ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2022.1.041

Meta-Heuristics Based Inverse Kinematics of Robot Manipulator's Path Tracking Capability Under Joint Limits

Ganesan Kanagaraj[1,✉], **S A R Sheik Masthan**[2], **Vincent F Yu**[3]

[1,2]Department of Mechatronics Engineering, Thiagarajar College of Engineering, Anna University, Madurai, Tamil Nadu, India
[3]Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan
gkmech@tce.edu[1,✉],sarsmech@tce.edu[2],vincent@mail.ntust.edu.tw[3]

Abstract

In robot-assisted manufacturing or assembly, following a predefined path became a critical aspect. In general, inverse kinematics offers the solution to control the movement of manipulator while following the trajectory. The main problem with the inverse kinematics approach is that inverse kinematics is computationally complex. For a redundant manipulator, this complexity is further increased. Instead of employing inverse kinematics, the complexity can be reduced by using a heuristic algorithm. Therefore, a heuristic-based approach can be used to solve the inverse kinematics of the robot manipulator end effector, guaranteeing that the desired paths are accurately followed. This paper compares the performance of four such heuristic-based approaches to solving the inverse kinematics problem. They are Bat Algorithm (BAT), Gravitational Search Algorithm (GSA), Particle Swarm Optimization (PSO), and Whale Optimization Algorithm (WOA). The performance of these algorithms is evaluated based on their ability to accurately follow a predefined trajectory. Extensive simulations show that BAT and GSA outperform PSO and WOA in all aspects considered in this work related to inverse kinematic problems.

Keywords: *Inverse Kinematics, Redundant Robot Manipulator, Path Tracking, Meta-Heuristic Algorithm, BAT, Particle Swarm optimization, Gravitational Search, Whale Optimization.*

Received: 05 April 2022
Accepted: 15 June 2022
Online: 24 June 2022
Published: 30 June 2022

1 Introduction

Recent increase in the use of robot manipulators in industries has attracted a great deal of attention to control the robot manipulator. Their wide applications in industrial systems include welding, painting, assembly, etc. A welding torch or a paint sprayer will be attached as a tool in the end effector of the robot manipulator. This tool has to accurately follow a predefined path in order to perform a particular task. Robot manipulators are made of several connected links. These links are to be controlled accurately in order to follow the given reference trajectories precisely. However, it is typically not tractable to control them accurately because of their high nonlinearity and unmodeled uncertainties. Thus, many works have been carried out to resolve such difficulties.

Several methodologies have been used for solving the inverse kinematics problem. Quaternion transformation approach was proposed to solve the inverse kinematic problem wherein solution for a general 7-link 7R mechanism is presented [17, 18, 19]. [26] proposed a detailed derivation of inverse kinematics using exponential rotational matrices by breaking the 6R-chain in the middle to form two open 3R-chains. Later [53] used the same quaternion approach instead of the regular Newton-Euler and Lagrange method as it simplified the modelling of the kinematics and dynamics of rigid multi-body systems.

Similar approaches were followed for trajectory tracking also. Jacobian matrix-adaption method was employed to overcome the two major limitations in Jacobian-matrix-pseudo-inverse (JMPI) for tracking control of robot manipulators [7]. A decentralized control strategy with finite-time convergence is developed for the trajectory tracking of a space manipulator in [41]. In this work, the robot manipulator is considered as a number of decoupled subsystems. A similar decoupling mechanism was proposed in [50] for aircraft assembly using a multi-objective posture optimization algorithm. For aerial manipulation, a multi-stage Model Predictive Control based approach was proposed in which was verified using a 3 degrees of freedom manipulator [15].

Several iterative approaches were also experimented for developing a controller for robot manipulator to follow a predefined trajectory. Iterative Learning Control was developed to identify and calculate the robot kinematic parameters and proposed an algorithm for the accurate path tracking of industrial robots [52].

An adaptive control method for trajectory tracking of robot manipulators, based on new neuro-fuzzy modelling was proposed in [43]. The proposed control scheme used a three-layer neural fuzzy network to estimate the system uncertainties, and its performance was compared with the conventional computed torque PD control. In [48], an adaptive output feedback tracking controller was proposed to prove the uniform global stability for the robot dynamic model with unknown parameters. [8] used Differential Evolution algorithm for training the neural network to obtain the kinematic modeling of robot manipulators.

[45] used neural network to track the trajectory adaptively. A radial basis function network is investigated to the joint position control of an n-link robot manipulator. Andreev and Peregudova investigated the trajectory control using uniform asymptotic stability in closed-loop system by using the dynamic position-feedback controller with feedforward [1]. A two-link planar elbow robot manipulator was used to illustrate the results. Baek et al., presented a practical Adaptive Time Delay Control scheme which was applied to robot manipulators to achieve good tracking performance with tolerant fluctuation and fast convergence speed [3].

Several control algorithms were proposed for trajectory tracking mechanisms in industrial robots. To name a few, Discrete-Time Nonlinear Optimization control [21], Sliding mode controller [30], adaptive control [49]. A sinusoidal-input describing function model along was created along with a controller for a two-link robot manipulator in [9]. A trajectory algorithm using artificial neural network and Kalman filter was proposed by [27]. The efficiency of the algorithm was verified by simulated using a 3-link Manipulator. To overcome the computational difficulties and approximations involved with the analytical methods, a machine learning based algorithm was proposed to predict the inverse kinematic solutions for parallel manipulators [44].

The computational complexity of the conventional analytical and other Jacobian based inverse kinematics led to the use of heuristic and meta-heuristic-based approach for this inverse kinematics problem [5]. Many swarm intelligent and meta-heuristics algorithms were employed to solve various optimization problems in the field of engineering. A new mutated genetic algorithm was employed to solve the problem of independent job scheduling in grid computing [51]. Study on the inner dynamics of PSO algorithm using network visualization showed the self-adaptive approaches of PSO [35]. The versatility of these algorithms had led to the development of hybrid algorithms. To name a few, genetic algorithm with fuzzy for multi objective optimization [40], Particle Swarm Optimization with clustering algorithm for system modeling and identification [23], facility location-network design model using Firefly and Invasive Weed Optimization based fuzzy system [39]. For the inverse kinematics issue considered in this paper, various meta-heuristic algorithms such as Genetic Algorithm [34, 6], Particle Swarm Optimization Algorithm [10, 15], Cuckoo Optimization Algorithm [4], Genetic Algorithm, Gravitational Search Algorithm [2], Artificial Bee Colony [14], Fire Fly Algorithm [38], Modified Firefly [22], Bat Algorithm [28], were employed.

Heuristic and meta-heuristic algorithm-based motion planning and tracking were also employed. [25, 29, 42, 46] proposed an adaptive genetic algorithm for tracking the trajectory of a robot manipulator. Online optimizations method was proposed by [16] for trajectory tracking to overcome the limitation of the traditional methods.

Literature survey shows the usage of meta-heuristic-based approaches for inverse kinematic problem. Since, the meta-heuristic approach does not include a Jacobian matrix and there is always a solution for forward kinematics, there are no singular configurations. At the same time, literature review shows that the usage of meta-heuristic approach for the robot manipulator trajectory following is very limited. The performance of these meta-heuristic algorithms is measured based on the faster convergence rate. From the survey, it was found that the usage of meta-heuristic-based approaches for inverse kinematics problem is limited and it was not extended to actual robot manipulators. This motivated us in using proposing a meta-heuristic-based approach to solve the inverse kinematic problem for an industrial robot manipulator.

This paper's contributions are summarized as follows. Four algorithms, viz.: BAT, GSA, PSO and WOA are experimented in this research work for solving the inverse kinematics problem for robot manipulators. Heuristic approach is proposed as it does not include a Jacobian matrix and there is always a solution for forward kinematics, there are no singular configurations as a consequence of inverse kinematics. The four proposed algorithms are compared based on their capability to generate solution that helps the robot manipulator to closely follow the pre-defined trajectory. Four predefined trajectories, viz.: linear, curvilinear, saw tooth and rose curve, are used to test the trajectory following capability of the heuristic algorithm. Wilcoxon test is employed to measure the performance of the trajectory following capability of the proposed algorithms based on three parameters viz., Minimum Average Error, Fast Convergence and Minimal variation in joint angles. The results are simulated and discussed. The remainder of the paper is structured as follows. A general overview of the problem is described in Section 2. Section 3 explains the proposed methodology and how it is used to carry out the simulation for trajectory following. Section 4 discusses about the experimental setup and the results obtained from the experiments. The paper is concluded with the key findings in section 5.

2 Problem Description

The position of the end effector of a robot manipulator can be controlled or varied by changing the link angles of the robot manipulator. Calculating these link

Table 1: Input and output of the algorithms.

Input	
1.	j^{th} coordinate point $(x_d, y_d, z_d)_j$ in the trajectory
2.	link angles $(\theta_1, \ldots \theta_l, \ldots \theta_{nL})_{j-1}$ generated for reaching the to $(j-1)^{th}$ point
Output	
link angles $(\theta_1, \ldots \theta_l, \ldots \theta_{nL})_j$ for reaching the j^{th} coordinate point $(x_a, y_a, z_a)_j$ in the generated trajectory	

angles for a particular end effector position in space is done generally by inverse kinematics. Inverse kinematics, in general, is computationally complex. The increased number of links in a redundant manipulator further increases this complexity [11]. The computational complexity of this link angle calculation can be reduced by employing heuristic algorithm instead of inverse kinematics. So, heuristic-based approach can be used to solve the inverse kinematics of the robot manipulator end-effector, which guarantees that the desired paths are followed precisely.

