Chapter

Use Improved Differential Evolution Algorithms to Handle the Inverse Kinetics Problem for Robots with Residual Degrees of Freedom

Trung Nguyen and Tam Bui

Abstract

In this study, the Self-adaptive strategy algorithm for controlling parameters in Differential Evolution algorithm (ISADE) improved from the Differential Evolution (DE) algorithm, as well as the upgraded version of the algorithms has been applied to solve the Inverse Kinetics (IK) problem for the redundant robot with 7 Degree of Freedom (DoF). The results were compared with 4 other algorithms of DE and Particle Swarm Optimization (PSO) as well as Pro-DE and Pro-PSO algorithms. These algorithms are tested in three different Scenarios for the motion trajectory of the end effector of in the workspace. In the first scenario, the IK results for a single point were obtained. 100 points randomly generated in the robot's workspace was input parameters for Scenario 2, while Scenario 3 used 100 points located on a spline in the robot workspace. The algorithms were compared with each other based on the following criteria: execution time, endpoint distance error, number of generations required and especially quality of the joints' variable found. The comparison results showed 2 main points: firstly, the ISADE algorithm gave much better results than the other DE and PSO algorithms based on the criteria of execution time, endpoint accuracy and generation number required. The second point is that when applying Pro-ISADE, Pro-DE and Pro-PSO algorithms, in addition to the ability to significantly improve the above parameters compared to the ISADE, DE and PSO algorithms, it also ensures the quality of solved joints' values.

Keywords: differential evolution (DE), particle swarm optimization (PSO), inverse kinematic (IK), degree of freedom (DOF), optimization

1. Introduction

The robot Inverse Kinematics problem involves finding the joints' variable values that match input parameters of position and direction of the end effector [1]. These matched variable values will ensure that subsequent robot control will follow the desired trajectory. This is one of the important issues in the robotic field because it is related to other aspects such as motion planning, dynamic analysis and control [2]. Traditionally, there are several methods to resolve inverse kinematics problem

for robots such as: geometry method is the method using geometric and trigonometric relationships to solve; the iterative method is often required inversion of a Jacobian matrix, etc. However, when applying these methods to solve the IK problem for robots, especially with redundant robots, it is often much more complicated and time-consuming. The reason is the nonlinearity of the formulas and the geometry between the workspace and the joint space. In addition, the difficult point is in the singularity, the multiple solutions of these formulas as well as the necessary variation of the formulas corresponding to the changes of different robot structures [3–5].

In addition to those existing methods of solving the IK problems, in recent years, the application of meta-heuristic optimization algorithms has become increasingly common. 8 optimization algorithms applied in [5] in the cases of a single point or a whole trajectory endpoint. The simulation results showed that the PSO algorithm can effectively solve the IK problem. In [6] the authors used algorithms such as ABC, PSO, and FA to solve the inverse kinematic requirement for Kawasaki RS06L 6-DoF robot in the task of picking and place objects. Ayyıldız et al. compared the results of all IK tests for a 4-DOF serial robot using 4 different algorithms: PSO, QPSO, GA and GSA [7]. Two versions of the PSO algorithm have been used to solve the IK problem for robots with a number of degrees of freedom from 9 to 180 [8]. In recent research [9], Malek et al. used PSO algorithm to handle inverse kinematics for a 7-DoF robot arm manipulator. The study mentioned both the requirements for the location and the direction of the endpoint, however, it only solved for 2 different end effector positions. Laura et al., in [10] used DE algorithm for the IK problem of 7-DoF robot. The problem was solved for specific points, but the quality evaluation parameters such as endpoint position deviation, execution time as well as the values of the joints' variable did not reach impressive quality. Ahmed El-Sherbiny et al. [11] proposed to use ABC variant algorithm for solving inverse kinematics problem in 5 DoFs robot arm. Serkan Dereli et al. [12] used a quantum behave partial algorithm (QPSO) for a 7-DoF serial manipulator and compare the results with other techniques such as firefly algorithm (FA), PSO and ABC.

In this study, the self-adaptive control parameters in Differential Evolution (ISADE) algorithm, that developed [13, 14] by authors, was applied to solve the problem of inverse kinematic for a 7-DOF serial robot. To compare the results, this IK problem was also handled by applying DE and PSO algorithms. In addition, the study also compared the results in the application of the above algorithms with the search space improvement of joints' variables (Pro-ISADE, Pro-PSO and Pro-DE) [15].

The remainder of the paper is divided into the following sections: Section II describes the experimental model. The theory of the PSO, DE and ISADE algorithms as well as the algorithms with improved search area, Pro-PSO, Pro-DE and Pro-ISADE, will then be presented in Section III. Section IV covers scenarios and object functions that will be applied to calculate the IK. The results after applying the algorithm are shown and compared in Section V. Finally, the conclusions are outlined in Section VI.

2. Testing model

The residual driven robots have many advantages such as easy escape from obstacles, flexible movement as well as a large operation space. However, their disadvantage is the complexity of the robot structure [16]. In this study, a serial redundant manipulator robot was used to evaluate the algorithm in resolving the inverse kinematics requirements. The simplified robot model was shown in the **Figure 1**. As in the figure, this serial robot manipulator is of type 7R (R: Revolute). The parameters of the D-H table of the robot are given in **Table 1**.

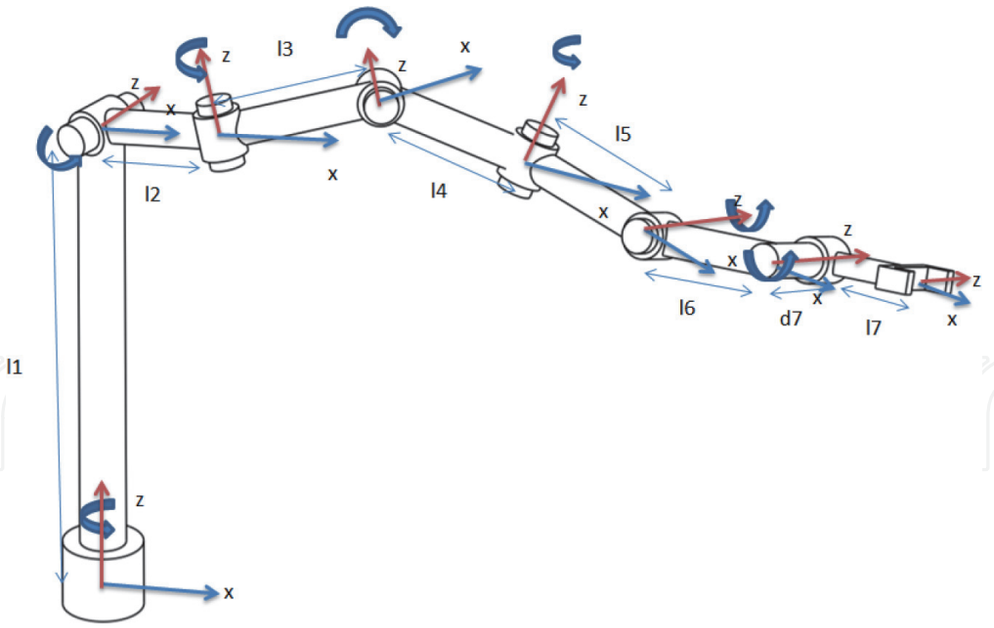


Figure 1.
The 7-DFO robot Scheme and coordinate systems used in the study.

Joint	θ(rad)	d(mm)	a(mm)	α(rad)
1	$-\pi < q_1 < \pi$	$d_1 = 500$	0	$-\pi/2$
2	$-\pi/2 < q_2 < \pi/6$	0	$l_2 = 200$	$\pi/2$
3	$-\pi/2 < q_3 < 2\pi/3$	0	$l_3 = 250$	$-\pi/2$
4	$-\pi/2 < q_4 < \pi/2$	0	$l_4 = 300$	$\pi/2$
5	$-\pi/2 < q_5 < \pi/2$	0	$l_5 = 200$	$-\pi/2$
6	$-\pi/2 < q_6 < \pi/2$	0	$l_6 = 200$	0
7	$-\pi/2 < q_7 < \pi/2$	$d_7=5$	$l_7 = 100$	0

Table 1.
D-H parameters.

