International Journal on Robotics, Automation and Sciences

Inverse Kinematics Analysis of Novel 6-DOF Robotic Arm Manipulator for Oil and Gas Welding Using Meta-Heuristic Algorithms

Bassey Etim Nyong-Bassey*, Ayebatonye Marttyns Epemu

***Abstract* – This research presents a comparison of the grey-wolf, improved grey-wolf, particle swarm, jellyfish and whale optimisation algorithms regarding the inverse kinematics solution of a newly designed 6-degrees of freedom robotic arm for oil and gas pipeline welding which has not been used in literature. Consequently, due to the robot's multiple joints with compounding combinatory possibilities of joint angles, the analysis of the inverse kinematics is relatively complex. In this research, the meta-heuristic algorithms, have been used to determine the robotic arm's inverse kinematics, essential for tracking a rectangular trajectory with six sets of waypoints in the 3D [X, Y, Z] space. The results were further analysed in terms of the accuracy of the position of the end effector from the accurate position of the rectangular target trajectory via a mean squared error cost function. Furthermore, the results of comparison between the meta-heuristic algorithms to position error from the inverse kinematics task demonstrated the superior performance of the grey-wolf algorithm over the particle swarm, improved grey-wolf, jellyfish, and whale optimisation algorithms.**

Keywords—Grey wolf, Robotic arm, 6 DoF, Inverse Kinematics, Meta-heuristic optimization.

I. INTRODUCTION

The use of robotic arms has been applied to a wide range of processes, including pick-and-place, welding, and assembly. It is essential that the end-effector of the robot follows the specified trajectory precisely and smoothly when performing automated tasks such as welding. To control the motion of a robotic arm, two factors must be considered and applied: inverse kinematics and trajectory planning [1].

The kinematic analysis (KA) describes the structure of a robotic manipulator mathematically. The KA focuses solely on the robot's movement, without considering the force which causes it. Furthermore, KA describes the relationship between the end effector and the base, as well as the intermediate links [2, 3]. There are two main categories of KA; forward and inverse kinematics. The forward kinematics (FK) determines the cartesian position of the end effector given a set of fully defined joint parameters. While, inverse kinematics (IK) utilizes joint angles, positions, and orientations to determine a given position and orientation of the end-effector. The IK method is more complicated and requires more constraints than the forward kinematics method. Most robotic arm designs and analyses use forward kinematics and inverse kinematics [4–6].

Robotics relies heavily on inverse kinematics solutions, especially for design, analysis, calibration, and control tasks [7]. Nevertheless, realising IK solutions analytically is difficult and cumbersome, especially for robots with higher degrees of freedom [8]. For the aforementioned reason, meta-heuristic algorithms such as particle swarm and a novel dual particle swarm have been successfully applied to solve the IK of 6-DOF [8] and 7-DOF [9] robots. In [10], a multi-

*Correspondence: Nyongbassey.bassey@fupre.edu.ng
Bassey Etim Nyong-Bassey* and Ayebatonye Marttyns Epemu are with the Department of Electrical/Electronic Engineering, Federal University of Petroleum Resources Effurun, Nigeria,

International Journal on Robotics, Automation and Sciences (2022) 4:13-22
https://doi.org/10.33093/ijoras.2022.4.3
Manuscript received: 24 April 2022 | Accepted: 16 June 2022 | Published: 8 July 2022

Published by MMU PRESS. URL: http://journals.mmupress.com/ijoras

objective particle swarm optimization algorithm with full parameters was presented to minimize manipulator position error, joint angle change, and attitude error. This transforms the manipulator's inverse kinematics resolution into a multi-objective optimization problem. In [11, 12], the inverse kinematics solution method based on improved particle swarm optimization for the 6-DOF robotic arm manipulator was presented and compared with a closed numerical solution. The results showed the improved particle swarm optimization method to be better than the numerical solution. In [13] multi-objective whale optimization was used to design an optimal trajectory for a humanoid robot tasked with carrying objects in an inclined plane.

Aside from the metaheuristic approach in [8,9] several alternative ways of achieving an inverse kinematic solution in robotics have been proposed in literature. One of these methods is based on the combination of the damping least square method of Jacobian matrix singular value decomposition and inverted Gaussian distribution presented in [14]. By avoiding unique points, the technique was employed to ensure the continuity of joint velocity. Another inverse kinematic algorithm for 7-DOF redundant manipulators with obstacles avoidance and singularities avoidance, developed on a hybrid of an analytical and numerical method was described in [15]. The algorithm transformed the inverse kinematics task into the optimization model of elbow orientation, while additional tasks were formulated as fitness evaluation functions. Also, in [16], a product of exponentials (POE) based analytical inverse kinematics-based method was used for the space station remote manipulator system. In addition to avoiding the Denavit-Hartenberg technique's singularity, the method also has better precision over the POE-based numerical solution.

Furthermore, deep artificial neural networks and deep learning for realising inverse kinematics resolution in robots were reported in [17, 18]. The authors in [18] showed that the deep neural network method could calculate the inverse kinematics of a 5-DOF robot with fewer input variables. A neural inverse kinematics method was also presented in [19]. The technique used the problem's hierarchical structure to identify valid joint angles based on the target position and the previous joint angle along the chain. In order to solve the inverse kinematics problem for robots with various numbers of degrees of freedom (DOF) (4, 5, 6 and 7), artificial neural networks (ANN) and adaptive neuro-fuzzy inference systems (ANFIS) were also employed in [20]. The performances of the methods were analysed in terms of precession and accuracy. A method based on reinforcement learning was applied to the continuous state and action model to find the inverse kinematics solution of a 5-DOF robotic arm [21]. The authors showed the technique can effectively resolve inverse kinematics.

The use of hybrid algorithms of reinforcement learning and metaheuristic methods such as Grey-wolf optimization to solve general optimization problems was emphasized in [22]. When the hybrid method was used to solve the inverse kinematics issue, it resulted in a better solution. A 6-DOF robotic manipulator's motion trajectories were generated via the use of the grey-wolf optimization [23]. The method determines certain parameters such as optimal trajectory with the least tracking error and joint increasing speed. Furthermore, the kinematic model and trajectory planning problem of an industrial robotic manipulator was based on hybrid optimization algorithms presented in [24]. As many as 18 metaheuristic algorithms such as particle swarm optimization, grey-wolf optimization, genetic algorithms etc., were evaluated with regard to forward, inverse kinematics and trajectory planning problems of an industrial robot. The performance of an improved grey-wolf optimizer as an effective tool in solving global optimization engineering problems in robotics was discussed in [25, 26].

The majority of the above-reviewed literature focused on undertaking the issue of inverse kinematics of robotic arm manipulator using metaheuristic algorithms such as particle swarm optimization, grey wolf optimization etc., or a hybrid of both metaheuristic algorithms in conjunction with deep learning, but rarely conducted a comprehensive and in-depth comparison of these meta-heuristic algorithms. Also, utilising deep neural network typically requires cumbersome planning data [27], while traditional closed-form IK solution is only possible for robots that have a simple structure [28].
This study presents the first application of the jellyfish optimiser (JFO) [29] for undertaking the inverse kinematics problem of a 6 DoF. Hence, extending the findings on the performance of the JFO for solving engineering problems.