2.1 Problem Statement

The efficiency of any robot manipulator in closely following the predefined trajectory can be measured in terms of its deviation from the predefined path or trajectory. This difference or deviation of the end effector from the predefined path at any point in the trajectory can be mathematically stated as:

$$Error(\epsilon) = \sqrt{(x_d - x_a)^2 + (y_d - y_a)^2 + (z_d - z_a)^2} \tag{1}$$

Here, $(x_d,\ y_d, z_d)$, $(x_a,\ y_a, z_a)$ denotes the Cartesian coordinates of desired and actual position of the end effector respectively. The problem is to find the link angles of each joint such that the error is minimum. Therefore, the problem can be transformed into an equivalent optimization problem as:

$$\begin{gathered} \text{Minimize } Error(\epsilon) \\ \text{subjected to the condition that} \\ \theta_{l,min} \leq \theta_l \leq \theta_{l,max} \end{gathered} \tag{2}$$

Here, θ_l represents the l^{th} link angle, $\theta_{l,min}$ and $\theta_{l,max}$ are the minimum and maximum value of the l^{th} link angle, $l = 1, 2, \ldots, nL$ with nL represents the number of links.

So, based on the above problem discussion, the objective of this research work is to experiment the proposed heuristic algorithms (BAT, GSA, PSO & WOA) for calculating the link angles of a robot manipulator to accurately follow a predefined trajectory.

3 Proposed Methodology

Four algorithms viz.: BAT, GSA, PSO and WOA, are employed in this research work to achieve the proposed objective. The efficiency of these algorithm is tested based on its closeness in following the generated trajectories. Table 1 shows the input and output of the algorithms.

Here, $j = 1, 2, \ldots, nPT$ and nPT is the number of points in the generated trajectory. The algorithm iterates repeatedly and gives the link angles $(\theta_1, \ldots \theta_l, \ldots \theta_{nL})_j$ of the robot manipulator to the reach the point that is given as input. Here $l = 1, 2, \ldots, nL$ and nL is the number of links in the robot manipulator. When these link angles $(\theta_1, \ldots \theta_l, \ldots \theta_{nL})_j$ are applied to the robot manipulator, it will orient in a particular configuration there by reaching a point $(x_a, y_a, z_a)_j$ The difference between $(x_d, y_d, z_d)_j$ and $(x_a, y_a, z_a)_j$ is the error for j^{th} point in the trajectory.

For smooth movement of the robot manipulator, there should be minimal changes in the link angles and there should not be abrupt variations in the link angles. To ensure this, the link angles generated for the previous point in the trajectory is also given as input to the algorithm.

3.1 Particle Swarm Optimization (PSO)

PSO algorithm was introduced by Kennedy and Eberhart [13]. It uses swarm intelligences for solving problems. It was inspired by the social behavior of the birds and fishes in finding their prey. A comprehensive study with the applications of PSO were presented in [24].

In PSO, each particle searches for the optimal solution thereby moving with a certain velocity. Each particle also remembers their best result as local best and the overall best of the entire population as global best. At each step, a particle has to move to a new position by adjusting its velocity so that its moves towards its local best $(P_{i,pbest})$ and the overall global best $(P_{i,gbest})$ record by the particle in the population. This is done iteratively until the optimal solution is achieved.

Each and every particle in the population represents the following set of parameter, $\langle Position\, P^t_{i,l} \,|\, Velocity\, V^t_{i,l} \rangle$. Here, the position vector $P^t_{i,l}$ represents the l^{th} link angles $(\theta_1,\ \ldots\ \theta_l, \ldots \theta_{nL})$ of i^{th} particle in t^{th} iteration. This is the required solution $(P_{i,l})$. The velocity $V^t_{i,l}$ represents an incremental change in the l^{th} link angle of i^{th} particle in t^{th} iteration. With nP as total number of particles and nL as total number of links, $i = 1, 2, \ldots, nP$ and $l = 1, 2, \ldots, nL$.

$$\begin{aligned} V^t_{i,l} =&\ K_{VF} V^{t-1}_{i,l} + C_p r_1 \left(P_{i,pbest} - P^{t-1}_{i,l}\right) \\ &+ C_g r_2 \left(P_{gbest} - P^{t-1}_{i,l}\right) \end{aligned} \tag{3}$$

$$P^t_{i,l} = P^{t-1}_{i,l} + V^t_{i,l} \tag{4}$$

The above equations represent the position and velocity of each particle in the population in which K_{VF} represents the current velocity factor, $r_1 \,\&\, r_2$ are uniformly distributed random number within the range $[0, 1]$, $C_p \,\&\, C_g$ represent the learning rates of local best and global best particles respectively. The values for these parameters are listed in are listed in Table 2.

3.2 Bat Algorithm (BAT)

developed by Xin-She Yang and Amir Hossein Gandomi [47]. It was inspired from echolocation behavior of bats with the varying pulse rate of emission and loudness which assists them in seeking for prey and/or avoids obstacles in complete darkness. All bats use echolocation to sense the distance. They fly randomly with a certain velocity and with a fixed frequency. During their flight, bats emit a sound pulse with particular loudness and listens to its echo that bounces back from the surrounding environment. Based on the difference in time between the emission of sound pulse and echo, they detect the distance of the prey, its orientation and even its motion. Based on their location the bats adjust its velocity and position thereby reaching the prey. The same principle is used by them to detect an obstacle and avoid them.

Each and every bat in the population represents the following set of parameters: $\langle Position\ P^t_{i,l}, Velocity V^t_{i.l}, frequency\ f_i,$ $Loudness\ A^t_i, Pulse\ rate\ \ r^t_i\rangle$. Here, the position vector $P^t_{i,l}$ represents the l^{th} link angles $(\theta_1,\ \ldots\ \theta_l, \ldots, \theta_{nL})$ of i^{th} bat in t^{th} iteration. The velocity $V^t_{i,l}$ represents an incremental change in the l^{th} link angle of i^{th} bat in t^{th} iteration. Bats emits wave with frequency in the range $[f_{min}\ f_{max}]$. A^t_i & r^t_i represents the loudness and pulse rate of the wave emitted by the i^{th} bat in t^{th} iteration. Of all these parameters, the position vector $P^t_{i,l}$ i.e., the link angles of the robot manipulator $(\theta_1, \ldots \theta_l, \ldots, \theta_{nL})$ represents the solution. Here, $i = 1, 2, \ldots, nB$ and $l = 1, 2, \ldots, nL$ the velocity of each bat and their position are calculated based on the following equations.

$$V^t_{i,l} = V^{t-1}_{i.l} + \left(P^{t-1}_{i.l} - P_{gbest}\right) f_i \tag{5}$$

$$P^t_{i,l} = P^{t-1}_{i,l} + V^t_{i,l} \tag{6}$$

To prevent the algorithm from getting stuck in a local minima / maximum, and to increase the exploration capability, a random walk is performed. Based on the pulse rate r^t_i, few bats are selected randomly to perform a random walk. A random position for the bat P_{new} is generated using equation (7). This newly generated position is selected based on the fitness (RMS Error) & the loudness A^t_i of the corresponding bat. The pulse rate r^t_i and loudness A^t_i of each bat is updated using equations (8,9) only if this new bat is accepted.

$$P_{new} = P_{gbest} + \varepsilon A^t \tag{7}$$

$$A^{t+1}_i = \omega A^t_i \tag{8}$$

$$r^{t+1}_i = r^0_i \left[1 - exp(-\gamma t)\right] \tag{9}$$

Here, ε is a random number in the range $[-1, 1]$, A^t is the average loudness of all the bats at time t, r^0_i is the initial pulse rate. Once a bat has found its prey, the pulse rate increases and the loudness decreases. This is achieved using the parameters ω, the pulse frequency increasing coefficient and γ, the pulse amplitude attenuation coefficient. The values of these parameters are listed in are listed in Table 2.

3.3 Gravitational Search Algorithm (GSA)

GSA is an optimization method based on Newtonian gravity and the laws of motion. This algorithm was developed by Rashedi et al. [37]. In the GSA, each particle in the search space is considered as a mass. Therefore, the GSA may be expressed as an artificial mass system. All masses in the search space attract each other according to Newton's gravity law and interact to exert force on each other with the force of gravity. Acting in the search space, masses exposed to these forces achieve the optimal solution.

