Apply Some Meta-Heuristic Algorithms to Solve Inverse Kinematic Problems of a 7-DoFs Manipulator Robot

Trung T. Nguyen[1], Tam N. Bui[1, 2], Watanabe. Dai [2], Tinh V. Nguyen [1], Linh N. Tao [1]
[1] School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam
[2] Shibaura Institute of Technology, Dept. of Machinery and Control Systems, Tokyo, Japan
Email: trung.nguyenthanh@hust.edu.vn

***Abstract*—In this research, Particle Swarm Optimization (PSO), Differential Evolution (DE) Algorithm and Searching Space improving DE & PSO Algorithm will be used for inverse kinematic solution of a 7-degree-of-freedom (DOF) serial manipulator. Firstly, the DH parameters of the robot manipulator are created, and transformation matrices are revealed. Afterward, the position equations are derived from these matrices. The end-effector position in the working space of the robotic manipulator is estimated using optimization algorithms. These algorithms were tested with two different end-effector motion scenarios. The first scenario uses 100 randomly selected points in the working space. The second scenario uses a spline trajectory including 100 points in the working space as well. According to the results, DE Algorithms has performed much more efficient than standard PSO Algorithms. The DE & PSO Algorithm using Searching Space Improvements can be used to optimize robots control easily.**

***Index Terms*—Differential Evolution (DE), Particle Swarm Optimization (PSO), Inverse Kinematic (IK) and Degree of Freedom (DOF)**

I. Introduction

Inverse kinematics (IK) is the use of kinematic equations to determine the joint variables of the robot to reach a desired position [1]. It is one of the most fundamental issues in robotics and it plays a very important role because it relates to other aspects such as trajectory planning, robot control and dynamic analysis [2]. However, it is much more complicated and time-consuming than forward kinematics because of non-linear equations. There are some methodologies to solve the IK problem for robots in general and the redundant robot in particular. For example, the geometric method is the method using geometric and trigonometric relationships [3], the iterative method often requires an inversion of a Jacobian matrix to solve the IK problem [4].In recent years, optimization techniques such as artificial neural network, Swarm intelligence algorithms and Evolutionary Computation algorithms have become extremely popular. Optimization algorithms have been applied [5] to solve the IK problem in the case of a single point or a whole endpoint trajectory. The simulation results show that the PSO and DE algorithm can effectively solve the IK problem. Laura et al. In [6] used DE algorithm for the IK problem of 7-DOF robot. This study has solved the IK problem to ensure the position and direction for the last stage. However, this study also only solves the single points. In [7] the authors used A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator in which QPSO is compared with PSO, ABC and FA. Most of these studies focus on the parameters of the optimal problem such as: distance errors, execution time, generations… There are not so many mentions about the variation in joint variable values over time. This can lead to disadvantages such as solutions are not feasible in reality and waste of search resources. The reason is that the joint variable values for continuous endpoints are dramatically changed. Therefore, in reality, the control process is very difficult to meet this requirement. The cause of this phenomenon is due to the multi solutions feature of the IK for redundant robots problem, while the optimal algorithm search is randomly generated in the robot workspace. In order to overcome these drawbacks, the authors of this research [8] proposed the Pro PSO algorithm improving PSO algorithm by limiting the initial searching space of joint variables. Pro DE and Pro PSO are applied and compared in solving IK problems for Scara RRRTR robots [9]. In [10], the authors focused on handling direct kinematics of a novel 3Prismatics-Rovolution-Revolution-Spherical type parallel manipulator with 6-DOFs. However, the author has not dealt with the problem of inverse kinetics for the model.

In this study, we will compare the results when apply DE, PSO as well as Pro DE, and Pro PSO to solve IK problems for a 7-DOF serial manipulator. The study solved the end-effector position when it moves along a spiral path created equidistantly or random points and made sure that the joint values do not change suddenly. The remainder of this paper is divided into the following Sections: Section 2 reviews the algorithms. Next, in Section 3 robot models used to test algorithms will be presented. Section 4 describes Scenarios to test the algorithms. The results after applying the algorithm are shown and compared in Section 5. Finally, conclusions and development directions are outlined in Section 6.