Also, the study aims to comparatively evaluate the inverse kinematic solution of meta-heuristic algorithms such as the grey-wolf optimization (GWO), improved grey wolf optimiser (I-GWO), particle swarm optimization (PSO), jellyfish optimiser (JFO) and whale optimisation (WO) techniques. The performances of the algorithms have been evaluated via the end-effector pose error of a newly designed 6-DoF robot in the MATLAB simulation environment, for welding applications in the oil and gas industry.

II. METHODOLOGY

A. Robotic Arm Forward Kinematics

The six degrees of freedom robotic arm manipulator for welding oil and Gas pipelines used as a case study, consists of six (6) revolute joints J1 – J6 and five links L1-L6 from the Base to the End-effector (EE) as shown in Figure 1.

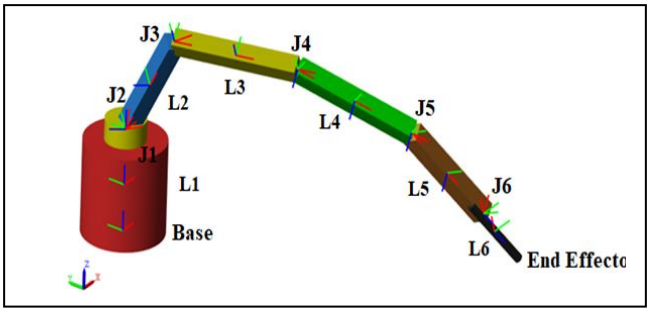


Figure 1. 6-DOF Robot arm for Oil and Gas Pipeline welding operations

The Denavit-Hartenberg (D-H) chart in Table I, presents the kinematics analysis of the robotic arm.

TABLE I. D-H PARAMETERS

Link	α_{n-1}(rad)	l_{n-1}(m)	θ_n(rad)	d_n(m)	Joint angle limit
T_1^0	$\alpha_0 = \pi/2$	$l_0 = 0$	θ_1	$d_1 = L1$	$-\pi \leq \theta_1 \leq \pi$
T_2^1	$\alpha_1 = 0$	$l_1 = 0$	θ_2	$d_2 = 0$	$-\pi/2 \leq \theta_2 \leq \pi/2$
T_3^2	$\alpha_2 = \pi/2$	$l_2 = 0$	θ_3	$d_3 = 0$	$-\pi/2 \leq \theta_3 \leq \pi/2$
T_4^3	$\alpha_3 = \pi/2$	$l_3 = 0$	θ_4	$d_4 = L4$	$-\pi/2 \leq \theta_4 \leq \pi/2$
T_5^4	$\alpha_4 = \pi/2$	$l_4 = 0$	θ_5	$d_5 = 0$	$-\pi/2 \leq \theta_5 \leq \pi/2$
T_6^5	$\alpha_5 = 0$	$l_5 = 0$	θ_6	$d_6 = L6$	$-3/4\pi \leq \theta_6 \leq 3/4\pi$

Generic presentation of Base to End-effector transformation matrix is as follows:

$$T_6^0 \equiv \begin{bmatrix} x_x & y_x & z_x & P_x \\ x_y & y_y & z_y & P_y \\ x_z & y_z & z_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The transformation matrix specific displacement of the EE to the Base reference frame of the robot in 3-dimensional Euclidean space $\{\boldsymbol{P_x}, \boldsymbol{P_y}, \boldsymbol{P_z}\}$ is given as follows:

$$\boldsymbol{P_x} = L6S\theta_3(C\theta_1 S\theta_2 + S\theta_1 C\theta_2) + L4S\theta_3(C\theta_1 S\theta_2 + S\theta_1 C\theta_2) \quad (2)$$

$$\boldsymbol{P_y} = -(L6C\theta_3 + L4C\theta_3 + L1) \quad (3)$$

$$\boldsymbol{P_z} = L6S\theta_4 C\theta_3(C\theta_1 S\theta_2 + S\theta_1 C\theta_2) - L4S\theta_3(C\theta_1 S\theta_2 + S\theta_1 C\theta_2) \quad (4)$$

B. Grey wolf Optimization

Recently, [30] introduced the grey-wolf optimizer (GWO) a novel swarm-based algorithm for solving Engineering problems. The GWO mimics the grey-wolf hunting behaviour and social hierarchy. To achieve optimality, the grey wolf employs three stages to hunt preys which include encircling, hunting and attacking with strict observance of social hierarchy within the wolf pack. Specifically, the social structure consists of four types of wolfs; the alpha (α), beta (β) delta (δ) and omega (ϖ) in order of importance with the downline wolf following the next superior wolf and so on. Thus, the alpha secures the wolf pack with the rest wolves in the pack following accordingly.

The pack's trajectory is governed in order of importance based on the leadership of the first, second and third best wolves; the alpha, the beta and the delta wolves which are respectively analogous to the three best optimal solutions, while the optimal solution with the least ranking are considered as the omega wolves [31], [32].

- Encircling

The encircling course is the first step deployed by the wolf pack in hunting prey and it is analogous to finding the optimisation search space. It is expressed mathematically as follows:

$$\vec{M} = |\vec{F}.\overrightarrow{X_p}(i) - \overrightarrow{X_w}(i)| \quad (5)$$

$$\overrightarrow{X_w}(i+1) = \overrightarrow{X_p}(t) - \vec{N} * \vec{M} \quad (6)$$

Where, i signifies the current iteration time step, $\vec{N}, \vec{M}$ and $\vec{F}$ are vectorized coefficients, $\overrightarrow{X_w}$ and $\overrightarrow{X_p}$ denote the wolf's and prey's vector positions respectively. The vector coefficients $\vec{M}$ and $\vec{N}$ are presented mathematically as follows:

$$\vec{M}=2\vec{a} * \overrightarrow{r_a} \quad (7)$$

$$\vec{N} = 2\overrightarrow{r_b} \quad (8)$$

Here, the magnitude of vector $\vec{a}$ linearly decreases from 2 down to 0 but has been improved with an exponential decaying term to speed up the learning process in the course of the simulation while, $\overrightarrow{r_a}$ and $\overrightarrow{r_b}$ are random vectors ϵ[0, 1] s.