Each and every particle mass in the system is represented by its position $\left(\ P^t_{i,l}\right)$. The gravitational constant is initialized at the beginning and will be reduced with time as in equation (10) to control the search accuracy.

$$G^t = G^0 e^{(-\alpha t / I_{max})} \tag{10}$$

Here, G^t is the gravitational constant at time any t and G^0 is the initial gravitational constant, α is the constant exponent factor. Here, $t = 1, 2, \ldots, I_{max}$, where I_{max} is the maximum iteration count of the algorithm. The fitness (RMS Error) of each mass $\left(\ P^t_{i,l}\right)$ are calculated and from this the best fitness value, i.e., the minimum RMS error (err^t_{min}) and worst fitness value, i.e., the maximum RMS error (err^t_{max}) are selected. From these, the gravitational mass $\left(\ m^t_i\right)$ of the i^{th} particle in the system at time t is calculated using equations (11) and the normalized gravitational mass $\left(\ M^t_i\right)$ is calculated using equation (12).

$$m^t_i = \frac{err^t_i - err^t_{max}}{err^t_{max} - err^t_{min}} \tag{11}$$

$$M^t_i = \frac{m^t_i}{\sum_{i=1}^{nP} m^t_i} \tag{12}$$

Here, nP is the total number of particle mass in the system. Now the force between the i^{th} & k^{th} particle mass in the system $\left(\ F^t_{i,k}\right)$ and the total force $\left(\ F^t_i\right)$ acting on any i^{th} particle mass at any time t are calculated using equation (13) and (14) respectively.

$$F^t_{i,k} = G^t \left(\frac{M^t_i \cdot\ M^t_k}{eud^t_{i,k} + eps}\right)\left(\ P^t_{k,l} -\ P^t_{i,l}\right) \tag{13}$$

$$F^t_i = \sum_{k \in bestK, k \neq i}^{nP} rand \cdot F^t_{i,k} \tag{14}$$

Here, $P^t_{i,l}$ & $P^t_{k,l}$ are the position of i^{th} & k^{th} particle mass and $eud^t_{i,k}$ is the Euclidian distance between

Table 2: Parameters used for BAT, GSA, PSO and WOA.

Algorithm	Parameters	Value
BAT	Frequency Range $[f_{min},\ f_{max}]$	$[-0.05,\ 0.05]$
	Pulse frequency increasing coefficient (ω)	0.97
	Pulse amplitude attenuation coefficient (γ)	0.1
	Loudness (A_i^t)	Random number in the range [1 2]
	Initial Pulse Rate (r_i^t)	Random number in the range [0 1]
GSA	Initial gravitational constant $\left(G^0\right)$	100
	Constant exponent factor (α)	1
	Random best agents $(bestK)$	(100 to 2)%
PSO	Current Velocity Factor (K_{VF})	0.3033
	Learning rates $(C_p$ & $C_g)$	2.4 & 3.2
	Uniformly distributed random number $(r_1$& $r_2)$	Random number in the range [0 1]
WOA	Shape of spiral (b)	5
	Solution selection parameter (p)	Random number in the range [0 1]

them which is calculated using equation (15). eps is a constant value added to avoid divide by zero condition, $rand$ is a random number between 0 and 1, $bestK$ is the set of for k agents with best fitness values, i.e., minimum RMS errors.

$$eud_{i,k}^t = ||P_{k,l}^t - Pi, l^t|| = \sqrt{\sum_{l=1}^{nL} (\theta_{k,l}^t - \theta_{i,l}^t)^2} \qquad (15)$$

Now the new solution or the update in position of the masses in the system are updated using the following equations.

$$P_{i,l}^t = P_{i,l}^{t-1} + V_{i,l}^t \qquad (16)$$

$$V_{i,l}^t = rand \cdot V_{i,l}^t + A_{i,l}^t \qquad (17)$$

$$A_{i,l}^t = \frac{F_i^t}{M_i^t} \qquad (18)$$

Here, $V_{i,l}^t$ & $A_{i,l}^t$ are the velocity and acceleration of the i^{th} agent at time t.

3.4 Whale Optimization Algorithm (WOA)

Mirjalili and Lewis [32] developed WOA which mimics the intelligent hunting behavior of humpback. This foraging behavior is called bubble-net feeding method which is observed only in humpback whales. The humpback whales dive down approximation 12 m and then create the bubble in a spiral shape around the prey. They then swim upward the surface following the bubbles and hunt the prey. Humpback whales can find the place of prey and encircle them. The WOA algorithm considers current best search agent position be the target prey or close to the optimum point, and other search agents will try to update their position towards the best search agent. Applications of WOA were presented in this work [20].

Each and every agent in the population is represented by its position vector $\left(P_{i,l}^t\right)$. Here, the position vector represents the l^{th} link angles $(\theta_1, \ldots\ \theta_l, \ldots \theta_{nL})$ of i^{th} agent in t^{th} iteration. This is the solution vector. Here, $i = 1, 2, \ldots, nA$ and $l = 1, 2, \ldots, nL$. nA is the total number of agents in the population and nL is the dimension of the solution, i.e., it represents the number of links of the robot manipulator in this work. The position of each agent is calculated and updated based on the following equations.

$$P_{i,l}^t = \begin{cases} P_{gbest}^t - (A * dist) & \text{if } p < 0.5 \\ dist' * e^{b*lrand} * cos\,(2\pi * lrand) + P_{gbest}^t & \text{if } p \geq 0.5 \end{cases} \qquad (19)$$

$$dist = 2 \cdot rand \cdot P_{gbest}^t - P_{i,l}^t \qquad (20)$$

$$dist' = \left|P_{gbest}^t - P_{i,l}^t\right| \qquad (21)$$

Here, $P_{i,l}^t$ is the position of the i^{th} agent with l dimension at time t, P_{gbest}^t is the best agent so far at time t, $rand$ & p are random numbers in the range [0 1]. A and $lrand$ are calculated as follows.

$$A = 2 \cdot rand \cdot a1 - a1 \qquad (22)$$

$$lrand = 1 + (a2 - 1) \cdot rand \qquad (23)$$

Here, $a1$ is a linearly decreasing vector from 2 to 0 over the course of iterations and $a2$ is a linearly decreasing vector from -1 to -2 over the course of iterations, b represents the shape of the spiral.

3.5 Algorithm Parameters

The values of the optimization parameters common to the algorithms are set as follows; Population size $n = 60$; the maximum iteration count used for trajectory following testing $I_{max} = 100$; the minimum acceptable error used for convergence testing is 0.1×10^{-4}. The parameters related to BAT, GSA, PSO and WOA used in this work are listed in Table 2.

3.6 Framework

The steps involved in carrying out the simulation are as follows. The 3D model of YASHKAWA MH5 robot is imported into MATLAB simulation environment [31].

1. Generate the trajectory for the selected curve (Linear / Curvilinear / Saw Tooth / Rose Curve)

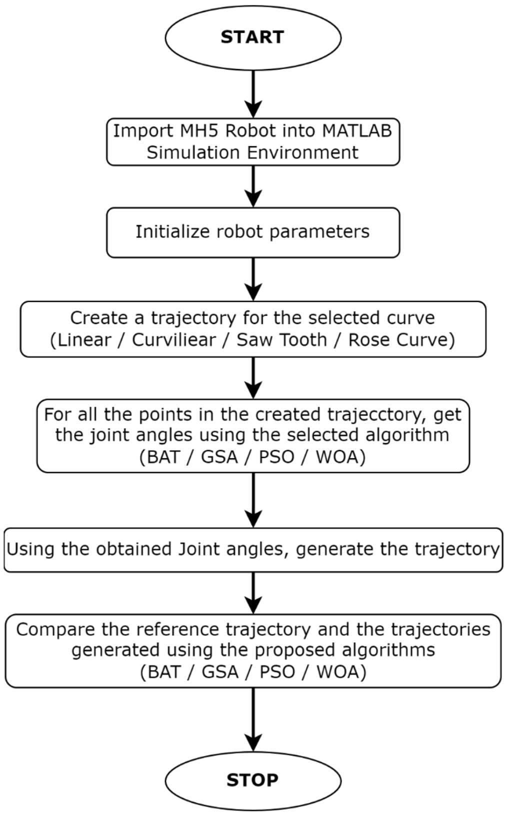
START
Import MH5 Robot into MATLAB Simulation Environment
Initialize robot parameters
Create a trajectory for the selected curve (Linear / Curviliear / Saw Tooth / Rose Curve)
For all the points in the created trajecctory, get the joint angles using the selected algorithm (BAT / GSA / PSO / WOA)
Using the obtained Joint angles, generate the trajectory
Compare the reference trajectory and the trajectories generated using the proposed algorithms (BAT / GSA / PSO / WOA)
STOP

Figure 1: Flowchart of the Simulation Framework.

2. Now each and every point $(x_d,\ y_d, z_d)_j$ in the generated trajectory, including waypoints, are given as input to the algorithms (BAT, GSA, PSO & WOA). Here $j = 1, 2, \dots, nPT$ and nPT is the number of points in the generated trajectory. The output from the algorithm are the link angles $(\theta_1, \dots, \theta_l, \dots, \theta_{nL})_j$ to reach the j^{th} point $(x_d,\ y_d, z_d)_j$. Here nL represents the number of links in the robot manipulator.

3. Now, using the link angles $(\theta_1, \dots, \theta_l, \dots, \theta_{nL})_j$, generated by the algorithm, the actual point reached by the robot manipulator $(x_a,\ y_a, z_a)_j$ is calculated.

4. The trajectory obtained using the points $(x_a,\ y_a, z_a)_j$ (obtained from BAT / GSA / PSO / WOA) is plotted and compared with the expected trajectory using the points $(x_d,\ y_d, z_d)_j$. and the error between these trajectories are generated.

5. The results are analyzed, interpreted and final conclusion is arrived.

The entire framework of the simulation carried out in this work presented in the flowchart in Figure 1.