Manuscript received February 1, 2021; revised May 26, 2021.

doi: 10.18178/ijmerr.10.9.498-504

II. ALGORITHMS

A. PSO

Particle swarm optimization was developed by Kenney and Eberhart based on observing the moving characteristics of bird flock and fish school [11] [12]. In this algorithm the individual of the population is called particle. The particle of the population (Called swarm) can move within its space and offer a potential solution. Particles can memorize the best condition, find and exchange information to other members. Each particle in the population has two characteristics: position and velocity. Starting with the particle population, each particle monitors its coordinates, updates its position and speed according to the best solution for each iteration. The velocity and position values are shown in the following equation:

$$v_{id}(t+1) = wv_{id}(t) + c_1 rand[p_{id}(t) - x_{id}(t)] + c_2 rand[g_{id}(t) - x_i(t)]$$
$$x_{id}(t+1) = x_{id}(t) + v_{it}(t) \quad (1)$$

Where x_i, v_i are the position and velocity of the particle i-th respectively; d is number of dimension; w is the inertia weight factor, c_1 and c_2 are cognitive learning rate and social learning rate respectively; p_i is the pbest value of i-th particle; g_i is gbest value of the population

Color figures will be appearing only in online publication. All figures will be black and white graphs in print publication.

B. DE

Differential Evolution (DE) algorithm is initially developed by Storn and Price based on the theory of evolution. The DE algorithm applied the principle of mutation, crossover, and selection of human populations to solve problems. The core idea behind this algorithm is that the more fitness, the more changes to the surviving of individuals in the social collection. The generation will repeat until the terminal condition is met. A mutant vector is generated according to:

$$V = X_{best} + F(X_{r1} - X_{r2}) \quad \text{(DE/best/1)} \quad (2)$$

In Eqs (2), F is Scaling factor, r1,r2 is random solution, $r1, r2, r3, r4, r5 \in \{1, 2, 3, ..., Np\}$ and $r1 \neq r2 \neq i$, X_{best} is population filled with the best member.

After the mutation phase, the crossover operation is applied to each pair of the generated mutant vector V_i, and its corresponding target vector X_i for the purpose of generating a trial vector:

$$U_i = \begin{cases} V_i, \text{if } (\text{rand}[0,1] \leq CR \text{ or } (j=j_{rand}) \\ X_i, otherwirse \end{cases} \quad (3)$$

Where CR is a user-specified crossover constant in the range [0,1], j_{rand} is a randomly chosen integer in the range [1, n] to ensure that the trial vector U_i will differ from its corresponding target vector X_i by at least one parameter.

DE algorithm compares the trial vector U_i with the target vector $X_{i,G}$ of the current population according to Greedy Selection scheme which enables the vector with a better fitness objective function to become more favorable, then the one with better fitness function is allowed to enter the next generation. The Greedy selection scheme used is:

$$X_{i,G+1} = \begin{cases} V_{i,G} \text{ if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, \text{otherwirse} \end{cases} \quad (4)$$

C. Pro DE and Pro PSO

The disadvantage of many studies using optimization algorithms to solve the IK problem of redundant robots is to focus on the results related to the optimal running process such as execution time, number of generation ... but have not yet considered the feasibility of the found joint variable values. One of the workarounds to improve the continuity of these values is limiting the initialization domain of X. This helps the program to achieve the dual goals of increasing calculation speed, accuracy and ensuring the continuity of the value of joint variables. In this algorithm, firstly the robot moves from any position to the first point of the trajectory. With this first point, the initialization values for the particles are randomly selected in the full range of motion of joints. In addition, the target function in this case has the form:

$$Func.1 = a * \sqrt{\sum_{k=1}^{n} (q_i^k - q_0^k)} + b * \sqrt{(x_i - x_{ei})^2 + (y_i - y_{ei})^2 + (z_i - z_{ei})^2} \quad (5)$$

Where the values q_0^k *and* q_i^k (i=1) are the joint variable values at the original position and the 1st point on the trajectory respectively; (x_i, y_i, z_i) is the end-effector coordinates for the i-point (i=1) found by the algorithm, (x_{ei}, y_{ei}, z_{ei}) is the desired end-effector coordinates; a, b are coefficient penalties. Cost function as (5) ensures the energy spent on the joints to reach the 1st desired position is minimized. Besides, it also minimizes the distance error between the actual and desired end-effector position. The stopping condition of the trajectory points is that the Cost Func.1 value is less than the value of ε or the number of iterations reaches 500 and the number of times algorithm running <10.

