Hind Z. Khaleel
Control and Systems Engineering Department, University of Technology, Baghdad, Iraq,
60175@uotechnology.edu.iq

Received on: 01/05/2019
Accepted on: 17/04/2019
Published online on: 25/07/2019

Enhanced Solution of Inverse Kinematics for Redundant Robot Manipulator Using PSO

Abstract- *Kinematics of the robot is divided into two parts: the forward kinematics, which evaluates the end-effector's position from joint angles, and the inverse kinematics, which demonstrates the joint angles from the end-effector's position. The solution of the inverse kinematics problem is too difficult and complicated for the redundant robot arm manipulator. A Particle Swarm Optimization (PSO) algorithm is an effective method to solve global optimization problems. This paper presents the solution of inverse kinematics problem of a three-link redundant manipulator robot arm using PSO without using the inverse kinematics equations. The circle, square and triangle generated trajectories using PSO are enhanced as compared with the trajectories of other works. The enhanced PSO algorithm is successfully found the best generating three joint angles and the best generating end-effector's position of a three-link robot arm. Then according to these joints and positions the circle, square and triangle path trajectories, results are smoother than the path trajectories of other work. This enhanced solution of inverse kinematics using PSO algorithm is too fast due to the short elapsed time in every iteration of trajectory. Besides that, these velocities results have been given evaluated and give an indication that the three-link robot is moving fast during the PSO algorithm. The elapsed time of circle trajectory equals to 20.903981 seconds, the elapsed time of square trajectory equals to 11.747171 seconds and the elapsed time of triangle trajectory equals to 15.729663 seconds. MATLAB R2015b program is used in order to simulate all results. The main benefit of this work is to solve two problems: 1) inverse kinematics is too complex equations of the three-link robot. The solutions of best joint angles using PSO are computed within joint limits without using inverse kinematics equations. 2) Another problem, this work is enhanced three trajectories with respect to the best joint angles and reaches 96% percent as compared with another work. The error is too small according to the start and goal PSO generated points for each trajectory.*

Keywords- *Three-link, inverse kinematics, PSO.*

How to cite this article: H.Z. Khaleel, "Enhanced Solution of Inverse Kinematics for Redundant Robot Manipulator Using PSO," *Engineering and Technology Journal*, Vol. 37, Part A, No. 7, pp. 241-247, 2019.

1. Introduction

In modern life, robots are able to impact on many applications such as industrial manufacturing, healthcare transportation, and exploration of space and underwater. Robots can assist humans in their complex tasks [1,2]. Manipulator pointed to the robotic arm. Manipulator redundant robot is consisted of links connected by joints like a chain. Kinematics problem is represented by the manipulator motion without the perception of forces and torques [3]. In general, the inverse kinematics problem of the robot is to get the values of the joint positions given end-effector's position and orientation [1]. There are many artificial intelligence methods solved an inverse kinematics problem of manipulator robot. Some of them are: the numerical algorithm which is based on the fuzzy logic approach that is used for solving the inverse kinematic problem of a SCARA robot. This method is usually difficult, computationally expensive and always gives an approximate solution with large error especially in the corners of robot workspace. In general, one of the most drawbacks of fuzzy logic is that it requires high computation time [4]. The inverse kinematics problem is solved using conventional Genetic Algorithm GA and the Continuous GA (CGA) for robot manipulators. The number of generation convergence speed of the CGA and the average execution time is better than the conventional GA algorithm. The average execution times for these GA algorithms are too long [5]. Many researches have done using Neural Networks. One of the latest was done using back-propagation neural network algorithm of multi-neural networks and solved I.K. problem of the Reis Robot–RV12L robot manipulator. However, the design was too complex with eight large trained Neural Networks [6]. Genetic Algorithm (GA) and Neural Network (NN) approach solved the problem of inverse kinematics for the three-link redundant robot. The

http://dx.doi.org/10.30684/etj.37.7A.4
2412-0758/University of Technology-Iraq, Baghdad, Iraq