The homogeneous transformation matrix can be used to obtain the forward kinematics of the robot manipulator, using the DH parameters in Eq. (1) [17].

$$T_{i-1i} = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_i \\ S\theta_i C\alpha_i & C\theta_i C\alpha_i & -S\alpha_i & -d_i S\alpha_i \\ S\theta_i S\alpha_i & S\theta_i S\alpha_i & -C\alpha_i & -d_i C\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where S and C denote the sine and cosine functions.
The position and orientation of the end-effector can be determined by Eq. (2):

$$T_{07} = T_{01} * T_{12} * T_{23} * T_{34} * T_{45} * T_{56} * T_{67} = \begin{bmatrix} n_x & s_x & a_x & x_5 \\ n_y & s_y & a_y & y_5 \\ n_z & s_z & a_z & z_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

With:

$$T_{01} = \begin{bmatrix} cq1 & 0 & -sq1 & 0 \\ sq1 & 0 & cq1 & 0 \\ 0 & -1 & 0 & l1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_{12} = \begin{bmatrix} cq2 & 0 & sq2 & l2cq2 \\ sq2 & 0 & -cq2 & l2sq2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$T_{23} = \begin{bmatrix} cq3 & 0 & -sq3 & l3cq3 \\ sq3 & 0 & cq3 & l3sq3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$T_{34} = \begin{bmatrix} cq4 & 0 & sq4 & l4cq4 \\ sq4 & 0 & -cq4 & l4sq4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$T_{45} = \begin{bmatrix} cq5 & 0 & -sq5 & l5cq5 \\ sq5 & 0 & cq5 & l5sq5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$T_{56} = \begin{bmatrix} cq6 & -sq6 & 0 & l6cq6 \\ sq6 & cq6 & 0 & l6sq6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$T_{67} = \begin{bmatrix} cq7 & -sq7 & 0 & l7cq7 \\ sq7 & cq7 & 0 & l7sq7 \\ 0 & 0 & 1 & d7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Where, T_{07} is matrix to produce a Catesian coordinate for any seven joint values. In the Eq. (10), $(x_{\mathrm{E}}, y_{\mathrm{E}}, z_{\mathrm{E}})$ denote the elements of position vector whereas, $(n_x, n_y, n_z, s_x, s_y, s_z, a_x, a_y, a_z)$ are the rotational elements of transformation matrix. In this study, only position vectors were used to calculate the distance error. After the computation, the end-effector coordinate in the manipulation space is determined by:

$$
\begin{aligned}
x_E = {} & d7sq5cq4sq1sq3 - cq1cq2cq3 + cq1sq2sq4 - cq5cq3sq1 + cq1cq2sq3 - l4cq4sq1sq3 \\
& - cq1cq2cq3 - l5sq5cq3sq1 + cq1cq2sq3 + l2cq1cq2 - l3sq1sq3 - l5cq5cq4sq1sq3 \\
& - cq1cq2cq3 + cq1sq2sq4 + l7cq7sq6sq4sq1sq3 - cq1cq2cq3 - cq1cq4sq2 \\
& - cq6cq5cq4sq1sq3 - cq1cq2cq3 + cq1sq2sq4 + sq5cq3sq1 + cq1cq2sq3 \\
& + l6sq6sq4sq1sq3 - cq1cq2cq3 - cq1cq4sq2 + l7sq7cq6sq4sq1sq3 - cq1cq2cq3 \\
& - cq1cq4sq2 + sq6cq5cq4sq1sq3 - cq1cq2cq3 + cq1sq2sq4 + sq5cq3sq1 \\
& + cq1cq2sq3 - l6cq6cq5cq4sq1sq3 - cq1cq2cq3 + cq1sq2sq4 + sq5cq3sq1 \\
& + cq1cq2sq3 + l3cq1cq2cq3 - l4cq1sq2sq4 \\
y_E = {} & l4cq4cq1sq3 + cq2cq3sq1 - d7sq5cq4cq1sq3 + cq2cq3sq1 - sq1sq2sq4 - cq5cq1cq3 \\
& - cq2sq1sq3 + l5sq5cq1cq3 - cq2sq1sq3 + l2cq2sq1 + l3cq1sq3 + l5cq5cq4cq1sq3 \\
& + cq2cq3sq1 - sq1sq2sq4 - l6sq6sq4cq1sq3 + cq2cq3sq1 + cq4sq1sq2 \\
& - l7cq7sq6sq4cq1sq3 + cq2cq3sq1 + cq4sq1sq2 - cq6cq5cq4cq1sq3 + cq2cq3sq1 \\
& - sq1sq2sq4 + sq5cq1cq3 - cq2sq1sq3 - l7sq7cq6sq4cq1sq3 + cq2cq3sq1 \\
& + cq4sq1sq2 + sq6cq5cq4cq1sq3 + cq2cq3sq1 - sq1sq2sq4 + sq5cq1cq3 - cq2sq1sq3 \\
& + l6cq6cq5cq4cq1sq3 + cq2cq3sq1 - sq1sq2sq4 + sq5cq1cq3 - cq2sq1sq3 \\
& + l3cq2cq3sq1 - l4sq1sq2sq4 \\
z_E = {} & l1 - l2sq2 + d7sq5cq2sq4 + cq3cq4sq2 + cq5sq2sq3 - l5cq5cq2sq4 + cq3cq4sq2 \\
& - l6sq6cq2cq4 - cq3sq2sq4 - l3cq3sq2 - l4cq2sq4 - l6cq6cq5cq2sq4 + cq3cq4sq2 \\
& - sq2sq3sq5 - l7cq7cq6cq5cq2sq4 + cq3cq4sq2 - sq2sq3sq5 + sq6cq2cq4 \\
& - cq3sq2sq4 + l7sq7sq6cq5cq2sq4 + cq3cq4sq2 - sq2sq3sq5 - cq6cq2cq4 \\
& - cq3sq2sq4 - l4cq3cq4sq2 + l5sq2sq3sq5
\end{aligned}
\tag{10}
$$

When solving the problem of inverse kinematics, with the endpoint coordinates as on the left side of Eq. (10), we need to find the values of the matching variable q. However, according to the Equation, the number of equations is much less than the number of variables. This makes it very difficult to find a unique and exact matching solution. In this study, ISADE algorithm and ISADE with searching space improvement algorithm (Pro-ISADE) were used to solve the IK problem for the robot. To compare the results, the study also used some other optimization algorithms such as PSO, DE as well as Pro-PSO and Pro-DE to solve the same IK problem for the robot above.

3. Applied algorithms and object functions

3.1 PSO

Particle swarm optimization was developed flying Kenney and Eberhart [18, 19] based on observing the moving characteristics of bird flock and fish school. In this algorithm the individual of the population is called particle. The particle of the population (Called swarm) can move in its space and offer a potential solution. Particles can memorize best condition and 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 and updates position and speed according to the best solution for each iteration. The velocity and position values are shown in the following equation:

$$
\begin{aligned}
& v_{id}(t+1) = \mathrm{w}v_{id}(t) + c_1 rand\left[p_{id}(t) - x_{id}(t)\right] + c_2 rand\left[g_{id}(\mathrm{t}) - \mathrm{x}_i(\mathrm{t})\right] \\
& \mathrm{x}_{id}(t+1) = x_{id}(t) + v_{it}(t)
\end{aligned}
\tag{11}
$$

In particular, x_{xi}, 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; pi is the *pbest* value of *i_th* particle; What is *gbest* value of the population.

3.2 DE

Differential Evolution (DE) algorithm is a population-based stochastic optimization algorithm recently introduced. DE works with two populations; old generation and new generation of the same population. The population is randomly initialized within the initial parameter bounds individuals in the population has two characteristics: position and velocity. Starting with the individual population, each individual monitors its coordinates and updates position and speed according to the best solution for each iteration. Velocity values (V) is randomly created in one of eight ways:

$$\begin{aligned}
&V = X_{r1} + F_w(X_{r2} - X_{r3})\\
&V = X_{best} + F_w(X_{r1} - X_{r2})\\
&V = X_{r1} + F_w(X_{r2} - X_{r3}) + F_w(X_{r4} - X_{r5})\\
&V = X_{best} + F_w(X_{r1} - X_{r2}) + F_w(X_{r3} - X_{r4})\\
&V = X + F_w(X_{r1} - X_{r2}) + F_w(X_{best} - X)\\
&V = X + F_w(X_{r1} - X_{r2}) + F_w(X_{r3} - X_{r4}) + F_w(X_{best} - X)\\
&V = X_{r1} + F_w(X_{r2} - X_{r3}) + F_w(X_{best} - X_{r1})\\
&V = X + F_w(X_{r2} - X_{r3}) + F_w(X_{r1} - X)
\end{aligned} \tag{12}$$

In particular, F is Scaling factor, r_1, r_2, r_3, r_4, r_5 is random solution, $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, 3, \ldots, N_p\}$ and $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq$ i, X_{best} is population filled with the best member.