- Hunting

In contrast to a practical hunting scenario, the optimal position of the prey (analogous to the global minimum or optimum) is unknown, therefore, rough estimates of the alpha, beta and the delta solutions are used as representative solutions. Furthermore, the alpha, the beta, and the delta wolfs' average positions are indicative of the prey's location as in (9). Thereafter,

the positions of the alpha, the beta and the delta wolf are updated towards the prey's location as in (10):

$$\overrightarrow{X_p}(i+1) = (\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3})/3 \quad (9)$$

Where,

$$\begin{cases} \overrightarrow{X_1} = \overrightarrow{X_\alpha} - \vec{N}_1 * (\vec{M}_\alpha) \\ \overrightarrow{X_2} = \overrightarrow{X_\beta} - \vec{N}_1 * (\vec{M}_\beta) \\ \overrightarrow{X_2} = \overrightarrow{X_\delta} - \vec{N}_1 * (\vec{M}_\delta) \end{cases} \text{ and } \begin{cases} \vec{M}_\alpha = |\vec{V}_\alpha \overrightarrow{X_\alpha} - \overrightarrow{X_w}| \\ \vec{M}_\beta = |\vec{V}_\beta \overrightarrow{X_\beta} - \overrightarrow{X_w}| \\ \vec{M}_\delta = |\vec{V}_\delta \overrightarrow{X_\delta} - \overrightarrow{X_w}| \end{cases} \quad (10)$$

- Attacking

In reality, the wolf attacks the prey once it is close enough. Similarly, in GWO, the grey wolf decreasing its movement as it inches towards the prey is analogous to vector $\vec{a}$, the learning rate being annealed from 2 to down to 0 as the optimal solution (prey) is approached.

C. Improved Grey wolf Optimisation (I-GWO)

The improved grey wolf optimiser (I-GWO) was proposed to enhance the standard GWO performance by updating and eliminating R wolfs from the wolf pack using the fitness score and replacing the eliminated wolfs with an equal number of randomly generated wolfs. Furthermore, the I-GWO which mimics the survival of the fittest biological evolution principle has fewer algorithmic parameters than GWO and is also easier to implement [33].

D. Particle Swarm Optimization (PSO)

Particle swarm optimisation (PSO) is a meta-heuristic optimizer whereby particles (individuals) indicate a lush feeding site (optimal solution) in an exploration space [34]. To activate the PSO, random population particles have random velocity and flight are lunched into the problem search space. Every particle modifies its flight trajectory p_b using self-experience as well as experience from adjacent particles as well as tracking the optimal flight trajectory g_b of the particle with the best fitness score [34].

The particles position and velocity modification are thus achieved as thus:

$$v_{i,u}^{(n+1)} = K * [v_{i,u}^{(n)} + c_1 r_1 * \left(P_{b_i} - p_{i,u}^{(n)}\right) + c_2 r_2 * (g_{b_i} - p_{i,u}^{(n)})] \quad (11)$$

$$p_{i,u}^{(n+1)} = p_{i,u}^{(n)} + v_{i,u}^{(n+1)} \quad (12)$$

Here, $v_{i,u}^{(n+1)}$ is particle *i's* velocity of in g- dimension in iteration $n+1$ and $p_{i,u}^{(n+1)}$ denotes the position of particle *i* in dimension g in iteration $n+1$, while $c_1$and c_2 respectively signifies cognitive and social acceleration constants.

The constriction factor Cf is presented mathematically as follows:

$$Cf = \frac{2}{\left|2 - Y\sqrt{Y^2 - 4Y}\right|} \quad (13)$$

Where, $Y = c_1 + c_2, Y > 4$

Consequently, the velocity range of a particle is $[-Vmax, Vmax], i = 1, 2, \ldots, k;$ is the quantity of swarm particles in $u = 1,2, \ldots, d$ problem dimensional space [34].

D. *Whale Optimisation Algorithm*

The whale optimisation algorithm (WOA) [35] is a meta-heuristic optimisation algorithm developed based on the principles of bubble-net foraging social conduct of humpback whales [36]. Three phases are involved in a whale bubble-net foraging; prey encircling process, bubble-net attack and hunting prey.

- Encircling prey

This involves the encircling of prey (solution) by the humpback whales (search agent) upon sighting. Nonetheless, since the optimal position towards which the whale should hunt is unknown ab-initio, the current best whale's position (candidate) is used while the other candidates update their position in the direction of the best whale's position. The mathematical presentation of the prey encircling process is as follows:

$$\vec{D} = |\vec{C} * \overrightarrow{\emptyset^*}(k) - \vec{\emptyset}(k)| \quad (14)$$

$$\vec{\emptyset}(k+1) = \overrightarrow{\emptyset^*}(k) - \vec{A} * \vec{D} \quad (15)$$

Where, k denotes the most recent iteration, $\vec{A}$ and $\vec{D}$ are vector factors, the optimal solution derived via the position vector with the current optimal solution is denoted as $\overrightarrow{\emptyset^*}(k)$ while the position vector is represented as $\vec{\emptyset}(k)$.

The vector factors $\vec{A}$ and $\vec{C}$ are evaluated as follows:

$$\vec{A} = 2\vec{b} * \overrightarrow{r_c} - \vec{b}) \quad (16)$$

$$\vec{C} = 2.\overrightarrow{r_d} \quad (17)$$

Where, $\overrightarrow{r_c}$ and $\overrightarrow{r_d}$ denotes random factors between 0 and 1, and $\vec{b}$ is reduced gradually from 2 to 0 during the iteration to guarantee on the one hand exploration and on the other hand exploitation.

- Bubble-net-attack

The constricted spiral movement the whale exhibits in the prey's route is represented mathematically as follows:

$$\vec{\emptyset}(k+1) = \overrightarrow{D'} * e^{\gamma l} * \cos(2\pi l) + \overrightarrow{\emptyset^*}(k) \quad (18)$$

$$\overrightarrow{D'} = |\overrightarrow{\emptyset^*}(k) - \vec{\emptyset}(\mathrm{k})| \quad (19)$$

Where, $\overrightarrow{D'}$ represents the variance between the n^{th}whale and the whale with the best position, γ is the constant factor governing the logarithmic spiral form or circular movement, while l denotes a random factor between -1 and 1.

- Hunting prey

Further, to the bubble-net foraging behaviour, the humpback whale can also engage in a random search for prey. Hence, the random search behaviour is expressed as follows:

$$\vec{D} = |\vec{C}\overrightarrow{\emptyset_r}(k) - \vec{\emptyset}(k)| \quad (20)$$

$$\vec{\emptyset}(k+1) = \overrightarrow{\emptyset_r}(k) - \vec{A} * \vec{D} \quad (21)$$

Where, $\overrightarrow{\emptyset_r}(k)$ is the position of a random agent.

Therefore, if |A| > 1 agents' position is updated based on a random agent's position, and by the position of the otherwise best solution if |A| < 1, to guarantee exploration and as well as exploitation respectively. Thereafter, the process is terminated only if the stopping criterion is satisfied.

E. *Jelly Fish Optimisation (JFO)*

More recently, Chou and Troung [29] proposed a meta-heuristic algorithm called the jellyfish optimizer (JFO) which is fundamentally based on the foraging behaviour of the jellyfish in its oceanic habitat [37]. The foraging mechanism is enabled primarily by two movement types; ocean current and swarm movements and a time-based switching operation presented as follows:

- Ocean Current Motion

This movement type involves the jellyfish drifting in the direction of the ocean current during the foraging activity. The magnitude by which the jellyfish drifts in search of an ideal search space is governed by the error difference in position between the individual jellyfish and the averaged positions of all the oceanic jellyfish. This magnitude of drift is presented mathematically as follows:

$$\begin{aligned}\overrightarrow{drift} &= 1/N \sum \overrightarrow{drift_k} \\ &= 1/N \sum (x_* - a_c x_k) \\ &= x_* - a_c \sum x_k/N) \\ &= x_* - a_c \mu \end{aligned} \quad (22)$$

Where, x_* represents the jellyfish which has the current-best position in the swarm; N is the overall quantity of jellyfishes in the swarm; a_c is an attraction factor; μ is the average position of all jellyfishes in the swarm.