sine wave and the circle trajectories are also simulated in MATLAB program. The simulation of trajectories results has little accuracy and the generated circle trajectory is not smooth in all areas of trajectory [7]. Particle Swarm Optimization (PSO) is derived from the natural-inspired swarm behavior of some animals such as bird flocks and fish schoolings. It has many advantages such as simple implementation, the global optimal solution is founded with high probability and efficiency, fast converges, no overlap and mutate, the computational time is short. PSO is good for solving difficult problems in order to find accurate mathematical models [8]. In this paper, the inverse kinematics problem is solved and enhanced the trajectories of the three-link robot manipulator using PSO.

2. Forward Kinematics and Trajectory Equations of Redundant Robot Manipulator

In this section, the forward kinematics and trajectory equations are illustrated as in the following:

I. Forward Kinematics and Joints Limitations

The forward kinematics problem of the robot manipulator has evaluated the end-effector's position and orientation from the given joint angles values [4, 5]. In this paper, the three-link redundant robot manipulator is used as shown in Figure 1.

Figure 1 shows that the three lengths of robot manipulator (a_1, a_2, a_3) with three joint angles (TH_1, TH_2, TH_3). Three links are connected with three joints. Three-link redundant robot manipulator forward kinematics equations can be written as in Eqs. (1) and (2) [7,9].

$$x_E = a_1\cos(TH_1) + a_2\cos(TH_1 + TH_2) + a_3\cos(TH_1 + TH_2 + TH_3)$$
$$y_E = a_1\sin(TH_1) + a_2\sin(TH_1 + TH_2) + a_3\sin(TH_1 + TH_2 + TH_3)$$
(1)

Where (x_E, y_E) refers to the three-link robot end-effector's position. While (TH_E) refers to three-link robot end-effector's orientation as in Eq. (2):

$$TH_E = TH_1 + TH_2 + TH_3$$
(2)

The link parameter table of a three-link redundant robot manipulator is illustrated in Table 1.

The four parameters in Table I are called: a_i=link length, α_i=link twist, d_i=link offset, and θ_i=joint angle. The three joint angles are: $\theta_1^* = TH_1$, $\theta_2^* = TH_2$ and $\theta_3^* = TH_3$. The * symbol means that the angles are changed values with certain ranges. The three joints limitations ranges are ($0 < TH_1 < \pi$, $-\pi < TH_2 < 0$, and $-\pi/2 < TH_3 < \pi/2$).

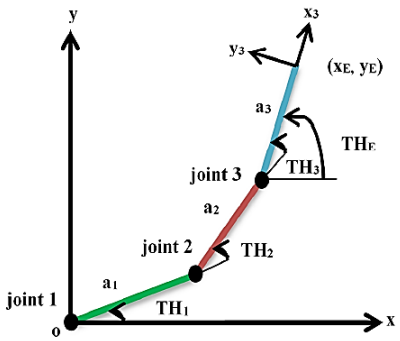

Figure 1: Three-link redundant robot manipulator [7]

Table 1: Link parameter table of the three-link redundant robot manipulator

Link no.	a_i (m)	α_i (deg.)	d_i (m)	θ_i (deg.)
1	2	0	0	θ_1^*
2	2	0	0	θ_2^*
3	2	0	0	θ_3^*

II. Trajectory Equations

Three-link robot redundant performs three trajectories. These trajectories Eq. (3-6) are illustrated as below:

The first circle trajectory equation is expressed as in Eq. (3) [7, 9]:

$$y_p = y_c + r\sin\varphi$$
(3)

Where (x_p, y_p) is the desired circle end-effector's position of the robot arm, (x_c, y_c) is circle's center position, r is the radius of the circle and the angle as (φ) is [0:2π]. The desired circle of end-effector's orientation is presented as in Eq. (4) [7, 9]:

$$TH_d = \tan^{-1}(y_p / x_p)$$
(4)

Parametric Cartesian space trajectory equations are used to express the square and the triangular trajectories of the three-link robot as shown in Figure 2.

