sensors

MDPI

Article

Intelligent Trajectory Tracking Behavior of a Multi-Joint Robotic Arm via Genetic–Swarm Optimization for the Inverse Kinematic Solution

Mohammad Soleimani Amiri and Rizauddin Ramli *

Department of Mechanical and Manufacturing Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, Bangi 43600, Selangor, Malaysia; msa0911@gmail.com
* Correspondence: rizauddin@ukm.edu.my

Abstract: It is necessary to control the movement of a complex multi-joint structure such as a robotic arm in order to reach a target position accurately in various applications. In this paper, a hybrid optimal Genetic–Swarm solution for the Inverse Kinematic (IK) solution of a robotic arm is presented. Each joint is controlled by Proportional–Integral–Derivative (PID) controller optimized with the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), called Genetic–Swarm Optimization (GSO). GSO solves the IK of each joint while the dynamic model is determined by the Lagrangian. The tuning of the PID is defined as an optimization problem and is solved by PSO for the simulated model in a virtual environment. A Graphical User Interface has been developed as a front-end application. Based on the combination of hybrid optimal GSO and PID control, it is ascertained that the system works efficiently. Finally, we compare the hybrid optimal GSO with conventional optimization methods by statistic analysis.

Keywords: robotic arm; Genetic Algorithm; Particle Swarm Optimization; PID control

Citation: Soleimani Amiri, M.; Ramli, R. Intelligent Trajectory Tracking Behavior of a Multi-Joint Robotic Arm via Genetic–Swarm Optimization for the Inverse Kinematic Solution. *Sensors* **2021**, *21*, 3171. https://doi.org/10.3390/s21093171

Received: 28 March 2021
Accepted: 29 April 2021
Published: 3 May 2021

1. Introduction

With the advancements in robotic technology, numerous types of robots have become involved in our daily life and help humans in many different areas. As one of the most common robots, the multi-joint manipulator robotic arm plays an important role in automotive, agriculture and bio-medical sectors due to its flexibility, robustness and accuracy [1–3].

The identification of the Inverse Kinematic (IK) plays an important role in the precision control of trajectory tracking [4,5]. Various IK solutions have been carried out for robotic arms [6,7]. For instance, Xu et al. [8] presented a combination brain of a computer interface and computer vision to move a robotic arm end-effector to a desired point by using a depth camera. A six degree of freedom (6DoF) robot with initialized commands from a user's brain signals combined with a point clouds model was verified with five healthy candidates without specific user training, showing acceptable accomplishment for complex tasks. Fang et al. [9] established a visual communication method using deep neural networks, in which the movements of a human arm were monitored and determined by the Denavit–Hartenberg (D-H) technique. Narayan et al. [10] presented a 5DoF robotic arm with a three-finger gripper and validated the IK in a simulation platform. Ye et al. [11] dealt with 5DoF manipulator forward-IK problems using Ferrari's and redundant Euler methods and validated them in a simulation model. Wei at al. [12] applied a neural network to a robotic arm and used environment feedback to reach a specific target point inspired by animal and human biological neural networks. They validated their approach using the penalty function to avoid the robot from reaching specific points. The results show the end-effector reached the target successfully. Ren et al. [13] developed generative neural networks to solve the IK for a robotic arm. They determined the IK by the D-H technique and Moveit, which is an application of a Robot Operating System (ROS) to control and monitor a robot.

Traditionally, the IK has been utilized to establish joint configurations of manipulators based on the end-effector position. However, the traditional IK methods cannot consider the continuity of configurations, collision avoidance and kinematic singularities that arises when attempting to follow the end-effector path [14]. In addition, solving IK problems is a difficult challenge because manipulators with more than 5DoF result in an infinite number of possible solutions for joint trajectories that determine the same position in the Cartesian space [15]. Traditional analytical solutions cannot directly calculate the one-to-many possible relationships in the Cartesian space. Therefore, evolutionary algorithms such as optimization methods are used to solve IK problems quantitatively [11,16]. For instance, Starke et al. [17] studied a mimetic evolutionary algorithm, which was a combination of the Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and gradient-based optimization to address the IK solution for various industrial and anthropomorphic robots.

One of the approaches used in this paper is to combine GA and PSO in order to develop an optimal solution for the IK and Proportional–Integral–Derivative controller (PID) controller tuning. This optimization method has been presented in several works for various applications [18–20]. For instance, Dziwinski, et al. [21] presented a fuzzy-logic controller in which a combination of PSO and GA was used in parallel to improve PSO performance by adding crossover and mutation to the GA to avoid becoming trapped in local optima. Farand et al. [22] developed a combination of GA and PSO to reduce computational time and accuracy in comparison with other known methods such as GA and PSO for high-dimensional and complex functions.

The PID controller is one of the most common used classical control systems in different industries because of its flexibility, satisfactory results [23,24], ease of implementation in a control system and wide usage in industries [25,26]. In order to enhance the accuracy and robustness of classical PID control, one of the techniques is to combine it with optimization methods [27,28]. Belkadi et al. [29] worked on PSO with a random initial value to tune the parameters of the PID controller by minimizing the trajectory error. They verified their controller in a simulation model and compared it with conventional methods by numerical analysis. Phu et al. [30] used optimization with sliding mode control based on the Bolza–Meyer criterion to minimize the vibration effect. In another work, Suhaimin et al. [31] used a PID controller for a 5DoF robotic arm and controlled its joints for the point-to-point trajectory tracking of end-effectors.

The contribution of this paper is the development of an optimal hybrid IK and PID controller for joint trajectory tracking, using the Genetic–Swarm Optimization technique. The applicability of the proposed technique for the IK solution of end-effectors and steady-state error for control is compared with conventional optimization approaches such as the GA and PSO. In addition, a 5DoF robotic arm is selected for analysis due to its simple structure, flexible action, small volume, convenient operation and so on; this device is widely used in many fields and industries [32].