After calculating the 1st point of the trajectory, the remaining points are calculated with a search limitation around the joint's variables found by its trajectory point before. By this limitation, the program's searching space will be limited while ensuring the continuity of the joint variables. In this case, the target function will still be the same as the function of the 1st point, but it has coefficient a = 0.

III. TESTING MODEL

The 6-DOF and 7-DOF robotic manipulators are widely used in current researches, because they are popular types in the field of robotics. These manipulators have many advantages, such as the ability of escaping easily from the obstacle, the flexible movement and having a larger working space. Despite all these advantages, their structures are extremely complex. In this study, a 7-DOF manipulator was used to compare the results of the algorithms. The simplified robot model was shown in the Fig. 1. D-H table parameters of robots are

given in Table I. The homogeneous transformation matrix can be used to obtain the forward kinematics of the robot manipulator, using the DH parameters in (6).

$$T_{i-1}^{i} = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_i \\ S\theta_i C\alpha_i & C\theta_i C\alpha_i & -S\alpha_i & -d_i S\alpha_i \\ S\theta_i S\alpha_i & S\theta_i S\alpha_i & -C\alpha_i & -d_i C\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

where S and C denote the sine and cosine functions.

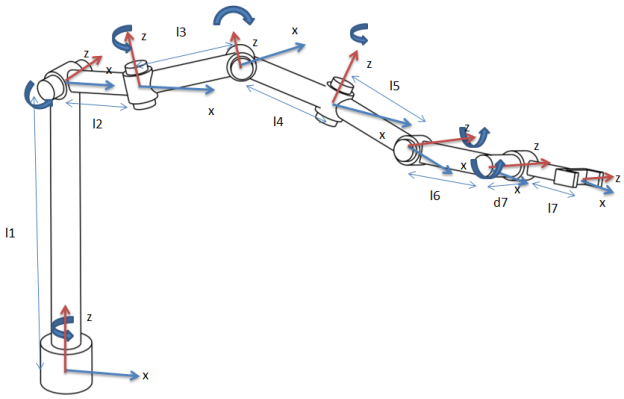


Figure 1. 7-DOF serial robotic manipulator.

TABLE I. D-H PARAMETER

θ(rad)	d(mm)	a(mm)	α(rad)
$-\pi < q_1 < \pi$	l1=500	0	- π/2
$-\pi/2 < q_2 < \pi/3$	0	l2=200	π/2
$-\pi < q_3 < \pi$	0	l3=250	- π/2
$-\pi/2 < q_4 < \pi/2$	0	l4=300	π/2
$-\pi/2 < q_5 < \pi/2$	0	l5=200	- π/2
$-\pi < q_6 < \pi$	0	l6=200	0
$-\pi/2 < q_7 < \pi/2$	d7=5	l7=100	0

The paragraph description the position and orientation of the end-effector can be determined by (7):

$$T_0^7 = T_0^1 * T_1^2 * T_2^3 * T_3^4 * T_4^5 * T_5^6 * T_6^7 \quad (7)$$

With:

$$T_0^1 = \begin{bmatrix} cq1 & 0 & -sq1 & 0 \\ sq1 & 0 & cq1 & 0 \\ 0 & -1 & 0 & l1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} cq2 & 0 & sq2 & l2cq2 \\ sq2 & 0 & -cq2 & l2sq2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^3 = \begin{bmatrix} cq3 & 0 & -sq3 & l3cq3 \\ sq3 & 0 & cq3 & l3sq3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^4 = \begin{bmatrix} cq4 & 0 & sq4 & l4cq4 \\ sq4 & 0 & -cq4 & l4sq4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^5 = \begin{bmatrix} cq5 & 0 & -sq5 & l5cq5 \\ sq5 & 0 & cq5 & l5sq5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$T_5^6 = \begin{bmatrix} cq6 & -sq6 & 0 & l6cq6 \\ sq6 & cq6 & 0 & l6sq6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

