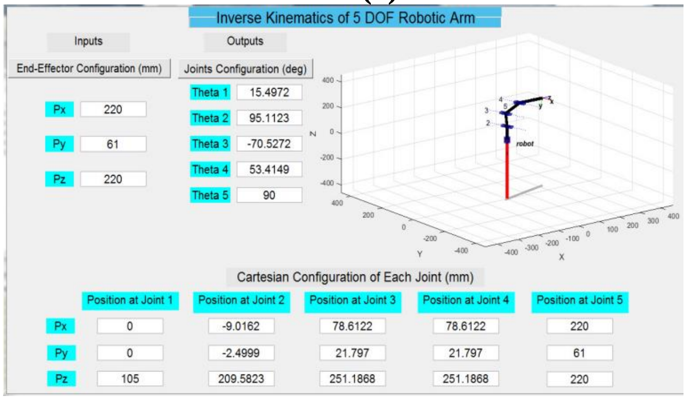
Outputs — Joints Configuration (deg)

Theta 1	165.1461
Theta 2	89.9426
Theta 3	-65.6204
Theta 4	53.6778
Theta 5	90

Cartesian Configuration of Each Joint (mm)

	Position at Joint 1	Position at Joint 2	Position at Joint 3	Position at Joint 4	Position at Joint 5
Px	0	-0.10167	-88.1809	-88.1809	-230
Py	0	0.026965	23.3871	23.3871	61
Pz	105	209.9999	251.1868	251.1868	220

Case (2)



Inverse Kinematics of 5 DOF Robotic Arm

Inputs — End-Effector Configuration (mm)

Px	220
Py	61
Pz	220

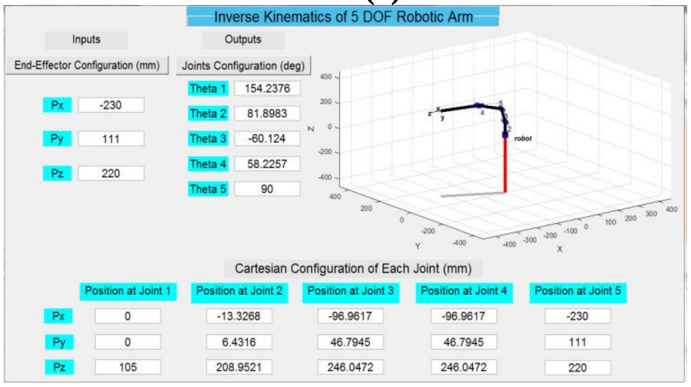
Outputs — Joints Configuration (deg)

Theta 1	15.4972
Theta 2	95.1123
Theta 3	-70.5272
Theta 4	53.4149
Theta 5	90

Cartesian Configuration of Each Joint (mm)

	Position at Joint 1	Position at Joint 2	Position at Joint 3	Position at Joint 4	Position at Joint 5
Px	0	-9.0162	78.6122	78.6122	220
Py	0	-2.4999	21.797	21.797	61
Pz	105	209.5823	251.1868	251.1868	220

Case (3)



Inverse Kinematics of 5 DOF Robotic Arm

Inputs — End-Effector Configuration (mm)

Px	-230
Py	111
Pz	220

Outputs — Joints Configuration (deg)

Theta 1	154.2376
Theta 2	81.8983
Theta 3	-60.124
Theta 4	58.2257
Theta 5	90

Cartesian Configuration of Each Joint (mm)

	Position at Joint 1	Position at Joint 2	Position at Joint 3	Position at Joint 4	Position at Joint 5
Px	0	-13.3268	-96.9617	-96.9617	-230
Py	0	6.4316	46.7945	46.7945	111
Pz	105	208.9521	246.0472	246.0472	220

Case (4)

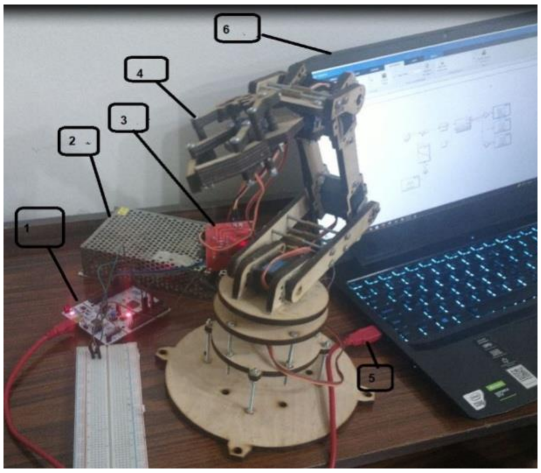
Fig. 12 GUI for four cases.

Table 4 The error of the end-effector position.

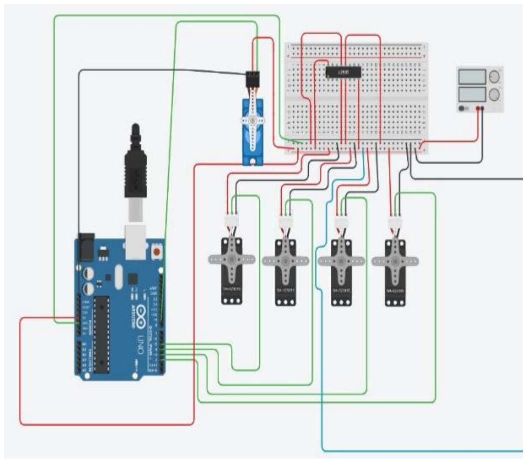
Case No.	True end effector position (mm)	Joint angle measured (degree)	True end effector position (mm)	Absolut error=((true-measured)/true)*100%
1	P_x=220 P_y=161 P_z=220	θ_1=36.1973 θ_2=69.9029 θ_3=-43.1499 θ_4=52.2469, θ_5=90	P_x=219.9986 P_y=161.0146 P_z=219.9934	0.0006 0.009 0.003
2	P_x=-230 P_y=61 P_z=220	θ_1=165.1461 θ_2=89.9426 θ_3=-65.6204 θ_4=53.6778, θ_5=90	P_x=-230.0103 P_y=60.9861 P_z=219.9964	0.0045 0.023 0.00016
3	P_x=220 P_y=61 P_z=220	θ_1=15.4972 θ_2=95.1123 θ_3=70.5272 θ_4=53.4149, θ_5=90	P_x=220.0553 P_y=60.9854 P_z=220.2386	0.0251 0.0239 0.1085
4	P_x=-230 P_y=111 P_z=220	θ_1=154.2376 θ_2=81.8983 θ_3=-60.124 θ_4=58.2257, θ_5=90	P_x=-230.0493 P_y=111.0121 P_z=220.1713	0.0214 0.0109 0.0778

6. ROBOT ARM PROTOTYPE AND CONTROL CIRCUIT

The robot arm prototype was manufactured from wood according to the design developed for this work, and the main parts are shown in Fig. 13 (a). Additionally, a schematic diagram of the control circuit is presented in Fig. 13 (b). Table 5 provides details on the control circuit components and their specifications.



(a)



(b)

Fig. 13 (a) Real photo of robot arm's control circuit;1: Arduino Mega 2560 board, 2: Power supply, 3: Servo motor driver, 4: Robot arm, 5: USB 2.0 cable, 6: Laptop;(b)Schematic diagram of the control circuit.

Table 5: Specifications of control circuit components.

No.	Part	Qty.	Specifications	Function
1	Servo Motor (360)	5	Torque: 17 kg.cm Current: 130mA Step angel: 1.8 Motor speed: 800 r.p.m	Robot forward and backward motion.
2	Arduino Mega 2560 board	1	A microcontroller board based on the AT mega 2560, with 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator.	Control program development.
3	Dual H-bridge motor driver using L298N	5	L298N Dual H Bridge DC Stepper Motor Drive Controller Board Module for Arduino the L298 Stepper Controller makes it easy to drive either two dc motor or a bipolar stepper motor. This is a very high quality board and is very compact for designs where space really matters.	Contact the motor to the control circuit
4	Power supply	1	5 DC volt, 10 mA	DC power source
5	USB 2.0 cable	1	Data transfer speed is 480 Mb/sec	Connect the Arduino board to PC.
6	Adaptor	1	Input 100-240V, 50/60 Hz, 0.2A Output 5V	Connect the Arduino board to power source

7. CONTROL SYSTEM DESIGN

a. Controller Design

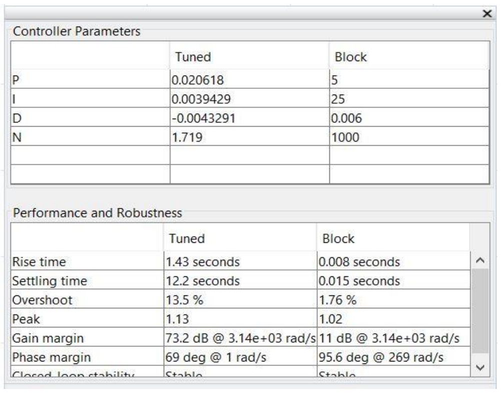
To control a motion process, the precise position of the object needs to be measured and maintained. The designed system should respond to the applied input with a suitable overshoot, settling time, and a zero steady state error as possible. Two controller will be used, PID and FL conrollers.

7.1.1 PID Controller Design

The PID algorithm is a widely used feedback controller in industry. It is a robust and easy-to-comprehend algorithm that provides excellent control effectiveness regardless of the dynamic behavior of a specific process plant.

$$G_{pid}(s) = k_p + k_i/s + k_d.s \tag{52}$$

The calculating algorithm offered in Equation (52) includes three separate coefficients: p, i, and d. These coefficients can be explained in terms of time, where p is for existing errors, and i is for the accumulation of past errors. Coefficient d means approaching error based on the uninterrupted pace of change [48]. Parameters of the PID controller were tuned using Simulink instead of traditional tuning methods like ZN method. Five PID controllers were developed, one for each DC servo motor associated with each joint. In order to assess the performance of the PID controller, simulations are presented. The parameters of the five developed PIDs are shown in Fig. 14.