Therefore,

, $Df = a_c \mu$ (23)

Where, Df denotes the average position of the jellyfish to the jellyfish with the current-best position.

More so, the JFO assumes that the jellyfish swarm contains jellyfishes which are distributed normally in all dimensions with a standard deviation of $\pm\beta\sigma$ around the average μ position which encompasses a certain likelihood of the entire jellyfish, thus:

$$Df = \beta\sigma.r1(0,1) \quad (24)$$

Set $\sigma = r2.(0,1)$ (25)

Where, r1 and r2 are random values generated between 0 and 1.

Consequently, the jellyfish's new position is obtained as follows:

$$X_i(t+1) = X_i(t) + r1(0,1).\overrightarrow{drift} \quad (26)$$

Furthermore, the jellyfish's new position is then updated as follows:

$$X_i(t+1) = X_i(t) + r1(0,1).(x_* - \beta\sigma.r1(0,1)) \quad (27)$$

- Jellyfish Swarm Motion

The jellyfish swarm movement is categorized as; active (type A) or passive (type B). The active motion is predominant during the swarm's initial formative period. Nonetheless, in time the jellyfish displays the passive motion by updating its position towards a random jellyfish in the swarm with better direction (evaluated via a fitness score) to a foraging location.

The type A motion is presented as follows:

$$X_i(t+1) = X_i(t) + \gamma.r1(0,1).(\mathrm{UB} - \mathrm{LB}) \quad (28)$$

Where γ is a motion factor regarding the distance about the jellyfish's location, and UB and LB represent the upper and lower bounds of the search environment.

Similarly, type B motion is denoted mathematically as follows:

$$\overrightarrow{Step} = X_i(t+1) - X_i(t) \quad (29)$$

And,

$$\overrightarrow{Step} = r1(0,1).\overrightarrow{direction} \quad (30)$$

$$\overrightarrow{direction} = \begin{cases} X_j(t) - X_i(t) & if\ f(X_i(t)) \geq f(X_j(t)) \\ X_i(t) - X_j(t) & if\ f(X_i(t)) < f(X_j(t)) \end{cases} \quad (31)$$

Where, j is the index of the j^{th} jellyfish in the swarm and f is a fitness function for evaluating the position X.

Also, the next position the jellyfish attains is updated as follows:

$$X_i(t+1) = X_i(t) + \overrightarrow{Step} \quad (32)$$

- Time Control Mechanism

The time control mechanism is responsible for switching between both Type A and B motions for exploration and exploitation of the ocean current in course of searching for nutritious food (optimal solution). The time control mechanism "$c(t)$" is achieved using a constant factor and a time-based randomly controlled function r with values between 0 and 1, and presented mathematically as follows:

$$c(t) = \left(1 - \frac{t}{t_{max}}\right).(2.r2 - 1) \quad (33)$$

Where, t, t_{max} indicate the defined iteration and maximum iteration time respectively, $r2$ is a randomised factor between 0 and 1.

During foraging both type A and B are performed by the jellyfish inside the swarm if r2 (0, 1) > (1 −c (t)), and if r2(0, 1) < (1 −c (t)) respectively. Initially, type A motion is selected more frequently mimicking exploration, but as time evolves the factor, (1 −c (t)) increases and the chance of (1 −c (t)) being greater than r2(0,1). Hence, type B motion will be selected more frequently than type A to mimic exploitation.

III. Results and Discussion

The meta-heuristic algorithmic parameters are presented in Table II. Specifically, the maximum iteration and population of search agents were selected modestly as 50 and 10 respectively for uniformity and consistency [34]. The meta-heuristic algorithms were utilised in determining the optimal joint angles which enables the end effector accurately trace the rectangular trajectory via the six sets of waypoints shown in Table III. The mean squared error (MSE) fitness score is used statistically to determine the accuracy of the effector's desired reference position and its actual position controlled by the meta-heuristic algorithms as follows:

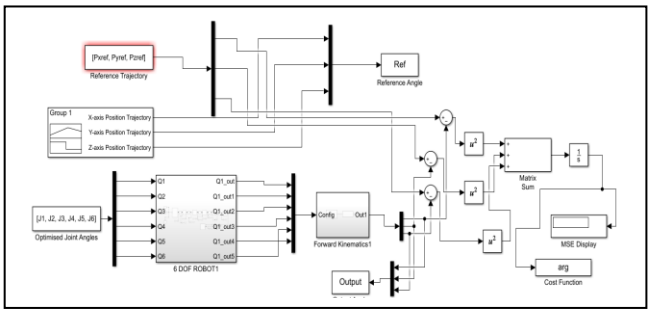
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{P}_{(i)ref} - \boldsymbol{P}_{(i)})^2 \qquad ,\mathrm{i} \in [x\ y\ z] \quad (34)$$

Where, $P_{(i)ref}$ and $P_{(i)}$ respectively denotes to the reference and actual coordinate positions of the end effector.

TABLE II. Meta-Heuristic Algorithmic Parameters

Parameters	Meta-Heuristic Algorithms				
	GWO	PSO	I-GWO	WO	JFO
Number of Agents, N	50	50	50	50	50
Convergence iteration	10	10	10	10	10
learning rate $\vec{a}$ [35]	2	-	2	-	-
Weights, w [38]	-	0.5	-	-	-
Acceleration, c1 [38]	-	1.4	-	-	-
Velocity c2 [38]	-	1.4	-	-	-

The MATLAB/Simscape/Simulink model for Joint angle optimization solution to the inverse kinematics robotic manipulator arm problem is shown in Figure 2.



. Figure 2. Robotic Arm Inverse Kinematics Architecture for optimised Joint angles

TABLE III. Waypoint for the Reference Rectangular Trajectory

Waypoint	Reference Pose Trajectory (m)		
	P_{xref}	P_{yref}	P_{zref}
1	0.25	0.01	0.11
2	0.25	0.01	0.11
3	0.25	0.11	0.11
4	0.15	0.11	0.11
5	0.15	0.01	0.11
6	0.25	0.01	0.11

The optimized joint angles resulting from GWO, PSO, I-GWO, WO and JFO are shown in Figures 3, 4 – 7 respectively. Furthermore, Tables IV, V - VIII present the summary of the results for the End Effector pose, Joint angles, and position error for the GWO, PSO, IGWO, WO and JFO methods respectively.

From Figures 4, 5 - 8, it is clear that the metaheuristic algorithms all constrained the joint angles within the pre-defined limits. Furthermore, the GWO in contrast to the rest algorithms has the least fitness score, which indicates better accuracy with respect to position error between the rectangular trajectory waypoints and the actual end effector's pose.