Position values new (U) shown in the following Equation:

$$U = X. * FM_{mpo} + V. * FM_{mui} \tag{13}$$

FM_{mui} are all random numbers < 0.9, FM_{mpo} is inverse mask to FM_{mui}.

3.3 ISADE

In the [13, 14], we suggested to develop a new version of DE algorithm that can automatically adapt the learning strategies and the parameters settings during evolution. The main ideas of the ISADE algorithm are summarized below.

3.3.1 Mutation operator

ISADE probabilistically selects one out of several available learning strategies in the mutation operator for each individual in the current population. In this research, we select three learning strategies in the mutation operator as candidates: "DE/best/1/bin", "DE/best/2/bin" and "DE/rand to best/1/bin" that are respectively expressed as:

$$DE/best/1: \; V_{i,j}^G = X_{best,j}^G + F * \left(X_{r1,j}^G - X_{r2,j}^G\right) \tag{14}$$

$$DE/best/2 : V_{i,j}^G = X_{best,j}^G + F * \left(X_{r1,j}^G - X_{r2,j}^G\right) + F * \left(X_{r3,j}^G - X_{r4,j}^G\right) \quad (15)$$

$$DE/rand\,to\,best/1 : V_{i,j}^G = X_{r1,j}^G + F * \left(X_{best,j}^G - X_{r1,j}^G\right) + F * \left(X_{r2,j}^G - X_{r3,j}^G\right) \quad (16)$$

Where: $i = \{1, 2, \ldots, NP\}; j = \{1, \ldots, D\}$ are current population and design variable, respectively.

"DE/Randtobest/1/bin" strategy usually demonstrates good diversity while the *"DE/best/1/bin"* and *"DE/best/2/bin"* strategy show good convergence property, which we also observe in our trial experiments.

3.3.2 Adaptive scaling factor F and crossover control parameter CR

In the ISADE algorithm, the author suggested to use the sigmoid function to control neighborhood parameter. we sort the particles by estimating their fitness. A ranked particle is labeled with ranked number and assigned F that corresponds with its number. The formula for F by sigmoid function as following:

$$F_i = \frac{1}{1 + \exp\left(\alpha * \frac{i - \frac{NP}{2}}{NP}\right)} \quad (17)$$

Where: α, i denote the gain of the sigmoid function, particle of the i_{th} in NP, respectively.

For better performance of ISADE, the scale factor F should be high in the beginning to have much exploration and after curtain generation F needs to be small for proper exploitation. Thus, we proposed to calculate the F as follow:

$$F_{iter}^{mean} = F_{\min} + (F_{\max} - F_{\min})\left(\frac{iter_{\max} - iter}{iter_{\max}}\right)^{n_{iter}} \quad (18)$$

Where: $F_{max}, F_{min}, iter, iter_{max}$ *and* n_{iter} are the lower boundary condition of F, upper boundary condition of F, current generation, maximum generation and nonlinear modulation index, respectively.

The author introduced a novel approach of scale factor F_i of each particle with their fitness in Eq. (15). Thus, in one generation the value of $F_i^{iter} (i = 1, \ldots, NP)$ are not the same for all particles in the population rather they are changed in each generation. The final value of scale factor for each generation is calculated as follow:

$$F_{iter}^i = \frac{F_i - F_{iter}^{mean}}{2} \quad (19)$$

Where $iter = 1, \ldots, iter_{max}$ *and* $i = 1, \ldots, NP$

The control parameter CR is adapted as following:

$$CR_i^{G+1} = \begin{cases} rand_2 if\ rand_1 \le \tau \\ CR_i^G\ othewise \end{cases} \quad (20)$$

The ISADE algorithm was summarized as in the **Figure 2**.

3.4 Cost functions and Algorithms with searching space improvement

As mention in the introduction part, the disadvantage of many studies using optimization algorithms to solve the IK problem of redundant robots is to focus on

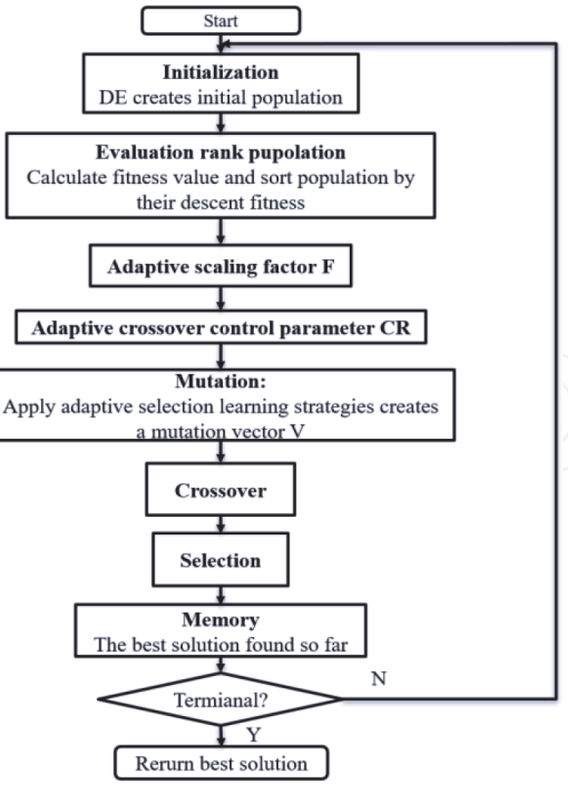


Figure 2.
ISADE Flowchart.

the results related to the optimal running process such as execution time, number of generation … but have not yet considered the feasibility of the joints' variable values. In order to overcome these drawbacks, the author of this research [15] proposed this algorithm that is explained as following: The solution to improve the continuity of joints' values constrains the initialization domain of X. This help the program to achieve the dual goal of increasing calculation speed, accuracy and ensuring continuity for the value of joints' variables. In this algorithm, firstly the robot from any position moves 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 (RoM) of joints. In addition, the target function in this case has the form:

$$Func.1 = a * \sqrt{\sum_{k=1}^{5} \left(q_i^k - q_{10}^k\right)} + b * \sqrt{\begin{array}{l}(x_i - x_{ei})^2 + (y_i - y_{ei})^2 + (z_i - z_{ei})^2 + (Rx_i - Rx_e)^2 + \\ (Rx - Rx_e)^2 + (Ry_i - Ry_e)^2 + (Rz_i - Rz_e)^2\end{array}} \quad (21)$$

where the values q_i^k *and* q_i^k (i = 1) are the joints' variable values at the original position and 1st point on the trajectory, respectively; (x_i, y_i, z_i) and (x_{ei}, y_{ei}, z_{ei}) are the End-effector coordinates for the i-point (i = 1) found by the algorithm and the desired End-effector coordinates; (Rx_i, Ry_i, Rz_i) and (Rx_{ei}, Ry_{ei}, Rz_{ei}) are corresponding rotation cosine angles performing orientation of the end-effector which are found by Algorithm and orientation of the desired end-effector; a, b are penalty coefficients. Cost function as Eq. (21) ensures the energy spent in 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 condition to stop for points of trajectory is that the Cost Func.1 value is less than value of e or the number of iterations reaches 600 and the number of times algorithm running <10.

After calculating for 1st point of the trajectory, the remaining points are calculated with a search limitation around the previous optimal joints' values. By using this suggested range, the program's search space will be limited while ensuring the continuity of the joint variables. In this case, the target function is still the same as the function of 1st point, but it has coefficient a = 0.

4. Scenarios

4.1 Scenario 1

In Scenario 1, an endpoint in the workspace were randomly selected; the PSO; DE and ISADE algorithms were then applied to solve the required problem. The purpose of this Scenario is to compare the convergence speed of the three algorithms. In this case, since the initial and the desired endpoints can be far apart, the Pro-PSO; Pro-DE and Pro-ISADE algorithms cannot be applied.

4.2 Scenario 2

In this case, the robot was required to move the endpoint through 100 points in the robot's working space one after another. These points were selected at random for the purpose of testing the effectiveness of each algorithm with many distinct points. Similar to the previous case, in this Scenario we also only applied the algorithms PSO, DE and ISADE with the solution space of the matching variable which limits the motion of these joints.

4.3 Scenario 3

The manipulator robot was required to move the end effector following a certain trajectory. The selected trajectory is spiral, and it is described by the following function:

$$\begin{cases} x_E = 200 * cos\,(2 * z_E/100) \\ y_E = 200 * sin\,(2 * z_E/100) \\ z_E = n * pi \end{cases} \quad (22)$$

Where: (x_E, y_E, z_E) is the desired endpoint coordinate on the trajectory. With 6 algorithms of PSO, Pro ISO, DE, Pro-DE and ISADE, Pro-ISADE, the comparison of the results on the same graph is not favorable. Therefore, the study divided this case into two smaller Scenarios:

- Scenario 3.1: Results when using ISADE algorithm comparing with results from PSO and DE algorithms.