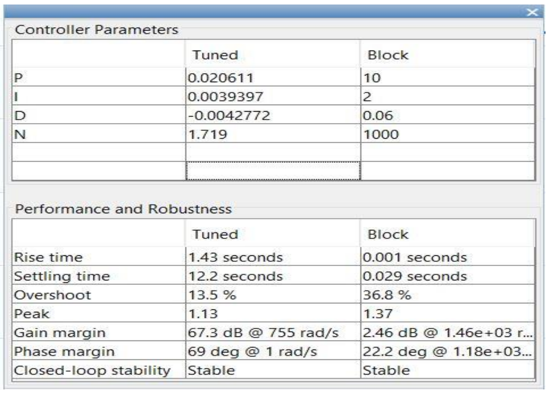

Controller Parameters

	Tuned	Block
P	0.020618	5
I	0.0039429	25
D	-0.0043291	0.006
N	1.719	1000

Performance and Robustness

	Tuned	Block
Rise time	1.43 seconds	0.008 seconds
Settling time	12.2 seconds	0.015 seconds
Overshoot	13.5 %	1.76 %
Peak	1.13	1.02
Gain margin	73.2 dB @ 3.14e+03 rad/s	11 dB @ 3.14e+03 rad/s
Phase margin	69 deg @ 1 rad/s	95.6 deg @ 269 rad/s
Closed-loop stability	Stable	Stable

(a)

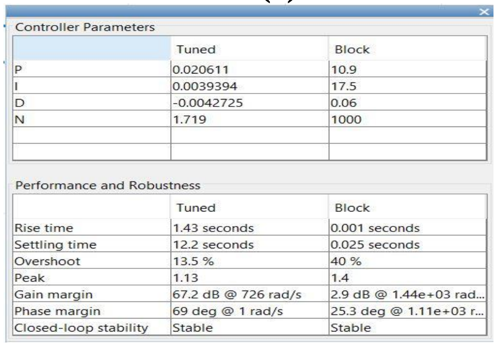

Controller Parameters

	Tuned	Block
P	0.020611	10
I	0.0039397	2
D	-0.0042772	0.06
N	1.719	1000

Performance and Robustness

	Tuned	Block
Rise time	1.43 seconds	0.001 seconds
Settling time	12.2 seconds	0.029 seconds
Overshoot	13.5 %	36.8 %
Peak	1.13	1.37
Gain margin	67.3 dB @ 755 rad/s	2.46 dB @ 1.46e+03 r...
Phase margin	69 deg @ 1 rad/s	22.2 deg @ 1.18e+03...
Closed-loop stability	Stable	Stable

(b)

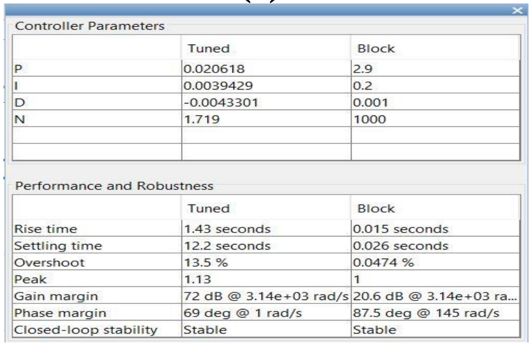

Controller Parameters

	Tuned	Block
P	0.020611	10.9
I	0.0039394	17.5
D	-0.0042725	0.06
N	1.719	1000

Performance and Robustness

	Tuned	Block
Rise time	1.43 seconds	0.001 seconds
Settling time	12.2 seconds	0.025 seconds
Overshoot	13.5 %	40 %
Peak	1.13	1.4
Gain margin	67.2 dB @ 726 rad/s	2.9 dB @ 1.44e+03 rad...
Phase margin	69 deg @ 1 rad/s	25.3 deg @ 1.11e+03 r...
Closed-loop stability	Stable	Stable

(c)

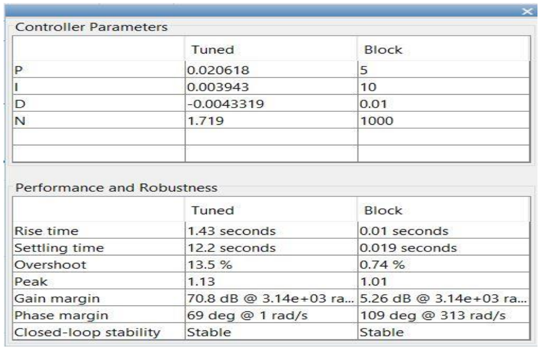

Controller Parameters

	Tuned	Block
P	0.020618	2.9
I	0.0039429	0.2
D	-0.0043301	0.001
N	1.719	1000

Performance and Robustness

	Tuned	Block
Rise time	1.43 seconds	0.015 seconds
Settling time	12.2 seconds	0.026 seconds
Overshoot	13.5 %	0.0474 %
Peak	1.13	1
Gain margin	72 dB @ 3.14e+03 rad/s	20.6 dB @ 3.14e+03 ra...
Phase margin	69 deg @ 1 rad/s	87.5 deg @ 145 rad/s
Closed-loop stability	Stable	Stable

(d)


Controller Parameters

	Tuned	Block
P	0.020618	5
I	0.003943	10
D	-0.0043319	0.01
N	1.719	1000

Performance and Robustness

	Tuned	Block
Rise time	1.43 seconds	0.01 seconds
Settling time	12.2 seconds	0.019 seconds
Overshoot	13.5 %	0.74 %
Peak	1.13	1.01
Gain margin	70.8 dB @ 3.14e+03 ra...	5.26 dB @ 3.14e+03 ra...
Phase margin	69 deg @ 1 rad/s	109 deg @ 313 rad/s
Closed-loop stability	Stable	Stable

(e)

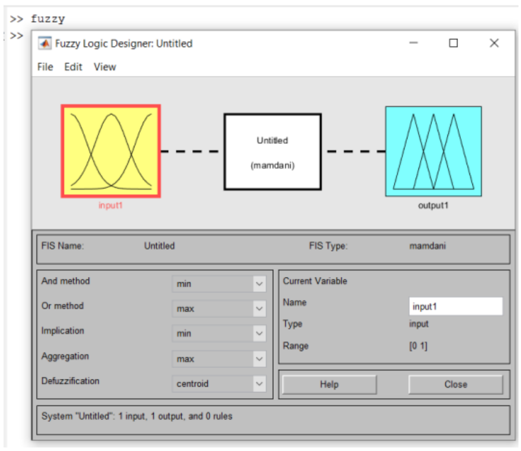
Fig. 14 Controller parameters of: (a) PID1, (b) PID2, (c) PID3, (d) PID4, and (e) PID5.

7.1.2 FL Controller Design

FLC controller system is easy to understand and design, and it performs better than other types of controllers. By executing basic principles guiding the system's behavior. FL enables the modeling of complicated systems that originate from information and mastery by combining substitutional ways of thinking using a higher level of the inference procedure, which is divided into four parts as follows: Fuzzification of input variables, rule evaluation, aggregation of rule outputs, and defuzzification [50-51]. The implementation of the FLC is as follows:1) Identifying FLC input and output. A FLC has two inputs: error E(t) and change of error ΔE(t), and one output, which is the designed control signal. 2) Fuzzifying input and output variables. Each variable of FLC inputs has seven fuzzy sets ranging from negative big (NB) to positive big (PB). 3) Determining the input-output relationship and designing an inference mechanism rule. The Mamdani (Pro Max) inference is used. 4)

Defuzzifying the output variable of the fuzzy mechanism. Different defuzzification methods were used and compared to obtain the control signal. Figure 15 shows the fuzzy membership functions of E(t). FLC Design and Simulation; the input and output functions were adjusted: input-1 represents the set point shown in Fig.s 16 and input-2 for the current position, while the output is for the output signal as shown in Fig. 17. Rules representation is shown in Fig. 18 and Table 6. For the error signal E(t) as an input and control signal C(t) as an output, NB stands for negative big, NM means negative mean, Z means zero, PS means positive small, PM means positive mean, and PB means positive big. To vary the error (ΔE) as input, N means negative, and P means positive, [50, 52].

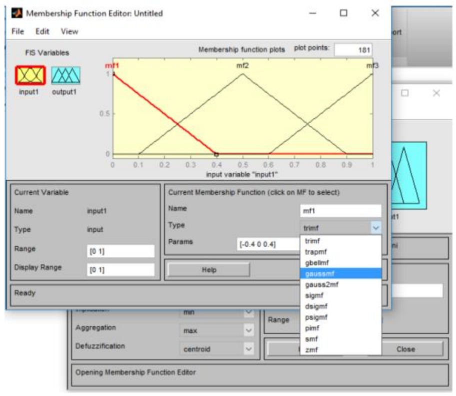
These are the linguistic variables that describe each of the time-varying fuzzy controller inputs and outputs that are used to define the rule base of the fuzzy controller. The Simulink model for the FLC is displayed in Fig. 19. A FLC controller was developed for each DC servo motor related to each joint.



```
>> fuzzy
>>
```

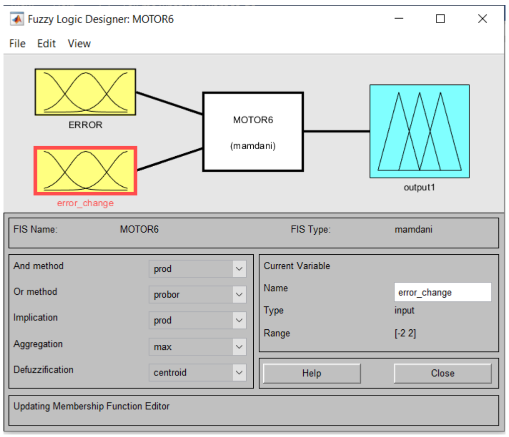
FIS Name: Untitled — FIS Type: mamdani
And method: min; Or method: max; Implication: min; Aggregation: max; Defuzzification: centroid
Current Variable — Name: input1; Type: input; Range: [0 1]
System "Untitled": 1 input, 1 output, and 0 rules

Fig. 15 Fuzzy library in MATLAB.



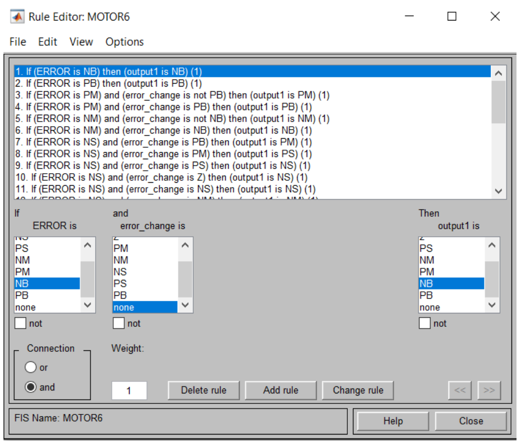
Current Variable — Name: input1; Type: input; Range: [0 1]; Display Range: [0 1]
Current Membership Function — Name: mf1; Type: trimf; Params: [-0.4 0 0.4]
Ready
Opening Membership Function Editor

Fig. 16 The input function adjustment.



FIS Name: MOTOR6 — FIS Type: mamdani
And method: prod; Or method: probor; Implication: prod; Aggregation: max; Defuzzification: centroid
Current Variable — Name: error_change; Type: input; Range: [-2 2]
Updating Membership Function Editor

Fig. 17 The set point representation.



```
1. If (ERROR is NB) then (output1 is NB) (1)
2. If (ERROR is PB) then (output1 is PB) (1)
3. If (ERROR is PM) and (error_change is not PB) then (output1 is PM) (1)
4. If (ERROR is PM) and (error_change is PB) then (output1 is PB) (1)
5. If (ERROR is NM) and (error_change is not NB) then (output1 is NM) (1)
6. If (ERROR is NM) and (error_change is NB) then (output1 is NB) (1)
7. If (ERROR is NS) and (error_change is PB) then (output1 is PM) (1)
8. If (ERROR is NS) and (error_change is PM) then (output1 is PS) (1)
9. If (ERROR is NS) and (error_change is PS) then (output1 is NS) (1)
10. If (ERROR is NS) and (error_change is Z) then (output1 is NS) (1)
11. If (ERROR is NS) and (error_change is NS) then (output1 is NS) (1)
```

Connection: [✓ and]; Weight: 1
FIS Name: MOTOR6

Fig. 18 Rules representation.

Table 6 Rule base description.

No.	Rule Description

1	If (ERROR is Z) and (error_change is NB) then (output1 is NS)
2	If (ERROR is Z) and (error_change is NM) then (output1 is z)
3	If (ERROR is Z) and (error_change is NS) then (output1 is z)
4	If (ERROR is Z) and (error_change is Z) then (output1 is z)
5	If (ERROR is Z) and (error_change is PB) then (output1 is PS)
6	If (ERROR is Z) and (error_change is PM) then (output1 is z)
7	If (ERROR is Z) and (error_change is PS) then (output1 is z)

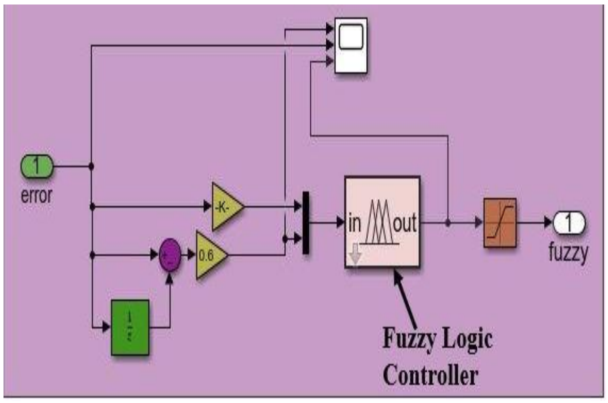


Fig. 19 Simulink model block diagram of FLC.