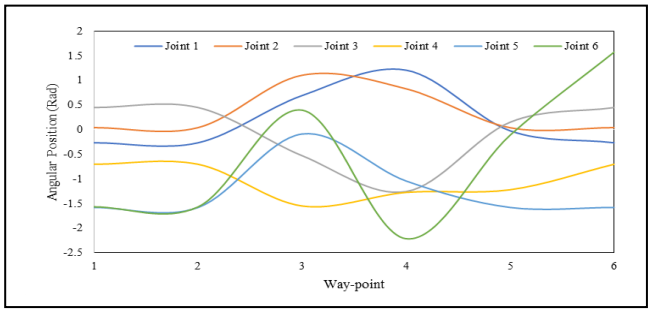


Figure 3: GWO optimised robotic joint angles solution for the inverse kinematics

E-ISSN: 2682-860X

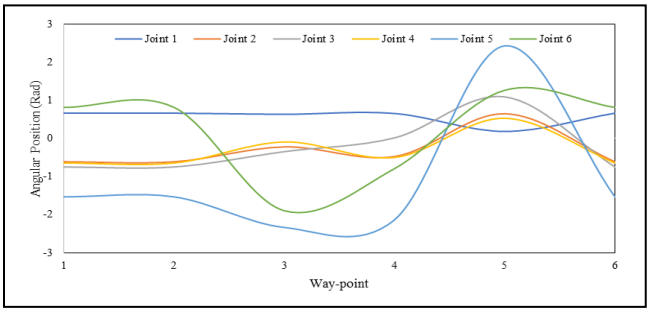


Figure 4: PSO optimised robotic joint angles solution for the inverse kinematics

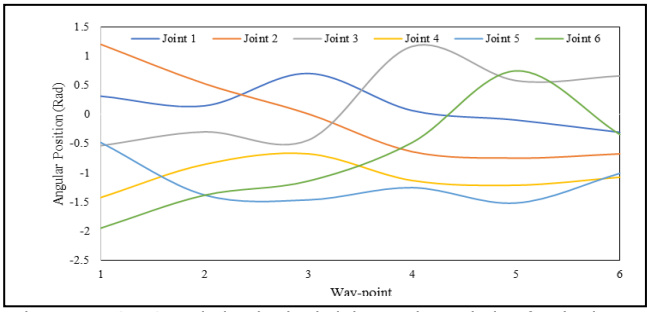


Figure 5: I-GWO optimised robotic joint angles solution for the inverse kinematics

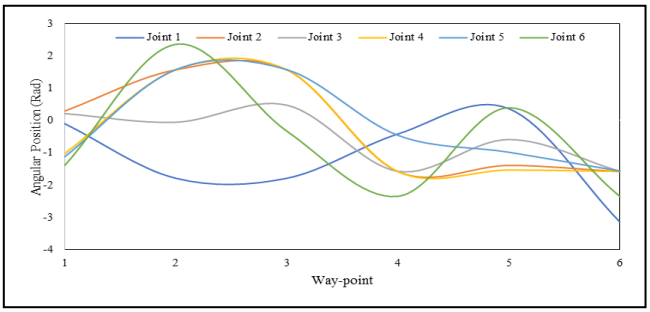


Figure 6: WO optimised robotic joint angles solution for the inverse kinematics

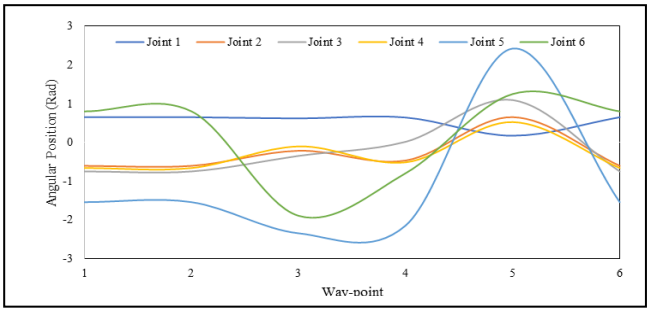


Figure 7: JFO optimised robotic joint angles solution for the inverse kinematics

TABLE IV. GWO OPTIMIZED JOINT ANGLES WITH END-EFFECTOR POSE PERFORMANCE

Waypoint	Robot output End-Effector Pose (m)			Optimized input Joint Angles (Rads)						MSE
	P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	
1	0.2507	0.0110	0.1063	-0.2629	0.0476	0.4568	-0.6999	-1.5708	-1.5655	5.0e-06
2	0.2507	0.0110	0.1063	-0.2629	0.0476	0.4568	-0.6999	-1.5708	-1.5655	5.0e-06
3	0.2505	0.1110	0.1084	0.6918	1.1125	-0.5200	-1.5385	-0.0918	0.3931	1.3e-06
4	0.1503	0.1095	0.1115	1.2061	0.8376	-1.2490	-1.2661	-1.0387	-2.2125	8.4e-07
5	0.1480	0.0080	0.0937	-0.0205	0.0466	0.1592	-1.2125	-1.5708	-0.1159	6.5e-05
6	0.2507	0.0110	0.1063	-0.2629	0.0476	0.4568	-0.6999	-1.5708	1.5655	5.1e-06

TABLE V. PSO OPTIMIZED JOINT ANGLES WITH END-EFFECTOR POSE PERFORMANCE

Waypoint	Robot output End-Effector Pose (m)			Optimized input Joint Angles (Rads)						MSE
	P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	
1	0.2402	0.0355	0.0275	0.6618	-0.6046	-0.7480	-0.6508	-1.5375	0.8059	2.5e-3
2	0.2402	0.0355	0.0275	0.6618	-0.6046	-0.7480	-0.6508	-1.5375	0.8059	2.5e-3
3	0.2347	0.0593	0.1953	0.6323	-0.2143	-0.3468	-0.0959	-2.3409	-1.8909	3.4e-03
4	0.1954	0.0960	0.1631	0.6535	-0.4663	0.0144	-0.5050	-2.1388	-0.7890	1.7e-03
5	0.2125	0.0552	0.0624	0.1841	0.6549	1.0880	0.5284	2.4293	1.2547	2.7e-03
6	0.2402	0.0355	0.0275	0.6618	-0.6046	-0.7480	-0.6508	-1.5375	0.8059	2.5e-03

TABLE VI. I-GWO OPTIMIZED JOINT ANGLES WITH END-EFFECTOR POSE PERFORMANCE

Waypoint	Robot output End-Effector Pose (m)			Optimized input Joint Angles (Rads)						MSE
	P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	
1	0.2491	0.0101	0.1085	0.3159	1.2096	-0.5296	-1.4234	-0.4857	-1.9495	9.8e-07
2	0.2501	0.0111	0.1102	0.1509	0.5347	-0.2931	-0.8518	-1.3822	-1.3838	4.3e-07
3	0.2487	0.1100	0.1094	0.7060	0.0160	-0.4342	-0.6715	-1.4661	-1.1421	6.7e-07
4	0.1499	0.1096	0.1095	0.0690	-0.6311	1.1690	-1.1304	-1.2562	-0.4820	1.6e-07
5	0.1504	0.0100	0.1102	-0.0936	-0.7404	0.5840	-1.2128	-1.5195	0.7478	7.7e-07
6	0.2496	0.0090	0.1096	-0.3049	-0.6685	0.6643	-1.0735	-1.0133	-0.3422	4.2e-07

