*International Journal of Advanced Technology and Engineering Exploration, Vol 9(97)*
*ISSN (Print): 2394-5443 ISSN (Online): 2394-7454*
*http://dx.doi.org/10.19101/IJATEE.2021.876125*

Research Article

# Inverse kinematics solutions of a newly designed three-link robotic manipulator for the casting process using the ant lion optimizer

**Mahendra Kumar Jangid[1*], Sunil Kumar[2] and Jagtar Singh[3]**
Research Scholar, Department of Mechanical Engineering, Sant Longowal Institute of Engineering & Technology, Longowal, Punjab, India[1]
Assistant Professor, Department of Mechanical Engineering, Sant Longowal Institute of Engineering & Technology, Longowal, Punjab, India[2]
Professor, Department of Mechanical Engineering, Sant Longowal Institute of Engineering & Technology, Longowal, Punjab, India[3]

Received: 07-June-2022; Revised: 14-December-2022; Accepted: 17-December-2022


## Abstract
***Conventional methods make determining inverse kinematics solutions of a multi-degree of freedom robot difficult. In recent years, soft computing methods have been used, and they are very easy to compute the solutions of inverse kinematics. In this study, the ant lion optimizer (ALO) was used to solve the inverse kinematics of a three-link robotic manipulator, and the results have been compared to other optimization algorithms such as particle swarm optimization (PSO), grey wolf optimization (GWO), and the sine-cosine algorithm (SCA). In the beginning, Denavit-Hartenberg (D-H) parameters for the robotic manipulator are constructed, as well as transformation matrices. The entire transformation matrix is then used to find the end-effector position equations. The ALO, PSO, GWO, and SCA algorithms are used to predict the end-effector location of this robotic manipulator in the workspace. The location error (difference between the real and goal positions) is estimated using a fitness function. The fitness function was used to find the inverse kinematic solutions by reducing the position error. These algorithms were put to the test in this study using two separate scenarios. Position error and solution time for a single point in the workspace were calculated in case I, while position error and solution time for twenty randomly selected locations in the workspace were estimated in case II. After computation, the ALO, PSO, GWO, and SCA give position errors of 6.557 e-06, 0.00835, 0.006881, and 0.00993 meters respectively, for case I. The solution times for the ALO, PSO, GWO, and SCA are 0.88, 14.34, 2.01, and 1.31 seconds, respectively, for case I. Similarly, better results were found for case II as compared to case I in terms of position error and solution time. By comparing case-II to case-I, case-II confirms the quality of the ALO when compared to other optimization algorithms (PSO, GWO, and SCA). In terms of position error and solution time, the ALO algorithm performs significantly better than the PSO, GWO, and SCA algorithms.***

## Keywords
***ALO, Inverse kinematics, Three-link robotic manipulator, PSO, GWO, SCA, Optimization algorithms.***

## 1.Introduction
In a fourth industrial revolution the demands of the robotic manipulators are increasing because it reduces lead time and enhances the production. The non-linear dynamic behaviour of the robotic manipulator makes controlling it a tough undertaking. The findings of the forward kinematics solution of a robotic manipulator are simple, but the inverse kinematics solution, which involves calculating joint angles for the desired position or cartesian space to joint space, is difficult.

If the number of degrees of freedom (DOF) of the robotic manipulator increases findings of inverse kinematics solution getting difficult and cannot be calculated using traditional methods like geometric [1, 2], iterative [3], or algebraic methods [4].

The inverse kinematics solution becomes more sophisticated as the number of the DOF of the robotic manipulator grows [5]. Many soft computing approaches, such as nature-inspired meta-heuristic optimization algorithms like particle swarm optimization (PSO) [6, 7], improved PSO [8], adapted PSO [9] and inertia weight-particle swarm

*Author for correspondence

optimization (IW-PSO) [10] have recently been used to find inverse kinematics solutions for robotic manipulators and [11], other soft computing techniques like- firefly algorithm (FA) [12], artificial bee colony (ABC) [13, 14], Quantum PSO [15, 16], neuro-genetic simulated annealing [17], neuro simulated annealing [18], artificial neural network (ANN) [19–21] and others in [22–26]. All these algorithms have different solution time and produce different results because of their different iterative nature, but in this study a new nature-inspired meta-heuristic algorithm is used to find the best solutions.
The primary goals of this research are to develop an inverse kinematics solution for a newly designed three-link robotic manipulator used for casting purpose using a soft computing method called ant lion optimizer (ALO). It is also compared to other soft computing approaches, that are discussed in the past studies.

The objectives are listed below in order of importance:

1. To find the end effector position by forward kinematics of three-link robotic manipulator using Denavit-Hartenberg (D-H) parameters.
2. To find inverse kinematics of three-link robotic manipulator solution using the ALO.
3. To make comparative analysis of the inverse kinematics solution with other techniques- PSO, grey wolf optimization (GWO) and sine cosine algorithm (SCA).
4. To make comparative analysis of position error and solution time using two different cases/scenarios.

To achieve these objectives first position of end effector was obtained using forward kinematics equations. The position error between desired and actual point was minimized using the ALO, PSO, GWO and SCA algorithms corresponding position, joint angles have been obtained. The position error and solution time of all these algorithms have been compared with two different cases. Case-I: for a single of end-effector position point in robotic manipulator workspace and Case-II: for 20 randomly selected end-effector position points in robotic manipulator workspace.

The following sections make up this article: section 1 includes the introduction of the work, section 2 includes literature review, and section 3 includes methods. The kinematic modelling and working procedure is covered in section 3, and the working mechanism of the ALO is explained in section 3.2. The fitness function is explained in Section 3.3. Sections 4 include simulation results and section 5 covers discussion and limitations. At the last conclusion and future work covered in section 6.

## 2.Literature review

The followings are some brief explanations of studies founds on the inverse kinematics solution of the robotic manipulator using different soft computing methods.

El-Sherbiny et al. [27] conduct a study on ANN, adaptive neuro-fuzzy inference system (ANFIS), and genetic algorithms (GA) that were used to investigate the inverse kinematic solutions of the Five-DOF robotic manipulator. In terms of cartesian position inaccuracy of the end effector and computation time, ANFIS outperforms ANN and GA, according to the research. Soylak et al. [28] conduct a study on ANN, ANN was used to investigate the inverse kinematics solution of a plasma cutting robot. To train the artificial neural network, the authors adopt the Levenberg–Marquardt training algorithm. The macneal schwendler corporation- automated dynamic analysis of mechanical systems (MSC-ADAMS) software is used to simulate this robot for a specified trajectory.

The ANN demonstrates effective results in terms of angle and position inaccuracy compared to established values. Deng and Xie [8] in this study, the effectiveness of adaptive particle swarm optimization (APSO) was evaluated using two serial robotic manipulators. In the meantime, a number of potent PSO variations that were developed using various techniques were chosen to contrast with APSO. The experimental findings show that the suggested fitness function, with APSO may effectively and precisely address the inverse kinematic, problem of multi-DOF robotic manipulators. The authors worked on an inverse kinematics solution for a puma-560 robotic manipulator using an improved or APSO. In comparison to other PSO, quantum behaved particle swarm algorithm (QPSO), and starling PSO, the author determined that APSO provides accurate and exact joint configurations for the required position and orientation.

Yiyang et al. [29] a complex nonlinear inertia weight gradually develops based on the idea of similarity is introduced to prevent the particle update rate from failing to adapt to each stage of the optimization process, making the search process more reliable. Additionally, the issue of local optimal solution is

also addressed. In order to do optimization search simultaneously, numerous populations are created, and the immigrant operator is suggested to broaden the variety of the particle population in each iteration. The results of this study, which used the Comau NJ-220 robot for test verification, reveal that the proposed enhanced PSO has higher algorithm robustness for generic robot kinematics, inverse solution issues and may significantly increase convergence accuracy, and speed. This approach offers a fresh approach to the problem of robot, inverse kinematics and offers a more reliable and effective kinematics base for the robot, motion planning.