The process of defuzzification yields a non-fuzzy action by converting a group of fuzzy sets into a crisp value. The most common approaches are as follows: (1) center of gravity (COG), (2) bisector of area (BOA), (3) denotes of maximum (MOM), (4) smallest of maximum (SOM), and (5) largest of maximum (LOM). The purpose of COG, where the crisp control value u_{COG} is the abscissa of the center of gravity of the fuzzy set, u_{COG} is calculated as in Equation (53):

$$u_{COG} = \frac{\sum_i \mu_c(X_i)x_i}{\sum_i \mu_c(x_i)} \qquad (53)$$

Where X_i is a point in the world of the conclusion (i = 1, 2,…...), and $\mu_c$xi is the membership value of the resulting conclusion set. Integrals take the place of summations for continuous sets.

The BOA defuzzification technique calculates the coordinates of the vertical line that's wears the area of the extracted membership function into two analogous areas as in Equation (54).

$$\left|\sum_{i=1}^{j} \mu_c(x_i) - \sum_{i=j+1}^{i_{max}} \mu_c(x_i)\right|, i < j < i_{max} \qquad (54)$$

Where i_{max} is the index of the largest abscissa max x_i, and BOA is considered a computationally difficult procedure.

Another option for obtaining the crisp value is to select a point with the most members. There may be several points with the highest membership value in the total inferred fuzzy set. As a result, calculating the average value of these points is a common method. Using the method called MOM, the crisp value is determined as in Equation (55):

$$u_{MM} = \frac{\sum_{i\in I} X_i}{|I|}, I = \{i|\mu_c(x_i) = \mu_{max}|\} \qquad (55)$$

Where i is the crisp set of indices i, μ_c (x_i)reaches its maximum μ_{max}, and | I | is its cardinality, a term used to describe the phenomenon of the number of members.
The leftmost point among the points with the highest membership in the inferred fuzzy set can also be chosen. This procedure is referred to as the SOM defuzzification method. The crisp value is determined as in Equation (56):

$$u_{som} = x_{min\,(I)} \tag{56}$$

Another option is choosing the rightmost point among the points with maximum membership to the overall inferred fuzzy set. This technique is called the LOM defuzzification technique, where the crisp value is calculated as in Equation (57):

$$u_{lOM} = x_{max\,(I)} \tag{57}$$

These methods have been suggested in the literature. Fig. 20 shows the step response of the developed FLC related to the first DC servo motor and joint as an example.

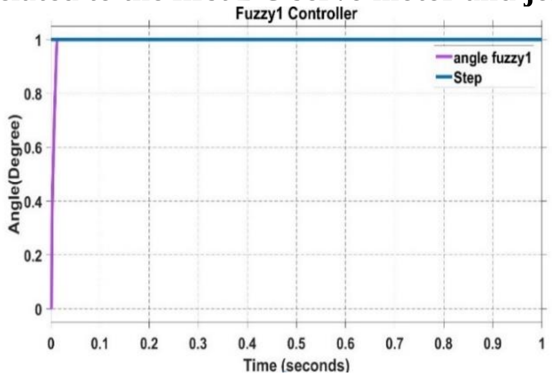


Fig. 20 The step response of the developed FLCs related to the first DC servo motor and joint.

7.2 Closed Loop Control Using PID and FLC Controllers

The PID control system, and FLC control system are combined together in one model for comparison, as shown in Fig. 21. The first part was the application of the PID control system on each motor as shown in Fig. 22 (a), and in the second part, the FLC controller was applied on each motor as shown in Fig. 22 (b).

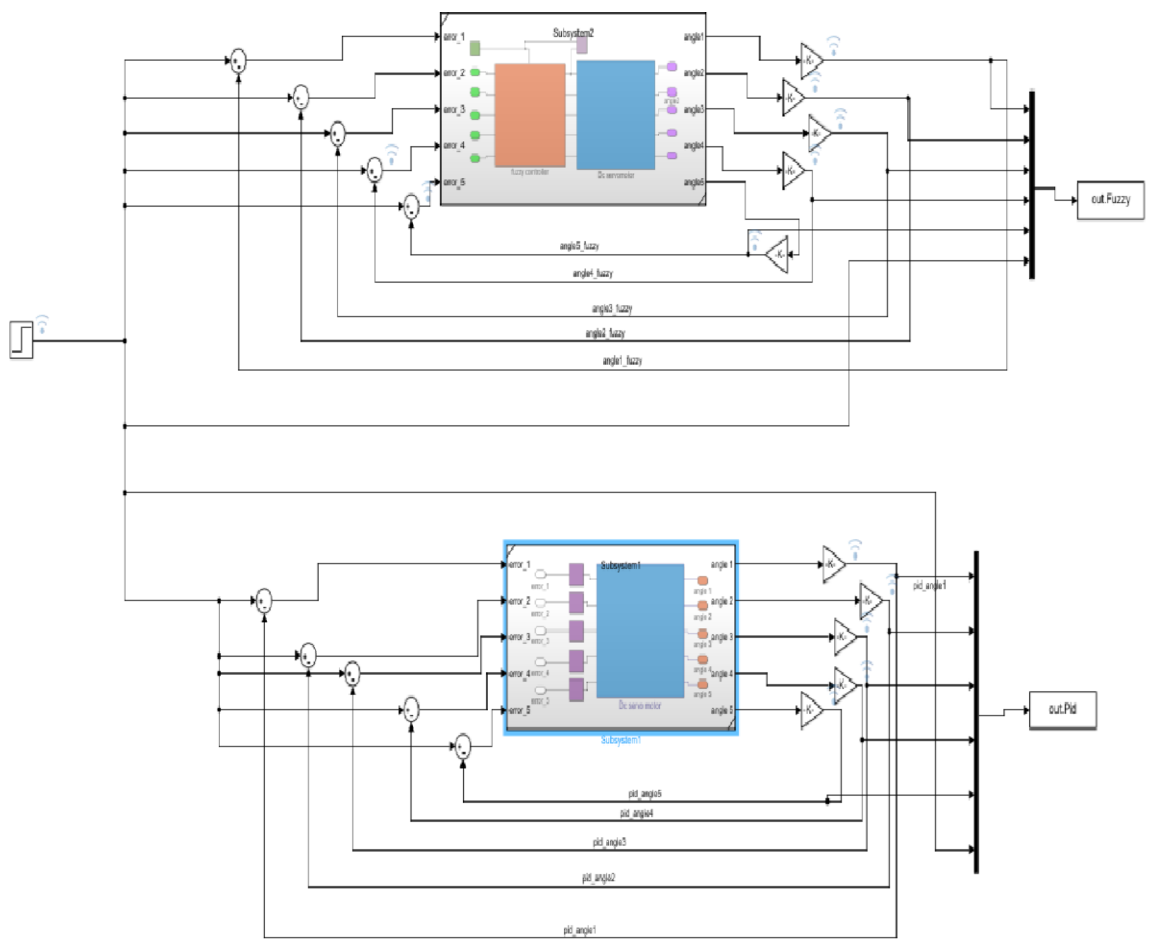
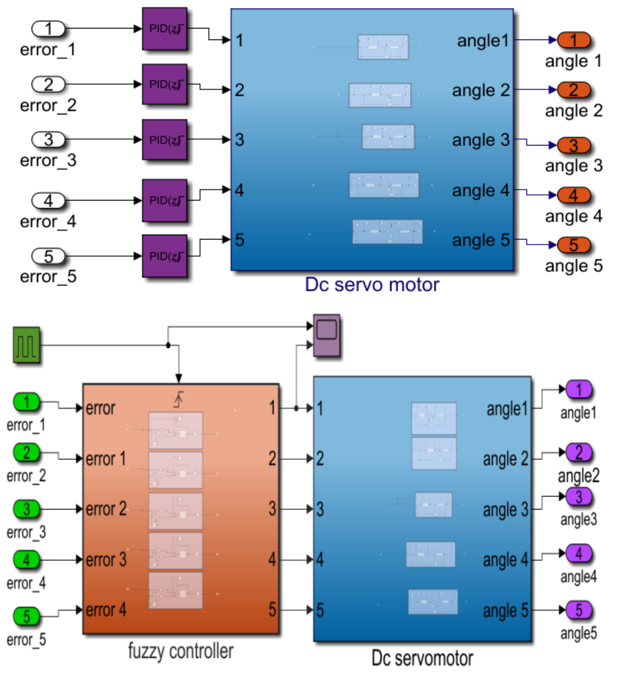


Fig. 21 Closed loops using PID and FLC control systems combined together.



(a) (b)

Fig. 22 (a) PID control systems structure, (b) Fuzzy control system structure.

7.3 Implementation of N Independent Joint Control

Independent joint control (IJC) controls the position of each joint independently [1]. This will help to ignore the dynamic coupling between joints. After testing the servo motor in the previous two stages and applying the controllers (PID and FLC), a complete system for the robotic arm was created that contains both controllers (PID and FLC) as shown in Fig. 23. It represents one motor to move one link. IJC for N joint robot manipulator (e.g., revolute or prismatic) joints is accomplished by using N block diagrams with respect to the number of the independent joints. The main system consists of several levels. As a case study, the independent joint control is implemented on 5 DOF robot arm.

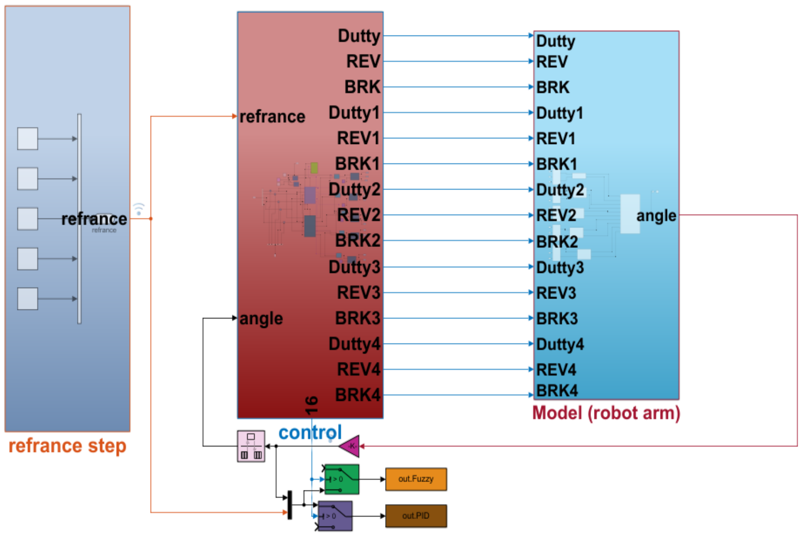


Fig. 23 IJC for 5 DOF using Simulink.

The main block diagram consists of three subsystems: subsystems 1, 2, and 3. The first subsystem "reference step" has the input values for each joint variable of the robot manipulator; these values represent the angles required that can be controlled by each one of the controllers independently. A step input is used for the five input angles. Similarly, subsystem 2 "control" has the disturbance input values that affect each link. In subsystem 3, "Model (robot arm)," five blocks are used, each one has an independent controller.