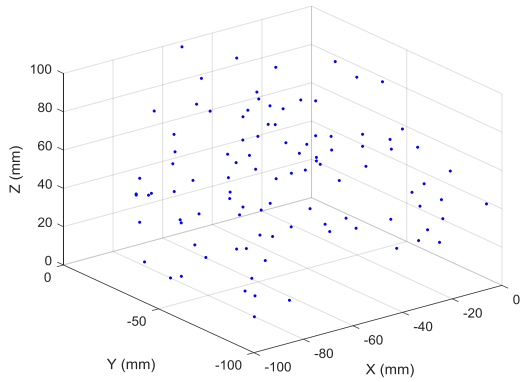
$$T_6^7 = \begin{bmatrix} cq7 & -sq7 & 0 & l7cq7 \\ sq7 & cq7 & 0 & l7sq7 \\ 0 & 0 & 1 & d7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

IV. SCENARIOS

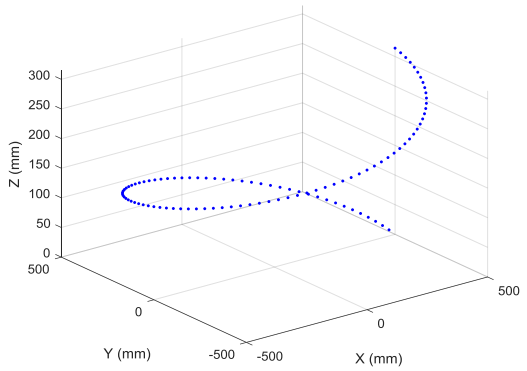
A. Scenario 1

For Scenario 1, robot is required to move the end-effector following 100 points selected randomly in working space (Fig. 2a) for the purpose of checking the effectiveness of each algorithm with distinct points. In this Scenario, the program will run in the following cases:

- Case 1.1: DE with the particle initialization domain is the RoM of joints.
- Case 1.2: PSO with the particle initialization domain is the RoM of joints



(a)



(b)

Figure 2. Testing Scenarios.
(a) Scenario 1: the randomly selected point; (b) Scenario 2: the defined spiral trajectory.

B. Scenario 2

In this Scenario, the robot is required to move the end-effector migration in turn over 100 points located in the workspace of the robot. The orbit is a Spline line described by the function:

$$\begin{aligned} z_e &= n\pi \\ x_e &= 500cos(2z/100) \\ y_e &= 500sin(2z/100) \end{aligned} \quad (9)$$

Where: (x_e, y_e, z_e) is the endpoint coordinates on trajectory, the program will run in the following cases:

- Case 2.1: DE with the particle initialization domain is the RoM of joints.
- Case 2.2: Pro DE with the particle initialization domain is around the previous joint variable.
- Case 2.3: PSO with the particle initialization domain is the RoM of joints.
- Case 2.4: Pro PSO with the particle initialization domain is around the previous joints' variable.

V. Comparative Results

A. Results of Algorithms in Scenario 1

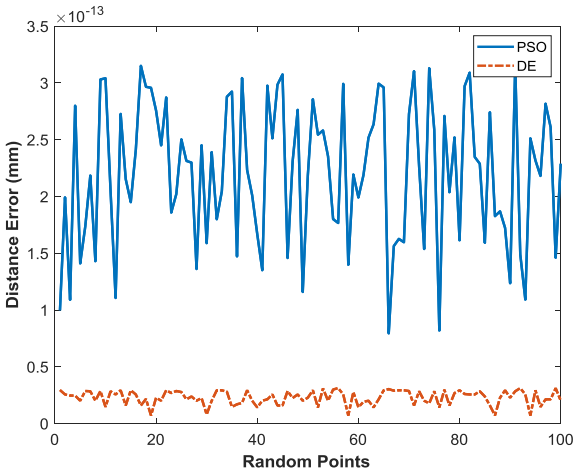


Figure 3. Distance error values in Scenario 1.