Rahkar [30] a brand-new metaheuristic optimization technique called battle royale optimization (BRO) is put forth in this study. The proposed strategy draws inspiration from the "battle royale" video game subgenre. In BRO, a population-based algorithm, each person is represented like a soldier or player who wants to relocate to the safest (best) location and ultimately survive. For the inverse kinematics solution of the 6-DOF PUMA 560 robot arm, the suggested approach has been evaluated with the well-known PSO technique and six recently proposed optimization algorithms. The experimental findings demonstrate that the suggested algorithm is an effective technique that yields favourable and competitive results, both in terms of convergence and accuracy.

Šegota et al. [31] in this study, a multilayer perceptron (MLP), a feed-forward form of ANN, is trained in order to compute the inverse kinematics of a robotic manipulator. First, the D-H approach is used to derive the direct kinematics of a robotic manipulator. Then, a dataset, of 15,000 points is produced using the estimated homogenous transformation, matrices. The best is then selected after training 10,240 distinct hyperparameter combinations on multiple MLPs. The topologies of the MLPs that produced the best results are provided, and each trained MLP is assessed using the R2 and mean absolute error metrics. As a result of the robotic manipulator's design, the final joint's regression performed comparably poorly compared to the first five joints (percentage error was less than 0.1 percent). Recently a new modified GWO algorithm was developed by Dereli [32]. The author replaces the ‘a’ parameter of the GWO algorithm with the fast parabolic descending (FPD) function that reduces the value of ‘a’ very fast and reaches zero with minimal time as compared to the older ‘a’, it decrees linearly so it is also taking more time. Serkan Dereli experiment on this fast parabolic descending- grey wolf optimization (FPD-GWO) algorithm to find the optimum solution of same function and inverse kinematics of the 7-DOF robotic manipulator. The author concluded that FPD-GWO gives very less position error as compared to GWO, FA, QPSO, ABC, and whale optimization algorithm (WOA).

So, in conclusion, there is very confined study founds related to inverse kinematics solution of three-link robotic manipulator using soft computing methods, all previous study was conducted on 7-DOF or 6-DOF robotic manipulator.

The purpose of this study is to find solutions of inverse kinematic of a newly designed three-link robotic manipulator that is used for casting process application using a soft computing method called the ALO. This algorithm was tested by two factors: position error of the end-effector and solution time and comparison with another optimization algorithm: - PSO, GWO and SCA that all were studied. The efficiency and effectiveness of these algorithms have been checked by comparative analysis of two cases: case-I and case-II.

## 3.Methods

The techniques for attaining the objectives mentioned in the introduction are addressed in this part.

### 3.1Robotic manipulator kinematic modelling

In the brass pellets production business, a three-link robotic manipulator for casting application was designed and developed using SolidWorks software as presented in *Figure 1(a)* and fabricated as shown in *Figure 1(b).* The actuator used for joints 1, 2 and 3 are epicyclic gear, trained stepper motors and these motors were controlled by a stepper motor driver with a power supply of 12 V and 5 Amp. A two-degree of freedom end-effector is also controlled by two small stepper motors for a mechanism that is used for turning and rotating of a small crucible.

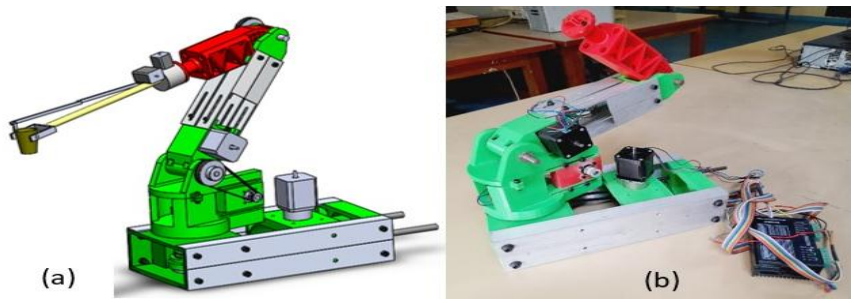


**Figure 1** Three-link robotic manipulator (a) CAD model (b) fabricated robot

The end-effector location of this robotic manipulator is determined using forward kinematics. The frame assignment on the robotic manipular using D-H parameters is the initial step in calculating forward kinematics. *Figure 2(a)* depicts the research frame assignment on a three-link robotic manipulator and also the end-effector in *Figure 2(b).*

Now the D-H parameter table is constructed using frame assignment and robot dimensions as in *Table 1*. After that forward kinematics transformation matrices from Equation 2 to Equation 5 have been made by putting values of D-H parameters from *Table 1* in Equation 1.

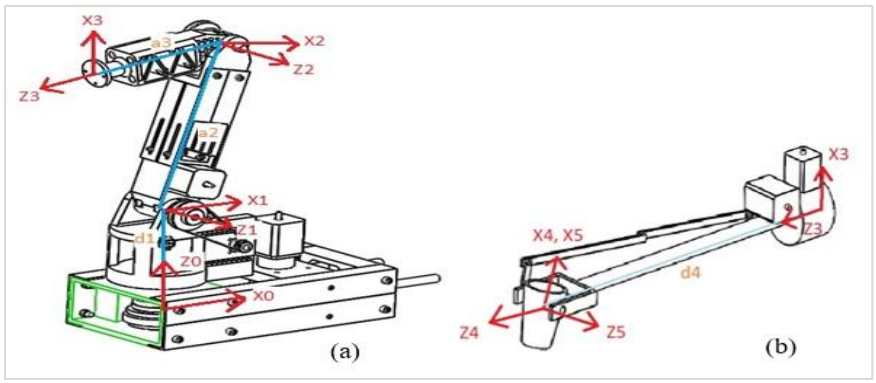


**Figure 2** D-H frame assignments on (a) three-link robotic manipulator and (b) end-effector

**Table 1** The D-H parameter table

| $\boldsymbol{Link\ i}$ | $\boldsymbol{a_i(m)}$ | $\boldsymbol{\alpha_i}(°)$ | $\boldsymbol{d_i(m)}$ | $\boldsymbol{\theta_i}(°)$ $\boldsymbol{(Range)}$ |
|---|---|---|---|---|
| 1 | 0 | $90^o$ | d1=0.19 | $\theta_1$ $(-135\ to\ 135)$ |
| 2 | a2=0.25 | 0 | 0 | $\theta_2$ $(-90\ to\ 90)$ |
| 3 | a3=0.16 | $-90^o$ | 0 | $\theta_3 - 90$ $(-90\ to\ 90)$ |
| 4 | 0 | 0 | d4=0.26 | $\theta_4$ $(-180\ to\ 180)$ |
| 5 | 0 | $90^o$ | 0 | $\theta_5$ $(-45\ to\ 45)$ |

End-effector position equations have been obtained by using the overall transformation matrix Equation 5. The end-effector position is represented by $P_X$, $P_Y$ and $P_Z$ as mention in Equation 6, Equation 7 and Equation 8 respectively.