For each IJC, an appropriate angle is specified in the menu. When the signal enters the closed-loop control from input one of Fig. 24, it passes through many blocks such as gain, sum, and PID control. The gain block is used for the multiplication process between the input and a constant value to increase or decrease the signal value. Next, the summation block is used to pass the error signal, which is the difference between the desired input and the output. The error enters the PID controller block. Finally, the output of the PID controller is used to move the motor of the desired link. When the disturbance is applied from the second input, it passes through the feedforward block. To display the impact of disturbance with and without feedforward control, the switch block is utilized to switch between the feedforward block and zero as a constant. The performance of each link using the PID and FLC controllers was compared. Fig.s 25 (a), (b), (c), (d), and (e) indicate the controllers' output response related to motors 1, 2, 3, 4, and 5.

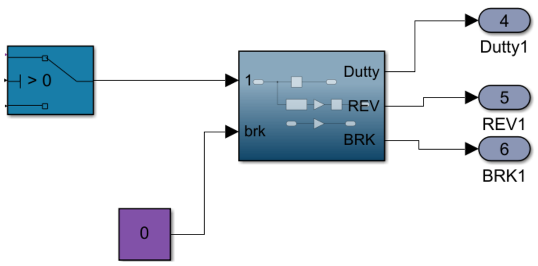
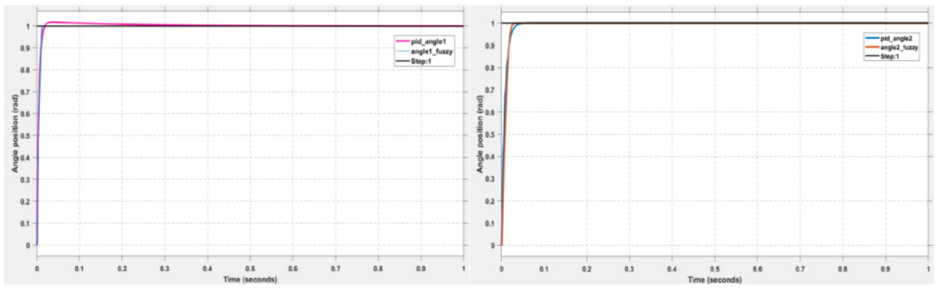
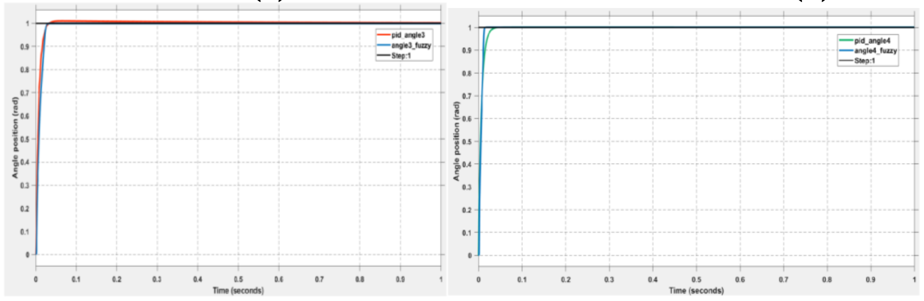


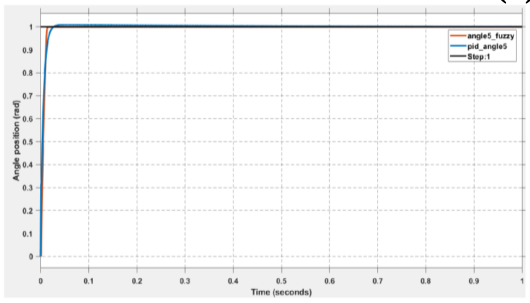
Fig. 24 One joint variable.



(a) (b)



(c) (d)



(e)

Fig. 25 Output responses of PID and FLC controllers for: (a) motor 1, (b) motor 2, (c) motor 3, (d) motor 4, and (e) motor 5.

The findings reveal that all controllers can complete the intended movement of the robot arm's servo motors. The time response parameters, including rise time, settling time, and steady-state error (SSE) of the PID controller and FLC system of the higher-order system transfer function of the DC servo motor of the processor, were also obtained. The FLC resulted in reduced rise time, stabilization time, SSE, and bypass than the PID controller. The comparison between the two controllers is shown in Table 7.

Table 8 displays the results of several defuzzification procedures, the BOA, MOM, and SOM techniques to the study, [23], which yield almost identical results.

Implementing a simple defuzzification approach resulted in system optimization due to the complexity of processes like fuzzification and defuzzification. Then the COG method should be avoided. When comparing the results to the results given in Table 4, it was found that the performance of the developed FLC gave better results in terms of rise time, settling time, the results obtained using the four defuzzification strategies are shown in Table 9. The results of the BOA, and SOM strategies are roughly equal, as this table demonstrates. In contrast, there are wide variations in the COG approach results. Due to complex operations such as fuzzification and particularly defuzzification, implementing a simplified defuzzification strategy optimizes the system. Comparing PID and FLC controllers' parameters.

Table 7 compared between PID and FLC controllers' parameters.

Motors (joint No.)	System properties				
	Controller Type	Rise time (rt)	Settling time (st)	Steady State Errors (SSE)	Overshoot
Motor1 (at joint 1)	PID	0.26514	0.015685	0.03125	0.03
	FLC	0.146253	0.011897	0.00612	0.016446
Motor2 (at joint 2)	PID	0.246084	0.030299	0.00108	0.008109
	FLC	0.225239	0.023342	0.000807	0.0023701
Motor3 (at joint 3)	PID	0.274915	0.026104	0.001985	0.069705
	FLC	0.107747	0.024641	0.001256	0.0015471
Motor4 (at joint 4)	PID	0.2714389	0.025763	0.014598	0.0028283
	FLC	0.094743	0.012818	0.00125	0.011673
Motor5 (at joint 5)	PID	0.212219	0.020554	0.00234	0.0176497
	FLC	0.0064693	0.012827	0.00254	0.0013509

Table 8 Comparison of the FLC parameters of the developed robot arm and the FlC in study of [23].

System output	Controllers					
Characteristics	Link 1 (FLC) to robot arm	Link 1 (FLC) in study [23]	Link 2 (FLC) to robot Arm	Link 2 (FLC) in study [23]	Link 3 (FLC) to robot arm	Link 3 (FLC) in study [23]
Rise time (rt)	0.146253	1.8507	0.225239	1.1373	0.107747	0.6444
Settling time (st)	0.011897	3.3064	0.02342	2.4064	0.024641	1.4413
Overshoot	0.016446	2.2594e-04	0.0023701	7.2402e-04	0.0015471	1.2071e-04
Steady State Errors (SSE)	0.00612	-0.001	0.00807	-0.002	0.001256	-0.003

Table 9 comparing different defuzzification strategies of FLC controllers.

System properties				
Defuzzification method	Rise time (rt)	Steady State Errors (SSE)	Overshoot	Settling Time (st)
COG	0.0086253	1.9507e-05	0.00044694	0.011897
SOM	0.0063461	0.01309	7.8937	0.99954
MOM	0.0066253	0.01304	0.00044694	0.011897
BOA	0.0064803	0.0033492	3.1163	0.99958

7.4 ROBOT ARM CONTROL USING FK AND IK IN MODEL SIMULATION

FK and IK blocks were applied to the control system connected to the robot arm as shown in Fig. 26.

7.4.1 Using forward kinematics

In the FK of a robotic arm, given the angles of the joints, the kinematics equations give the location of the tip of the arm. Matlab R2016B was used for model creation and simulation. The RVC feature, which includes the robotic 3D capabilities, is initialized in the Matlab tool. With the RVC feature, the robot arm can be developed, controlled, and manipulated [49]. The file startup_rvm.m calls several robot functions by calling the respective (.m files) of each function. A Matlab link file performs the process of link creation. All of the details about a robot link, including the motor, transmission, rigid body inertial, and kinematics characteristics, are contained in this file. In addition, there are classes and functions to receive the four parameters of the D-H convention and construct the link. The type of link is determined by its fifth parameter, which can be either prismatic (1) or revolute (0). The robot is constructed by joining all of the links in a serial fashion. The robot's dimensions and all of the links' information are provided by the Matlab file SerialLink.m, which connects the vector of link objects to form a serial-link robot object. Fig. 27(a) shows the output of SerialLink.m (Serial-Link robot). The D-H parameters of each link are also shown in Fig. 27(b). A robot with 5 links in revolute configuration is developed, and robot positions with joint angles (0, 0, 0, 0, 0) are presented in Fig. 28. Further, the D-H parameters (forward kinematics) of the 5 DOF robot arm with joint angles (0, -60, -30, -30, 0) and robot position are shown in Fig. 29(a) and (b), respectively.

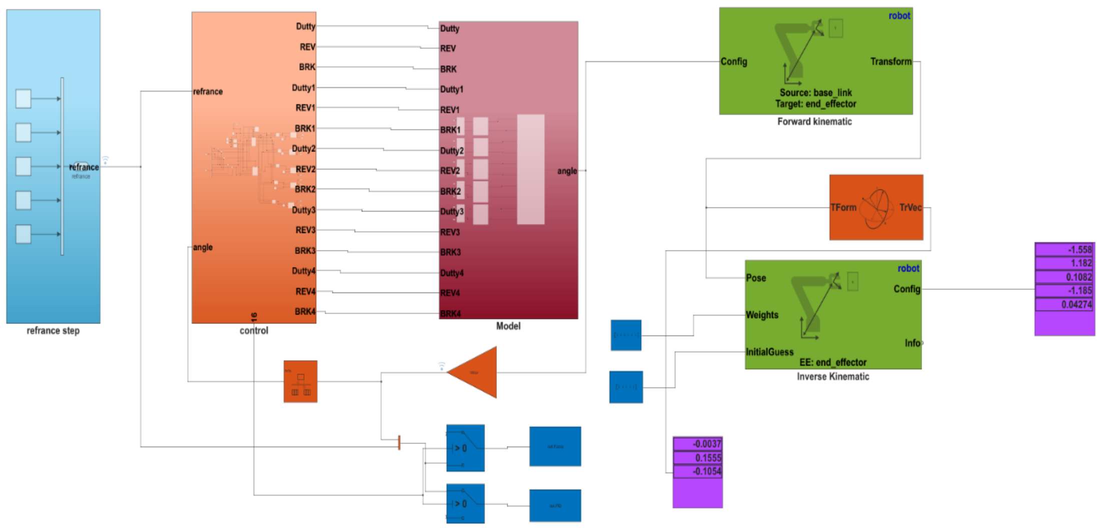
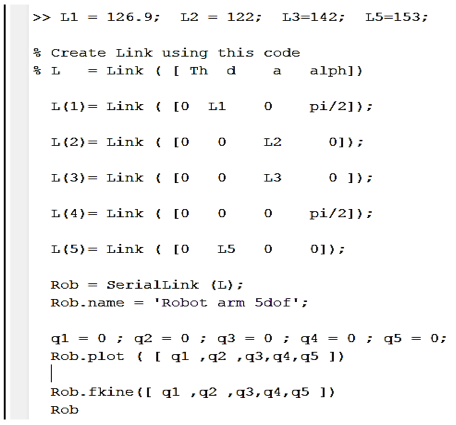
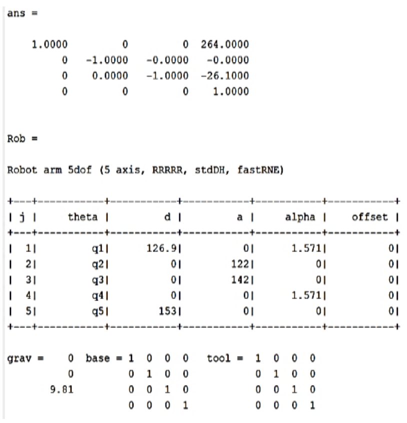