- Scenario 3.2: Compare the results using Pro-ISADE algorithm with the results getting from Pro-PSO and Pro-DE algorithms

And then results from Scenario 3.1 were be compared with the results from Scenario 3.2.

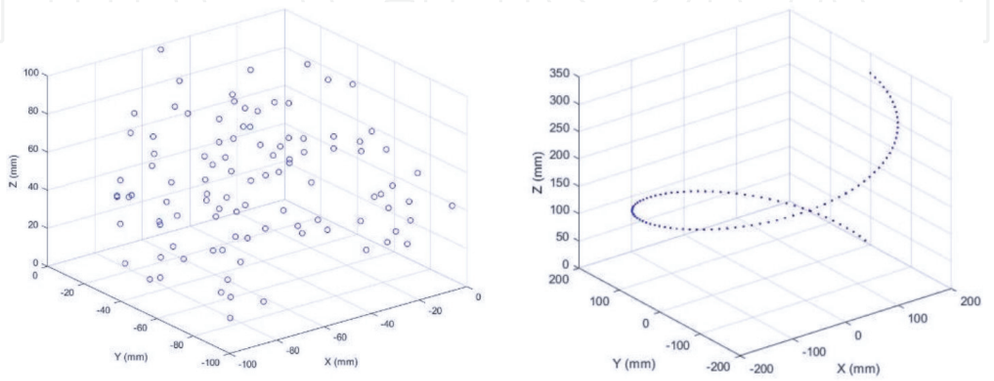
5. Simulation and results

5.1 Experimental setup

The main task of this study is to find the optimal value of the joints' variable to ensure the end effector of robots can reach the desired points. The desired point positions of the Scenario 2 and 3 are shown as the **Figure 3**. Research using the ISADE and Pro-ISADE algorithm, which were developed by the authors [13–15], to get simulation results of inverse kinematics problem and then compared it with the results when using PSO, DE and Pro-PSO, Pro-DE algorithms. When solving the IK problem for the 7-DoF serial robot manipulator, the study focused on three main aspects. The first of these is the sensitivity of the solution - in the other word, the amount distance error of end effector is minimum. The second criterion was the execution time. In order to avoid the endless loop, the maximum numbers of generation ($iter_{max}$)were set as 600, 600 and 130 for PSO (Pro-PSO), DE (Pro-DE) and ISADE (Pro-ISADE), respectively. And the final aspect is the searching space of joints' variables. Normally, Normally, almost all studies have been using the Range of Motion (RoM) of joints for its boundary space. Our algorithm [15] proposed to use the searching space of current generation is around previous optimal joints' values. In the **Table 2**, the ubs_{i+1} *and* lbs_{i+1} are the joints' upper and lower boundary of the current generation.C_1 *and* C_2 are weights of personal best and global best, respectively. w is the inertia weight. ρ is the number of run for each algorithm to choose the best result. Besides, after some trial runs for the algorithms, we noticed that our ISDE algorithm gave much better results than DE and the least was the PSO algorithm. Thus, when setting up the maximum distance error by the fitness value setting for the end effector position, the study set the value of $1e-14\ (m)$; $1e-15\ (m)$ and $1e-17\ (m)$ for PSO (Pro-PSO); DE (Pro-DE) and ISADE (Pro-ISADE), respectively or that can be seen in the **Table 2**. In this research, the proposed and other methods were tested in the two different Scenarios. Both the first and second Scenario was coded by Matlab version 2019a and run on the computer equipped with an Intel Core i5-4258U @2.4GHz processor and 8 GB Ram memory.

5.2 Scenario 1 results

After applying the inverse kinematic problem processing algorithms for a single endpoint, the results are shown in **Figures 4** and **5**. All algorithms are able to handle



a) b)

Figure 3.
Testing scenarios. (a) Scenario 2: 100 random points in workspace; (b) Scenario 3: 100 points on a spiral trajectory.

Algorithms	Max No. of Gen. max *iter*	Max Distance Err. (m)	Searching space $[\boldsymbol{ubs}_{i+1}\boldsymbol{lbs}_{i+1}]$	ρ	Mut. rate	Cross. rate	C_1	C_2	w	F_i
PSO	600	1e-14	*RoM*	10	**	**	1.5	1.5	$w = w_{start} + \left(\frac{iter}{iter_{max}}\right) * (w_{end} - w_{start})$	**
Pro-PSO	600	1e-14	$q_{oi} \pm \pi/100$		**	**				**
DE	600	1e-15	*RoM*		Scheme 2 in Eq. (10)	0.9	**	**	**	0.5
Pro-DE	600	1e-15	$q_{oi} \pm \pi/100$			0.9	**	**	**	0.5
ISADE	130	1e-17	*RoM*			*Eq.*(20)	**	**	**	*Eq.*(19)
Pro-ISADE	130	1e-17	$q_{oi} \pm \pi/100$				**	**	**	

Table 2.
Optimization parameters used in PSO, Pro-PSO, DE Pro-DE, and ISADE, Pro-ISADE.

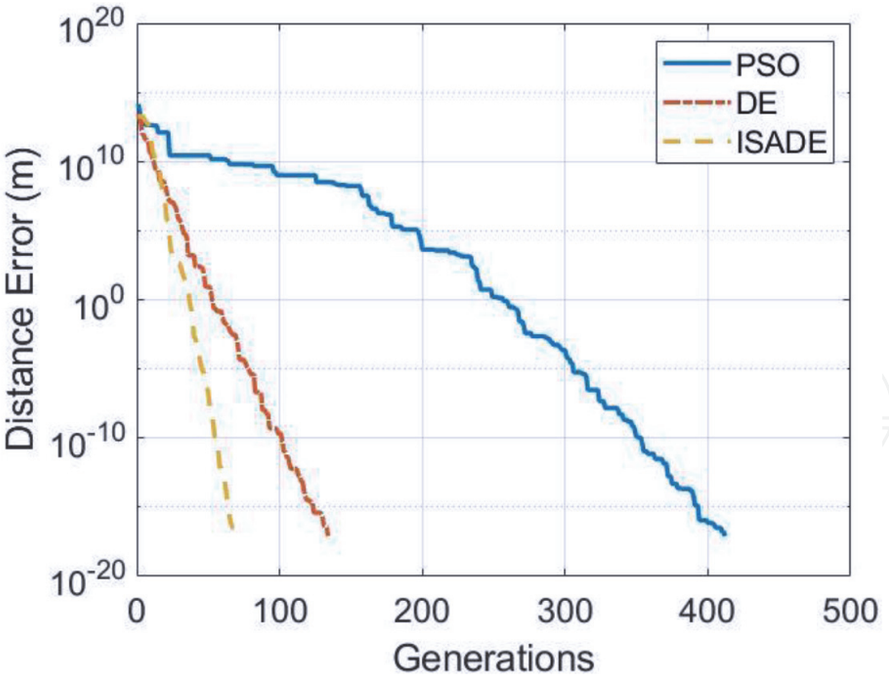


Figure 4.
End effector distance error vs. generations in Scenario 1.

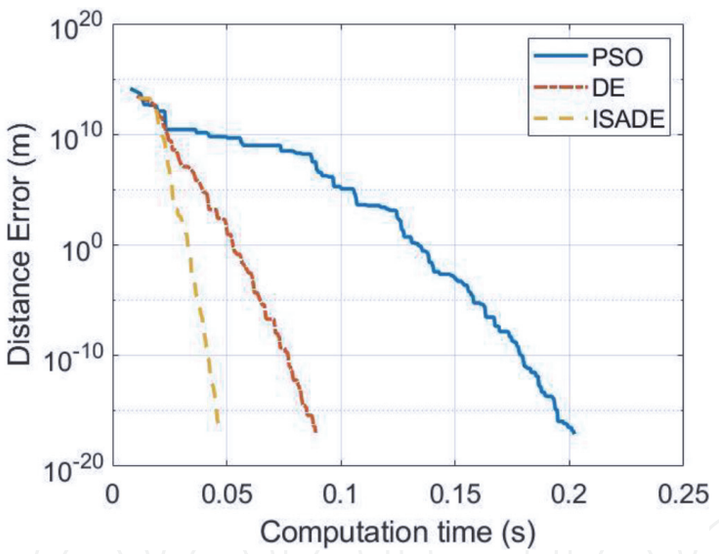


Figure 5.
End effector distance error vs. time in Scenario 1.