$$T_i^{i-1} = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$T_1^0 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_2^1 = \begin{bmatrix} C_2 & -S_2 & 0 & a2.C_2 \\ S_2 & C_2 & 0 & a2.S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$T_3^2 = \begin{bmatrix} S_3 & 0 & -C_3 & a3.S_3 \\ C_3 & 0 & S_3 & a3.C_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_4^3 = \begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$T_5^4 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$T_5^0 = T_1^0.T_2^1.T_3^2.T_4^3.T_5^4 = \begin{bmatrix} n_x & s_x & a_x & P_x \\ n_y & s_y & a_y & P_y \\ n_z & s_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$P_x = -C_1.(d4.(S_3)^2 - a2.C_2 + d4.C_2.C_3 - a3.C_2.S_3 + a3.C_3.S_3) \quad (6)$$

$$P_y = -S_1.(d4.(S_3)^2 - a2.C_2 + d4.C_2.C_3 - a3.C_2.S_3 + a3.C_3.S_3) \quad (7)$$

$$P_z = d1 + a2.S_2 + a3.C_{23} - d4.S_{23} \quad (8)$$

Where: $C_1 = Cos(\theta_1)$, $S_1 = Sin(\theta_1)$ ……. etc. and $C_{23} = Cos(\theta_2 - \theta_3)$, $S_{23} = Sin(\theta_2 - \theta_3)$

The working procedure of the whole study is presented by the block diagram in *Figure 3*. First a three-link robotic manipulator is designed and frame is assigned. After that, all D-H parameters of this robotic manipulator are created. The transformation matrixes are created between two consecutive joints using Equation 1. End-effector position in x, y and z direction are formed by multiplication of all these transformation matrixes that is shown by Equation 5, the end-effector position equations in x, y and z directions are given by Equation 6, Equation 7 and Equation 8 respectively. After that soft-computing methods are used to find the inverse kinematics solutions (findings of joint angles for a given position of end-effector) of the three-link robotic manipulator applied in casting. In this study ALO, PSO, GWO and SCA algorithms are used to find the inverse kinematics solutions. Two cases of this study are conducted to find the comparative effective method for solutions of inverse kinematics. Case-I: take a single point of end-effector position in the workspace (*Figure 7*), then during the working task in casting there is some difference in the actual position and target position of end-effector (*Figure 6*) this difference called the position error. The position error will be converted into a fitness function, so a point is selected manually in the workspace, after that these algorithms minimize this error and find the joint angles for a manual particular end-effector position in the workspace. Now, in Case II: 20 random points are selected as similar to Case I, but after finding the position error and solution time we take the average of all these 20 solutions. At the end comparing the results of both cases and all four algorithms we found that Case II with the ALO gives effective good results as compare to Case I and PSO, GWO and SCA algorithms in terms of position error and solution time.

### 3.2Ant lion optimizer

The ant lion belongs to the Neuroptera order and the Myrmeleontidae family. The lifespan of ant lions is up to 3 years, which include mainly two phases: larvae and adult. Ant lions spend the most of their lives as larvae, with the remainder as adults (three to five weeks). The larval phase is when they hunt the most, whereas the adult phase is when they reproduce. The ant lion optimizer imitates the larval phase of ant lion hunting behaviour. The larvae of the ant lion dig cone-shaped holes of varying sizes in the soil or ground. The ant lion stays at the lowest position of the cavity as shown in *Figure 4 (a)* and wait for ants to slide on the loose sands and drop into the pits. When an ant or insect is trapped in cone shape pits the prey tries instantly to escape from the trap but the ant lion catches them by throwing sands near the border of the cavity. The ant lion drags the prey after grabbing it and swallow it as shown in *Figure 4 (b).*

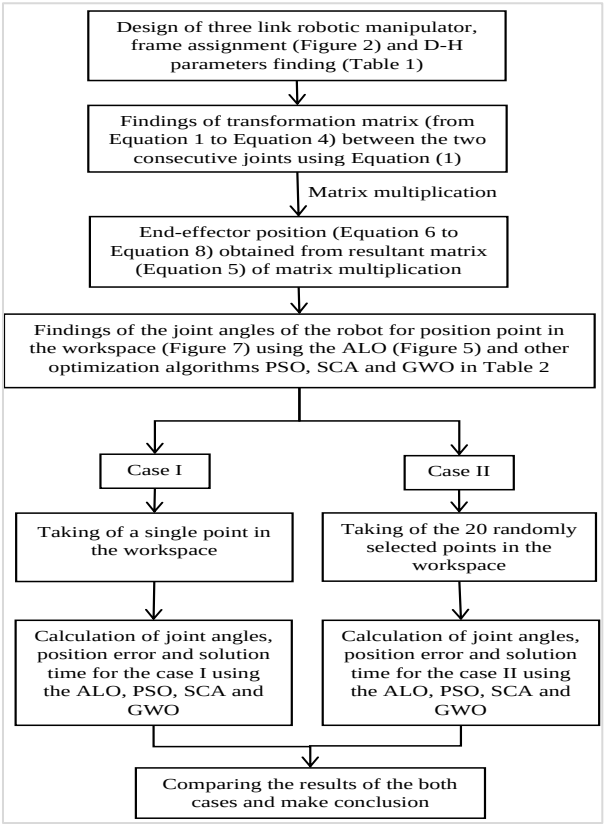


**Figure 3** Working procedure

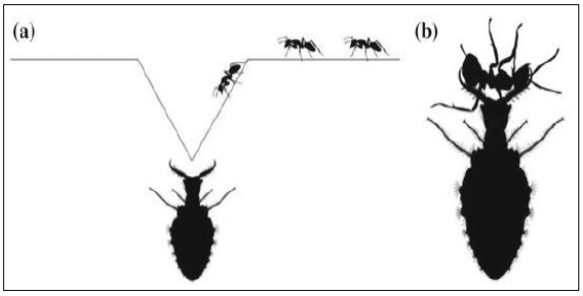


**Figure 4** Ant lion hunting mechanism [33] (a) the ant lion waiting for the ants or prey (b) catching the prey

As a mimic of this natural hunting process of ant lion, ant lion optimizer was developed by Mirjalili [34]. In this optimizer ants act as seeking agents, walking across conclusion space and being hunted by ant lions in order to improve their fitness. The ant lion's position is enhanced in relation to the chosen ant lion based on the elite and roulette wheel. First, the fitness functions of ants and ant lions picked at random in search space are calculated. Every iteration, the roulette wheel's operator selects an ant lion for each ant. The ant lion's location is revised with the help of two random walks throughout the roulette-selected oligarchy and ant lion. The latest position of the ants can be deduced by computing their fitness functions and differentiating particular ant lions. If an ant suit fits better than its associated ant lion on the next iteration, its placement is identified as a new location for the ant lion. In addition, if the finest ant lion reported in the present iteration is more fit than the elite, the oligarchy will be modified. These steps are repeated until the number of iterations reaches zero. The ALO flowchart is shown in *Figure 5*.

### 3.2.1Initialization of ant and ant lion locations, as well as assessment of their fitness functions

The ALO defines two types of dweller ants and ant lions. In the conclusion region, ants are the seeking agents, while ant lions hide in the area, follow them down, and grab their site in order to get fitter. Ant and ant lion's position in one decision variable can be stated in Equation 9 and Equation 10 respectively.

$$\text{Ant's position} = [A_1, A_2 \dots \dots A_n \dots \dots A_N] \quad (9)$$

$$\text{Ant lion's position} = [AL_1, AL_2 \dots AL_n \dots AL_N] \quad (10)$$

Where: $A_N$ and $AL_N$ are nth decision variable of ant and ant lion respectively, and N is number of selected variables.

Position matrix of ant and ant lion is presented by Equation 11.

$$M_{ant} = \begin{Bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} & \cdots & A_{1,N} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} & \cdots & A_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{m,1} & A_{m,2} & \vdots & A_{m,n} & \vdots & A_{m,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{M,1} & A_{M,2} & \vdots & A_{M,n} & \vdots & A_{M,N} \end{Bmatrix} M_{antlion}$$

$$= \begin{Bmatrix} AL_{1,1} & AL_{1,2} & \cdots & AL_{1,n} & \cdots & AL_{1,N} \\ AL_{2,1} & AL_{2,2} & \cdots & AL_{2,n} & \cdots & AL_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ AL_{m,1} & AL_{m,2} & \vdots & AL_{m,n} & \vdots & AL_{m,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ AL_{M,1} & AL_{M,2} & \vdots & AL_{M,n} & \vdots & AL_{M,N} \end{Bmatrix} \quad (11)$$

Where: $M_{ant}$ and $M_{antlion}$ are position matrix of ant and ant lion respectively, $A_{M,N}$ and $AL_{M,N}$ are $n^{th}$ selected variable of the $m^{th}$ ant and ant lion respectively.