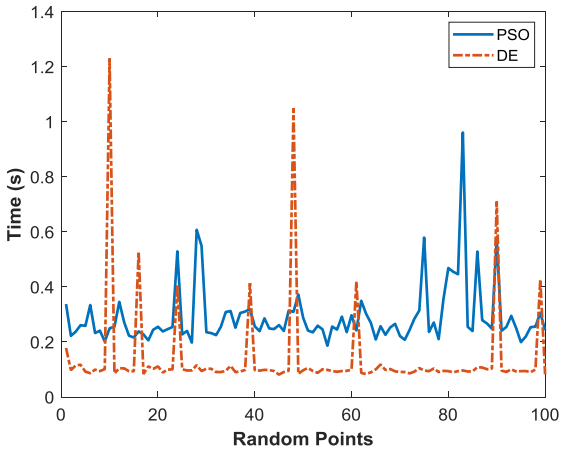


Figure 4. Optimization execution time obtained in Scenario 1

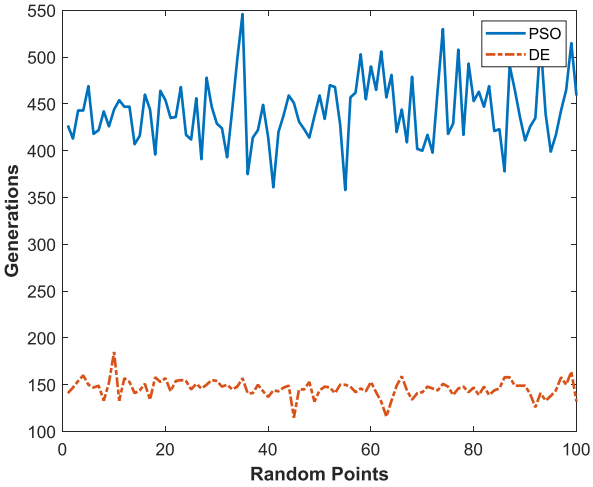


Figure 5. Optimization number of iterations in Scenario 1.

In the first Scenario, 100 points in the working space of the robot were chosen randomly and inputted to the inverse kinematics requirement. Results after using the algorithms were shown in Fig. 3, 4, 5 and Table II. From these we found that the results of PSO algorithm (case 1.2) is not as good as DE (case 1.1). Although the accuracy when applying the DE algorithm is 10 times higher than the PSO algorithm, specifically, average distance error between the reference and the simulated trajectories when applying the DE and PSO algorithm were about $2.3x10^{-14}$ (mm) and $2.2x10^{-13}$ (mm), respectively, the number of PSO generations is nearly three times more than DE. Table II and Fig. 5 clearly show this. The average number of generations for DE algorithm was around 146, while this number for PSO was 442. This also leads to an inevitable consequence of execution time when applying PSO algorithm to the inverse kinetics problem was longer than when applied using the DE algorithm. The average execution times were 0.1416 (s) and 0.2901 (s) for DE and PSO, respectively. In the summery, in the first Scenario, the DE algorithm has proven better than the PSO algorithm in terms of end effector accuracy, the execution time and needed generations. Next, the study will try in the case of the end effector follows a particular orbit in the working space.

TABLE II. Comparative Results of All Algorithms in Scenario 2.

	Case 2.1 (Pro DE)	Case 2.2 (DE)	Case 2.3 (Pro PSO)	Case 2.4 (PSO)
Avg. error	2.17e-14	2.2e-14	2.4e-13	2.33e-13
STD	6.23e-15	5.77e-15	5.42e-14	6.6413e-14
Avg. iteration	136	170	453	458
Avg. execution time	0.1136	0.1473	0.2537	0.3464

TABLE III. COMPARATIVE RESULTS OF TWO ALGORITHMS IN SCENARIO 1.