3.7 Solution Representation and Fitness Evaluation

This section explains how the solution, i.e., link angles $(\theta_1, \dots, \theta_l, \dots \theta_{nL})$ generated by the algorithm are transformed into a cartesian point (x_a, y_a, z_a) in space, and based this how the fitness is evaluated. If we apply the link angles $(\theta_1, \dots, \theta_l, \dots \theta_{nL})$ to a robot manipulator, it will take a configuration and its end effector will reach a particular point in space. This end effector position calculation based on the link angles $(\theta_1, \dots \theta_l, \dots, \theta_{nL})$ is called forward kinematics. This can be calculated using the transformation matrix [12]. The general form of a standard transformation matrix from link $l-1$ to link l is given by

$$ {}^{link\ l-1}_{link}T = \begin{bmatrix} cos\theta_l & -sin\theta_l & 0 & a_{l-1} \\ sin\theta_l cos\alpha_{l-1} & cos\theta_l cos\alpha_{l-1} & -sin\alpha_{l-1} & -sin\alpha_{l-1}d_l \\ sin\theta_l sin\alpha_{l-1} & cos\theta_l sin\alpha_{l-1} & cos\alpha_{l-1} & cos\alpha_{l-1}d_l \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Where, a_{l-1} is the link length, α_{l-1} is the twist angle, d_l is the joint offset, θ_l is the joint angle or the link angle of the corresponding link. Here $l = 1, 2, \dots, nL$ and nL represents the number of links in the robot manipulator. The transformation matrix in equation (24) is calculated for base to link 1 $\left({}^{base}_{link\ 1}T\right)$, link 1 to link 2 $\left({}^{link\ 1}_{link\ 2}T\right)$, till link $nL-1$ to the end effector $\left({}^{link\ nL-1}_{end\ effector}T\right)$.

Multiplying all these matrices will give the transformation matrix from base to the end effector of the robot manipulator.

$$ {}^{base}_{end\ effector}T = {}^{base}_{link\ 1}T \cdot {}^{link\ 1}_{link\ 2}T \cdots {}^{link\ nL-1}_{end\ effector}T \tag{25}$$

In resulting 4×4 matrix ${}^{base}_{end\ effector}T$ from equation (24), the values in the location $(1,4)$, $(2,4)$, $(3,4)$ give the (x, y, z) coordinate of the end effector of robot manipulator. Thus, the link angles $(\theta_1, \dots \theta_l, \dots, \theta_{nL})$ are transformed into coordinates (x_a, y_a, z_a).

The input to this algorithm is any j^{th} point $(x_d, y_d, z_d)_j$ from the trajectory. This is the desired coordinate that the robot manipulator has to reach. From the randomly generated link angles $(\theta_1, \dots \theta_l, \dots, \theta_{nL})$, the coordinate point actually reached by the robot manipulator (x_a, y_a, z_a) is calculated. The fitness of this solution is calculated based on its deviation from the desired position $(x_d, y_d, z_d)_j$. This deviation is calculated based on the Root Mean Square (RMS) error, i.e., the fitness of the solution for any j^{th} point in the trajectory is calculated as

$$ RMSerror_j = \sqrt{(x_a - x_d)_j^{\ 2} + (y_a - y_d)_j^{\ 2} + (z_a - z_d)_j^{\ 2}} \tag{26}$$

4 Results and Discussions

The simulation environment and the test cases used for measuring the efficiency of the proposed methodology are explained in this section. MATLAB (R2020b) environment is used to simulate the proposed work. YASKAWA's MH5 robot manipulator is considered to carryout the stated objective. The actual robot and the simulated robot are shown in Figure 2.

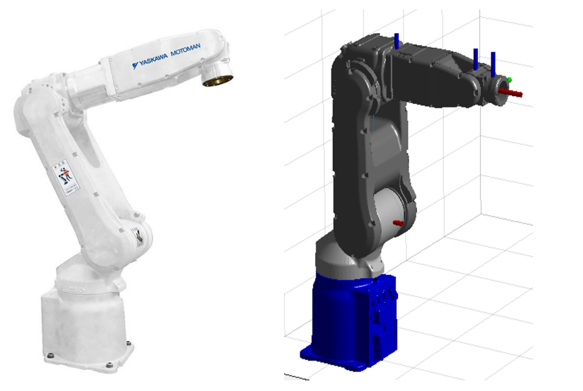

Figure 2: (a) YASKAWA's MH5 robot manipulator Physical robot (b) Robot simulated in MATLAB environment.

Table 3: YASKAWA's MH5 Robot Manipulator Parameters [33].

Name of the Line	Link Angle Range (°)	Maximum Speed (°/s)
S-Axis: Swivel Base	$-170 \leq \theta_1 \leq +170$	376
L-Axis: Lower Arm	$-065 \leq \theta_2 \leq +150$	350
U-Axis: Upper Arm	$-136 \leq \theta_3 \leq +255$	400
R-Axis: Arm Roll	$-190 \leq \theta_4 \leq +190$	450
B-Axis: Wrist Bend	$-135 \leq \theta_5 \leq +135$	450
T-Axis: Tool Flange	$-360 \leq \theta_6 \leq +360$	720

This robot has 6 links (nL). Therefore the six link angles ($\theta_1, \theta_2 \ldots, \theta_6$) corresponding to the 6 links are adjusted simultaneously to control the position and orientation of the robot's end effector. The other parameters related to this robot manipulator are shown in Table 3.

The performance of the proposed algorithms for trajectory following are assessed using four different trajectories, viz., linear, curvilinear, sawtooth and rose trajectories. The trajectories simulated in MATLAB environment are shown in Figure 3.

Each trajectory consists of j points, where, $j = 1, 2, \ldots, nPT$, and nPT is the number of points in the trajectory. In this paper, nPT is considered as 17, i.e., we generate 15 intermediate points between the start and endpoint of the trajectory, plus one start point and one end point. The number of intermediate points in the trajectory is selected as 15 for the sake of simplicity and ease of comparison. The number of points in the trajectory can be decided based on the application. The error between the expected trajectory and the trajectory generated by BAT, GSA, PSO and WOA are calculated for these 17 points.

The objective of this work is to find the joint angles of the robot manipulator to reach the points in the trajectory with minimum error. So, the proposed work emphasis on following the trajectory with minimum error and does not depend on the number of points in the trajectory. The algorithm is tested for its closeness in following the trajectory. All experiments are deployed on the PC with Intel (R) Core (TM) i3, M370 @2.40GHz, 8GB RAM using MATLAB 2020b.

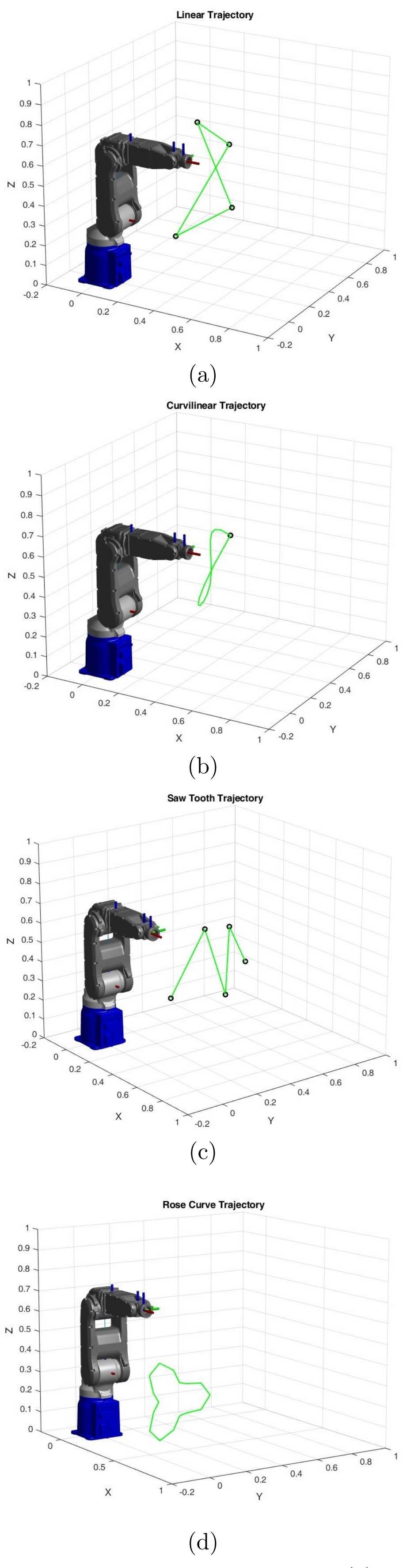

Figure 3: Expected reference trajectory (a) linear (b) curvilinear (c) sawtooth (d) rose curve.