These parametric Cartesian space trajectory eqs. (5) and (6) can be written as [10, 11]:

$$X(u) = X_a + u(X_b - X_a)$$
(5)
$$Y(u) = Y_a + u(Y_b - Y_a)$$
(6)

Where (X_a, Y_a): start point; (X_b, Y_b): endpoint (goal point), (u= 0:1). (X, Y): coordinates of end-effector's robot.

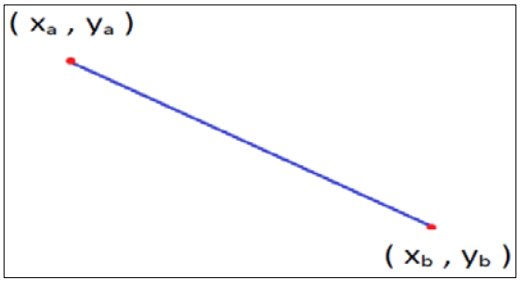

Figure 2: Trajectory line path [10, 11]

3. Solving Inverse Kinematics for Redundant Robot Manipulator using PSO

This section describes the introduction of PSO and the enhanced solving inverse kinematics for redundant robot manipulator using PSO.

I. Introduction of Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) was derived from the natural-inspired swarm behavior of some animals such as bird flocks and fish schoolings. Russell Eberhart and James Kennedy have invented the PSO algorithm. The natural behavior of swarms has been used in order to solve the optimization problem [9]. The basic idea of PSO is initialized by randomly generating a swarm of particles. Each particle (i) has position namely (x_i), with a certain velocity namely (v_i). Swarm of particles is executed a random iterative search in the space of possible n-dimensional solution vectors. This algorithm is related to the objective function value f = (x_i), where f is the function to be minimized. After iterations, the optimal solution is accomplished. At every iteration, the positions of particles are updated. They depend on two main elements: first is the best historical position direction achieved by each particle individually denotes the individual element (pbest). Second is the position achieved by any particle in the swarm, denotes the global element (gbest). In each iteration, these two elements are updated. The following two main Eqs. (7) and (8) of the PSO algorithm are:

$$v_{i,j}^{k+1} = v_{i,j}^{k} + c_1 r_1 \left(xbest_{i,j}^{k+1} - x_{i,j}^{k}\right) + c_2 r_2 (xgbest_j^{k} - x_{i,j}^{k}) \quad (7)$$

$$x_{i,j}^{k+1} = x_{i,j}^{k} + v_{i,j}^{k+1} \quad (8)$$

Where k: denotes the k^{th} generation

$x_{i,j}^{k}$: j^{th}Component of the i^{th}particle's position vector

$x_{i,j}^{k+1}$: j^{th}Component of the i^{th}new particle's position vector

$v_{i,j}^{k}$: j^{th}Component of the i^{th}particle's velocity vector

$v_{i,j}^{k+1}$: j^{th}Component of the i^{th}new particle's velocity vector

r_1, r_2: Random numbers values between 0 and 1

c_1, c_2: called cognition and social constants respectively.

The constants values range of (0, 2).

$xbest_i$: Best positions experienced of i[th] particle

$xgbest_j$: j^{th}Component of whole swarm [12, 13, 14, 15].