	Max. Iteration	Position error (m)	Calculation time (s)
PSO	85	2.6815e-04	0.0941
DE	85	5.7514e-10	0.0715
ISADE	85	2.8422e-13	0.0490

Table 3.
Comparison of ISADE with other algorithms.

the inverse kinetics problem, but the best results have been obtained with the ISADE algorithm as shown in **Table 3**.

Figures 4 and **5** show convergence speed of algorithms corresponding to the number of iterations and processing time, respectively. The results show that the

processing speed of the ISADE algorithm is the best, followed by the DE algorithm and finally with the PSO algorithm. In **Table 3** the study of selecting stop conditions for algorithms is the maximum number of iterations of 85 rounds. After 10 runs, the best results are shown in the table. The ISADE algorithm gives the best processing results in terms of both quality and speed. The endpoint deviation can reach 2.8422e-13 (m) in 0.049 (s) time. For the PSO algorithm, it can handle the reverse kinematic problem for the end point with an accuracy of 2.6815e-4 in a period of 0.0941 (s). and, 5.7514e-10 (m) and 0.0715 (s) are the accuracy of end effector and execution time for DE algorithm.

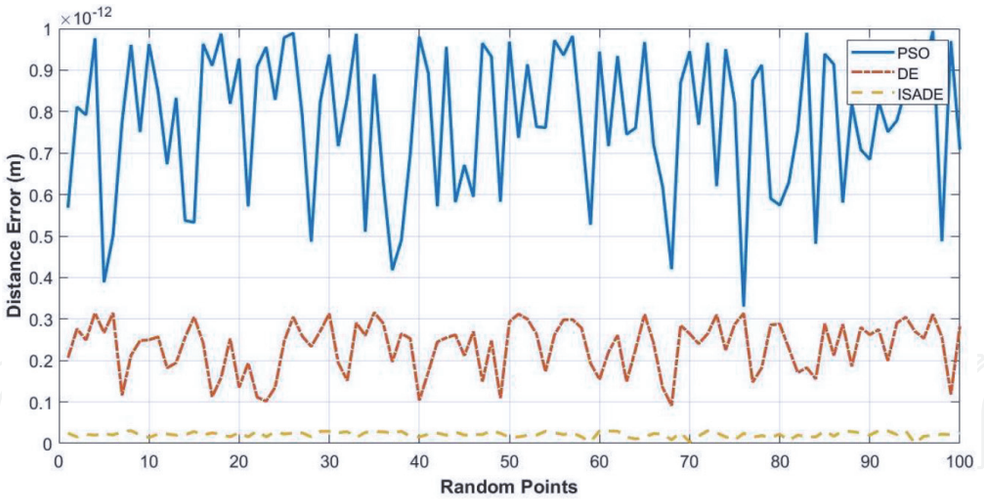
5.3 Scenario 2 results

As mentioned above, in this Scenario 2, algorithms was used to resolve inverse kinematics problem for 100 randomly chosen points within the workspace of the robot. When processed at each point, the end effector started at the same initial position of [0 0 0 0 0 0 0] for 7 serial joints values. Because the end effector points all come from the same starting point to go to each of the 100 points, the study only used the ISADE algorithm and compares with the results from PSO and DE algorithms without using the Pro-ISADE algorithm as well as Pro-DE and Pro-PSO.

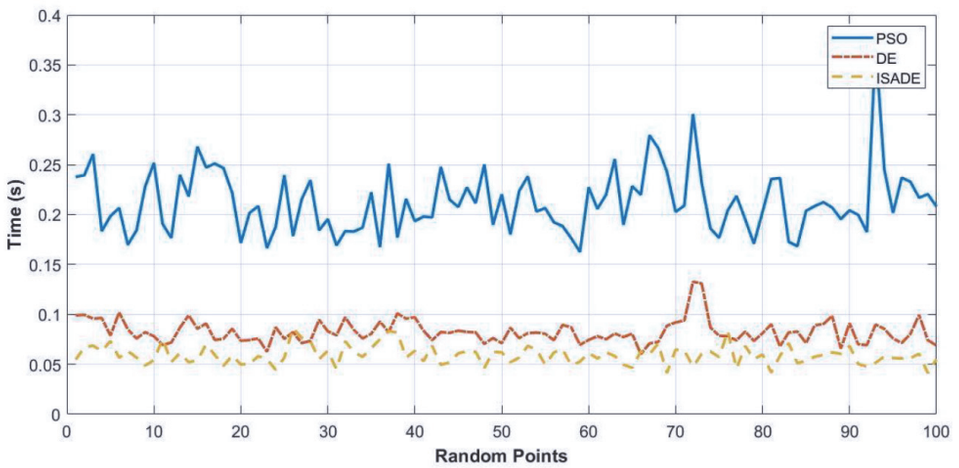
The 100 randomly selected points were shown in the **Figure 3a**. Results when applying ISADE and the other algorithm were presented in the **Figure 6**. As shown in the Figure, all algorithms have solved problem well. In particular, with the ISADE algorithm, although the fitness value in experimental setup required 1000 and 100 times higher than the required by applying the PSO and DE algorithms, respectively, it was not only guaranteed required precision but also showed faster processing speed and fewer iterations compared to the 2 other algorithms. Specifically, as shown in **Figure 6b** and **c** and especially **Table 4**, the average execution time when using ISADE to solve IK of each points was around 0.0685 second, while this value of the PSO and DE algorithm were on average 0.2307 (s) and 0.0978 (s) respectively. The main reason for this, as seen in **Figure 6b** and **Table 4**, was the number of generations to reach the optimal values much higher in PSO algorithm and slightly higher in DE algorithm, compared to in ISADE algorithm. Specifically, the PSO algorithm needed an average of 413.24 and the DE algorithm needed average of 124.45 loops to find a solution, while the ISADE algorithm used an average of 85.63 loops. Another remarkable thing is although there was not much difference in the number of iterations to solve the problem between the two algorithms DE and ISADE, but the ISADE algorithm still gave a processing speed of 1.42 times higher than DE algorithm though required 100 times more accuracy for the ISADE algorithm. This demonstrated the very high efficiency of the ISADE algorithm when it was applied to handle inverse kinematics problem for this robot. In short, in the optimization study for randomly chosen points in working space, the ISADE algorithm presented the best algorithm to resolve the IK requirement in term of accuracy, iteration and execution time.

5.4 Scenario 3 results

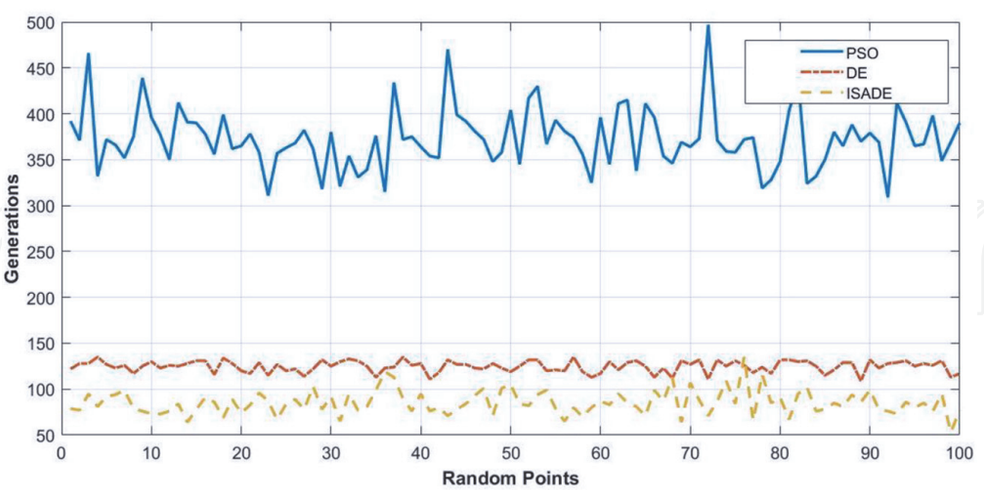
In Scenario 2, the end effector moved through the 100 points located on a specific trajectory that was defined in Eq. (22) and shown in **Figure 3b**. The main difference between Scenario 2 and Scenario 3 is that, instead of after solving each IK problem for each point, the end effector goes back to the original point to continue processing for the next points like in Scenario 2, in Scenario 3 the end effector starts from previous point in order to calculate for the next point. Stemming from this feature, the searching space of joints' variable also starts previous optimal joints' values. However, depending on the searching space we have 2 smaller cases such as:



a)



b)



c)

Figure 6.
Results for Scenario 2. (a) Distance error. (b) Execution time. (c) Number of generations.

- Scenario 3.1: Searching spaces for joints' variables are *RoMs*. Then, like the Scenario 2, the study compared the results when using the ISADE algorithm with the results when using the PSO and DE algorithms.