	Case 1.1 (DE)	Case 1.2 (PSO)
Avg. error	2.2e-14	2.2e-13
STD	6.29e^{-15}	6.26e^{-14}
Avg. iteration	146	442
Avg. execution time	0.1416	0.2901

B. *Results of Algorithms in Scenario 2*

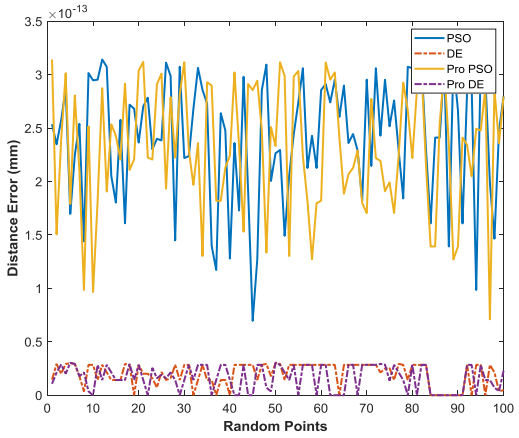


Figure 6. Distance error values in Scenario 2.

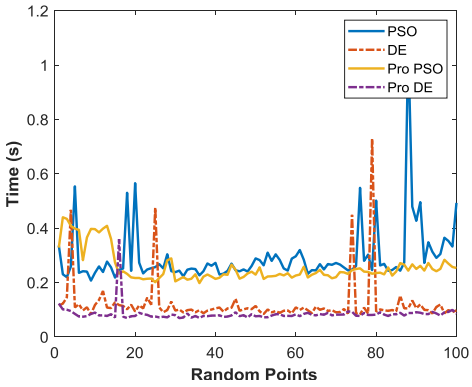


Figure 7. Optimization execution time obtained in Scenario 2

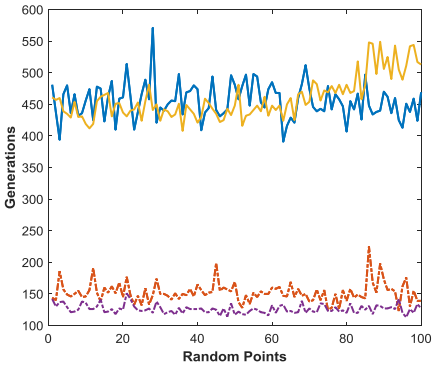


Figure 8 . Optimization number of iterations in Scenario 2.