II. Enhanced Solving Inverse Kinematics for Redundant Robot Manipulator using PSO

This work presents the solutions of inverse kinematics without using the inverse kinematics equations. The inputs initializations are the randomly three joint angles within the three joints limitations ranges (section (2.1)) of the three-link robot arm. After applying the PSO algorithm procedure as mention in the previous section (3.1) with the fitness, function eq. (9). The position and velocity of particles (θ_1, θ_2, and $\theta 3$) are adapted depending to their velocity and positions equations (7, 8) within the joint angles limit. The outputs are three best joint angles. These three best joint angles are interring as inputs to the forward kinematics eqs. (1,2) in order to find the best end-effector positions and orientation. After that, the best end-effector positions and orientations are inputs to every three trajectories (circle, triangle, square) eqs. (3-6) in order to achieve the best three generated end-effector positions and joint angles as shown in Figure 3. Each joint angle variable of three-link redundant robot manipulator arm is expressed as a particle: $\theta_i = (\theta_1, \theta_2, \theta_3)$. Each particle has a velocity during the particle movement: $v_I = (v_{\theta 1}, v_{\theta 2}, v_{\theta 3})$. The aim of the PSO algorithm is to minimize the objective function by adjusting joint angles or (θ_i) and get the best solution of joint angles. The minimization of fitness function is represented as in eq. (9) [13]:

$$f(\theta) = ||(\theta_T - \theta_i)|| \quad (9)$$

Where θ_T is the target joint angles and θ_i is the estimated joint angles for i=1:3. If the min error fitness function is equal or less than 0.0001 and the iterations times for each trajectory is equal or less than the values in the below flowchart, then finish the algorithm or if it is larger than 0.0001 choose another joint angle within joints limitations. Ite1, Ite2, and Ite3 are the PSO

iteration times for circle, square and triangle trajectories individually as shown in Figure 3.

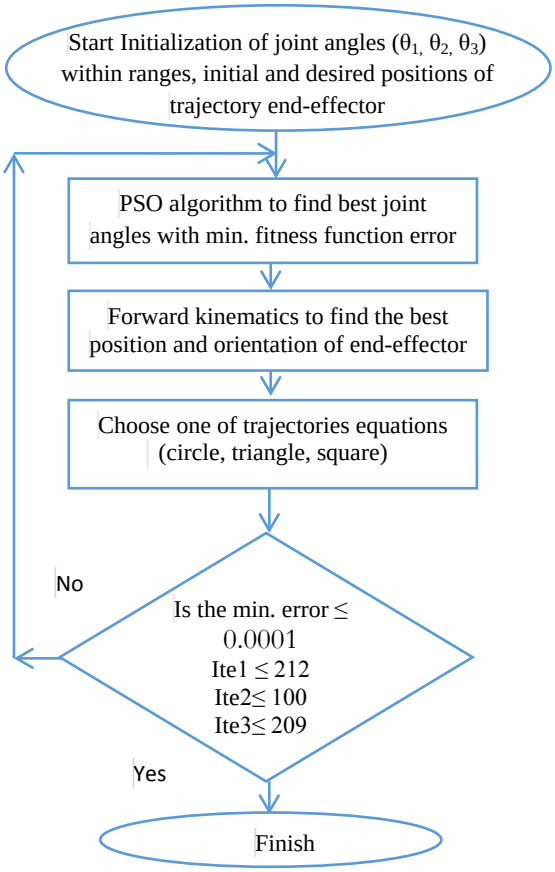

Figure 3: Flowchart of the enhanced work using PSO algorithm

4. Simulation Results

In this work, three trajectories (circle, triangle, and square) are simulated of the three-link robot manipulator. The circle trajectory is simulated according to the eq. (3) as in Figure 4.

The square and triangle trajectories are also simulated due to parametric Cartesian space trajectory eqs. (5) and (6) as shown in Figures 5 and 6.

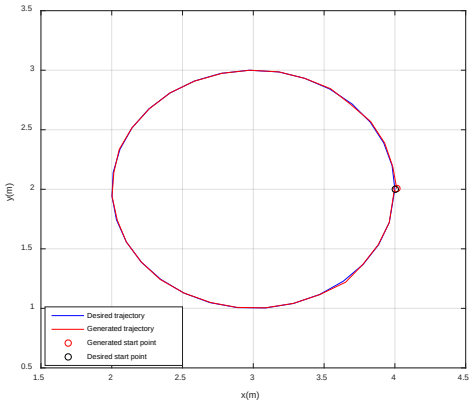

Figure 4: End-effector's three-link robot of two circle trajectories: generated with PSO and desired