	PSO	DE	ISADE
Fitness value	1e-14	1e-15	1e-17
Avg. error	7.3016e-13	2.2938e-13	2.1644e-14
STD	2.0415e-13	5.991e-14	6.2125e-15
Avg. iteration	413.24	124.45	85.63
Avg. execution time	0.2307	0.0978	0.0685

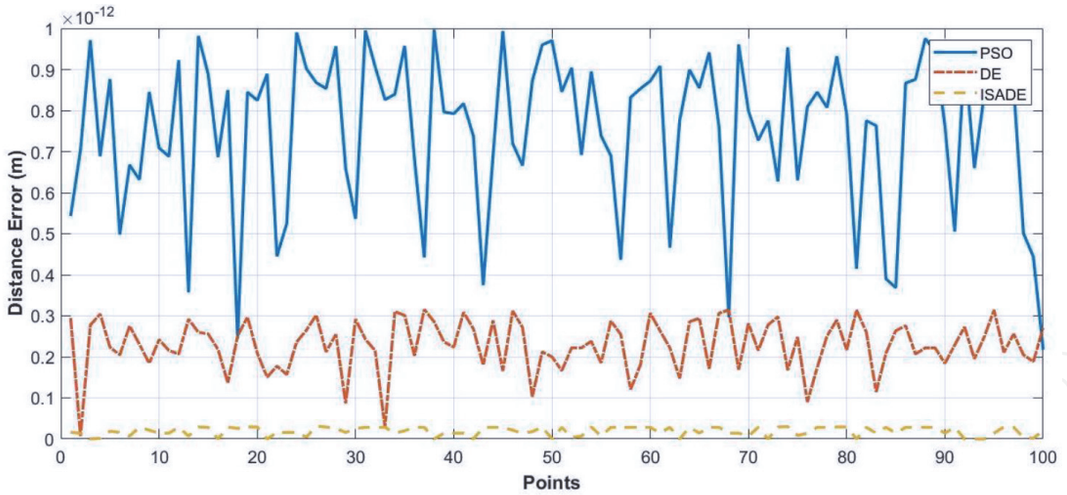
Table 4.
Comparative results in case 2.

- Scenario 3.2: Searching spaces for joints' variables are around the previous optimal joints' values. The study compared the results when using Pro-ISADE algorithm with when using Pro-PSO and Pro-DE algorithms.

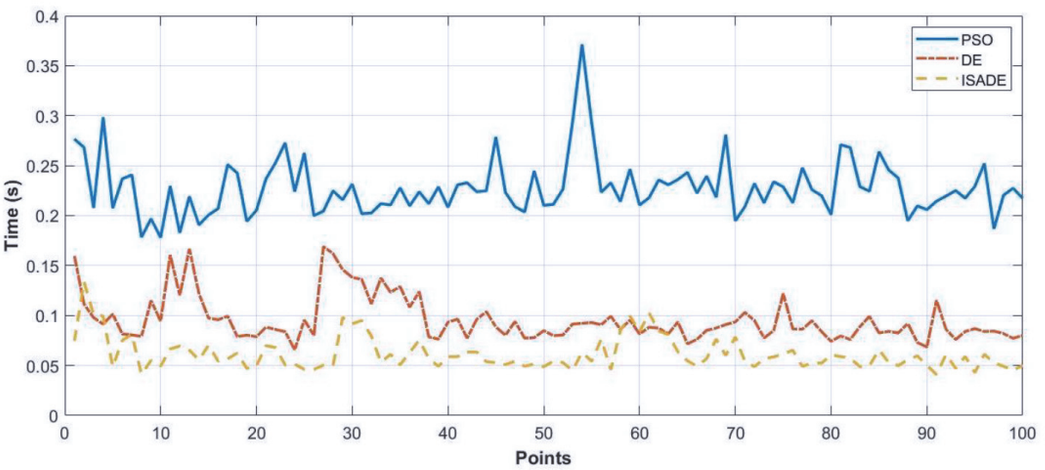
The results were presented in the **Figure 7** and **Table 5**. Similar to the Scenario 2, although the experimental installation required the ISADE (and Pro-ISADE) algorithm to be 100 and 1000 times more accurate than the algorithm DE (Pro-DE) and PSO (Pro PSSO), respectively, all of 6 algorithms gave appropriated solutions for all the points in the trajectory. It can be seen that, in both cases 3.1 and 3.2 the ISADE and Pro-ISADE algorithms showed the best ability to resolve the inverse kinematics problems in all 3 aspects: accuracy, execution time and number of generations. More specifically, in Scenario 3.1, when searching space for joints' variables were *RoMs*, the average achieved accuracies for ISADE was around 2.0748e-14 (m) that is much better than the values of 7.5404e-13 (m) and 2.2260e-13 (m) corresponding for PSO and DE algorithms. Although the ISADE algorithm was set to a fitness value to achieve such higher accuracy, the execution time of the algorithm was still below the time of PSO and DE algorithm. These average execution time values were 0.0679 (s); 0.0845 (s) and 0.3478 (s) second for ISADE, DE and Pro algorithm, respectively. The above results can be partly explained based on the number of necessary iterations that each algorithm was needed to find the optimal values of joints variables. From **Figure 6c**, it showed that, when solving the IK problem for almost points in the spiral trajectory, the ISADE method used the least number of iterations. The **Table 5** presented more clearly, on the average each point in the trajectory the ISADE needed 85.19 generations to find the optimal values, these means number for DE and PSO algorithm are 125.44 and 391.1

In Scenario 3.2, the searching space for joints' variables were around previous optimal values that were set up as in the **Table 2**. Similar to the Scenario 3.1, all of the comparison parameters gotten from using Pro-ISADE algorithm were better than that values from Pro-DE and Pro-PSO algorithms. These parameters are described in the as well as **Table 5**. In order to comparison between Scenario 3.1 with Scenario 3.2, all average parameters was shown in the **Table 5**. From all comparison, the proposed ISADE or Pro-ISADE were always proved the best solution to solve the inverse kinematics requirements for the manipulator robot. Moreover, **Table 5** also showed that, the Pro-ISADE had better performance compared to ISADE. By using Pro-ISADE algorithm, it reduced all of parameters including distance error, execution time and number of generations.

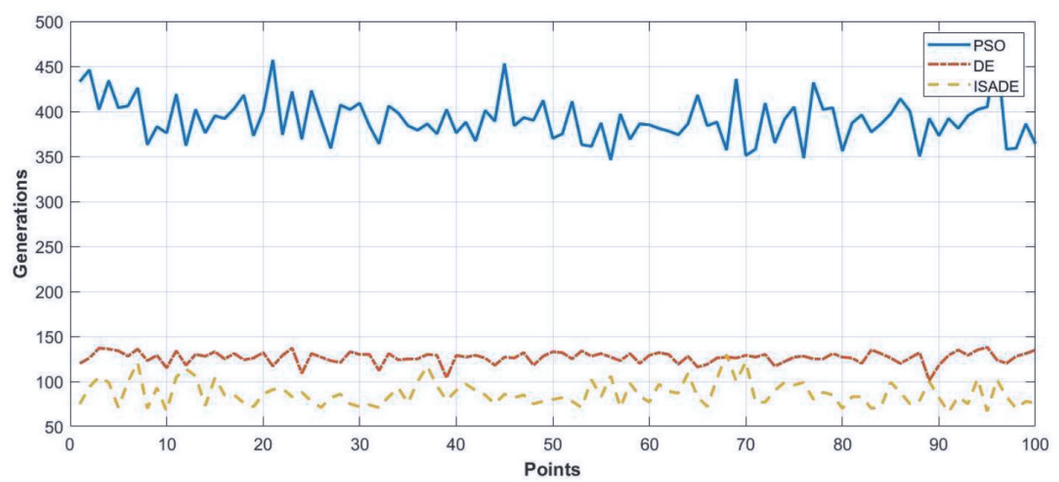
Another very important result gotten from Scenario 3.2 is the quality of joints' values. **Figure 8** show the joints' value in two cases of using ISADE in Scenario 3.1 and using Pro-ISADE in Scenario 3.2. It is clear that the joints' value in the Scenario 3.1 were change dramatically. On the contrary, the values of joints in Scenario 3.2 changed continuously and slowly. The quality of joints variable values as **Figure 9b**,



a)



b)



c)

Figure 7.
Results for Scenario 3.1. (a) Distance error. (b) Execution time. (c) Number of generations.

that received by using Pro-ISADE, will ensure feasibility in the next stages of calculation and design for the robot. These values, along with the values of speed, acceleration, as well as the weight parameters of the stages, will be used in the dynamic problem as well as in future control.