Fig. 26 Robot arm subsystem using FK and IK.



```
>> L1 = 126.9;  L2 = 122;  L3=142;  L5=153;

% Create Link using this code
% L    = Link ( [ Th  d    a    alph])

  L(1)= Link ( [0  L1     0    pi/2]);

  L(2)= Link ( [0   0    L2      0]);

  L(3)= Link ( [0   0    L3      0 ]);

  L(4)= Link ( [0   0     0    pi/2]);

  L(5)= Link ( [0   L5    0     0]);

  Rob = SerialLink (L);
  Rob.name = 'Robot arm 5dof';

  q1 = 0 ; q2 = 0 ; q3 = 0 ; q4 = 0 ; q5 = 0;
  Rob.plot ( [ q1 ,q2 ,q3,q4,q5 ])
  |
  Rob.fkine([ q1 ,q2 ,q3,q4,q5 ])
  Rob
```



```
ans =

    1.0000         0         0  264.0000
         0   -1.0000   -0.0000   -0.0000
         0    0.0000   -1.0000  -26.1000
         0         0         0    1.0000


Rob =

Robot arm 5dof (5 axis, RRRRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|      126.9|          0|      1.571|          0|
|  2|         q2|          0|        122|          0|          0|
|  3|         q3|          0|        142|          0|          0|
|  4|         q4|          0|          0|      1.571|          0|
|  5|         q5|        153|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  1  0  0           0  1  0  0
        9.81         0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```

(a) (b)

Fig. 27 (a) Forward kinematics with joint angles (0, 0, 0, 0, 0), (b) The D-H parameters of each link.

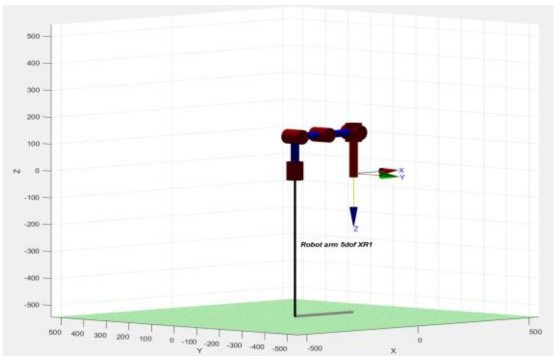
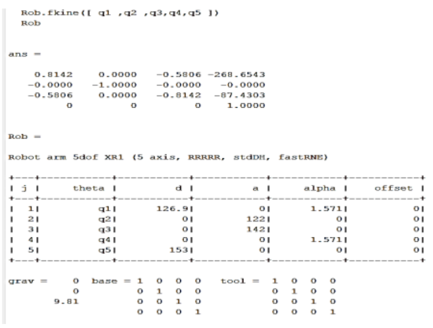
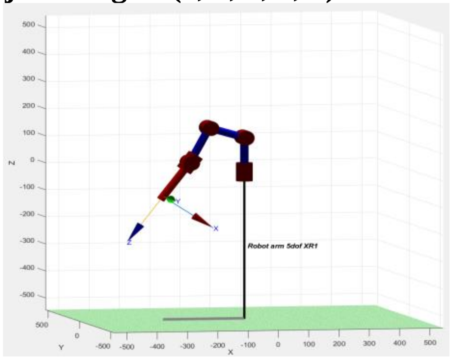


Fig. 28 Robot position with joint angles (0, 0, 0, 0, 0).



```
  Rob.fkine([ q1 ,q2 ,q3,q4,q5 ])
  Rob


ans =

    0.8142    0.0000   -0.5806 -268.6543
   -0.0000   -1.0000   -0.0000   -0.0000
   -0.5806    0.0000   -0.8142  -87.4303
         0         0         0    1.0000


Rob =

Robot arm 5dof XR1 (5 axis, RRRRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|      126.9|          0|      1.571|          0|
|  2|         q2|          0|        122|          0|          0|
|  3|         q3|          0|        142|          0|          0|
|  4|         q4|          0|          0|      1.571|          0|
|  5|         q5|        153|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  1  0  0           0  1  0  0
        9.81         0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```



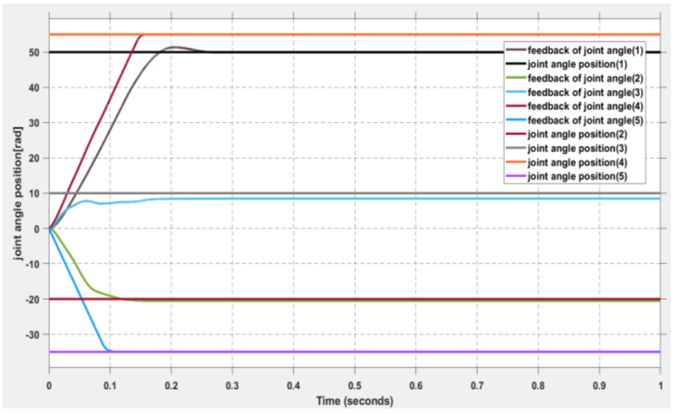
(a) (b)

Fig. 29 5 DOF robot arm with joint angles (0, -60, -30, -30, 0): (a)Forward kinematics, (b) Robot position.

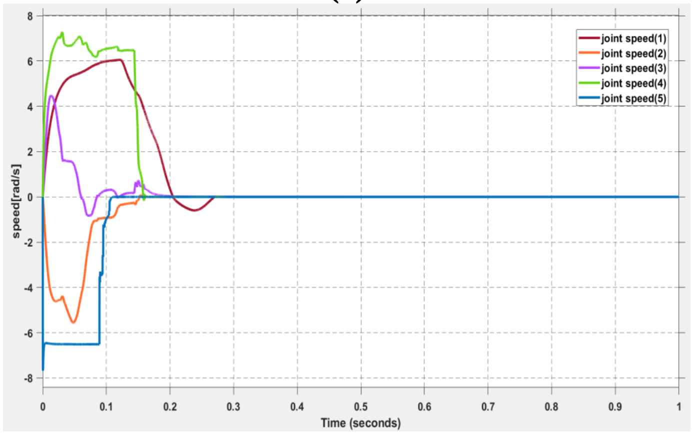
7.4.2 Using Inverse Kinematics

In IK given a desired location for the tip of the robotic arm, what should the angles of the joints be so as to locate the tip of the arm at the desired location. Typically, there are multiple solutions available, and occasionally, solving the problem can be challenging. To operate the robotic arm, this common robotics issue must be resolved to perform tasks it is designated to do with the robotic arm, and given the desired coordinate, the problem is finding the joint angles involved. Fig. 30 shows the simulation results: (a) input and output values of joint angles; (b) output values of

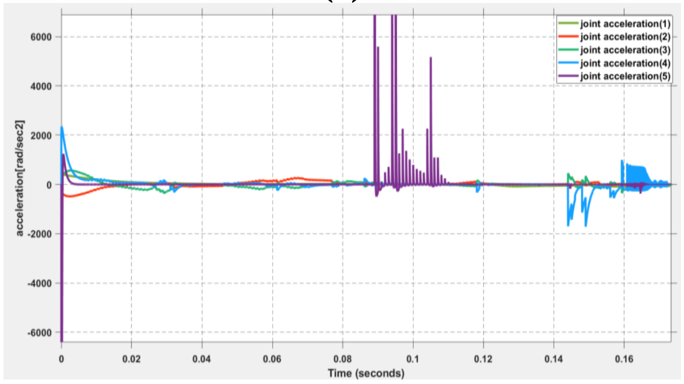
joint speed; (c) output values of joint acceleration; (d) output values of joint torque; (e) end effector position.



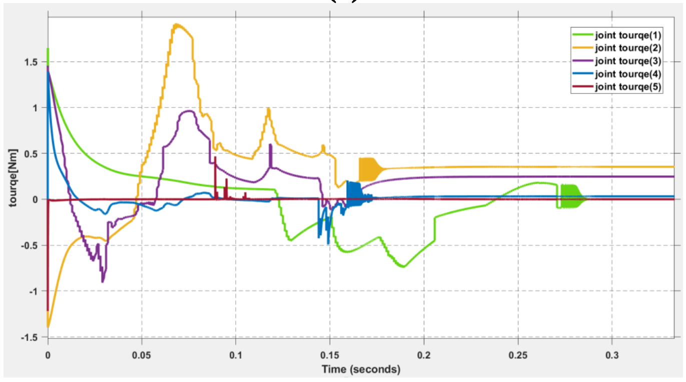
(a)



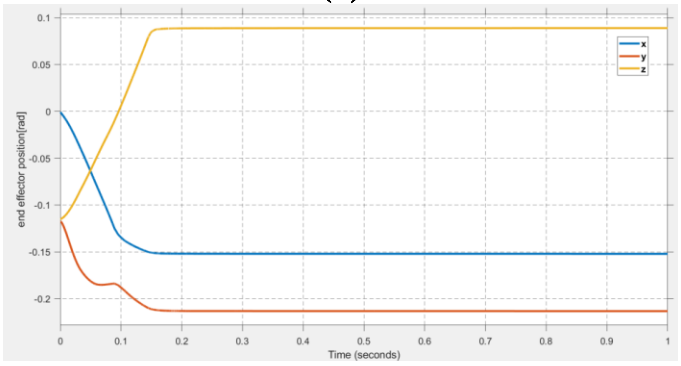
(b)



(c)



(d)



(e)

Fig. 30 The simulation results of IK. (a)Input and output values of joints angles, (b)output values of joints speed, (c) output values of joints acceleration, (d) output values of joints torque,(e) end effector position.

7.5 Real Time Control

To characterize the system and estimate the transfer function of the DC motors, the system identification process, which depends on Data Driven Control (DDC) [53], is applied to create a simple control system and identify the plant model from the I/O data, then using the identified model to design a controller and execute it. DDC is an approach that requires measured input-output data and is used to design the controller when a plant model is not available. DDC contains four steps: data acquisition to get the data set, system identification to get the plant model, controller design to control the system, and real-time testing on the hardware. The hardware setup includes a DC motor, [47], connected to an Arduino mega board with a L293 DC motor driver.

In data collection: The host PC is connected to the Arduino mega board using Simulink®, which makes it possible to generate an executable code and run it on the required hardware. The Simulink library for use with Arduino hardware is shown by Fig. 31.

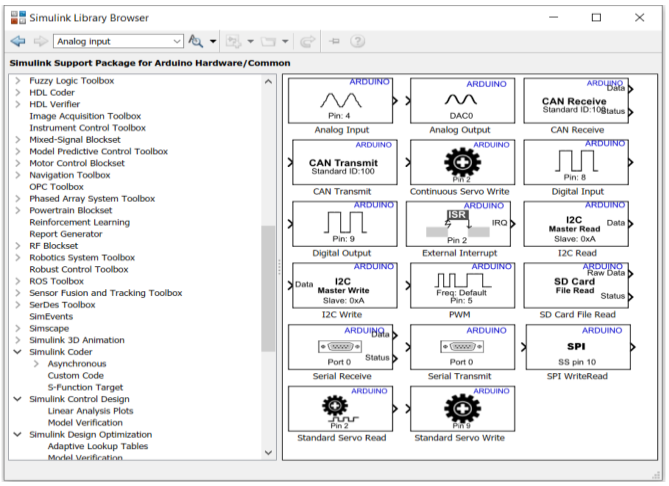


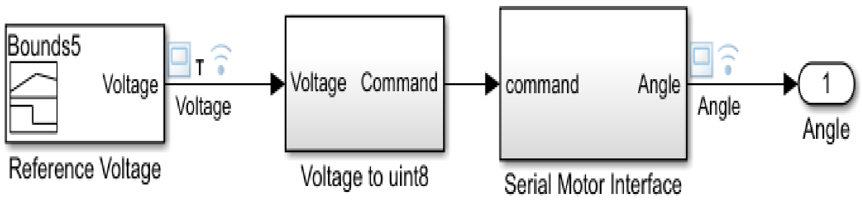
Fig. 31 Simulink block library; Simulink support package for Arduino hardware.