Fitness function of ant and ant lion is presented in Equation 12.

$$F_{ant} = \begin{Bmatrix} f([A_{1,1} & A_{1,2} & \cdots & A_{1,n} & \cdots & A_{1,N}]) \\ f([A_{2,1} & A_{2,2} & \cdots & A_{2,n} & \cdots & A_{1,N}]) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f([A_{m,1} & A_{m,2} & \vdots & A_{m,n} & \vdots & A_{m,N}]) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f([A_{M,1} & A_{M,2} & \vdots & A_{M,n} & \vdots & A_{M,N}]) \end{Bmatrix} F_{antlion}$$

$$= \begin{Bmatrix} f([AL_{1,1} & AL_{1,2} & \cdots & AL_{1,n} & \cdots & AL_{1,N}]) \\ f([AL_{2,1} & AL_{2,2} & \cdots & AL_{2,n} & \cdots & AL_{1,N}]) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f([AL_{m,1} & AL_{m,2} & \vdots & AL_{m,n} & \vdots & AL_{m,N}]) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f([AL_{M,1} & AL_{M,2} & \vdots & AL_{M,n} & \vdots & AL_{M,N}]) \end{Bmatrix} \quad (12)$$

Where: $F_{ant}$ and $F_{antlion}$ are fitness function matrix of ant and ant lion respectively.

### 3.2.2 Digging trap and sliding ants toward ant lion

#### 3.2.2.1 Digging

Every ant is assigned to an ant lion using the roulette wheel operator in this procedure. Every iteration, each ant can only be imprisoned in one ant lion. Because a solution with a higher fitness function has a better probability of being chosen using a roulette wheel operator, an ant lion with a large trap can chase down more aunts.

#### 3.2.2.2 Skidding ants with respect to ant lion

When an ant falls into the confine, it continually strives to escape, but the ant lion throws the stand outward centre of the pit, allowing the ant lion to slide sown. Equation 13 to Equation 15 presents parameters of skidding ants with respect to ant lion.

$$\alpha(i) = \frac{c(i)}{R} \quad (13)$$

$$\beta(i) = \frac{d(i)}{R} \quad (14)$$

$$R = 10^{a\frac{(i)}{(I)}} \quad (15)$$

Where: $\alpha(i)$ a modified vector of minimum of all chosen variables at i[th] iteration; $c(i)$ a vector of minimum of all selected variables at i[th] iteration; R is a ratio defined by Equation (16); $\beta(i)$ is a modified vector of maximum of all decision variables at i[th] iteration and $d(i)$ is a modified vector of maximal of all dependent variables at i[th] iteration

$$a = \begin{cases} 2 & If\ i > 0.1I \\ 3 & If\ i > 0.5I \\ 4 & If\ i > 0.75I \\ 5 & If\ i > 0.9I \\ 6 & If\ i > 0.95I \end{cases} \quad (16)$$

Where: $a$ is a constant drive from iteration number. $i$ and $I$ are iteration counter and iterations number respectively. Convergence of the ALO happens the radius of random walk decreases, when iteration no. in Equation 16 increases.

### 3.2.3 Catching ants inside pits and random walk of ants

#### 3.2.3.1 Seizing ants inside ditch

Ant lion baits have the ability to influence ant's actual movement. Every iteration of a mathematical model of these characteristics governs the threshold of an ant stochastic process, allowing the ant to walk-in twitchy on all sides of the designated ant lion bait. The upper and lower thresholds of the ant's stochastic process can be computed using the following Equation 17 and Equation 18 for each domain and iteration:

$$\alpha_m(i) = Antlion_l(i) + \alpha(i) \quad (17)$$

$$\beta_m(i) = Antlion_l(i) + \beta(i) \quad (18)$$

Where: $\alpha_m(i)$ is a vector of minimum of conclusion variables for m[th] ant in the i[th] iteration; $Antlion_l(i)$ is a location of a selected ant lion in the i[th] iteration; $\beta_m(i)$ is a vector of minimum of all conclusion variables for m[th] ant. Equation (17) and Equation (18) show the ant's random walk is in hyperspace, that is explained by vector $\alpha$ and $\beta$ on all sides of the roulette wheel selected ant lion.

#### 3.2.3.2 Ants roam stochastically in search of food in nature:

which can be represented by the Equations 19 and Equation 20 below:

$$X(t) = [0, cumsum(2r(1) - 1), cumsum(2r(2) - 1), \ldots . cumsum(2r(i) - 1) \ldots cumsum(2r(I) - 1)] \quad (19)$$

$$r(i) = \begin{cases} 1 & if\ rand > 0.5 \\ 0 & if\ rand \leq 0.5 \end{cases} \quad (20)$$

Where: $X(t)$ is vector of random walk position; $cumsum$ is cumulative sum; $r(i)$ is a function that can be calculated by Equation 20 and $rand$ (in Equation 20) is a randomly generated value with a uniform distributed between 0 and 1. Equation 21 represent standardised nonstationary destination of the n[th] response factor at the i[th] iteration.

$$P_n(i) = \frac{(X_n(i) - E_n) + (\beta_n(i) - \alpha_n(i))}{F_n - E_n} \quad (21)$$

Where: $P_n(i)$ is standardised nonstationary destination of the n[th] response factor at the i[th] iteration; $X_n(i)$ is nonstationary destination of the n[th] response factor at the i[th] iteration before standardisation; $E_n$ is the lower bound of conditional variance for the n[th] response factor; $F_n$ is upper bound of conditional variance for the n[th] response factor; $\alpha_n(i)$ is lower bound of the n[th] response factor at the i[th] iteration; $\beta_n(i)$ is upper bound of the n[th] response factor at the i[th] iteration.

### 3.2.4 Elitism, grabbing prey and rebuild the imprison

#### 3.2.4.1 Elitism

Snobbishness is an essential feature of metaheuristics, which use optimization procedures to find the optimal solution. The best ant lion in each iteration saved in the ALO is termed prestige ant lion, and the average of both prestige ant lion and stochastic trudges around the roulette wheel sorted ant lion is considered the new ant lion candidate. Equation 22 represents location of picked m[th] ant lion in i[th] iteration.

$$Ant_m^i = \frac{R_l^i + R_e^i}{2} \quad (22)$$

Where: $Ant_m^i$ is the location of picked m[th] ant lion in the i[th] iteration; $R_l^i$ is $l^{th}$ roulette wheel picked ant lion random walk all sides at the i[th] iteration and $R_e^i$ is random walk all sides of the the elite ant lion at the i[th] iteration.