	PSO	Pro-PSO	DE	Pro-DE	ISADE	Pro-ISADE
Scenario 1						
Fitness value	1e-9	Not applied	1e-10	Not applied	1e-12	Not applied
Avg. error (m)	2.4151e-09	Not applied	6.9655e-10	Not applied	6.8362e-11	Not applied
STD (m)	5.8117e-10	Not applied	2.0075e-10	Not applied	2.3796e-11	Not applied
Avg. iteration	357.91	Not applied	76.54	Not applied	64.34	Not applied
Avg. execution time (s)	0.2931	Not applied	0.1115	Not applied	0.0455	Not applied
Scenario 3.1 (Italic values) and Scenario 3.2						
Fitness value	*1e-14*	1e-14	*1e-15*		*1e-17*	1e-12
Avg. error (m)	*7.4140e-13*	7.4650e-13	*2.2260e-13*	2.2950e-13	*2.0748e-14*	2.0103e-14
STD (m)	*1.9574e-13*	1.9736e-13	*6.5615e-14*	6.1330e-14	*1.0414e-14*	9.8913e-15
Avg. iteration	*429.950*	407.8800	*125.4400*	114.2700	*85.1900*	75.2300
Avg. execution time (s)	*0.3604*	0.2576	*0.1015*	0.0845	*0.0679*	0.0554

Italics were used to differentiate the results of Scenario 3.1 and 3.2.

Table 5.
Comparative results between all cases.

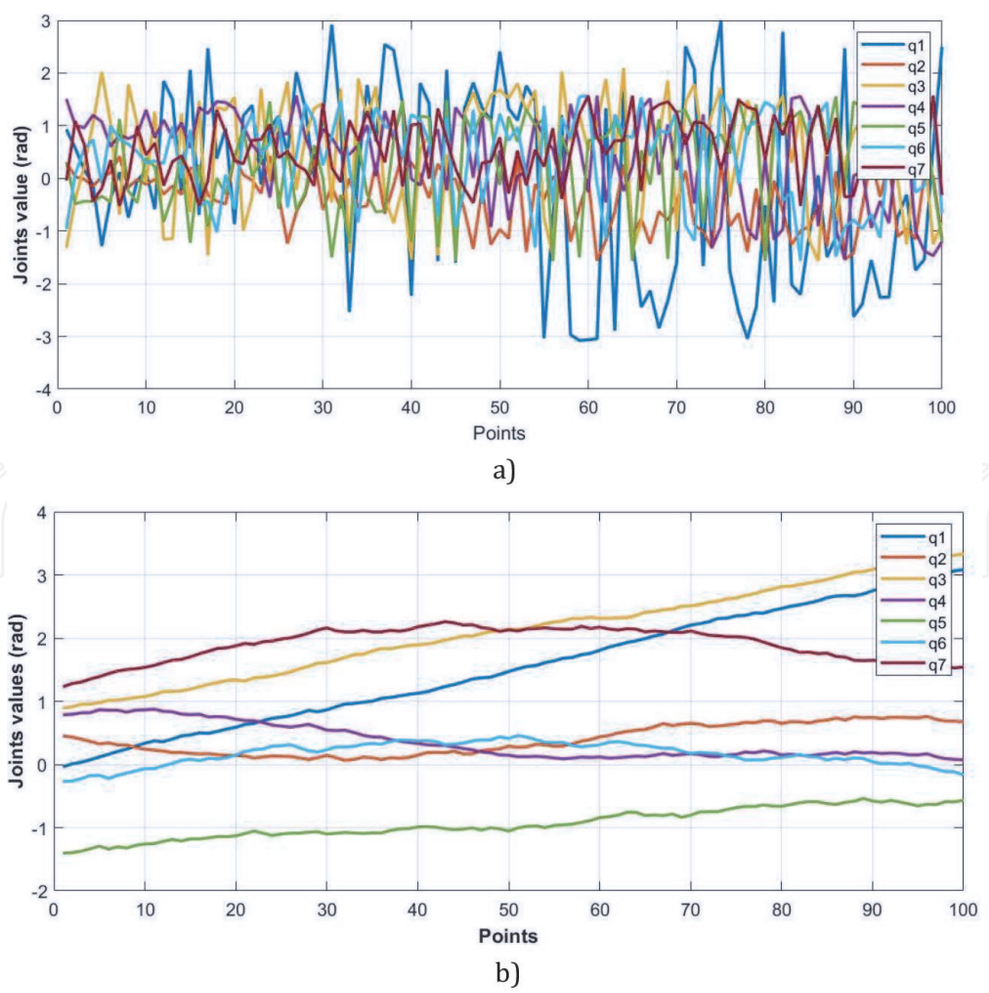


Figure 8.
Joint variables' results. (a) Joint variables' values in Scenario 3.1 using ISADE algorithm. (b) Joint variables' values in Scenario 3.2 using Pro-ISADE algorithm.

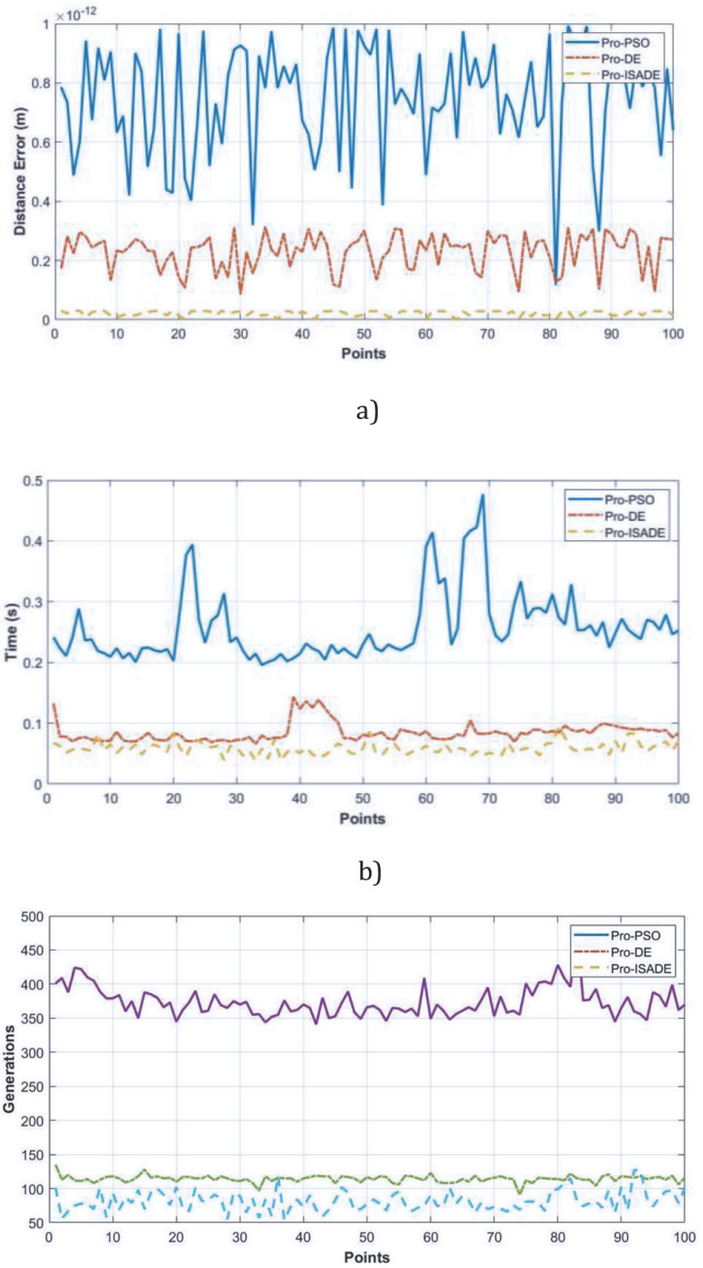


Figure 9.
Results for Scenario 3.2. (a) Distance error. (b) Execution time. (c) Number of generations.

In short, after comparing the results of Scenario 3.1 and 3.2, it is possible to conclude that the ISADE algorithm and Pro-ISADE are the best solutions to solve the IK problem for the robot in all aspects: endpoint accuracy, execution time and number of generation. The Pro-ISADE algorithm not only guarantees the above parameters, it also ensures the quality of the joints' variables to serve the next computational and design stages.

Table 5 summarizes results of the average error, the standard deviation of error (STD), the average iteration and the average execution time of all Scenarios. As in the table, the algorithms of ISADE and Pro-ISADE got the better results than the other algorithms.