TABLE VII. WO OPTIMIZED JOINT ANGLES WITH END-EFFECTOR POSE PERFORMANCE

Waypoint	Robot output End-Effector Pose (m)			Optimized input Joint Angles (Rads)						MSE
	P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	
1	0.2491	0.0048	0.0588	-0.1020	0.2997	0.2110	-1.0147	-1.1253	-1.3937	8.8e-04
2	0.2890	0.0116	0.1231	-1.7919	1.5708	-0.0560	1.5708	1.5708	2.3562	5.6e-04
3	0.2890	0.0114	0.1236	-1.7922	1.5708	0.4702	1.5708	1.5708	-0.3329	3.8e-03
4	0.1547	0.0368	0.0039	-0.4222	-1.5708	-1.5708	-1.5708	-0.4627	-2.3562	5.6e-03
5	0.1157	0.0843	0.0579	0.3623	-1.3813	-0.5904	-1.5228	-0.9866	0.3901	3.1e-03
6	-0.0740	0.0201	0.0980	-3.1416	-1.5708	-1.5708	-1.5708	-1.5708	-2.3562	3.5e-02

TABLE VIII. JFO OPTIMIZED JOINT ANGLES WITH END-EFFECTOR POSE PERFORMANCE

Waypoint	Robot output End-Effector Pose (m)			Optimized input Joint Angles (Rads)						MSE
	P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	
1	0.2266	0.0443	0.1585	0.1771	1.4394	-0.6167	-0.8487	-1.4403	-1.4829	1.4e-03
2	0.1987	-0.1457	0.2313	-0.5448	1.5045	-0.9748	-0.8154	-1.1328	2.1743	14e-03
3	0.3891	0.0280	-0.0303	-1.7579	1.3302	1.4655	0.3363	1.5676	0.8013	15e-03
4	0.4353	0.1017	0.3228	0.3662	0.2428	-0.3079	-0.6212	0.5280	0.6917	42e-03
5	-0.0056	0.0978	-0.0602	1.6378	-1.2959	-1.4735	-0.8724	-1.4269	-2.2491	20e-03
6	0.0538	0.0765	0.0980	1.1184	-0.6143	-0.7021	-1.4692	-1.5613	-2.2210	14e-03

The 2D performance comparison of the GWO with the PSO optimization methods with regards to the accuracy of the position error is shown in Figures 8(a) and 8(b) while Figures 6(a) and 6(b) show the 3D representations respectively. From Figure 5(b), a mismatch can be observed in 3D GWO optimized plot. However, by isolating the Z-axis as in the 2D plot in Figure 8(a) reveals that the cause of the error is due to position error in the Z-axis which causes the depth mismatch. Nevertheless, this type of error will be addressed in future work using a suitable controller.

Furthermore, the trajectory trace of the end effector optimized by PSO shows greater distortion than the GWO counterpart. From Figures 9(a) and 9(b), both the 2D and 3D plots respectively both show an unacceptable mismatch between the reference trajectory and that of the end effector, indicating the impact of the position error and the ineffectiveness of PSO to obtain optimized joint angles for this particular Robot.

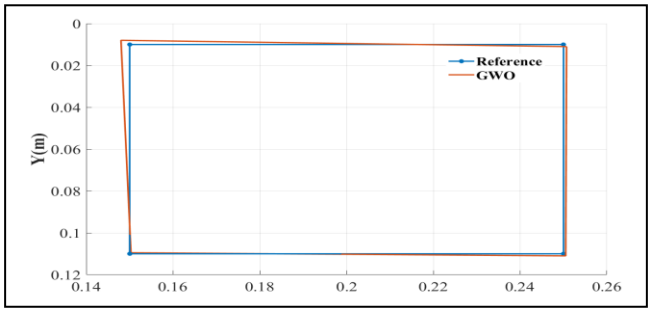


Fig 8(a): Performance of GWO IK solution in 2D

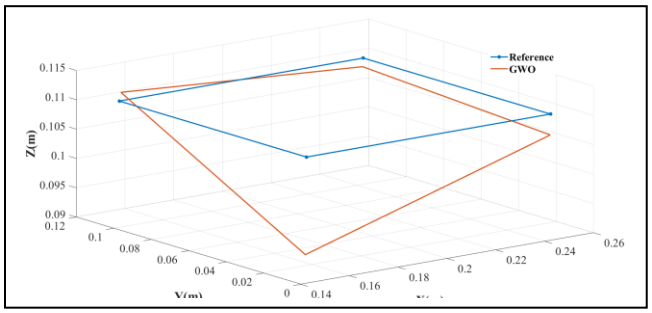


Fig 8(b): Performance of GWO IK solution in 3D

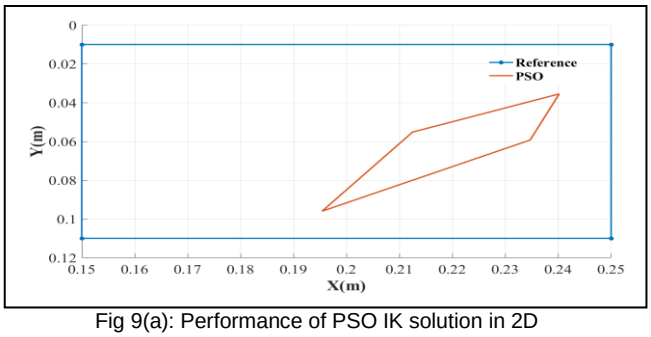


Fig 9(a): Performance of PSO IK solution in 2D

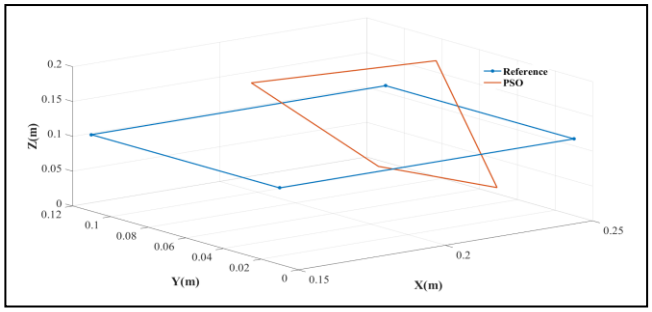


Fig 9(b): Performance of PSO IK solution in 3D

The robot's trajectory trace in 2D and 3D derived using the I-GWO are shown in Figures 10 (a) and 10 (b) respectively. From Figures 10 (a) and 10 (b) plot the I-GWO was only accurate in the 2D plane.