#### 3.2.4.2 Catching the prey and redesign the imprison

Ant lion isolating ants fall into the incarcerates base in the last stage of the ALO algorithm, where they are seized by the ant lion's jaw and devoured. When an ant's objective function is superior to that of its ant lion counterpart, it can grab the victim. The stalked ant lion moves to the ant lion that has been stalked. The following Equation 23 demonstrates this procedure.

$$Antlion_l^i = Ant_l^i \ \text{If}\ f(Ant_m^i) > f(Antlion_m^i) \quad (23)$$

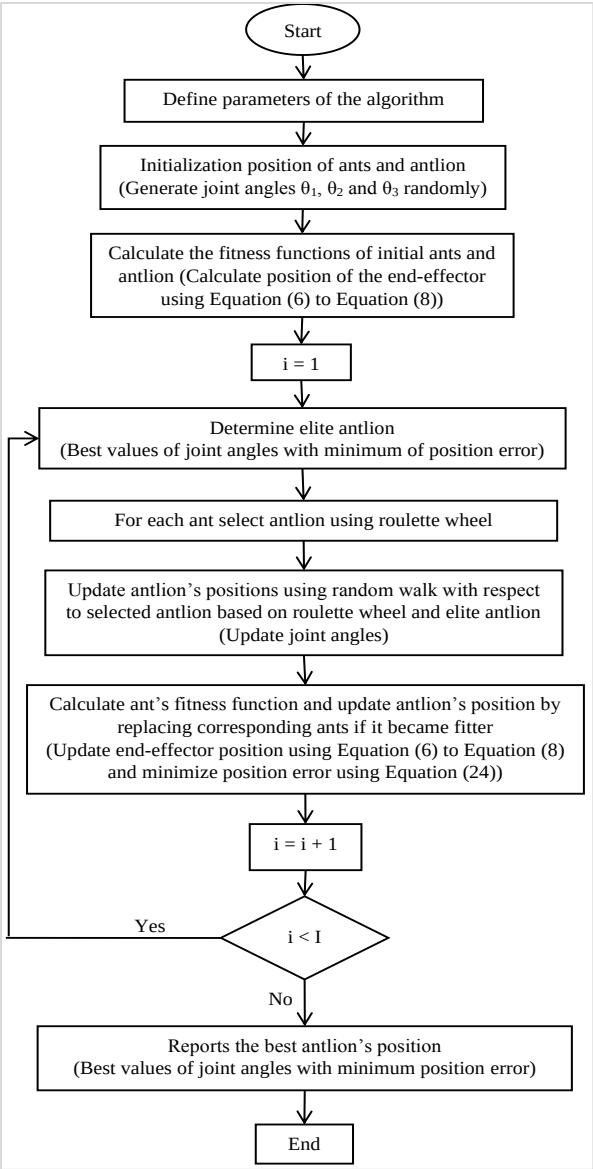


**Figure 5** The ALO flowchart (where: I= number of iterations and i = iteration counter)

**Fitness function**

The major purpose of this research is to minimise position error while obtaining the robotic manipulator joint angles for the intended path/position. As we can see in *Figure 6* during casting operation for the pouring of liquid metal a predefined trajectory that have been performed by robotic manipulator end-effector. The accurate trajectory is obtained by minimizing position error (difference between target position ($P_2$) and actual position ($P_1$)) as presented in Equation 24. Jangid et al. [35] did similar research on improving tracking performance in a casting process (Equation 24).

$$\text{error} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2} \quad (24)$$

## 4.Simulation results

The ALO method was evaluated for inverse kinematics solutions against other optimization techniques such as PSO, GWO, and SCA. This research was conducted in two situations, with a comparison drawn at the conclusion. All algorithms are written in MATLAB 2016a and run on a PC with an Intel(R) Core (TM) i3-7020U processor running at 2.30 GHz and 4 GB of RAM. The position error and solution time are all taken into account in all scenarios.

### 4.1Case-I

In first case a single point is selected from work space (*Figure 7*) of the robotic manipulator as ($P_X$, $P_Y$, $P_Z$) = (0.1225, 0.0392, 0.5627) these values have been obtained by putting ($\theta_1$, $\theta_2$, $\theta_3$) = ($60^o$, $45^o$, $45^o$) values in Equation 6 to Equation 8. Although three-link robotic manipulator have fixed number of solutions for inverse kinematics with in workspace as shown in *Figure 7*.

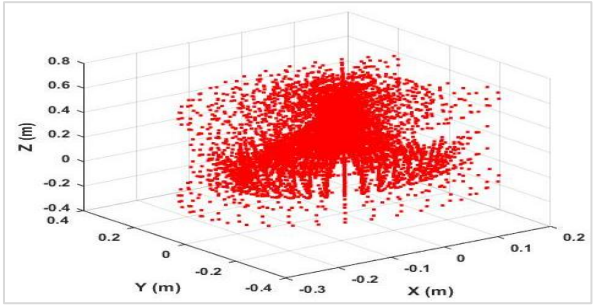


**Figure 7** Three-dimensional workspace of the robotic manipulator

The optimum values of joint angles for the case-I were obtained from different optimization algorithms that are shown in *Table 2*. These values are obtained by selecting the best minimum position error and minimum solution time after running the algorithm 10 times.

The optimum values of end-effector position and its position error compared to other optimization algorithms are shown in *Table 3*. We can see as well in *Figure 8*, that effective results are produced by the ALO in terms of position error compared to other optimization algorithms. These solutions are obtained by selecting different particles size (algorithm parameter) for each algorithm for each time after running the algorithm for 10 times then effective results are taken.

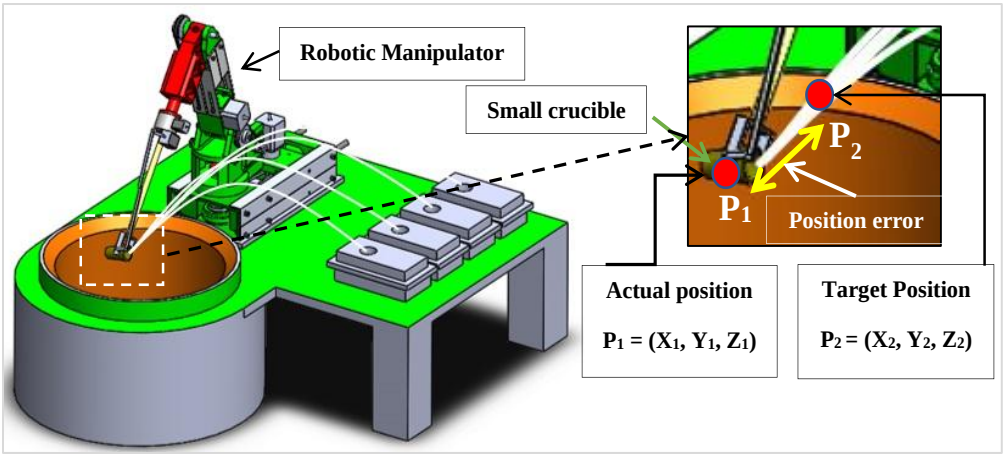


**Figure 6** CAD model presenting the robotic manipulator end-effector, there is a position inaccuracy (error) while poured liquid metal from a small pot

**Table 2** Joint angle values obtained from different optimization algorithms

| Angles | Range (In degree) | Manuel values | PSO | SCA | GWO | ALO |
|---|---|---|---|---|---|---|
| $\theta_1$ | -135° to 135° | 60° | 22.3098 | -112.8017 | -122.1671 | 85.1327 |
| $\theta_2$ | -90° to 90° | 45° | -17.3056 | 12.87852 | -61.81894 | 7.30068 |
| $\theta_3$ | -90° to 90° | 45° | -2.5887 | -4.739529 | 13.57607 | 88.9821 |