As mentioned at the beginning of this article, intelligent optimization techniques have been using more and more popular in difficult and complex tasks including the IK problem for redundant manipulator robots. **Table 6** shows some studies used meta-heuristic optimization algorithms to resolve the inverse kinematics task for

different robot models. The Table presents: the used algorithm for the IK calculation, selected manipulators for the test, the algorithms that are used to comparison. For example, El-Sherbiny et al. [11] used the Adaptive Neuro Fuzzy Inference System (ANFIS) algorithm to calculate the IK problem of a 5 DOF robot, and then compared results with GA algorithm. Both algorithms could get the appropriate solutions, but ANFIS algorithm proved to be the best one. The comparison also shows that a number of studies [12, 22, 23], using optimal algorithms such as PSO, ABC, Q-PSO ... handle the inverse kinetic requirements for the model of 7 degrees of freedom. All the used algorithms have proven the ability to handle the problem, but it is not difficult to see that most of these studies have the lower accuracy and processing speed than the ISADE as well as the Pro-ISADE algorithm proposed in this study.

Research	**Robot arm**	**Results of Used algorithm**	**Results of Compared algorithm**	**Average of**
Rokbani et al. [20]	3-DOF	10 Firefly	60 Firefly	
		1.27e−17	1.78e−18	Position error (m)
		1.21e−03	7.15e−3	Execution time (s)
Ayyıldız and Çetinkaya [7]	4-DOF	PSO	GA	
		7.70e−06	3.96e−04	Position error (m)
		0.0196	0.1753	Execution time (s)
El-Sherbiny et al. [11]	5-DOF	Adaptive Neuro Fuzzy Inference System (ANFIS)	GA	
		5.426e−03	7.64e−04	Position error (m)
		0.0308	83.1239	Execution time (s)
Shi and Xie [21]	6-DOF	Adaboost NN	—	
		0.00267	—	
		0.3	—	
Dereli and Köker [22]	7-DOF	Random IW-PSO	Global–Local Best IW-PSO	
		6.20e−03	3.64e−03	Position error (m)
		1.6	1.2	Execution time (s)
Serkan Dereli [12]	7-DOF	Q-PSO	PSO; ABC; Firefly	
		6.69347e-11	1.4547e-3	Position error (m)
		0.2195	0.4806	Execution time (s)
Serkan Dereli [23]	7-DOF	firefly	PSO, ABC	
		6.53e−05	5.45e−04	Position error (m)
		0,9204	0,4441	Execution time (s)
Our study	7-DOF	ISADE, Pro-ISADE	PSO, DE, Pro-PSO, Pro-DE	
		2.0103e-14	*7.4140e-13*	Position error (m)
		0.0554	*0.3604*	Execution time (s)

Italics were used to differentiate the results of Scenario 3.1 and 3.2.

Table 6.
Comparison with some other studies.

6. Conclusions

In this research, inverse kinematics problem for a 7 degree of freedom serial robot manipulator was implemented to prove the accuracy and efficiency of the self-adaptive control parameters in Differential Evolution (ISADE) and the ISADE algorithm with searching space improvement (Pro-ISADE) algorithm. To evaluate the effectiveness of the two algorithms above, the results obtained from the ISADE algorithm as well as Pro-ISADE were compared with the results from the PSO (Pro-PSO) and DE (Pro-DE) algorithm. Experiments were performed with three Scenarios. In the first Scenario, an endpoint in the workspace is randomly selected. The purpose of this Scenario is to compare the convergence speed of the three algorithms. In the second Scenario, algorithm was used to calculate inverse kinematics of the robot for 100 points randomly selected in the working space. The aim of this Scenario 2 is to test the accuracy and efficiency of the algorithm when the end effector started at the same position, it went to any point in working space. Meanwhile, in the third Scenario, the algorithms solved the inverse kinematics problem when the end effector of the robot moved point to point that are located on a spiral trajectory in the workspace. The implementation experiments have shown, the ISADE algorithm gave much better results than other algorithms in term of: accuracy, execution time and number of generation. Besides, by improving the searching boundary for joints' variable, the Pro-ISADE, Pro-DE and Pro-PSO also improve the accuracy as well as processing speed and especially the quality of the value of the joints variable compared to the ISADE, DE and PSO, respectively. These optimal joints' values ensure the feasibility of the dynamic and control problem in the future. In short, with ISADE algorithm as well as Pro-ISADE, they have handled the inverse kinematic requirement very effectively both in term of accuracy and computation time. The Pro-ISADE algorithm not only improves the above two factors, but also improves the quality of the joints' variables.

Conflict of interest

"The authors declare no conflict of interest."

Author details

Trung Nguyen[1*] and Tam Bui[1,2]

1 Hanoi University of Science and Technology, Hanoi, Vietnam

2 Shibaura Institute of Technology, Tokyo, Japan

*Address all correspondence to: trung.nguyenthanh@hust.edu.vn

References

[1] Köker R, Çakar T. A neuro-genetic simulated annealing approach to the inverse kinematics solution of robots: a simulation based study. Eng Comput. 2016;32:553–565.

[2] Huang HC, Chen CP, Wang PR (2012) Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators. IEEE international conference on systems, man, and cybernetics. Seoul, Korea, pp 3105–3110

[3] Rubio JJ, Bravo AG, Pacheco J, Aguilar C (2014) Passivity analysis and modeling of robotic arms. IEEE Lat Am Trans 12(8):1381–1389

[4] Kou YL, Lin TP, Wu CY (2014) Experimental and numerical study on the semi-closed loop control of a planar robot manipulator. Math Probl Eng 2014:1–9. doi:10.1155/2014/769038

[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, 2018.

[6] Mahanta GB, Deepak BBVL, Dileep M, et al. Prediction of inverse kinematics for a 6-DOF Industrial robot Arm using soft computing techniques. In: Soft computing for problem solving. Singapore: Springer; 2019. p. 519–530.

[7] Ayyıldız M, Çetinkaya K. Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robotmanipulator.Neural Comput Appl. 2016;27:825–836.

[8] Thomas Joseph Collinsm, Wei-Min Shen. Particle Swarm Optimization for High-DOF Inverse Kinematics. 3rd International Conference on Control, Automation and Robotics: IEEE, 2017.

[9] Malek Alkayyali, Tarek A. Tutunji, PSO-based Algorithm for Inverse Kinematics Solution of Robotic Arm Manipulators, 20th international Conference on Research and education in Mechatronics (REM), 2019

[10] 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

[11] El-Sherbiny A, El-Hosseini MA, Haikal AY. A new ABC variant for solving inverse kinematics problem in 5 DOF robot arm. Appl Soft Comput. 2018;73:24–38.

[12] Serkan Dereli; Ra¸sit Köker; Ameta-heuristic proposal for inverse kinematics solution of 7-DOF serial roboticmanipulator: quantum behaved particle swarm algorithm; Artificial Intelligence Review, 2019

[13] Bui, T. Pham, H. Hasegawa, H., Improve self-adaptive control parameters in differential evolution for solving constrained engineering optimization problems. J Comput Sci Technol Vol.7, No.1 (2013), pp.59-74. DOI:10.1299/jcst.7.59.

[14] Coello, C.A.C., Use of a self-adaptive penalty approach for engineering optimization problems, Computers in Industry 41 (2000), pp.113-127.

[15] Thanh-Trung Nguyen, Ngoc-Linh Tao, Van-Tinh Nguyen, Ngoc-Tam Bui, V. H 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)

[16] Serkan Dereli, Ra¸sit Köker, Ameta-heuristic proposal for inverse kinematics solution of 7-DOF serial roboticmanipulator: quantum behaved particle swarm algorithm; Artificial Intelligence Review, 2019

[17] Craig JJ (2005) Introduction to robotics mechanics and control, 3rd edn. Pearson Education, Upper Saddle River, NJ

[18] Kennedy, J.; Eberhart, R. Particle swarm optimization. In: IEEE international conference on neural networks, (1995) pp 1943–1948

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

[20] Rokbani N, Casals A, Alimi AM (2015) IK-FA, a new heuristic inverse kinematics solver using firefly algorithm. Comput Intell Appl Model Control 575:553–565

[21] ShiQ,Xie J (2017)Aresearch on inverse kinematics solution of 6-dof robot with offset-wrist based on adaboost neural network. In: IEEE international conference on CIS and RAM, 18–20 November, Ningbo, China

[22] Dereli S, Köker R (2018) IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator. Sigma J Eng Nat Sci 36:77–85

[23] Serkan Dereli & Raşit Köker, Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of theresults in terms of speed and accuracy, Inverse Problems in Science and Engineering, 2020, VOL. 28, NO. 5,