The host PC must communicate with the Arduino board to send voltage commands. Hence, a Simulink model is built to enable this communication. Further, the Arduino

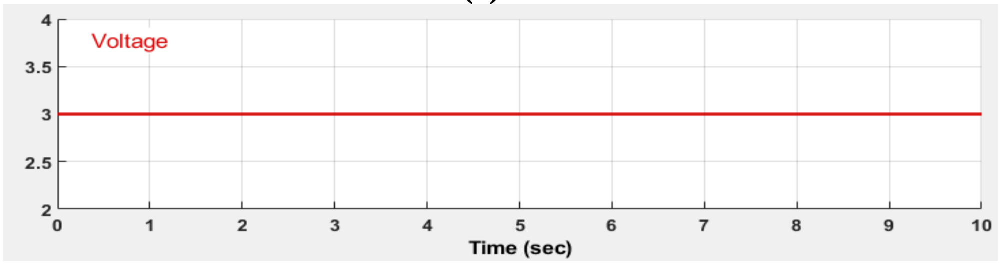
board sends voltage commands to the actuator, and the resulting position of the actuator is measured. A second Simulink model is created to permit the data transition.

The Simulink model running on the host PC includes three blocks. The first is the "Reference Voltage" block, through which the required signal can be created. Then there is the "Voltage to Unit8" block, which transforms the signal into an appropriate style to suit the IP. The third block is the "Serial Motor Interface" block that sends the signal to the selected serial port of the PC. This model is shown in Fig. 32(a), and the voltage signal is shown in Fig. 32(b). A step function with a value of 3 volts is chosen to excite the actuator.

In the model running on the Arduino mega board shown in Fig. 33, the Simulink function block "Serial Receive" receives the voltage signal sent from the PC through the serial port. After that, the block "Voltage Command To Pins" reads the signal from the block, "Serial Receive" and sends it to the "Analogue Output (PWM) Forward" block, which directs the signal to the appropriate PWM pin. The serial communication protocol is used to make the host PC able to communicate with the Arduino board.



(a)



(b)

Fig. 32: (a) Simulink model that will run on the host PC, (b) Voltage signal.

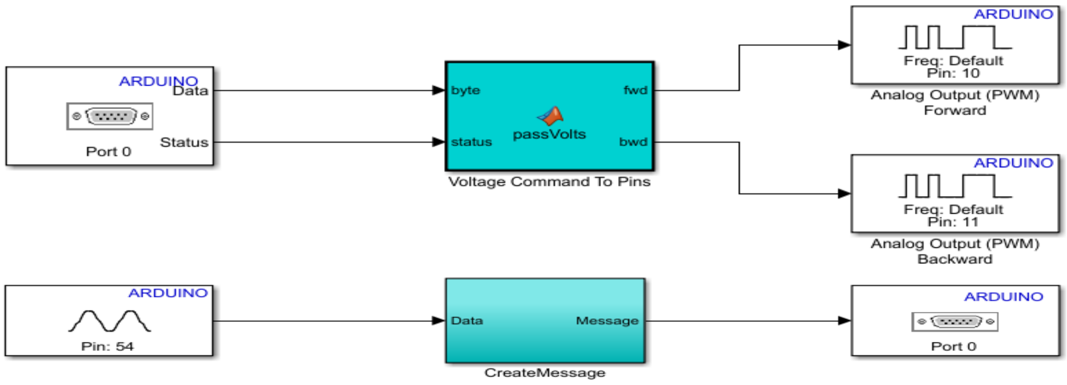
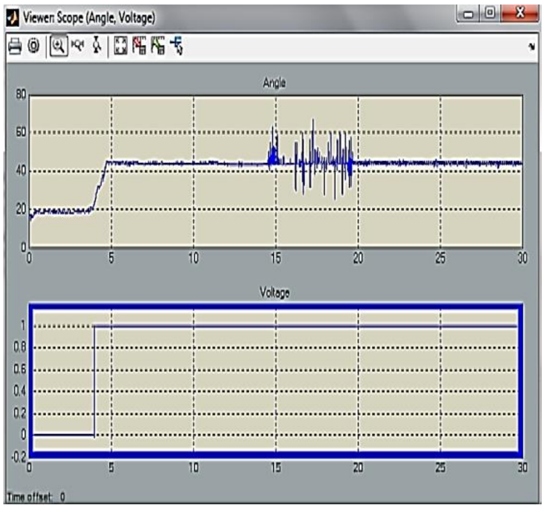


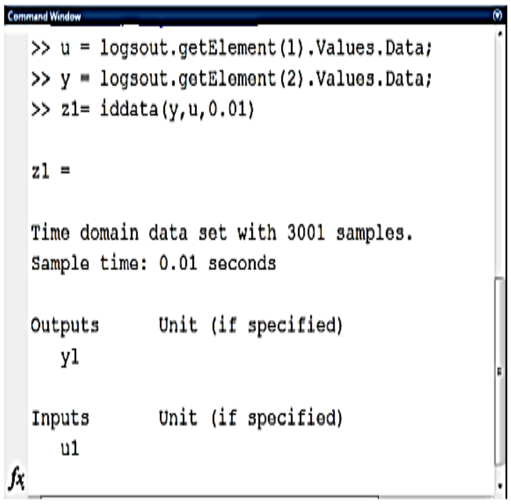
Fig. 33 Simulink model that will run on the Arduino board.

Different signals can be built by the signal builder block in Matlab. A system identification experiment is executed for the control of servo DC motors using PID and FL. Using the Simulink program 1 (program on the host PC), the Arduino card communicates with the PC, sending and receiving the input and the output voltage to the computer, respectively. The Simulink program was used to record and save the input voltage and the corresponding output position voltage. Program 2 code was uploaded to the Arduino card to send the input voltage and receive the output voltage to and from the actuator, respectively. The hardware setup in Fig. 13(a) is used in the experiment. Then run the Simulink model. The input voltage and output tracking angle are shown in Fig. 34 (a).

Then identify element 1 (voltage) and element 2 (angle), which appear in the command window, as shown in Fig. 34(b). By writing the (ident) command in the command window and pressing enter, the system identification tool window opens, then select the import data, the data object, then the data import. Hence, the import data window opens as shown in Fig. 35(a), select the object z1, and select the import icon to obtain the object in the system identification tool windows as shown in Fig. 35(b).

In the Import Data dialog box with the (time-domain signals) option, the options were specified as in Fig. 35(a). The imported data was divided into two parts, as in Fig. 35(b), for model estimation and for model validation. Matlab can compare many systems' performance and give which is the best one.
In the system identification tool, the transfer function model is estimated after specifying the number of poles and zeros of the transfer function block, as shown in Fig. 36(a). By clicking on transfer function models, the estimation process will start as in Fig. 36(b).

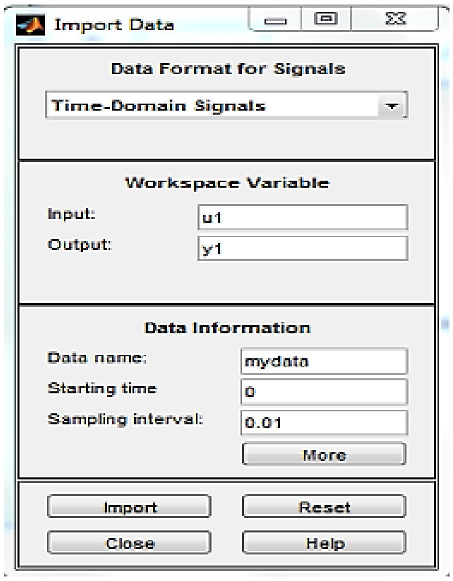




```
>> u = logsout.getElement(1).Values.Data;
>> y = logsout.getElement(2).Values.Data;
>> z1= iddata(y,u,0.01)

z1 =

Time domain data set with 3001 samples.
Sample time: 0.01 seconds

Outputs      Unit (if specified)
   y1

Inputs       Unit (if specified)
   u1
```

(a) **(b)**

Fig. 34: (a) Input voltage and output tracking angle, (b) Input/output identification in command window


Import Data

Data Format for Signals

Time-Domain Signals

Workspace Variable

Input: u1

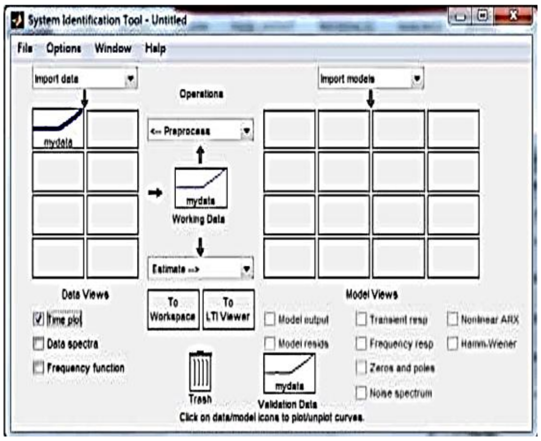
Output: y1

Data Information

Data name: mydata

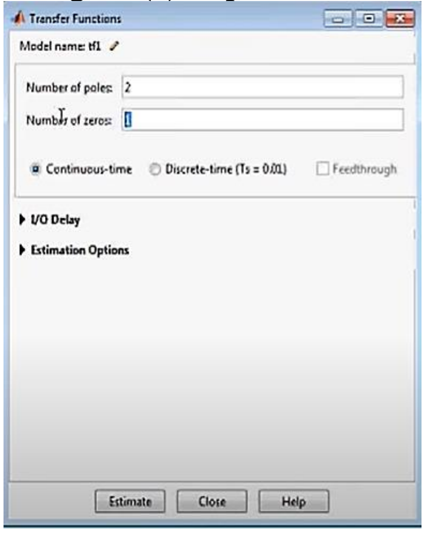
Starting time: 0

Sampling interval: 0.01



(a) (b)

Fig. 35 (a) Import data window, (b) System identification tool window.