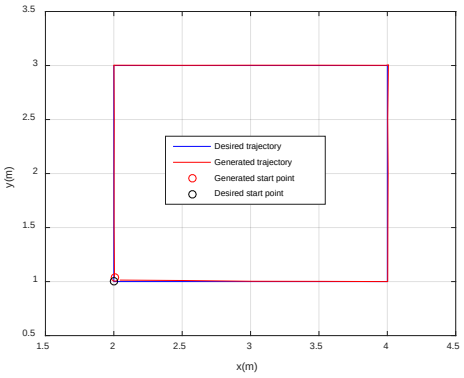

Figure 5: End-effector's three-link robot of two square trajectories: generated with PSO and desired

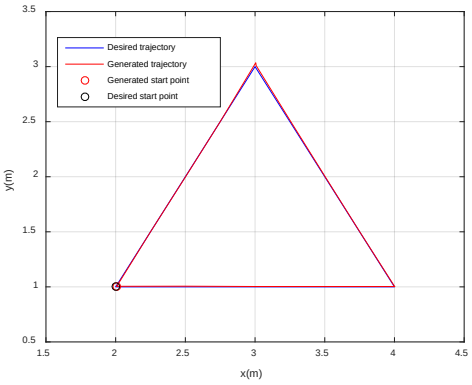

Figure 6: End-effector's three-link robot of two triangle trajectories: generated with PSO and desired

The three joint angles of the desired and the generated circle trajectories are simulated of the three-link robot for three trajectories (circle, triangle, and square) and all results of the generated three joint angles are close to the desired three joint angles as shown in Figures 7-9. Where in these Figures the x-axis presents the iterations and the y-axis marked as the angles in radian. TH1d, TH2d, and TH3d are the first, second and third desired joint angles respectively. While TH1g, TH2g, and TH3g are the first, second and third generated of joint angles. The

three trajectories (circle, triangle, and square) numbers of iterations are: (32, 20, and 15) respectively.

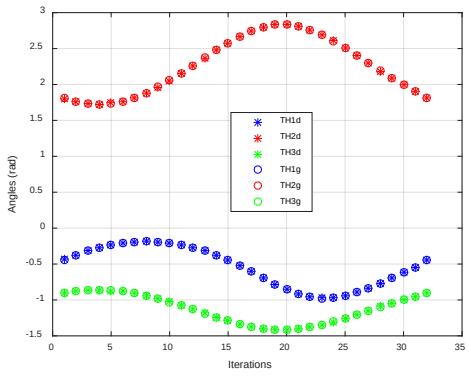

Figure 7: Joint angles of the generated and desired circle trajectory of the three-link robot

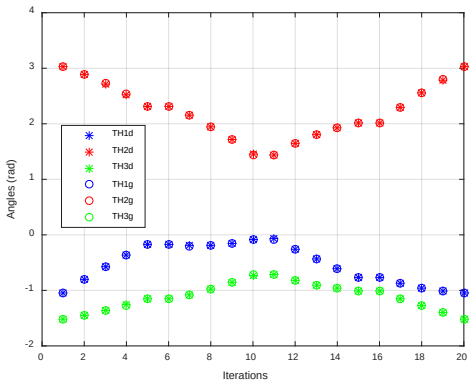

Figure 8: Joint angles of the generated and desired square trajectory of the three-link robot

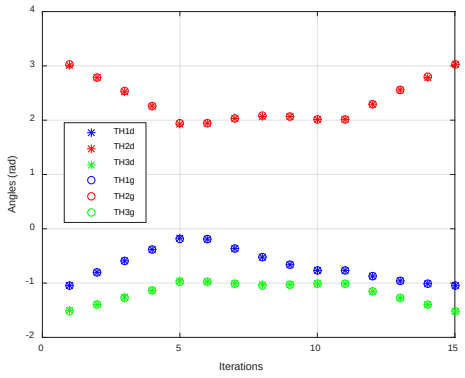

Figure 9: Joint angles of the generated and desired triangle trajectory of the three-link robot