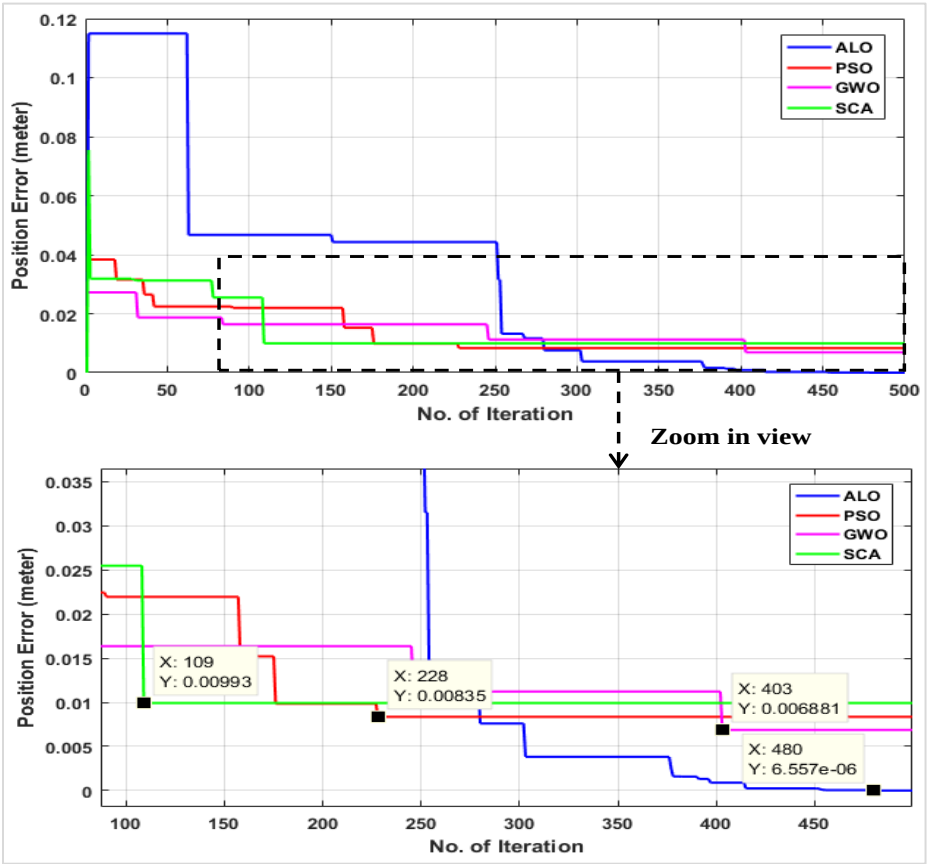


**Figure 8** Position error verses iteration graph for case-I

Now standard particles size for effective results are given in *Table 3.* All these algorithms are run at a maximum iteration of 500. The PSO algorithm gives an effective solution at iteration nearly 200 and solution time nearly 14 seconds. When the GWO is used, it gives the solution at iteration nearly 400 and solution time nearly 2 seconds. The SCA gives the solution at iteration nearly 100 and solution time nearly 1.3 seconds and the last ALO gives the effective solution at iteration nearly 450 and solution time nearly 0.8 seconds. As a result, in terms of position/location inaccuracy and time to solve, ALO produce very good results as compared to PSO, GWO, and SCA. The GWO algorithm gives good results as compared to PSO and SCA. It was also found that when the number of particles increases in GWO, it reduces the position error and also increases the solution time. When the number of particles/agents increases in ALO there was not very much effects/difference founds on the results of position/location error but solution time increases.

**Table 3** Position of the end-effector obtained from optimization algorithms

| **End effector position (In meter)** | **Manual values** | **PSO** | **SCA** | **GWO** | **ALO** |
|---|---|---|---|---|---|
| $P_X$ | 0.1225 | 0.1261 | 0.1141 | 0.1197 | 0.12251 |
| $P_Y$ | 0.0392 | 0.0416 | 0.0347 | 0.0444 | 0.0392 |
| $P_Z$ | 0.5627 | 0.5698 | 0.5652 | 0.5613 | 0.5627 |
| Error (m) | | 0.00835 | 0.00993 | 0.006881 | 6.557e-06 |

### 4.2Case-II

In this case, 20 randomly points have been selected from the workspace (*Figure 7*) that are showing graphically in *Figure 9*. These points are used to verify the quality of the algorithms used in case-I because this randomly selected point gives different values of position/location error and solution time as compared to the previous one, so with help of this we can compare the quality of these algorithms (*Table 4*). So, as we can see in *Figure 10* ALO gives much better results as compared to others. The GWO algorithm also gives some good results as compared to PSO and SCA. *Figure 11* shows the solution time graph for 20 randomly selected points as we can see ALO gives very less minimum solution time as compared to others. The PSO gives the worst solution time in this study. *Table 5* contains all of the results collected from the Case II. In comparison to PSO, SCA, and GWO, ALO clearly provides superior outcomes in terms of position inaccuracy and solution time.

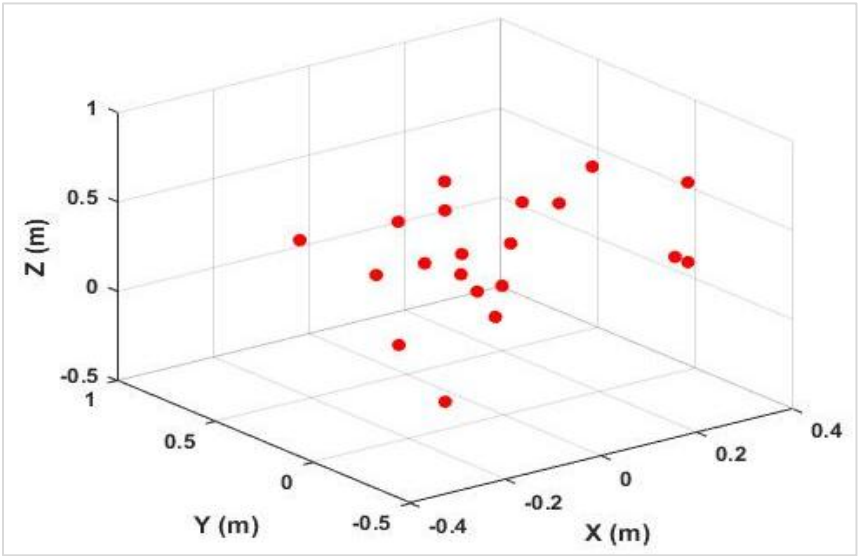


**Figure 9** 20 randomly selected points in workspace

**Table 4** ALO vs. other optimization techniques

| | Particle size | Max. iteration | Solution iteration | Position error (m) | Solution time (s) |
|---|---|---|---|---|---|
| PSO | 300 | 500 | 229 | 0.00835 | 14.34 |
| SCA | 150 | 500 | 109 | 0.00993 | 1.31 |
| GWO | 150 | 500 | 404 | 0.006881 | 2.01 |
| ALO | 2 | 500 | 480 | 6.557e-06 | 0.88 |