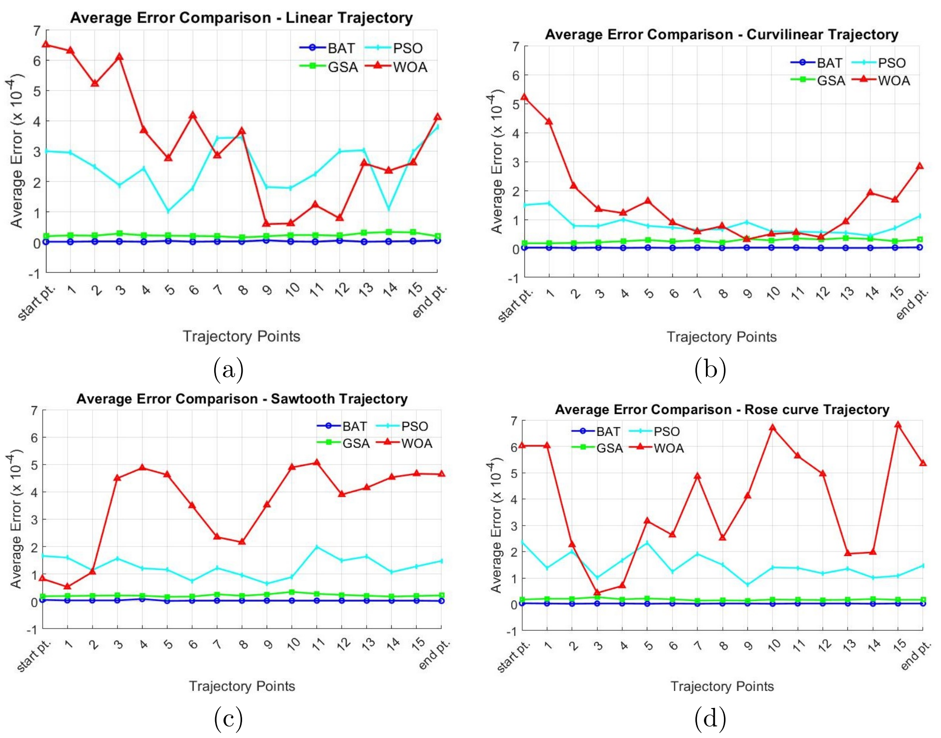

Figure 4: Average error plots (a) linear trajectory (b) curvilinear trajectory (c) sawtooth trajectory (d) rose curve trajectory.

Table 4: Average Error $(\times 10^{-4})$ between expected and generated trajectory.

Algorithm	Trajectory Type	Intermediate points in the Trajectory																
		Start	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	End
Linear	BAT	0.02	0.02	0.03	0.03	0.02	0.05	0.02	0.03	0.03	0.07	0.03	0.02	0.06	0.02	0.03	0.04	0.06
	GSA	0.20	0.23	0.22	0.29	0.23	0.22	0.21	0.20	0.16	0.20	0.24	0.24	0.22	0.31	0.34	0.33	0.20
	PSO	3.00	2.95	2.48	1.88	2.43	1.03	1.79	3.43	3.45	1.82	1.79	2.25	2.99	3.03	1.11	2.97	3.79
	WOA	6.50	6.30	5.21	6.09	3.68	2.76	4.17	2.85	3.65	0.60	0.62	1.23	0.79	2.60	2.35	2.62	4.11
Curvilinear	BAT	0.03	0.03	0.02	0.03	0.02	0.03	0.02	0.03	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.03	0.04
	GSA	0.18	0.18	0.19	0.21	0.25	0.29	0.24	0.28	0.21	0.33	0.28	0.35	0.31	0.36	0.33	0.25	0.31
	PSO	1.50	1.56	0.78	0.77	1.00	0.78	0.72	0.65	0.66	0.91	0.59	0.58	0.56	0.54	0.44	0.70	1.12
	WOA	5.21	4.36	2.15	1.35	1.22	1.63	0.89	0.58	0.77	0.31	0.50	0.55	0.39	0.92	1.92	1.67	2.83
Sawtooth	BAT	0.06	0.04	0.04	0.04	0.09	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02
	GSA	0.19	0.20	0.21	0.22	0.21	0.17	0.18	0.26	0.21	0.26	0.35	0.28	0.24	0.21	0.18	0.20	0.22
	PSO	1.66	1.60	1.14	1.57	1.21	1.16	0.75	1.22	0.96	0.65	0.89	1.99	1.49	1.64	1.07	1.28	1.48
	WOA	0.83	0.53	1.07	4.49	4.87	4.62	3.49	2.35	2.16	3.52	4.89	5.06	3.90	4.15	4.53	4.66	4.64
Rose curve	BAT	0.04	0.03	0.02	0.03	0.03	0.02	0.03	0.02	0.03	0.03	0.02	0.03	0.03	0.03	0.02	0.03	0.03
	GSA	0.18	0.21	0.21	0.27	0.19	0.22	0.19	0.14	0.15	0.14	0.18	0.17	0.16	0.17	0.20	0.17	0.17
	PSO	2.36	1.38	2.01	1.01	1.67	2.33	1.24	1.91	1.50	0.74	1.40	1.38	1.17	1.35	1.01	1.08	1.47
	WOA	6.02	6.02	2.26	0.43	0.70	3.16	2.63	4.86	2.51	4.11	6.70	7.09	4.95	1.92	1.97	6.81	5.34

Table 5: Statistical analysis of average error $(\times 10^{-4})$, bold face represents best values.

Algorithm	Trajectory Type	Mean	Median	Standard Deviation	Minimum	Maximum
Linear	BAT	**0.03**	**0.03**	**0.01**	**0.02**	**0.07**
	GSA	0.23	0.22	0.05	0.16	0.34
	PSO	2.33	2.48	0.79	1.03	3.79
	WOA	2.63	2.85	1.88	0.60	6.50
Curvilinear	BAT	**0.03**	**0.03**	**0.00**	**0.02**	**0.04**
	GSA	0.26	0.28	0.06	0.18	0.36
	PSO	0.77	0.72	0.31	0.44	1.56
	WOA	1.17	1.22	1.35	0.31	5.21
Sawtooth	BAT	**0.03**	**0.03**	**0.02**	**0.02**	**0.09**
	GSA	0.22	0.21	0.04	0.17	0.35
	PSO	1.23	1.22	0.35	0.65	1.99
	WOA	2.98	4.15	1.49	0.53	5.06
Rose curve	BAT	**0.03**	**0.03**	**0.00**	**0.02**	**0.04**
	GSA	0.18	0.18	0.03	0.14	0.27
	PSO	1.41	1.38	0.44	0.74	2.36
	WOA	3.13	4.11	2.02	0.43	6.81

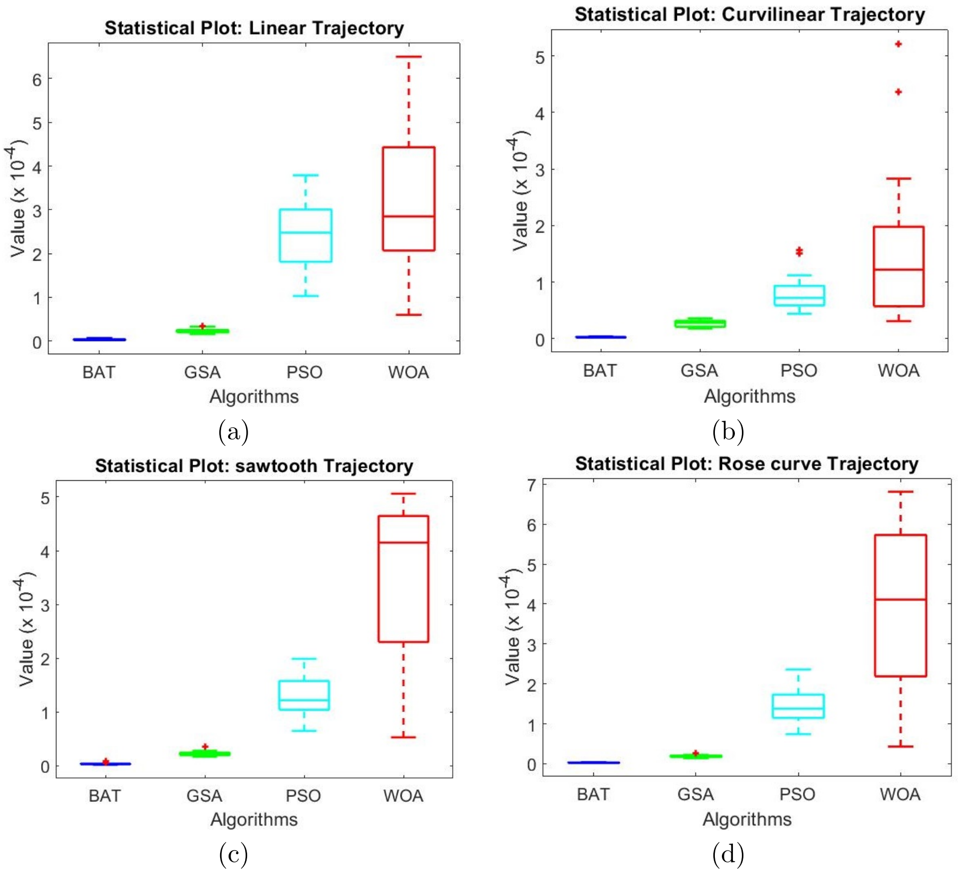

Figure 5: Statistical plot of average error for different trajectories (a) Linear (b) Curvilinear (c) Sawtooth (d) Rose curve.

4.1 Trajectory Following Testing

Each of the 4 different trajectories are tested with four different algorithms. So, a total of 16 different test cases are carried out for trajectory following experiment. To investigate the robustness, each of these 16 different cases are run for 20 times and the average objective values are recorded. The performance of the algorithm is measured in terms of its closeness in following the trajectory.