PSO algorithm is implemented in MATLAB program R2015b. The number of variables equals to 3 according to three joint angles ($\theta_1, \theta_2, \theta_3$), the population size =100. Simulation of the PSO algorithm is performed in order to obtain the best joint angles ($\theta_1, \theta_2, \theta_3$) of three-link robot arm within their angles ranges as illustrated in previous sections. The minimum fitness functions of PSO results that are computed from eq. (9) of three trajectories are simulated as in the following Figures 10-12.

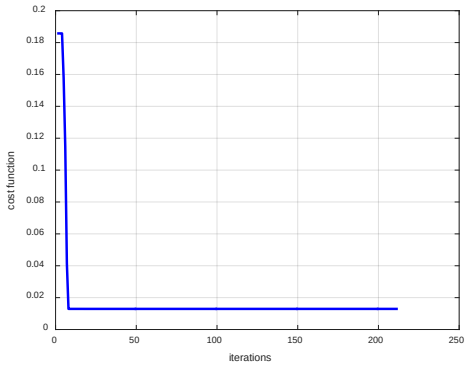

Figure 10: The fitness function of PSO generated circle trajectory

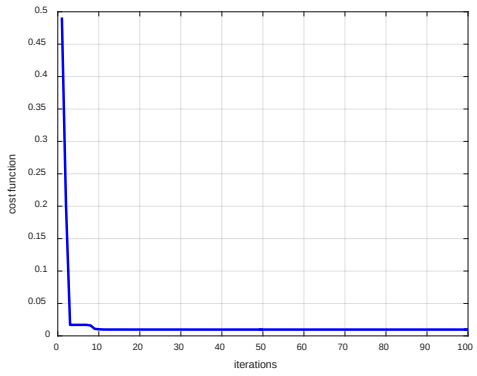

Figure 11: The fitness function of PSO generated square trajectory

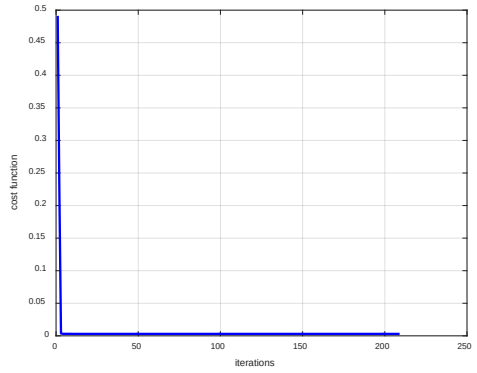

Figure 12: The fitness function of PSO generated triangle

The velocity is modulated and evaluated while the PSO is running as in equations (10, 11) below.
The time (t) is computed as:

$$t = \frac{\text{elapsed time}}{\text{no.of iterations}} \quad (10)$$

The velocity (vi) while PSO is running, is obtained from the following equation:

$$vi = \frac{\Delta\, \theta_i(j)}{t} \qquad (11)$$

Where $\Delta\, \theta_i(j)$: the change of joint angle. It is the difference between the next joint angles and old joint angles while i=1…3 and j=1: number of iterations. The three velocities for each three-link robot joint angle are simulated with iterations where, the first, second and third velocities marked as (v1, v2, v3) respectively. Three colors are consisting of three velocities: blue line denoted v1, red color marked v2 and green color labeled v3 as in the Figures 13-15.