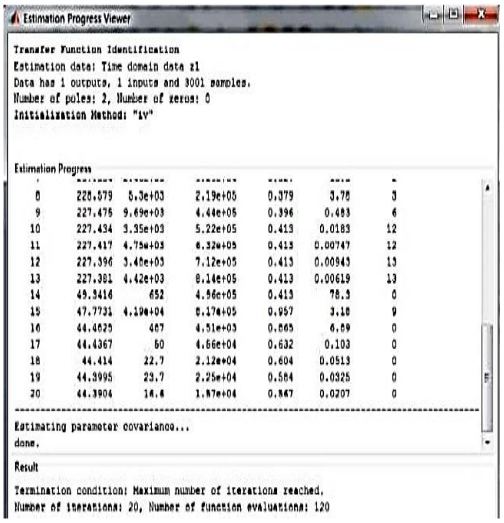

Transfer Functions

Model name: tf1

Number of poles: 2

Number of zeros: 0

[✓ Continuous-time] Discrete-time (Ts = 0.01) Feedthrough


Estimation Progress Viewer

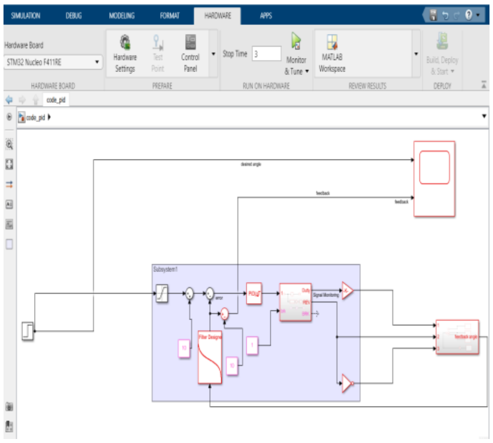
```
Transfer Function Identification
Estimation data: Time domain data z1
Data has 1 outputs, 1 inputs and 3001 samples.
Number of poles: 2, Number of zeros: 0
Initialization Method: "iv"

Estimation Progress
     8     228.579   8.3e+03    2.19e+05   0.379      3.78     3
     9     227.476   9.69e+03   4.44e+05   0.396      0.483    6
    10     227.434   3.35e+03   5.22e+05   0.413      0.0183  12
    11     227.417   4.75e+03   6.32e+05   0.413      0.00747 12
    12     227.396   3.48e+03   7.12e+05   0.413      0.00943 13
    13     227.381   4.42e+03   8.14e+05   0.413      0.00619 13
    14     49.3416        652   4.96e+05   0.413     78.3      0
    15     47.7731   4.19e+04   8.17e+05   0.957      3.18     9
    16     44.4625        467   4.51e+03   0.865      6.89     0
    17     44.4367         60   4.66e+04   0.632      0.103    0
    18      44.414       22.7   2.12e+04   0.604      0.0513   0
    19     44.3995       23.7   2.25e+04   0.584      0.0325   0
    20     44.3904       14.4   1.87e+04   0.567      0.0207   0
-------------------------------------------------------------------------
Estimating parameter covariance...
done.

Result
Termination condition: Maximum number of iterations reached.
Number of iterations: 20, Number of function evaluations: 120
```

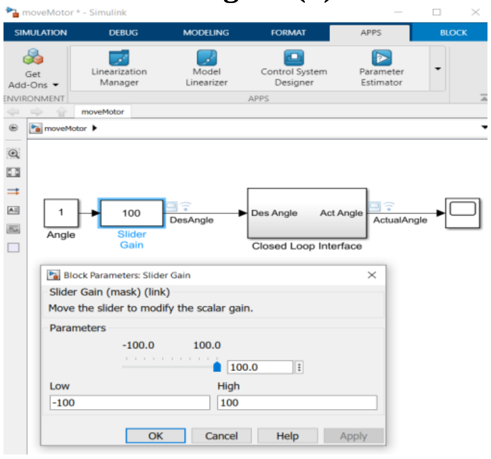
(a) (b)

Fig. 36 (a) Transfer function toolbox, (b) Estimation progress viewer.

In real-time testing for PID and FLC controller using a servo DC motor;To implement PID controller and FLC controllers on hardware system, and to control the position of servo DC motor of robot arm, it is needed to control the input voltage supplied to the motor, the control process of input voltage has been done by comparing the input voltage with the output voltage measured by the potentiometer, the desired position is set and the controller under testing transmits the control signal as PWM to rotate the motor to the corresponding position, the arduino card reads the value of the current position of motor as voltage from the output potentiometer, and sends this value to the controller, the controller compares the output voltage with reference voltage of the desired position and give suitable output signal corresponding to difference between them, if the difference equals to zero the controller's signal will equal to zero, the PID algorism outputs the control signal that rotates the motor to the desired position.

Also, a special code (Simulink) was uploaded to the Arduino card to send the input angle and receive the output angle to the DC motor, and a special code on the host computer which was used to send the reference position data (angle) and collect the actual position feedback and check how the motor tracked the reference. Fig. 37 (a) and (b) show the Simulink model on the Arduino board and the host computer Simulink model and slider gain controller. By using typical steps of real-time testing, click on the gain slider to produce different angles degree and look to the result by clicking on angle look point to obtain the output compared with the input reference for the PID controller as shown in Fig. 38 (a), and the desired and actual angle graph of the FLC controller reference angle are shown in Fig. 38 (b).





Block Parameters: Slider Gain

Slider Gain (mask) (link)

Move the slider to modify the scalar gain.

Parameters

-100.0 100.0

100.0

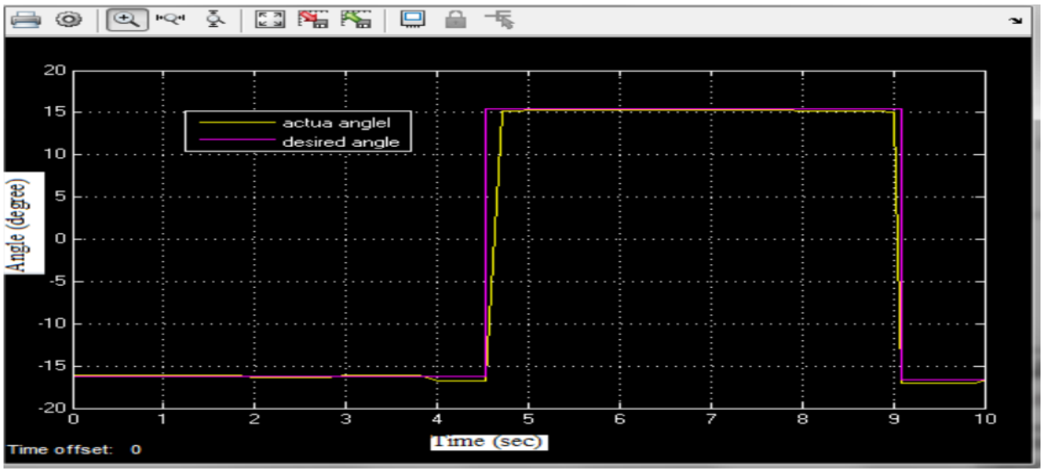
Low -100

High 100

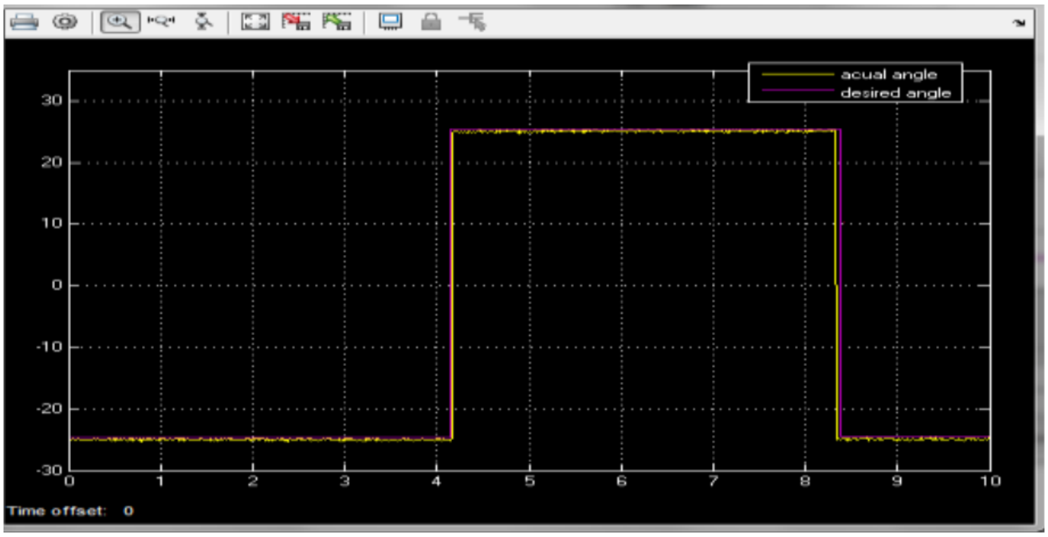
OK Cancel Help Apply

(a) (b)

Fig. 37 (a) Simulink model on arduino board, (b) Host computer Simulink model and slider gain controller.



(a)



(b)

Fig. 38 Desired and actual angle graph in real time testing using DC motor of the: (a) PID controller, (b) FLC controller.

CONCLUSIONS

Modeling and controlling of a 5-DOF robot arm through providing systematic rules for analyzing forward and inverse kinematics solutions for the robot manipulator with revolute joints using DH parameters are performed, then the mathematical model of the DC motor is analyzed, and friction equations are introduced. Then, the problem of control technique was discussed. A PID controller was designed and tuned using Simmechanics instead of using conventional tuning methods ZN. FLC was designed based on Mamadani's (pro Max) inference. Five different defuzzification methods were used and compared to obtain the control signal. These are (1) center of gravity (COG), (2) bisector of area (BOA), (3) means of maximum (MOM), (4) smallest of maximum (SOM), and (5) largest of maximum (LOM). From results, BOA, MOM, and SOM techniques yield almost identical optimum results; however, the COG strategy is not suitable. Implementing a simple defuzzification approach resulted in system optimization due to the complexity of processes like fuzzification and defuzzification. A feedforward method was used to overcome the disturbances, which loaded on each motor. Then the results of using the two controllers for controlling the robot manipulator were compared in terms of overshoot, transient response, and steady state error. Matlab and Simulink, which are extensively utilized in control applications, were used to display simulations.

In the control of the 5-DOF robot arm to reach the specified location with minimum error while meeting certain specifications, the final position for each motor was set using the independent joint control method. The Mamdani method was applied in FLC using 49 rules to control the robot arm. In order to adjust the PID controller's parameters p, i, and d. By doing this, the PID may be tuned while overcoming the tuning limitation of classical tuning methods. When it comes to time response behavior, the FLC outperforms the PID controller, FLC yielded a better performance in rise and settling time and reduced steady-state error and overshoot. Real time control was applied; a data driven control approach was used to design the controller

The system performance was good as the output follows the reference signal in a good manner, and the FL gave better results.

In future work, different fuzzy approaches can be studied. The Takagi-Sugeno model may be used in place of the Mamdani approach and compared with it. A future work may focus on different types of controllers like adaptive fuzzy controller, and extend the system to more degrees-of-freedoms. This may improve the results obtained and minimize errors between the actual arm and the simulation. As with other robots, future developments such as path planning and computer vision are expected for this robotic arm.