For each and every point in the reference trajectory as input, the algorithms are executed for a maximum iteration count $I_{max} = 100$. At the end of 100^{th} iteration, the best solution is taken as the output. The trajectory is generated for all the four curves considered in this paper using the proposed algorithms. Each of this case is run for 20 times and the average value of error between the expected and the generated trajectories are recorded. The plot of average error obtained for these algorithms are shown in Figure 4 and the values are listed in Table 4.

Average error values in Table 4 shows that BAT and GSA closely follows the reference trajectory with minimum error when compared to other algorithms considered in this paper. Both BAT and GSA produced consistent result at all times. This is evident from the plot as there are very less variation in the average error. This is further investigated using the statistical analysis which are listed in Table 5.

Table 5 shows that GSA's performance is next best. Both BAT and GSA were able to provide solution, with consistent error values, for all points in the trajectory. Where as PSO and WOA have a higher standard deviation value. This is evident from the box whisker plot shown in Figure 5.

4.2 Convergence Testing

The objective of every optimization algorithm is to converge to a global optimal solution. This convergence capability of BAT, GSA, PSO and WOA for the inverse kinematics problem is tested as follows.

In trajectory following testing, the stopping condition was the maximum iteration count $I_{max} = 100$, i.e., the algorithm is executed for 100 iterations and at the end of 100^{th} iteration, the best solution is taken as the output. In convergence testing, a minimum acceptable error is set as the stopping condition. The algorithm is executed till the minimum acceptable error is achieved. So, a random point in the trajectory is selected and given as input to the algorithm. The minimum acceptable error is set as 0.1×10^{-4}. To prevent the algorithm from running continuously, the iteration limit is set as 1000. Each algorithm is executed 20 times and the average value of loop count and execution time required to achieve this acceptable error are recorded and listed in Table 6.

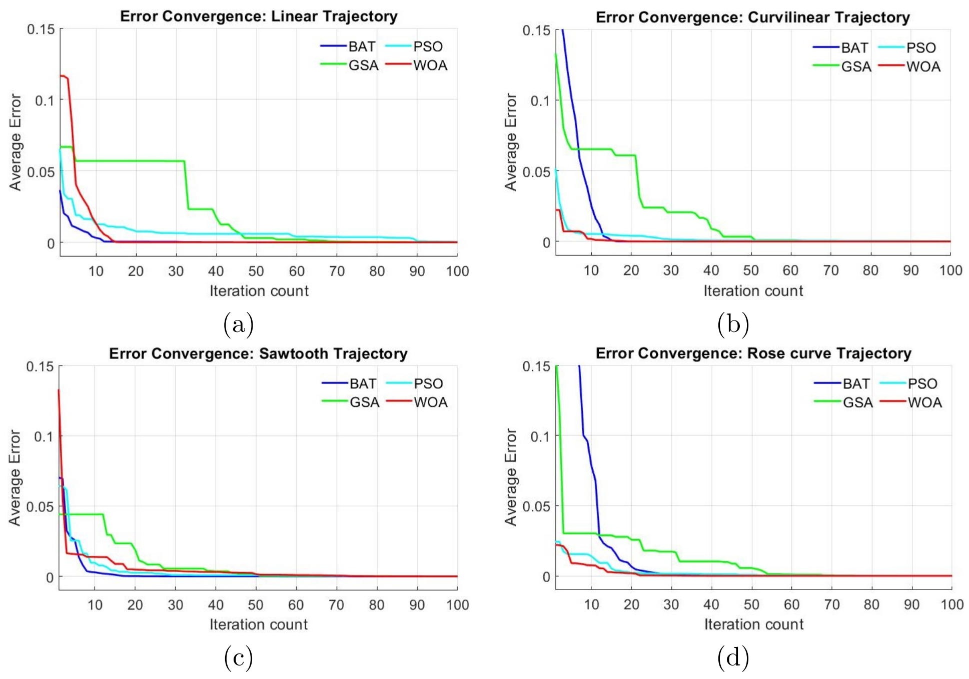

Figure 6: Error convergence plot comparison for (a) linear trajectory using (b) curvilinear trajectory (c) sawtooth trajectory (d) rose curve trajectory.

Table 6: Convergence analysis.

Trajectory Type	Parameter	Algorithm			
		BAT	GSA	PSO	WOA
Linear	Avg. loop count	22	88	96	46
	Avg. time (s)	0.48	1.89	1.99	1.02
Curvilinear	Avg. loop count	18	86	81	78
	Avg. time (s)	0.39	1.99	1.72	1.79
Sawtooth	Avg. loop count	21	87	116	69
	Avg. time (s)	0.47	1.87	2.49	1.57
Rose curve	Avg. loop count	38	84	135	65
	Avg. time (s)	0.82	1.93	2.91	1.48

In case of average loop count required to reach the acceptable level of error, BAT still maintains its lead. Though GSA was able to closely follow the trajectory with lesser average error when compared with WOA, the loop count required to achieve the results are more than that of WOA.

To visualize the convergence of error, the error values during each iteration for a random point in the trajectory is recorded. The average value of error after 20 executions for all the four trajectories are shown in Figure 6. This plot also confirms the PSO and GSA consumes more iterations to converge where as WOA and BAT converges faster.

4.3 Singularity Testing

Furthermore, to check whether singularity is avoided in the solution created by these proposed algorithms, the joint angles $(\theta_1, \ldots \theta_l, \ldots, \theta_6)$ are recorded. The variation of joint angles from the starting point to the end point in the trajectory are noted. The plot of joint angle variation for all the four trajectories considered in this paper are shown in Figure 7 and Figure 8.

The joint angle variation plot in Figure 7 and 8 shows that the joint angles generated for all the four trajectories using BAT, GSA, PSO and WOA are within the limits specified in Table 3 thereby avoiding singularity.

Investigating further into the joint angle variation plots, GSA shows the best results. They were able to create solution with minimal change in link angles throughout the trajectory. This is seen in all the four trajectories. Comparatively, more variation in the joint angles is seen in WOA, PSO for linear trajectory and in PSO for curvilinear trajectory.

4.4 Comparison using Wilcoxon test

Wilcoxon test is conducted to compare the performance of the proposed algorithms for robot manipulator trajectory tracking applications. The average error during trajectory tracking, average loop count and time required to reach a minimum acceptable error level, variation in the joint angles are considered as parameters for this test. The results are listed in Table 7.

The result of Wilcoxon test reveal that BAT, GSA are better in terms of the parameters considered in this paper.

5 Conclusion

In this paper, four meta-heuristic-based approaches (BAT, GSA, PSO & WOA) were used to make a robot manipulator follow four pre defined paths. The performance of these algorithms was measured based on their accuracy and precision in following the trajectory, average loop count and time required to create solution

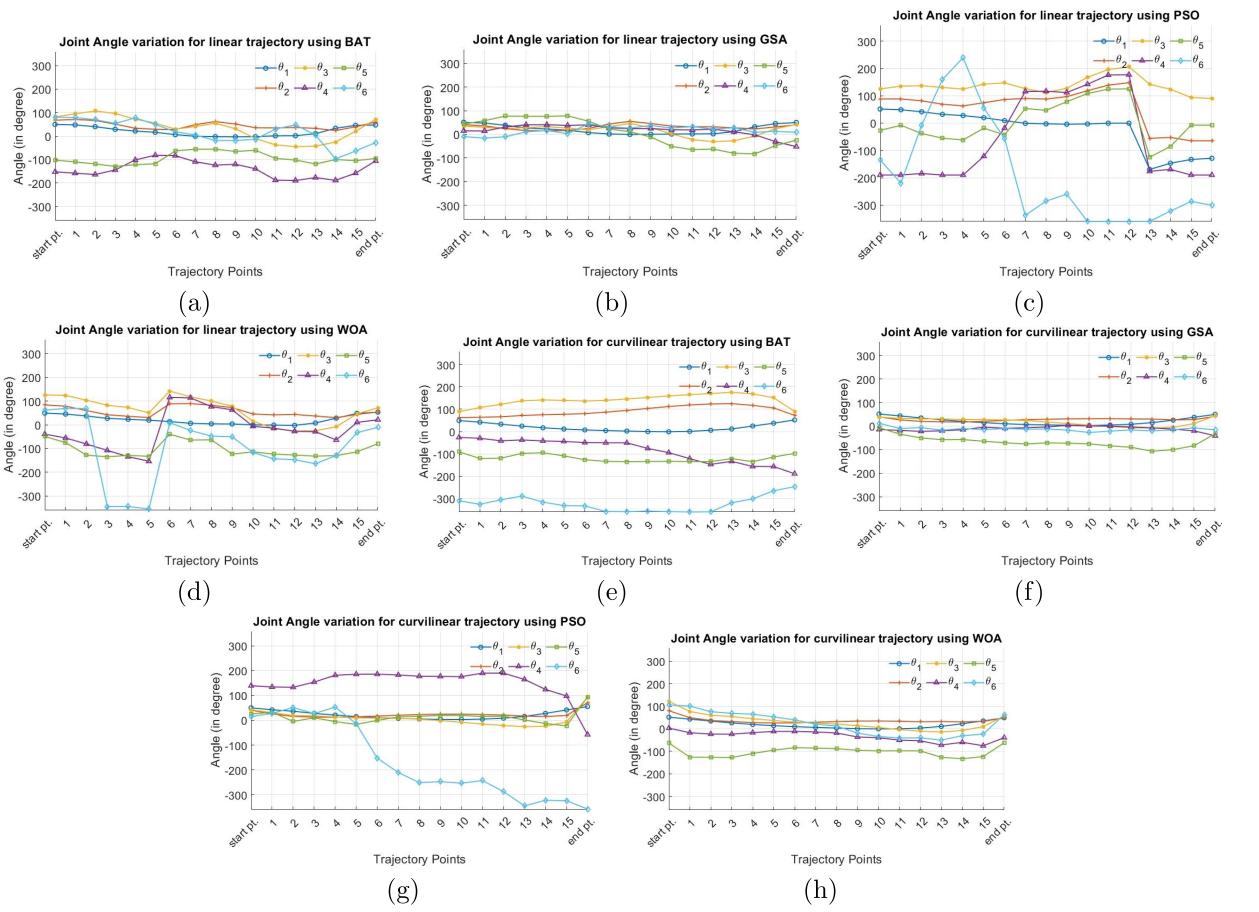

Figure 7: Generated joint angles for linear trajectory using (a) BAT (b) GSA (c) PSO (d) WOA Curvilinear trajectory using (e) BAT (f) GSA (g) PSO (h) WOA.