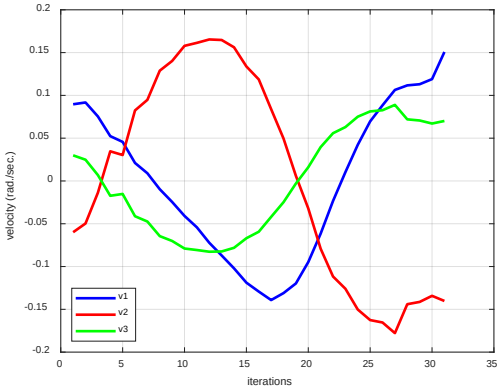

Figure 13: Velocities with iterations of circle trajectory

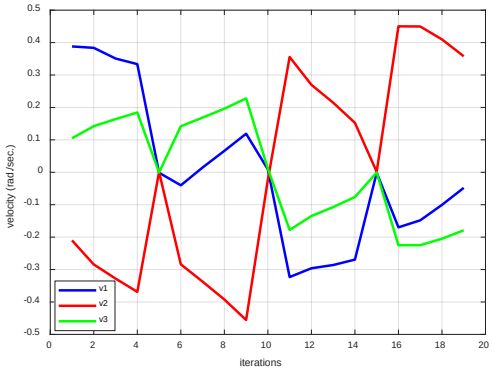

Figure 14: Velocities with iterations of square trajectory

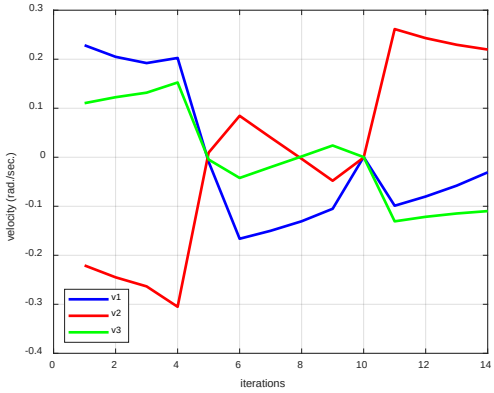

Figure 15: Velocities with iterations of triangle trajectory

Figures 13-15 show those three velocities for every three-link robot joint angle results of each trajectory that are depending on the eqs. (10, 11). These velocities results have been given an indication that the three-link robot is moving fast during the PSO algorithm. The main simulation results of all the enhanced work are illustrated as in Table 2. From all results of Table 2 below for minimum fitness functions errors and from Figures 10-12, the following can be summarized:

The best-generated end-effectors results of the three-link robot are obtained for three trajectories: circle, square and triangle. These results are too close to the desired trajectories as shown in Figures 4-6.

1- The best generated three joint angles ($\theta_1, \theta_2, \theta_3$) results of the three-link robot are evaluated for the three trajectories: circle, square and triangle. These angles are too near to the desired trajectories as in Figures 7-9.

2- The elapsed time is demonstrated in each trajectory and it is a too short time as in Table 2 results. This is due to the short elapsed time of the PSO algorithm implementation and is too fast.

3- The best generated results of three trajectories are enhanced as compared with another result that is presented by earlier studies as the following:

- The best-generated end-effector positions of circle trajectory are more accurate and smoother than another work in [7, 9] as shown in Figure 4.
- The simulation results of the best end-effector positions are more accurate and smoother than another work as in [16] for three trajectories: circle, square and triangle as shown in Figures 4-6. In the related work [7, 16] the generated circle, square and triangle trajectories are very rough trajectories in some areas of each trajectory in their simulation work.

Table 2: Work results of the three-link redundant robot manipulator

Trajectory	Desired start point end-effector's robot (m)	Generated start point with PSO end-effector's robot (m)	Desired goal point end-effector's robot (m)	Generated goal point with PSO end-effector's robot (m)	Min.fitness function error	Elapsed time (seconds)
Circle	(4,2)	(4.0131,2.0059)	(3.9999,1.9999)	(3.9957,1.9925)	0.0129	20.903981
Square	(2,1)	(2.005,1.0380)	(2,1)	(2.0435, 1.0138)	0.0097	11.747171
Triangle	(2,1)	(2.0080,1.0058)	(2,1)	(2.0042, 1.0050)	0.0030	15.729663

5. Conclusion

From the simulation results of this paper for the three-link robot manipulator, the following can be concluded:

1) The best three joint angles are computed using PSO algorithm within the limit joint angles without using inverse kinematics equations.
2) The best end-effector positions are evaluated.
3) The circle, square and triangle of PSO generated-trajectories are enhanced and are smoother than the trajectories of other related studies. In other related works, there are rough trajectories in some areas. This is the main novelty of this paper.
4) This PSO algorithm is too fast due to two reasons: the first is the short elapsed time in each trajectory and the second is the demonstration the velocities results for each three-link robot joint angle results for every trajectory.
5) The elapsed time of circle trajectory equals to 20.903981 seconds, the elapsed time of square trajectory equals to 11.747171 seconds and the elapsed time of triangle trajectory equals to 15.729663 seconds.
6) The minimum fitness function results are acceptable for three trajectories.
7) Three trajectories are optimum trajectories with respect to best end-effectors and best joint angles.

References

[1] S. Bruno, K. Oussama, "Springer Handbook of Robotics," Springer, 2nd edition, Cham, 2016.

[2] B. Mordechai, M. Francesco, "Elements of Robotics," Springer, Cham, 2018.

[3] W.S. Mark, H. Seth, and V. Mathukumalli, "Robot Modeling and Control," John Wiley & Sons, Inc., 1st edition, 2005.

A. Francesco, B. Alberto, A. Riccardo, and F. Rodolfo, "A Fuzzy Logic to Solve The Robotic Inverse Kinematic Problem," *Applied Mechanics and Materials*, Vol. 772, pp. 488-493, 2015.

[4] M. Shaher, S.A. Zaer, and M.A. Othman, "Solution of Inverse Kinematics Problem using Genetic Algorithms," *Applied Mathematics & Information Sciences*, Vol. 10, No. 1, pp. 225-233, 2016.

[5] A.R. Firas, R.K. Azad, and J.H. Amjad, "Inverse Kinematics Solution of Robot Manipulator End-Effector Position Using Multi-Neural Networks," *Eng. & Tech. Journal*, Vol. 34, No.7, pp. 1360-1368, 2016.

[6] Z.K. Hind, "Inverse Kinematics Solution for Redundant Robot Manipulator using Combination of GA and NN," *Al-Khwarizmi Engineering Journal*, Vol. 14, No. 1, pp. 136-144, 2018.

[7] A. Zeineb, G. Adel, B.B. Lazhar, H. Mohamed, and A.A.E. Nasser, "Review of Optimization Techniques Applied for The Integration of Distributed Generation From Renewable Energy Sources," *ELSEVIER, Renewable Energy*, Vol. 113, pp. 266-280, 2017.

[8] V.D. Adrian, "Neural Network based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm," ELSEVIER, The 7th Int. Conference on Interdisciplinarity in Engineering, 2014.

[9] B. Luigi, M. Claudio, "Trajectory Planning for Automatic Machines and Robots," Springer, 2008.

[10] J.H. Edwin, S. Gilbert, "Calculus Volume 3," OpenStax, 2016.

[11] S.Y. Xin, "Nature-Inspired Optimization Algorithms," ELSEVIER, 1st edition, 2014.

[12] R. Nizar, M.A. Adel, "Inverse Kinematics Using Particle Swarm Optimization, A Statistical Analysis," ELSEVIER, Int. Conference on Design and Manufacturing (IConDM), Vol. 64, pp. 1602-1611, 2013.

[13] Z. Panfeng, M. Xihui, M. Zhenshu, and D. Fengpo, "An Adaptive PSO-Based Method for Inverse Kinematics Analysis of Serial Manipulator," Int. Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), Chengdu, 2012.

[14] A. Kaveh, "Advances in Metaheuristic Algorithms for Optimal Design of Structures," chapter 2, Springer, 2014.

[15] V.D. Adrian, "ANFIS Based Solution to the Inverse Kinematics of a 3DOF Planar Manipulator," ELSEVIER, The 8th Int. Conference on Interdisciplinary in Engineering, pp. 526-533, 2014.