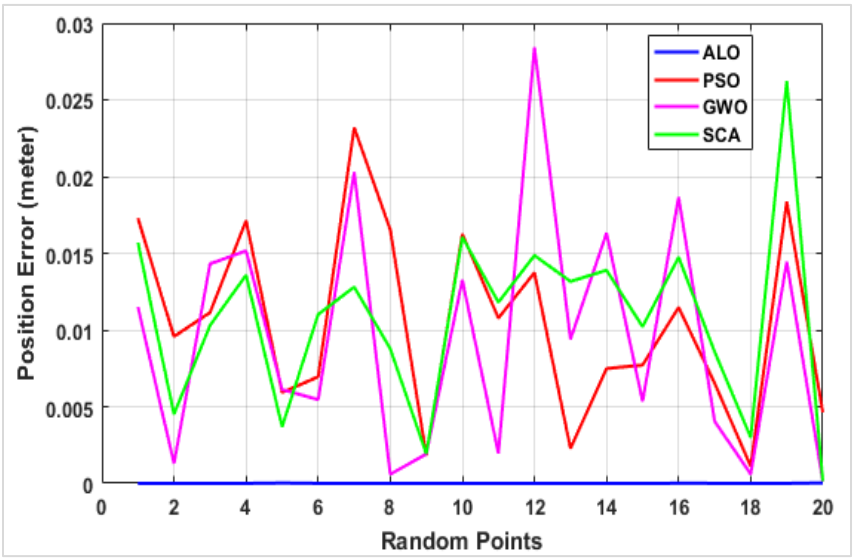


**Figure 10** Position error for 20 randomly selected points in workspace

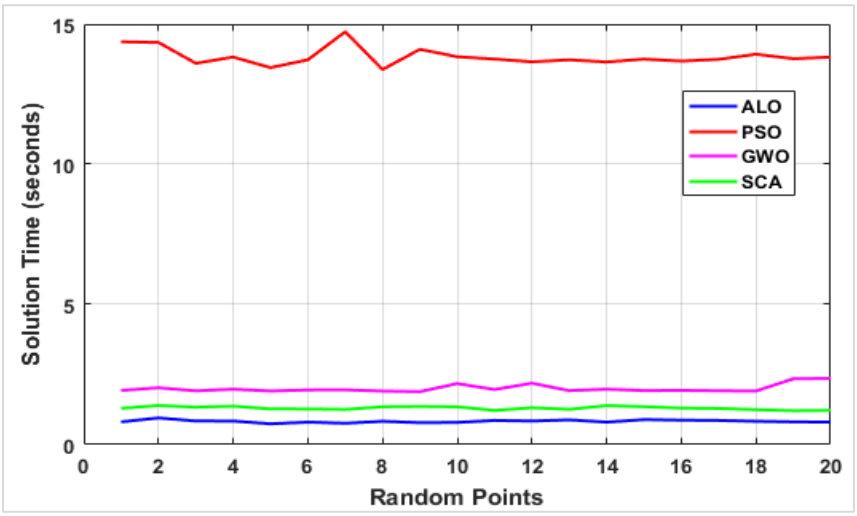


**Figure 11** Time to solve 20 randomly chosen points in the workplace

**Table 5** Comparison of ALO with other optimization algorithms for 20 randomly selected points

| | Particle size | Max. iteration | Position error (m) (Avg. of 20) | Solution time (s) (Avg. of 20) |
|---|---|---|---|---|
| PSO | 300 | 500 | 0.010494 | 13.84093 |
| SCA | 150 | 500 | 0.010748 | 1.305011 |
| GWO | 150 | 500 | 0.009461 | 2.006823 |
| ALO | 2 | 500 | 6.34e-06 | 0.832892 |

## 5.Discussion

The ALO gives very less position error and solution time as compare to PSO, GWO and SCA and it is also taking high number of iterations to get these results as compare to others as shown in *Figure 8.* The major findings of this study are as below:

- The ALO gives 99.92147 %, 99.90471 % and 99.93397 % less position error compared to PSO, GWO and SCA respectively for case-I.
- The ALO gives 93.86332 %, 56.21891 % and 32.82443 % less solution time compared to PSO, GWO and SCA respectively for case I.
- The solution iteration for case-I is nearly 480, 200, 400 and 100 for ALO, PSO, GWO and SCA respectively.
- The ALO gives 99.93958 %, 99.93298 % and 99.94101 % less position error compared to PSO, GWO and SCA respectively for case-II.
- The ALO gives 93.98239 %, 58.49698 % and 36.17739 % less solution time compared to PSO, GWO and SCA respectively for case-II.

*Table 6* shows some comparative research on inverse kinematics solution and their findings in terms of position inaccuracy and solution time. Previous studies found that Adaboost neural network (NN), ANFIS, FA and ABC some high position error as compare to the ALO. But the FA, ABC and Adaboost NN gives less solution time as compare to the ALO. On the other hand, APSO and Improved PSO gives some less position error as compare to the ALO but both consume high solution time as compare to the ALO.

A complete list of abbreviations is shown in *Appendix I.*

**Table 6** The ALO in comparison to other research published in the literature

| Author | Robotic arm | Technique used | Position error (m) | Solution time (s) |
|---|---|---|---|---|
| Shi and Xie (2017) [36] | 6-DOF | Adaboost NN | 2.67e−03 | 0.3e-03 |
| El-Sherbiny et al. (2017) [27] | 6-DOF | ANFIS | 5.426e−03 | 0.0308 |
| Dereli and Köker (2019) [12] | 7-DOF | FA | 6.53e−05 | 0.9204 |
| Dereli and Köker (2019) [13] | 7-DOF | ABC | 4.75e−04 | 0.2087 |
| Heng and Chong (2021) [8] | 6-DOF and7-DOF | APSO | 1.19e-08 (Best) | Less than 2t |
| Yiyang et al. (2021) [29] | Comau NJ-220 robot | Improved PSO | 8.00e-07 (best) | |
| Present study | 3-DOF | ALO | 6.34e-06 | 0.832892 |

### 5.1Limitations

The main limitations of this study are below-

- Each algorithm produces different results of position error and solution time when changing algorithm parameters.
- Solution time is also depending on processer speed of the computer system.
- Some algorithm gives high position error and less processing time compare to other algorithm vice-versa as in *Table 6*.

## 6.Conclusion and future work

An inverse kinematics solution of a three-link serial robotic manipulator used in casting was done and simulated in this work to test the accuracy and effectiveness of the ALO approach. When the count of the DOF rises, calculating inverse kinematics becomes even more complicated, which is when heuristic functions come in handy. The data obtained by ALO is compared with PSO, GWO, and SCA to ensure that the method is valid. The experiments of simulation have been carried out with two different cases. In case-I, a single point was selected in the workspace of the robot, and the test of position/location error and solution time was carried out that have compared with case-II. The algorithm's reliability has been demonstrated in case-II (drawn from the population 20 spots in the workspace).

In perspective of position error and solution time, the results demonstrate that ALO performs significantly better than the other algorithms tested in this study. In terms of position inaccuracy and solution time, ALO can be used in inverse kinematic solutions.
In the future, alternative optimization algorithms or artificial intelligence could be employed to solve the inverse kinematics of a robotic manipulator with several DOF robotic manipulator.

### Acknowledgment

The authors are thanks to NDF/ADF AICTE NEW DELHI for funding their study.

### Conflicts of interest

The authors have no conflicts of interest to declare.

## Author's contribution statement

**Mahendra Kumar Jangid:** Conceptualization, investigation, design and development, writing - original draft. **Sunil Kumar:** Writing - review and editing, analysis and interpretation of results. **Jagtar Singh:** Writing - review and editing, supervision.

## References


[1] Yahya S, Moghavvemi M, Yang SS, Mohamed HA. Motion planning of hyper redundant manipulators based on a new geometrical method. In international conference on industrial technology. 2009 (pp. 1-5). IEEE.

[2] Sheng L, Yiqing W, Qingwei C, Weili H. A new geometrical method for the inverse kinematics of the hyper-redundant manipulators. In international conference on robotics and biomimetics 2006 (pp. 1356-9). IEEE.

[3] Olsen AL, Petersen HG. Inverse kinematics by numerical and analytical cyclic coordinate descent. Robotica. 2011; 29(4):619-26.

[4] Perez A, Mccarthy JM. Sizing a serial chain to fit a task trajectory using clifford algebra exponentials. In proceedings of the international conference on robotics and automation 2005 (pp. 4709-15). IEEE.

[5] http://ethesis.nitrkl.ac.in/6980/1/2015_Panchanand_Phd_511ID101.pdf. Accessed 10 April 2022.

[6] Rokbani N, Alimi AM. Inverse kinematics using particle swarm optimization, a statistical analysis. Procedia Engineering. 2013; 64:1602-11.

[7] Huang HC, Chen CP, Wang PR. Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators. In international conference on systems, man, and cybernetics 2012 (pp. 3105-10). IEEE.

[8] Deng H, Xie C. An improved particle swarm optimization algorithm for inverse kinematics solution of multi-DOF serial robotic manipulators. Soft Computing. 2021; 25(21):13695-708.