Table 7: Statistical analysis using Wilcoxon test.

Evaluation Parameters	**Trajectory Type**	**Rank**			
		BAT	**GSA**	**PSO**	**WOA**
Minimum Average Error	Linear	1	2	3	4
	Curvilinear	1	2	3	4
	Sawtooth	1	2	3	4
	Rose curve	1	2	3	4
Fast Convergence	Linear	1	3	4	2
	Curvilinear	1	4	3	2
	Sawtooth	1	3	4	2
	Rose curve	1	3	4	2
Minimal variation in joint angles	Linear	2	1	4	3
	Curvilinear	3	1	4	2
	Sawtooth	3	1	4	2
	Rose curve	3	1	4	2
	Average Rank	1.58	2.08	3.58	2.75
	Final Rank	1	2	4	3

for a particular acceptable error and minimal variation in the joint angles.

The experimental results showed that BAT closely followed the trajectory with an average error of 0.03×10^{-4} for all the four trajectories. Followed by this, GSA was ranked 2 with an average error in the range $(0.08\ to\ 0.26) \times 10^{-4}$. The average loop count and time testing showed that BAT has superior performance with WOA, GSA and PSO in the second, third and fourth positions. Joint angle variation testing revealed that GSA outperformed all other algorithms considered in this paper. In short BAT has shown the potential of getting faster convergence and yielding global optimum solution for the inverse kinematics problem along with trajectory tracking. GSA could create solution with lesser variations in the joint angles for inverse

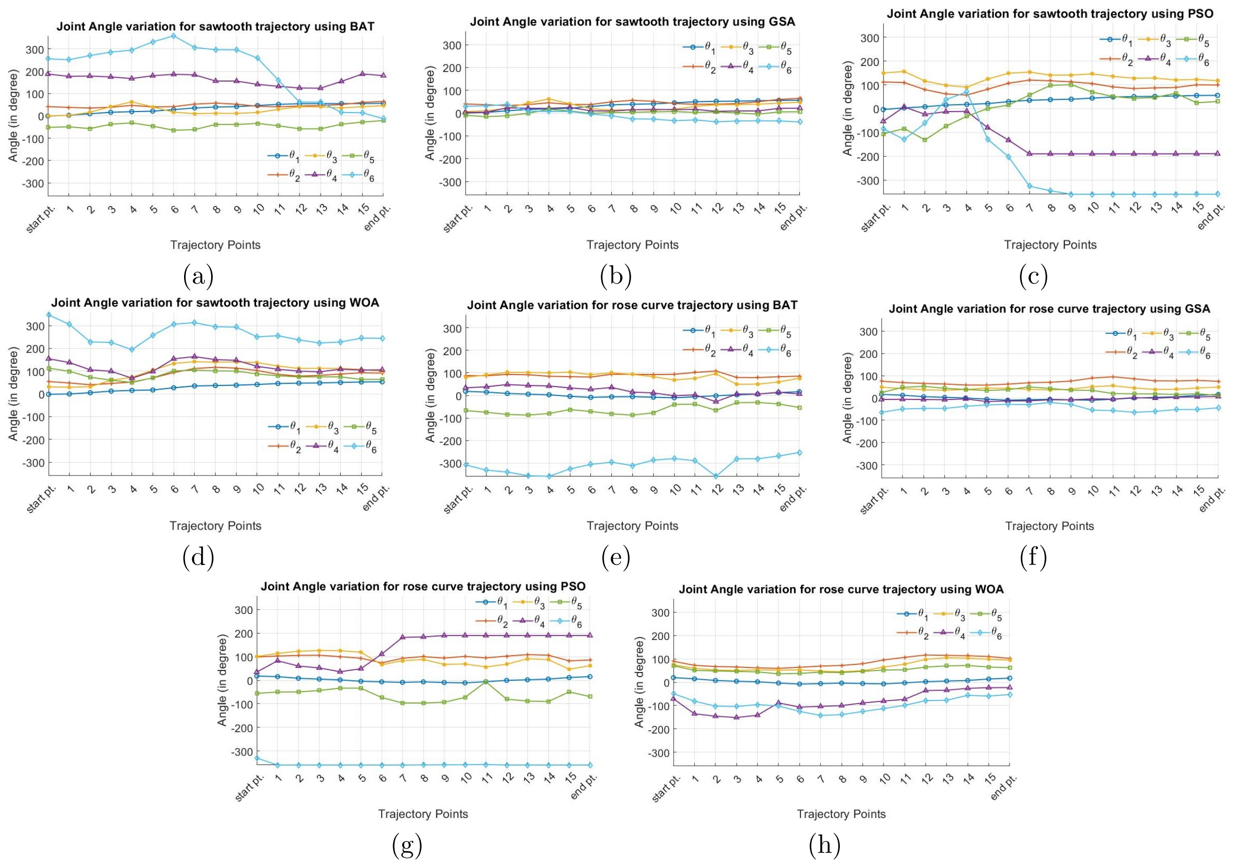

Figure 8: Generated joint angles for sawtooth trajectory using (a) BAT (b) GSA (c) PSO (d) WOA Rose curve trajectory using (e) BAT (f) GSA (g) PSO (h) WOA.

kinematics problem with minimal error.

The future scope of this work includes parameter tuning of the algorithms to get better accuracy. Furthermore, the environment considered in this paper is 3D with robot manipulator constrain only. No obstacles were considered in the robot workspace. Detection of obstacles, both static and dynamic, in the robot manipulator workspace can be taken as a future work. Detection of the obstacle, along with its dimension, its location in 3D space is required to alter the path of the robot manipulator. So, detecting the obstacle's presence, calculating its dimension, locating the obstacle and tracking its movement, in case of dynamic obstacle are needed to enable the robot manipulator to operate in a more constrained environment. Lastly, existing algorithms such as BAT, GSA, PSO and WOA are experiment in this work to determine the solution of robot manipulator for trajectory following. To gain advantage of the best quality in the existing meta-heuristic algorithms, hybrid algorithms can be developed and experiment for the inverse kinematic problem considered in this paper.

References

[1] Andreev, A., and Peregudova, O. Trajectory tracking control for robot manipulators using only position measurements. *International Journal of Control 92*, 7 (2019), 1490–1496.

[2] Ayyildiz, M., and Çetinkaya, K. Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-dof serial robot manipulator. *Neural Computing and Applications 27*, 4 (2016), 825–836.

[3] Baek, J., Cho, S., and Han, S. Practical time-delay control with adaptive gains for trajectory tracking of robot manipulators. *IEEE Transactions on Industrial Electronics 65*, 7 (2017), 5682–5692.

[4] Bayati, M. Using cuckoo optimization algorithm and imperialist competitive algorithm to solve inverse kinematics problem for numerical control of robotic manipulators. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 229*, 5 (2015), 375–387.

[5] Beyer, U., and Śmieja, F. A heuristic approach to the inverse differential kinematics problem. *Journal of Intelligent and Robotic Systems 18*, 4 (1997), 309–327.

[6] Chen, C.-Y., Her, M.-G., Hung, Y.-C., and Karkoub, M. Approximating a robot inverse kinematics solution using fuzzy logic tuned by genetic algorithms. *The International Journal of Advanced Manufacturing Technology 20*, 5 (2002), 375–380.

[7] Chen, D., Zhang, Y., and Li, S. Tracking control of robot manipulators with unknown models:

A jacobian-matrix-adaption method. *IEEE Transactions on Industrial Informatics 14*, 7 (2017), 3044–3053.

[8] Cheng, J., Zhang, G., Caraffini, F., and Neri, F. Multicriteria adaptive differential evolution for global numerical optimization. *Integrated Computer-Aided Engineering 22*, 2 (2015), 103–107.

[9] Chin, K. M., Teh, S.-H., Ho, J.-H., and Ng, H. K. Controller design and trajectory tracking of a two-link robotic orthosis via sinusoidal-input describing function model. *International Journal of Mechanical Engineering and Robotics Research 8*, 6 (2019).