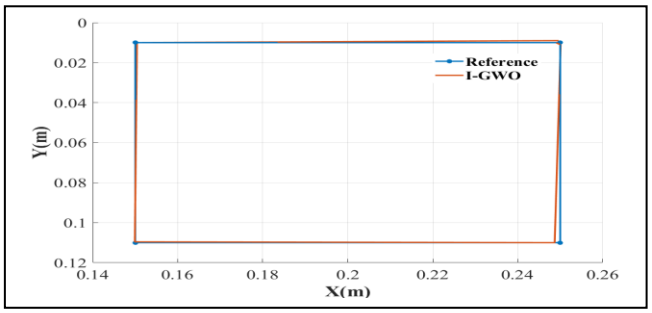


Fig 10(a): Performance of I-GWO IK solution in 3D

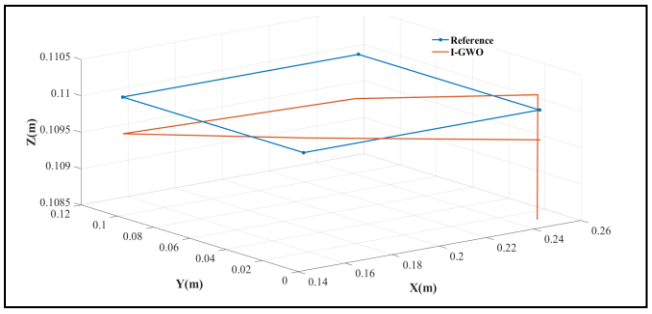


Fig 10(b): Performance of I-GWO IK solution in 3D

Figures 11 (a) and 11 (b) show the 2 D and 3 D plots resulting from the WO method. The WO was not able to

trace the trajectory as shown in both the 2D and 3D plots. This is as a result of the error not converging to a negligible value (i.e., less than 1e-06).as in the case of the GWO and I-GWO.

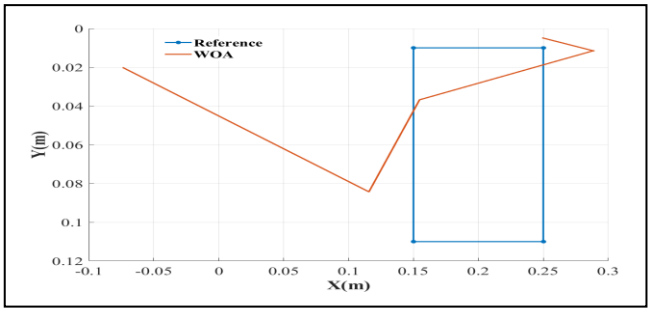


Fig 11(a): Performance of WO IK solution in 3D

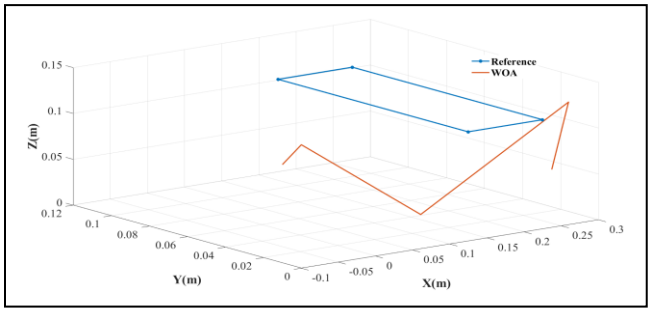


Fig 11(b): Performance of WO IK solution in 3D

Furthermore, the JFO trajectory traces in both 2D and 3 D are shown in Figures 12(a) and 12(b) respectively. The accuracy was the least among the meta-heuristic algorithms which have been compared.

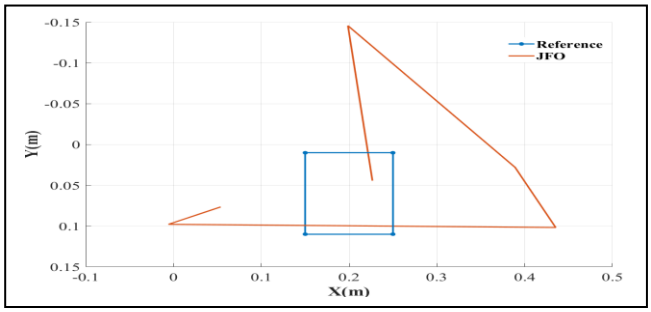


Fig 12(a): Performance of JFO IK solution in 3D

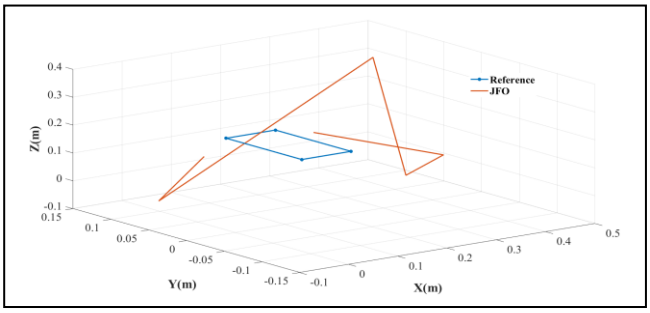


Fig 12(b): Performance of JFO IK solution in 3D

IV. Conclusion

The inverse kinematic performances of GWO PSO, I-GWO, WO and JFO was evaluated using the end effector pose error of a newly designed robot in the MATLAB simulation environment. The GWO, PSO, I-GWO, WO and JFO were recursively used to optimize the robotic arm's joint angles for tracing a rectangular trajectory. From the analysis of the results in 2D, the GWO had the lowest fitness score which was closely followed by the I-GWO for matching the rectangular trajectory. Nevertheless, in 3D the GWO and I-GWO showed errors in the Z-axis. The PSO, WO and JFO all had unacceptable trajectories which mismatched the rectangular trajectory Thus, from the investigation, only the GWO and I-GWO effectively solved the inverse kinematics problem of the newly designed robotic arm manipulator with 6 degrees-of-freedom for welding operations.

Future work will aim to practically implement the robot and also improve accuracy regarding the reference waypoints tracking using an innovative meta-heuristic algorithm.

References

[1] M.A.N. Huda, S.H Susilo, and P.M. Adhi, "Implementation of Inverse Kinematic and Trajectory Planning on 6-DOF Robotic Arm for Straight-Flat Welding Movement", Logic: Jurnal Rancang Bangun dan Teknologi, vol. *22, no.* 1, pp.51-61, 2022.

[2] M. Saraf, A. Agarwal, A. Chaudhary, and A. Ganthale, "Kinematic Modelling and Motion Mapping of Robotic Arms," Journal of Physics: Conference Series, vol. 1969, no. 1, 2021.

[3] M. T. Nguyen, C. Yuan, and J. H. Huang, "Kinematic Analysis of A 6-DOF Robotic Arm," IFToMM World Congress on Mechanism and Machine Science, vol. 73, no. 100, pp. 2965–2974, 2019.

[4] Š. Ondočko, T. Stejskal, J. Svetlík, L. Hrivniak, M. Šašala, and A. Žilinský, "Position Forward Kinematics of 6-DOF Robotic Arm," Acta Mech. Slovaca, vol. 24, no. 2, pp. 30–36, 2020.

[5] Š. Ondočko et al., "Inverse kinematics data adaptation to non-standard modular robotic arm consisting of unique rotational modules," Applied Sciences, vol. 11, no. 3, pp. 1–15, 2021.

[6] A. R. Al Tahtawi, M. Agni, and T. D. Hendrawati, "Small-scale robot arm design with pick and place mission based on inverse kinematics," Journal of Robotics and Control (JRC), vol. 2, no. 6, pp. 469–475, 2021.