[9] Sancaktar I, Tuna B, Ulutas M. Inverse kinematics application on medical robot using adapted PSO method. Engineering Science and Technology, an International Journal. 2018; 21(5):1006-10.

[10] Dereli S, Köker R. IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator. Sigma Journal of Engineering and Natural Sciences. 2018; 36(1):77-85.

[11] Alkayyali M, Tutunji TA. PSO-based algorithm for inverse kinematics solution of robotic arm manipulators. In international conference on research and education in mechatronics 2019 (pp. 1-6). IEEE.

[12] Dereli S, Köker R. Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy. Inverse Problems in Science and Engineering. 2020; 28(5):601-13.

[13] Dereli S, Köker R. Simulation based calculation of the inverse kinematics solution of 7-DOF robot manipulator using artificial bee colony algorithm. SN Applied Sciences. 2020; 2(1):1-11.

[14] El-sherbiny A, Elhosseini MA, Haikal AY. A new ABC variant for solving inverse kinematics problem in 5 DOF robot arm. Applied Soft Computing. 2018; 73:24-38.

[15] Dereli S, Köker R. A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. Artificial Intelligence Review. 2020; 53(2):949-64.

[16] Ayyıldız M, 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. 2016; 27(4):825-36.

[17] Köker R, Çakar T. A neuro-genetic-simulated annealing approach to the inverse kinematics solution of robots: a simulation based study. Engineering with Computers. 2016; 32(4):553-65.

[18] Köker R. A neuro-simulated annealing approach to the inverse kinematics solution of redundant robotic manipulators. Engineering with Computers. 2013; 29(4):507-15.

[19] Duka AV. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. Procedia Technology. 2014; 12:20-7.

[20] Park JK. Inverse kinematics based on fuzzy logic and neural networks for the WAM-titan II teleoperation system. Master's Thesis, University of Tennessee, 2007.

[21] Gao R. Inverse kinematics solution of robotics based on neural network algorithms. Journal of Ambient Intelligence and Humanized Computing. 2020; 11(12):6199-209.

[22] Gong M, Li X, Zhang L. Analytical inverse kinematics and self-motion application for 7-DOF redundant manipulator. IEEE Access. 2019; 7:18662-74.

[23] Tevatia G, Schaal S. Inverse kinematics for humanoid robots. In proceedings of international conference on robotics and automation. symposia proceedings (Cat. No. 00CH37065) 2000 (pp. 294-9). IEEE.

[24] Rokbani N, Mirjalili S, Slim M, Alimi AM. A beta salp swarm algorithm meta-heuristic for inverse kinematics and optimization. Applied Intelligence. 2022:1-26.

[25] Ghafil HN, Jármai K. Optimization algorithms for inverse kinematics of robots with MATLAB source code. In vehicle and automotive engineering 2020 (pp. 468-77). Springer, Singapore.

[26] Xu J, Wang W, Sun Y. Two optimization algorithms for solving robotics inverse kinematics with redundancy. Journal of Control Theory and Applications. 2010; 8(2):166-75.

[27] El-sherbiny A, Elhosseini MA, Haikal AY. A comparative study of soft computing methods to solve inverse kinematics problem. Ain Shams Engineering Journal. 2018; 9(4):2535-48.

[28] Soylak M, Oktay T, Turkmen İ. A simulation-based method using artificial neural networks for solving the inverse kinematic problem of articulated robots. Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering. 2017; 231(3):470-9.
[29] Yiyang L, Xi J, Hongfei B, Zhining W, Liangliang S. A general robot inverse kinematics solution method based on improved PSO algorithm. IEEE Access. 2021; 9:32341-50.
[30] Rahkar FT. Battle royale optimization algorithm. Neural Computing and Applications. 2021; 33(4):1139-57.
[31] Šegota SB, Anđelić N, Mrzljak V, Lorencin I, Kuric I, Car Z. Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator. International Journal of Advanced Robotic Systems. 2021; 18(4):1-11.
[32] Dereli S. A new modified grey wolf optimization algorithm proposal for a fundamental engineering problem in robotics. Neural Computing and Applications. 2021; 33(21):14119-31.
[33] Mani M, Bozorg-haddad O, Chu X. Ant lion optimizer (ALO) algorithm. In advanced optimization by nature-inspired algorithms 2018 (pp. 105-16). Springer, Singapore.
[34] Mirjalili S. The ant lion optimizer. Advances in Engineering Software. 2015; 83:80-98.
[35] Jangid MK, Kumar S, Singh J. Trajectory tracking optimization and control of a three link robotic manipulator for application in casting. International Journal of Advanced Technology and Engineering Exploration. 2021; 8(83):1255-67.
[36] Shi Q, Xie J. A research on inverse kinematics solution of 6-DOF robot with offset-wrist based on adaboost neural network. In international conference on cybernetics and intelligent systems (CIS) and conference on robotics, automation and mechatronics 2017 (pp. 370-5). IEEE.

**Mahendra Kumar Jangid** is currently a Ph.D. Research Scholar in Department of Mechanical Engineering SLIET Longowal, Punjab. He is doing Ph.D. under National Doctor Fellowship, AICTE New Delhi selected through GATE. His research interests are Robotic Manipulator, Mobile Robots, Kinematics and Dynamics.
Email: mahendra.jangid0007@gmail.com

**Dr. Sunil Kumar** is currentely an Assistant Professor at Department of Mechanical Engineering SLIET Longowal, Punjab. He received B.Tech. degree from REC Jalandhar, M.E. degree from TIET Patiala and Ph.D. from SLIET Lonowal. He has teaching experience of 17 years. His research interests are Robotics, Machine Design, Kinematics and Dynamics.
Email: sunil_thappa@yahoo.com

**Dr. Jagtar Singh** is currentely Professor at Department of Mechanical Engineering SLIET Longowal, Punjab. He received B.E. degree from Gulbarga University, M.E. degree from TIET Patiala and Ph.D. from NIT Kurukshetra. He is a life member of IWS (Indian Welding Society), ISTE (Indian Society for Technical Education) and SMES (SLIET Mechanical Engineering Society). His research interests are Wear, Additive Manufacturing, Farm Machinery, Cryogenic and Welding.
Email:jagtarsliet@gmail.com

## Appendix I

| S. No. | Abbreviation | Description |
|---|---|---|
| 1 | $a_i$ | Link Length |
| 2 | $\alpha_i$ | Link Twist |
| 3 | $d_i$ | Joint Offset/Distance |
| 4 | $\theta_i$ | Joint Angle |
| 5 | $C\alpha_i$ | $Cos\alpha_i$ |
| 6 | $S\alpha_i$ | $Sin\alpha_i$ |
| 7 | $P_x$ | End-Effector Position in x Direction |
| 8 | $P_y$ | End-Effector Position in y Direction |
| 9 | $P_z$ | End-Effector Position in z Direction |
| 10 | ABC | Artificial Bee Colony |
| 11 | ALO | Ant Lion Optimizer |
| 12 | ANFIS | Adaptive Neuro Fuzzy Inference System |
| 13 | ANN | Artificial Neural Network |
| 14 | APSO | Adaptive Particle Swarm Optimization |
| 15 | BRO | Battle Royale Optimization |
| 16 | CAD | Computer Aided Design |
| 17 | D-H | Denavit–Hartenberg |
| 18 | DOF | Degree of Freedom |
| 19 | FA | Firefly Algorithm |
| 20 | FPD | Fast Parabolic Descending |
| 21 | GA | Genetic Algorithms |
| 22 | GWO | Grey Wolf Optimization |
| 23 | MLP | Multi-Layer Perceptron |
| 24 | PSO | Particle Swarm Optimization |
| 25 | QPSO | Quantum Behaved Particle Swarm Algorithm |
| 26 | SCA | Sine Cosine Algorithm |
| 27 | WOA | Whale Optimization Algorithm |