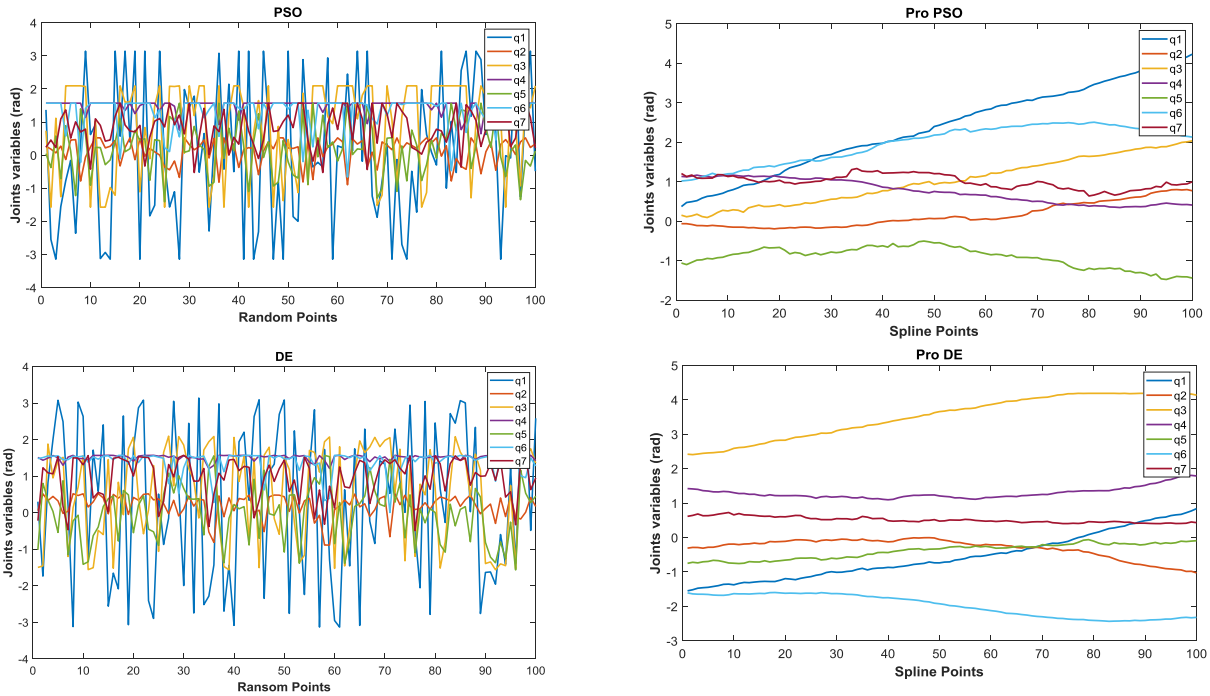


Figure 9. Joints variables in scenario 2

From Fig. 6, 7, 8 and Table III, it is noted that the distance errors between reference and simulated trajectories as well as the Standard Deviation of Errors (STD) of DE (case 2.1) and Pro De (case 2.2) with spiral trajectory for end effector are equivalent to about 10^{-14} (mm) and 6e-15 (mm) respectively, while that of PSO (case 2.3) and Pro PSO (case 2.4) are only about 10^{-13} (mm) and 6e-14 (mm) corresponding. Number of generations of Pro DE in order to get the optimum values is the best, the next is DE, Pro PSO and the final is with

PSO algorithm. It is also possible to see that the execution time of PSO algorithms is not better than the DE algorithms. Specifically, according to the Table III, the average execution time of Pro DE is 0.11s, Pro De is 0.15s, Pro PSO is 0.25s, PSO is 0.34s. The stability of proposed Pro DE & Pro PSO Algorithms is better than DE and PSO Algorithms. Its execution time is unstable and varies a lot. Fig. 9 shows the quality of inverse kinematics solutions. In addition to improving the quality of the optimization algorithm in terms of execution time, accuracy, number of generations, the joints value solutions in Pro DE and Pro PSO are very smooth. They are not as variable as the solutions gotten by DE and PSO algorithms. The Fig. 9 clearly shows this issue. The values of solutions when applying the Pro DE and Pro PSO algorithms as shown in this figure are of great importance in the next stages of the robot design and manufacturing process such as: dynamics problems, simulation problems and controller development, etc.

VI. Conclusion

In this study, inverse kinematics calculation of 7-degree of freedom serial robot was solved by using DE, PSO and the proposed algorithm (Pro DE, Pro PSO). The optimizing algorithms can handle the complexity and the difficulty of the inverse kinematics process. Experiments have been carried out in two different scenarios. In the first scenario, the results were obtained by predicting a single point manually determined have verified by the second scenario and the stability of the algorithm has demonstrated. The tests show that DE and Pro DE gave better results in terms of execution time, distance error and number of generations. Pro DE and Pro PSO algorithms result stable joint variables and continuously improve the execution time to DE and PSO algorithm. Pro DE and Pro PSO can be easily used for the inverse kinematics solution and optimizing robots' control of the developed manipulator.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Trung T. Nguyen modeled the robot system, calculated the forward kinematic and did programing; Tam N. Bui programed PSO, DE algorithms; Linh N. Tao and Tinh V. Nguyen programed Pro PSO, Pro DE algorithms; Huy V. Nguyen and Huong X. Nguyen set up and collected results; Watanabe Dai analyzed the experiment results; all authors contributed to write the manuscript.

References

[1] Köker R, Öz C, Çakar T, Ekiz H (2004) A study of neural network based inverse kinematics solution for a three-joint robot. *Rob Auton Syst* 49:227–234

[2] Küçük S, Bingül Z (2014) Inverse kinematics solutions for industrial robot manipulators with offset wrists Appl Math Model 38:1983–1999

[3] Lee CSG, Ziegler M (1984) Geometric approach in solving inverse kinematics of PUMA robots. *IEEE Trans Aerosp Electron Syst* 6:695–706

[4] Manocha D, Canny JF (1994) Efficient inverse kinematics for general 6R manipulators. *IEEE Trans Robot Autom* 10:648–657