REFERENCES

1. Tsai L-W., "Robot analysis: the mechanics of serial and parallel manipulators," ,The Mechanics of serial and parallel manipulators. p. 520, (1999). Wiely, ISBN: 978-0-471-32593-2.
2. Craig J. J., "Introduction to robotics: mechanics and control". Pearson Educacion, (2005).
3. Zhao Z.-Y., Tomizuka M., and Isaka S., "Fuzzy gain scheduling of PID controllers," IEEE Trans. Syst. Man. Cybern., Vol. 23, No. 5, pp. 1392–1398, (1993).
4. Wang X., Chen X., Jia W., Sun Y., and Pu H., "Forward kinematics analysis and 3-dimmision gait simulation of a MiniQuad walking robot," in 2007 International Conference on Mechatronics and Automation, pp. 1932–1937, (2007).
5. Elgazzar, Shadia. "Efficient kinematic transformations for the PUMA 560 robot." IEEE J. Robotics Autom. 1,pp.142-151, (1985).doi: 10.1109/JRA. 1985. 1087013
6. Chakraborty, B., Mukhopadhyay, R., Chattopadhyay, P.," Multi-objective optimization for complex trajectory tracking of 6-DOF robotic arm manipulators", In: Das, A.K., Nayak, J., Naik, B., Vimal, S., Pelusi, D. (eds) Computational intelligence in pattern recognition. CIPR 2022. Lecture Notes in Networks and Systems, vol 480. Springer, Singapore, (2022). https://doi.org/10.1007/978-981-19-3089-8_48
7. B. Melih Yilmaz , Enver Tatlicioglu Aydogan Savran, and Musa Alci" Self-adjusting fuzzy logic based control of robot manipulators in task space", IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, Vol. 69, No. 2,pp. 1620-1629, FEBRUARY (2022).
8. Feng, Zhi & Hu, Guoqiang & Sun, Yajuan & Soon, Jeffrey.," An overview of collaborative robotic manipulation in multi-robot systems", Annual Reviews in Control. 49, (2020). 10.1016/j.arcontrol.2020.02.002.
9. Li, Jianfei & Liu, Li & Wang, Yaobing & Liang, Wenyuan." Adaptive hybrid impedance control of robot manipulators with robustness against environment's uncertainties", pp. 1846-1851, (2015). 10.1109/ICMA.2015.7237767.
10. Chen J. and Tao G., "Adaptive control of robot manipulators in varying environments", Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA, pp. 211-216, (2022) doi: 10.1109/SIEDS55548.2022. 9799418.
11. Alassar A. Z., Abuhadrous I. M., and Elaydi H. A., "Modeling and control of 5 DOF robot arm using supervisory control," 2010 The 2nd International Conference

on Computer and Automation Engineering (ICCAE), Singapore, pp. 351-355, (2010), doi: 10.1109/ICCAE.2010.5451398.
12. Soh A. C., Alwi E. A., Rahman R. Z. A., and Fey L. H., “Effect of fuzzy logic controller implementation on a digitally controlled robot movement,” Kathmandu Univ. J. Sci. Eng. Technol., Vol. 4, No. 1, pp. 28–39, (2008).
13. Khoury G. M., Saad M., Kanaan H. Y., and Asmar C., “Fuzzy PID control of a five DOF robot arm,” J. Intell. Robot. Syst., Vol. 40, No. 3, pp. 299–320, (2004).
14. Anavatti S. G., Salman S. A., and Choi J. Y., “Fuzzy + PID controller for robot manipulator,” CIMCA 2006 Int. Conf. Comput. Intell. Model. Control Autom. Jointly with IAWTIC 2006 Int. Conf. Intell. Agents Web Technol.(2006), doi: 10.1109/CIMCA.2006.103.
15. Bekit B. W., Seneviratne L. D., Whidborne J. F., and Althoefer K., “Fuzzy PID tuning for robot manipulators,” in IECON’98. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 98CH36200), Vol. 4, pp. 2452–2457, (1998).
16. Delibaşı A., Türker T., and Cansever G., “Real–time DC motor position control by fuzzy logic and PID controllers using Labview.” Yildiz Technical University, (2010). http://www.yildiz.edu.tr/~adelibas/pdf/AD_101_mechrob04.pdf .
17. Visioli A., “Tuning of PID controllers with fuzzy logic,” IEE Proceedings-Control Theory Appl., Vol. 148, No. 1, pp. 1–8, (2001).
18. Banga V. K., Kaur J., Kumar R., and Singh Y., “Modeling and simulation of robotic arm movement using soft computing,” World Acad. Sci. Eng. Technol., vol. 51, pp. 616–619, (2011).
19. Tao Y., Zheng, Jiaqi, Lin, Yuanchang, Wang, Tianmiao, Xiong, Hegen, He, Guotian, Xu, Dong, “Fuzzy PID control method of deburring industrial robots,” J. Intell. Fuzzy Syst., Vol. 29, No. 6, pp. 2447–2455, (2015), DOI: 10.3233/IFS-151945.
20. Tavoosi J., Jokandan A. S., and Daneshwar M. A., “A new method for position control of a 2-DOF robot arm using neuro-fuzzy controller,” Indian J. Sci. Technol., Vol. 5, No. 3, pp. 2253–2257, (2012).
21. Baghli, F.Z., El Bakkali, L. (2016). Design and simulation of robot manipulator position control system based on adaptive fuzzy PID controller. In: Zeghloul, S., Laribi, M., Gazeau, JP. (eds) Robotics and Mechatronics. Mechanisms and Machine Science, vol 37. Springer, Cham. https://doi.org/10.1007/978-3-319-22368-1_24
22. Ghaleb N. M. and Aly A. A., “Modeling and control of 2-DOF robot arm,” Int. J. Emerg. Eng. Res. Technol., Vol. 6, No. 11, pp. 24–31, (2018).
23. Kabir U., Hamza M. F., Haruna A., and Shehu G. S., “Performance analysis of PID, PD and fuzzy controllers for position control of 3-DOF robot manipulator,” arXiv Prepr. arXiv1910.12076, (2019).
24. Ying Liu, Du Jiang, Juntong Yun, Ying Sun, Cuiqiao Li, Guozhang Jiang, Jianyi Kong, Bo Tao, and Zifan Fang, “Self-tuning control of manipulator positioning based on fuzzy PID and PSO algorithm”, Front. Bioeng. Biotechnol., 11 February 2022, Sec. Bionics and Biomimetics, Vol. 9, (2021), DOI: https://doi.org/10.3389/fbioe. 2021. 817723
25. El-Khatib M. F., and Maged S. A., “Low level position control for 4-DOF arm robot using fuzzy logic controller and 2-DOF PID controller,” 2021 Int. Mobile, Intelligent, Ubiquitous Comput. Conf. MIUCC 2021, pp. 258–262, (2021), DOI: 10.1109/MIUCC52538.2021.9447617.

26. Solouki M., Ansarin M., Torabi M., Nematia A., and Bakhshizadeh Y., "Optimization of PID controller with supervisory fuzzy control for industrial robots," J. Artif. Intell. Electr. Eng., Vol. 7, No. 28, pp. 27–36, (2019).

27. Amin Rashidifar M., Amin Rashidifar A., Ahmadi D., "Modeling and control of 5DOF robot arm using fuzzy logic supervisory control," International Journal of Robotics and Automation (IJRA), Vol. 2, No. 2, June (2013), pp. 56~68, ISSN: 2089-4856

28. Kharidege A. , Jianbiao D., And Zhang Y., " Performance study of PID and fuzzy controllers for position control of 6 DOF arm manipulator with various defuzzification strategies", MATEC Web of Conferences, 77, 01011, (2016), doi: 10.1051/ 010 (2016)

29. Dulaidi D., "Modeling and control of 6 DOF industrial robot using fuzzy logic controller." MSC thesis, University of Tun Hussein Onn Malaysia, (2014).

30. Tavoosi J., Alaei M., and Jahani B., "Neuro–fuzzy controller for position control of robot arm," Pap. Ref., No. 0113–795, pp. 12–17, (2011).

31. Zhu Q., "Real-time DC motor position control by (FPID) controllers and design (FLC) using labview software simulation," in 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), Vol. 2, pp. 417–420, (2010).

32. Kuo Y.-L., and Liu S.-M., "Position control of a serial manipulator using fuzzy-PID controllers," Int. J. Autom. Smart Technol., Vol. 5, No. 1, pp. 18–26, (2015).

33. Carignan C. R., Gefke G. G., and Roberts B. J., "Intro to space mission design: space robotics," in Seminar of Space Robotics, University of Maryland, Baltimore, Vol. 26, (2002).

34. Boyce R. and Mull J., "Complying with the occupational safety and health administration: Guidelines for the dental office," Dent. Clin. North Am., Vol. 52, No. 3, pp. 653–668, (2008).

35. Rahman A., Khan A. H., Ahmed T., and Sajjad M., "Design, analysis and implementation of a robotic arm- the animator", American Journal of Engineering Research (AJER):Vol. 2, Issue 10, pp-298-307, (2013). www.ajer.org.

36. Elfasakhany A., Yanez E., Baylon K., and Salgado R., "Design and development of a competitive low-cost robot arm with four degrees of freedom," Mod.Mech. Eng., vol. 1, No. 2, pp. 47–55, (2011), DOI: 10.4236/mme.2011.12007.

37. Crage J.J., "Introduction to robotics mechanics and control", 3rd Edition, Prentice Hall, (2005).

38. Spong M.W., Hutchinson S., and Vidyasagar M., "Robot modeling and control", 1st Edition, Jon Wiley & Sons, Inc, (2005).

39. Angeles J., "Fundamentals of robotic mechanical systems: theory, methods, and algorithms", 2nd Edition, (2003).

40. Gouda Mohamed M., Said Fatma El-Zahraa, Elmoushi El Sayed, and Abdelwahab Sabreen A.," Performance analysis and improvement of robot arm 5DOF using PID and fuzzy controllers: A comparative study", Prot said Engineering Research Journal, Vol. 26, No. 3, pp. 110-133, (2022). DOI: 10.21608/PSERJ.2022.132466.1177

41. Tokarz, Krzystof, and Kieltyka, Slawosz., "Geometric approach to inverse kinematics for arm manipulator", International conference on systems- Proceedings-1, (2010).

42. Meriam J.L., Kraige L.G.,"Engineering mechanics: Statics"7th edition. B07NWGC2L9, Wiley india Pvt. Ltd; Classic Edition (1 January 2013) ISBN: 978-0-470-61473-0. ISBN: 978-0-470-91787-9 (BRV).
43. Armstrong-Helouvry, B., Dupont, P., and Canudas-de-Wit, C., " A survey of models, analysis tools and compensation methods for the control of machines with friction", Automatica, Vol. 30, No. 7, pp. 1083-1138, (1994).
44. Olsson, H., Åström, K. J., Canudas de Wit, C., Gäfvert, M., & Lischinsky, P. "Friction models and friction compensation.", European journal of control, Vol. 4, No. 3, pp. 176-195, (1998).
45. Sciavicco, L., & Siciliano, B., "Modelling and control of robot manipulators.", Springer Science & Business Media, (2012).
46. Ziegler J.G. and Nichols N.B., "Optimum settings for automatic controllers," Transaction American Society of Mechanical Engineering, Vol. 64, pp. 759–768, (1942).
47. Venkateswara Rao V. M., "Performance analysis of speed control of DC motor using P, PI, PD and PID controllers", International Journal of Engineering Research & Technology (IJERT),Vol. 2, No. 5, pp. 60–66, (2013).
48. Ahmad A. Mahfouz, Mohammed M. K., and Farhan A. Salem, " Modeling, simulation and dynamics analysis issues of electric motor, for mechatronics applications, using different approaches and verification by MATLAB/Simulink", I.J. Intelligent Systems and Applications, Vol. 05, pp. 39-57, (2013). DOI: 10.5815/ijisa.2013.05.06.
49. The MathWorks Inc. MATLAB 7.0 (R14SP2). The MathWorks Inc., (2005).
50. Lee C.C., "Fuzzy logic in control systems: fuzzy logic controller-Part I", IEEE Transactions on Systems, Man and Cybernetics., Vol. 20, No.2, pp. 404–418, March/April (1990).
51. Mamdani E.H., "Applications of fuzzy logic to approximate reasoning using linguistic synthesis," IEEE Transactions on Computers, Vol. 26, No. 12, pp. 1182–1191, Dec. (1977).
52. Feng G., "A survey on analysis and design of model-based fuzzy control systems," IEEE Trans. Fuzzy Syst., Vol. 14, No. 5, pp. 676–697, (2006).
53. Hou, Zhongsheng & Wang, Zhuo., "From model-based control to data-driven control: Survey, classification and perspective", Information Sciences. Vol. 235. PP. 3–35, (2012). DOI: 10.1016/j.ins.2012.07.014.