[10] Chyan, G. S., and Ponnambalam, S. Obstacle avoidance control of redundant robots using variants of particle swarm optimization. *Robotics and Computer-Integrated Manufacturing 28*, 2 (2012), 147–153.

[11] DeMers, D., and Kreutz-Delgado, K. Inverse kinematics of dextrous manipulators. In *Neural Systems for Robotics*. Elsevier, 1997, pp. 75–116.

[12] Denavit, J., and Hartenberg, R. S. A kinematic notation for lower-pair mechanisms based on matrices.

[13] Eberhart, R., and Kennedy, J. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the sixth international symposium on micro machine and human science* (1995), Ieee, pp. 39–43.

[14] El-Sherbiny, A., Elhosseini, M. A., and Haikal, A. Y. A new abc variant for solving inverse kinematics problem in 5 dof robot arm. *Applied Soft Computing 73* (2018), 24–38.

[15] Emami, S. A., and Banazadeh, A. Simultaneous trajectory tracking and aerial manipulation using a multi-stage model predictive control. *Aerospace Science and Technology 112* (2021), 106573.

[16] Fang, J., Mei, T., Zhao, J., and Li, T. A dual-mode online optimization method for trajectory tracking of redundant manipulators. *Industrial Robot: An International Journal* (2016).

[17] Funda, J., Taylor, R. H., and Paul, R. P. On homogeneous transforms, quaternions, and computational efficiency. *IEEE transactions on Robotics and Automation 6*, 3 (1990), 382–388.

[18] Gan, D., Liao, Q., Wei, S., Dai, J., and Qiao, S. Dual quaternion-based inverse kinematics of the general spatial 7r mechanism. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 222*, 8 (2008), 1593–1598.

[19] Geradin, M., and Cardona, A. Kinematics and dynamics of rigid and flexible mechanisms using finite elements and quaternion algebra. *Computational Mechanics 4*, 2 (1988), 115–135.

[20] Gharehchopogh, F. S., and Gholizadeh, H. A comprehensive survey: Whale optimization algorithm and its applications. *Swarm and Evolutionary Computation 48* (2019), 1–24.

[21] Guo, J., Qiu, B., Hu, C., and Zhang, Y. Discrete-time nonlinear optimization via zeroing neural dynamics based on explicit linear multistep methods for tracking control of robot manipulators. *Neurocomputing 412* (2020), 477–485.

[22] Hernandez-Barragan, J., Lopez-Franco, C., Arana-Daniel, N., Alanis, A. Y., and Lopez-Franco, A. A modified firefly algorithm for the inverse kinematics solutions of robotic manipulators. *Integrated Computer-Aided Engineering 28*, 3 (2021), 257–275.

[23] Houcine, L., Bouzbida, M., and Chaari, A. Improved fuzzy clustering algorithm using adaptive particle swarm optimization for nonlinear system modeling and identification. *Iranian Journal of Fuzzy Systems 18*, 3 (2021), 179–196.

[24] Houssein, E. H., Gad, A. G., Hussain, K., and Suganthan, P. N. Major advances in particle swarm optimization: theory, analysis, and application. *Swarm and Evolutionary Computation 63* (2021), 100868.

[25] Huang, H.-C., Xu, S. S.-D., and Hsu, H.-S. Hybrid taguchi dna swarm intelligence for optimal inverse kinematics redundancy resolution of six-dof humanoid robot arms. *Mathematical Problems in Engineering 2014* (2014).

[26] Husty, M. L., Pfurner, M., and Schröcker, H.-P. A new and efficient algorithm for the inverse kinematics of a general serial 6r manipulator. *Mechanism and machine theory 42*, 1 (2007), 66–81.

[27] Joo, D., and Yeom, K. Improved hybrid trajectory tracking algorithm for a 3-link manipulator using artificial neural network and kalman filter [j]. *International Journal of Mechanical Engineering and Robotics Research 10*, 2 (2021), 60–66.

[28] Kanagaraj, G., Masthan, S. A. R. S., and Vincent, F. Y. Inverse kinematic solution of obstacle avoidance redundant robot manipulator by batalgorithms. *International Journal of Robotics and Automation 36*, 1 (2021).

[29] Karimi, J., and Pourtakdoust, S. H. Optimal maneuver-based motion planning over terrain and threats using a dynamic hybrid pso algorithm. *Aerospace Science and Technology 26*, 1 (2013), 60–71.

[30] Lafmejani, A. S., Masouleh, M. T., and Kalhor, A. Trajectory tracking control of a pneumatically actuated 6-dof gough–stewart parallel robot using backstepping-sliding mode controller and geometry-based quasi forward kinematic method. *Robotics and Computer-Integrated Manufacturing 54* (2018), 96–114.

[31] MathWorks India. Import rigid body tree model from urdf file, text, or simscape

multibody model - matlab importrobot, https://in.mathworks.com/help/robotics/ref/importrobot.html (Accessed 06/2022).

[32] Mirjalili, S., and Lewis, A. The whale optimization algorithm. *Advances in engineering software 95* (2016), 51–67.

[33] Motoman. Motoman mh5ls ii robot for assembly & handling — 5.0 kg, https://www.motoman.com/en-us/products/robots/industrial/assembly-handling/mh-series/mh5ls-ii (Accessed 06/2022).

[34] Nearchou, A. C. Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism and machine theory 33*, 3 (1998), 273–292.

[35] Pluháček, M., Kazikova, A., Kadavy, T., Viktorin, A., and Senkerik, R. Relation of neighborhood size and diversity loss rate in particle swarm optimization with ring topology. *Mendel 27*, 2 (2021), 74–79.

[36] Ram, R., Pathak, P. M., and Junco, S. Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling. *Mechanism and Machine Theory 131* (2019), 385–405.

[37] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. Gsa: a gravitational search algorithm. *Information sciences 179*, 13 (2009), 2232–2248.

[38] Rokbani, N., Casals, A., and Alimi, A. M. Ik-fa, a new heuristic inverse kinematics solver using firefly algorithm. In *Computational intelligence applications in modeling and control*. Springer, 2015, pp. 369–395.

[39] Sadat Asl, A., Fazel Zarandi, M., Sotudian, S., and Amini, A. A fuzzy capacitated facility location-network design model: A hybrid firefly and invasive weed optimization (fiwo) solution. *Iranian Journal of Fuzzy Systems 17*, 2 (2020), 79–95.

[40] Setayandeh, M., and Babaei, A. A novel method for multi-objective design optimization based on fuzzy systems. *Iranian Journal of Fuzzy Systems 18*, 5 (2021), 181–198.

[41] Shen, D., Tang, L., Hu, Q., Guo, C., Li, X., and Zhang, J. Space manipulator trajectory tracking based on recursive decentralized finite-time control. *Aerospace Science and Technology 102* (2020), 105870.

[42] Tarokh, M., and Zhang, X. An adaptive genetic algorithm for real-time robotic trajectory tracking. *IFAC Proceedings Volumes 39*, 15 (2006), 199–204.

[43] Theodoridis, D. C., Boutalis, Y. S., and Christodoulou, M. A. A new adaptive neuro-fuzzy controller for trajectory tracking of robot manipulators. *International Journal of Robotics and Automation 26*, 1 (2011), 64.

[44] Thomas, M. J., Sanjeev, M. M., Sudheer, A., and Joy, M. Comparative study of various machine learning algorithms and denavit–hartenberg approach for the inverse kinematic solutions in a 3-ppss parallel manipulator. *Industrial Robot: the international journal of robotics research and application 47*, 5 (2020), 683–695.

[45] Van Cuong, P., and Nan, W. Y. Adaptive trajectory tracking neural network control with robust compensator for robot manipulators. *Neural Computing and Applications 27*, 2 (2016), 525–536.

[46] Wang, G.-G., Chu, H. E., and Mirjalili, S. Three-dimensional path planning for ucav using an improved bat algorithm. *Aerospace Science and Technology 49* (2016), 231–238.

[47] Yang, X.-S., and Gandomi, A. H. Bat algorithm: a novel approach for global engineering optimization. *Engineering computations* (2012).

[48] Yarza, A., Santibanez, V., and Moreno-Valenzuela, J. Uniform global asymptotic stability of an adaptive output feedback tracking controller for robot manipulators. *IFAC Proceedings Volumes 44*, 1 (2011), 14590–14595.

[49] Yin, X., and Pan, L. Enhancing trajectory tracking accuracy for industrial robot with robust adaptive control. *Robotics and Computer-Integrated Manufacturing 51* (2018), 97–102.

[50] Yin, X., Pan, L., and Cai, S. Robust adaptive fuzzy sliding mode trajectory tracking control for serial robotic manipulators. *Robotics and Computer-Integrated Manufacturing 72* (2021), 101884.

[51] Younis, M. T., and Yang, S. Genetic algorithm for independent job scheduling in grid computing. In *Mendel* (2017), vol. 23, pp. 65–72.

[52] Zhao, Y. M., Lin, Y., Xi, F., and Guo, S. Calibration-based iterative learning control for path tracking of industrial robots. *IEEE Transactions on industrial electronics 62*, 5 (2014), 2921–2929.

[53] Zorić, N. D., Lazarević, M. P., and Simonović, A. M. Multi-body kinematics and dynamics in terms of quaternions: Langrange formulation in covariant form: Rodriguez approach. *FME Transactions 38*, 1 (2010), 19–28.