[5] Carlos Lopez-Franco, Jesus Hernandez-Barragan, Alma Y. Alanis, Nancy Arana-Daniel. A soft computing approach for inverse kinematics of robot manipulators. *Engineering Applications of Artificial Intelligence*, 74, 104- 120, 2018701

[6] Laura Cecilia Antonio-Gopar,Carlos Lopez-Franco∗, Nancy Arana-Daniel, Eric, Gonzalez-Vallejo and Alma Y. Alanis, Inverse Kinematics for a Manipulator Robot based on Differential Evolution Algorithm, *IEEE Latin American Conference on Computational Intelligence* (LACCI), 2018

[7] Dereli, S., Köker, R. A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. *Artif Intell Rev* 53, 949–964

[8] Thanh Trung Nguyen, Ngoc Linh Tao, Van Tinh Nguyen, Ngoc Tam Bui, Van Huy Nguyen, Dai Watanabe, Apply PSO Algorithm with Searching Space Improvements on a 5 Degrees of Freedom Robot, The 3rd International Conference on Intelligent Robotics and Control Engineering (IRCE 2020)

[9] Thanh Trung Nguyen, Van Huy Nguyen, Xuan Huong Nguyen, Comparing the Results of Applying DE, PSO and Proposed Pro DE, Pro PSO Algorithms for the 5-DOF Scara Robot Inverse Kinematics Problem, The 2020 International Conference on Advanced Mechatronic Systems (ICAMechS 2020)

[10] Zhumadil Zh. Baigunchekov and Rustem A. Kaiyrov, "Direct Kinematics of a 3-PRRS Type Parallel Manipulator," International Journal of Mechanical Engineering and Robotics Research, Vol. 9, No. 7, pp. 967-972, July 2020. DOI: 10.18178/ijmerr.9.7.967-972

[11] Kennedy J, Eberhart R (1995) Particle swarm optimization, IEEE international conference on neural networks, pp 1943–1948

[12] Eberhart R, Kennedy J (1999) A new optimizer using particle swarm theory, The sixth international symposium on micro machine and human science, pp 39–4

Trung T. Nguyen was born in Vietnam on September 20th 1985. Nguyen graduated as a mechanical engineer at Hanoi University of Science and Technology, Vietnam in 2008. After that, he studied and completed the master and doctor degrees at Shibaura Institute of Technology, Japan in 2015. His field of research in master and doctoral course was Bio-science and Technology.
From 2008 he was been recruited to be a lecturer at department of Manufacturing Technology, School of mechanical engineering, Hanoi University of Science and Technology. His interest is Biomechanics in rehabilitation, model, simulate and optimize the mechanisms and artificial intelligence. Dr. Nguyen has published papers in some journals and paper conferences in some reputation organization such as: The Japan Society of Mechanical Engineers (JSME), IEEE, KICS…

Ngoc-Tam Bui received the B.E (2008) degree in Mechanical Engineering from Hanoi University of Science and Technology (HUST). He received M.E (2011) and Ph.D. (2015) in the field of optimal system design at Shibaura Institute of Technology. He now is an assistant professor at Shibaura Institute of Technology, Japan. Besides, he works as a lecturer at School of Mechanical Engineering of HUST. His research interests include optimization system design, topology optimization, reliability analysis.

Dai WATANABE Associate Professor, Department of Machinery and Control Systems, College of Systems Engineering and Science. His research interests include Simulation, Finite Element Method, Model Validation, Damage, Dynamic Response.

Van-Tinh Nguyen received the B.E. (2012) from Hanoi University of Science and Technology (HUST), Vietnam, and M.Sc. (2016) from Shibaura Institute of Technology (SIT), Japan. Besides, he works as a lecturer at School of Mechanical Engineering of HUST. Currently, he is pursuing Dr. Eng. degree in Functional Control Systems at SIT. His research interests include optimization system design, multi-body systems, humanoid robot, and evolutionary algorithm.

Ngoc-Linh Tao received his B.E. (2010) at Hanoi University of Science and Technology, Vietnam, and M.E. (2013) from Taiwan University of Science and Technology, Taiwan. He graduated PhD in Functional Control System at Shibaura Institute of Technology, Japan. His research interests include 3D computer vision, intelligent algorithm, autonomous vehicles localization and tracking.