[7] A. Saadah, "Computing the Kinematics Study of a 6 DOF Industrial Manipulator Prototype by Matlab," Recent Innovations in Mechatronics, vol. 7, no. 1., 2021.

[8] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm," Artificial Intelligence Review, vol. 53, no. 2, pp. 949–964, 2020.

[9] H. Khan, H. H. Kim, S. J. Abbasi, and M. C. Lee, "Real-time inverse kinematics using dual particle swarm optimization DPSO of 6-DOF robot for nuclear plant dismantling," IFAC-PapersOnLine, vol. 53, no. 2, pp. 9885–9890, 2020.

[10] S. Luo, D. Chu, Q. Li, and Y. He, "Inverse Kinematics Solution of 6-DOF Manipulator Based on Multi-Objective Full-Parameter Optimization PSO Algorithm," Frontiers in Neurorobotics, vol. 16, pp. 1–12, 2022.

[11] [1 L. I. U. Yiyang, X. I. Jiali, B. A. I. Hongfei, and W. Zhining, "A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm," IEEE Access, vol. 9, pp. 32341–32350, 2021.

[12] A. Jiping, L. Xinhong, Z. Zhang, W. Man, G. Zhang, and W. Ding, "Application of An Improved Particle Swarm Optimization Algorithm in Inverse Kinematics Solutions of Manipulators," In IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), pp. 1680–1684, 2020.

[13] K. Sanprasit, and P. Artrit, 2020. Multi-Objective Whale Optimization Algorithm for Balance Recovery of a Humanoid Robot. International Journal of Mechanical Engineering and Robotics Research, 9(6), pp.882-893, 2020.

[14] X. Li, X. Zhang, H. Li, W. Yuan, and Y. Qiu, "Singularity processing algorithm for inverse kinematics of 6-DOF series robot," In IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), vol. 2020, pp. 696–701, 2020.

[15] M. Jin, Q. Liu, B. Wang, and H. Liu, "An Efficient and accurate Inverse Kinematics for 7-DOF Redundant Manipulators Based on a Hybrid of analytical and Numerical Method," IEEE Access, vol. 8, pp. 16316–16330, 2020.

[16] Y. Wang, X. Ding, Z. Tang, C. Hu, Q. Wei, and K. Xu, "Research Article A Novel Analytical Inverse Kinematics Method for SSRMS-Type Space Manipulators Based on the POE Formula and the Paden- Kahan Subproblem," International Journal of Aerospace Engineering, 2021.

[17] N. A. Mohamed, A. T. Azar, N. E. Abbas, M. A. Ezzeldin, and H. H. Ammar, "Experimental Kinematic Modeling of 6-DOF Serial Manipulator Using Hybrid Deep Learning," in Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020), pp. 283–295, 2020.

[18] S. K. Shah, R. Mishra, and L. S. Ray, "Solution and validation of inverse kinematics using Deep artificial neural network," Materials Today: Proceedings, vol. 26, pp. 1250–1254, 2020.

[19] R. Bensadoun, S. Gur, N. Blau, T. Shenkar, and L. Wolf, "Neural Inverse Kinematics," arXiv preprint, 2022.

[20] J. Demby, Y. Gao, and G. N. Desouza, "A Study on Solving the Inverse Kinematics of Serial Robots using Artificial Neural Network and Fuzzy Neural Network," IEEE International Conference on Fuzzy Systems, pp. 1–6, 2020.

[21] X. Shi, Z. Guo, J. Huang, Y. Shen, and L. Xia, 2020, "A distributed reward algorithm for inverse kinematics of arm robot," In 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE), pp. 92-96, 2020.

[22] A. Seyyedabbasi, R. Aliyev, F. Kiani, M. U. Gulle, H. Basyildiz, and M. A. Shah, "Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems," Knowledge-Based Systems, vol. 223, p. 107044, 2021.

[23] C. Choubey and J. Ohri, "Optimal Trajectory Generation for a 6-DOF Parallel Manipulator Using Grey Wolf Optimization Algorithm," Robotica, vol. 39, no. 3, pp. 411-427, 2021.

[24] G. Singh and V.K Banga, "Kinematics and Trajectory Planning Analysis Based on Hybrid Optimization Algorithms for an Industrial Robotic Manipulators," 2022.

[25] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems," Expert Systems with Applications, vol. 166, p. 113917, 2021.

[26] S. Dereli, "A new modified grey wolf optimization algorithm proposal for a fundamental engineering problem in robotics," Neural Computing and Applications, vol. 33, no. 21, pp. 14119-14131, 2021.

[27] B. E. Nyong-Bassey, D. Giaouris, C. Patsios, S. Papadopoulou, Papadopoulos, A. I., S. Walker, and S. Gadoue, "Reinforcement learning based adaptive power pinch analysis for energy management of stand-alone hybrid energy storage systems considering uncertainty", Energy, 193, 116622. 2020.

[28] C. Lopez-Franco, D. Diaz, J. Hernandez-Barragan, N., Arana-Daniel, and M. Lopez-Franco, "A Metaheuristic Optimization Approach for Trajectory Tracking of Robot Manipulators", Mathematics, 10(7), 1051, 2022.

[29] J. S. Chou and D. N. Truong, D.N, "A novel metaheuristic optimizer inspired by behaviour of jellyfish in ocean", Applied Mathematics and Computation, 389, p.125535, 2021.

[30] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer", Advances in engineering software, vol. 69, pp. 46-61, 2014.

[31] H. Kraiem, F. Aymen, L. Yahya, A. Triviño, M. Alharthi, and S. S. Ghoneim, "A comparison between particle swarm and grey wolf optimization algorithms for improving the battery autonomy in a photovoltaic system", Applied Sciences, vol. 11, no. 16, pp. 7732, 2021.

[32] Y. S. Kushwah, and R. Shrivastava, "Particle Swarm Optimization (PSO) Inspired Grey Wolf Optimization (GWO) Algorithm". Int. J. Math. Trends Technol., vol. 58, pp. 81-91, 2018.

[33] J. S. Wang, J. S. X. Li, "An improved grey wolf optimizer based on differential evolution and elimination mechanism. Scientific reports", 9(1), 1-21, 2019.

[34] B. E. Nyong-Bassey, and B. Akinloye, "Comparative study of optimized artificial intelligence based first order sliding mode controllers for position control of a DC motor actuator". Journal of Automation, Mobile Robotics and Intelligent Systems, pp. 58-71, 2014.

[35] Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing, 22(1), 1-15.

[36] Ning, G. Y., & Cao, D. Q. (2021). Improved whale optimization algorithm for solving constrained optimization problems. Discrete Dynamics in Nature and Society, 2021.

[37] M. Abdel-Basset, R. Mohamed, R.K. Chakrabortty, M.J. Ryan, and A. El-Fergany, "An improved artificial jellyfish search optimizer for parameter identification of photovoltaic models", Energies,14(7), 2021, p.1867.

[38] T. Zhang, W. Xin and W. Zhenlei, "A Novel Improved Grey Wolf Optimization Algorithm for Numerical Optimization and PID Controller Design", 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS). IEEE, 2018.