The rest of the paper is organized as follows: first, the kinematic and dynamic models of the 5DoF robotic arm are established using the D-H and Lagrangian method. Subsequently, an IK solution is determined by hybrid Genetic–Swarm Optimization (GSO) for angular trajectories of each joint reaching the target position. The joint angles determined by the IK are implemented in a closed-loop system using a PID controller, and the gains are tuned by the GA and PSO. The 3D models of the robotic arm are simulated in the Gazebo environment. Finally, a Graphical User Interface (GUI) is created to interact with the 3D model in the ROS environment.

2. Dynamic and Kinematic Model

The robotic arm consists of a base, four links, a wrist and gripper that are connected to each other by joints in series. Figure 1 represents a 5DoF robotic arm, in which the coordinate systems of joints and global frames are presented.

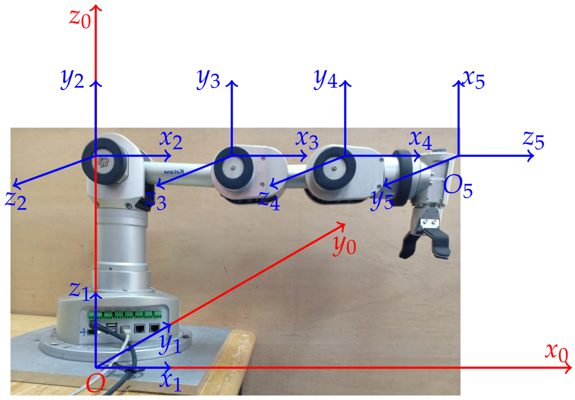


Figure 1. Configuration of the robotic arm, where x_0, y_0 and z_0 are axes of the reference frame.

There are various methods used to determine the dynamic equation of robot manipulators, such as Newton–Euler, Kane and Hamilton approaches. In this work, an energy-based Lagrangian method has been adopted to determine the relation between the torque and angle of joints; one of the advantages of the Lagrangian method is that, unlike the Newton–Euler method, it is not necessary to determine internal forces between joints; therefore, it is quicker and easier to obtain the equation of motion [33]. The Lagrangian equation is given as follows:

$$L = E_k - E_p \tag{1}$$

$$\tau_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}_i}) - (\frac{\partial L}{\partial \theta_i}) + B_i(\dot{\theta}_i) \tag{2}$$

where B_i is the joint friction coefficient; L is the Lagrangian function; T_i is the torque of each link, with $i = 1,2,3,4,5$; θ_i and $\dot{\theta}_i$ are the angular trajectory and velocity; and E_p and E_k are the total potential and kinetic energies, respectively. From [34], the equations of E_p and E_k are determined as follows:

$$E_p = \sum_{i=1}^{5} m_i g z_{di} \tag{3}$$

$$E_k = \sum_{i=1}^{5}[\frac{1}{2}m_i(\dot{x}_{di}^2 + \dot{y}_{di}^2 + \dot{z}_{di}^2) + \frac{1}{2}I_{xi}\dot{\theta}_i^2 + \frac{1}{2}I_i] \tag{4}$$

where m_i and I_i are the mass and inertia of each link; g is the gravity acceleration; and $(\dot{x}_{di}, \dot{y}_{di}, \dot{z}_{di})$ is the time derivative of the centroid position of each joint, where $i = 1,2,3,4,5$. According to the geometric relation, the centroid position (x_{di}, y_{di}, z_{di}) of every linkage is written as

$$X_{d_i} = \sum_{j=1}^{i-1}(R_{z_j}.{}^jX) + R_{z_i}.{}^iX_d \tag{5}$$

where ${}^jX \in \Re^{3\times1}$ is the position of joint $(i-1)_{th}$ according to the reference frame; $X_{d_i} \in \Re^{3\times1}$ is the position of the centroid point of link i_{th} relative to the reference frame; $R_{z_i} \in \Re^3$ is the rotation matrix around z-axes according to the coordinate system placed in the i_{th} joint; and ${}^iX_d \in \Re^{3\times1}$ represents the centroid position of the link i_{th} regarding the coordinate system located in the joint i_{th}. By substituting E_k and E_p in the Lagrangian function, the state space dynamic is determined in range of motion condition where while one joint is moving, the other ones are fixed, which is shown as follows:

$$\tau = M\ddot{\theta} + V\dot{\theta} + G(\theta) \tag{6}$$

where $\theta \in \Re^{5\times1}$ and $\ddot{\theta} \in \Re^{5\times1}$ are the angular rotation and acceleration; $\tau \in \Re^{5\times1}$ is the torque vector; and $M \in \Re^{5}$ is a matrix containing mass and inertia elements, which is shown as follows:

$$M = \begin{bmatrix} e_{m_1} & 0 & 0 & 0 & 0 \\ 0 & e_{m_2} & 0 & 0 & 0 \\ 0 & 0 & e_{m_3} & 0 & 0 \\ 0 & 0 & 0 & e_{m_4} & 0 \\ 0 & 0 & 0 & 0 & e_{m_5} \end{bmatrix} \quad (7)$$

where $e_{m_i} \quad i = 1,2,3,4,5$ represents the mass and inertia elements, expressed as follows:

$$e_{m_1} = I_{x1}; \quad e_{m_i} = l_i^2 \sum_{i=i+1}^{5} (m_i) + I_i + m_i l_{c_i}^2 \quad (8)$$

where l_{c_i} is the length of the centroid position for each link and l_i is the length of every link. $V \in \Re^{5}$ is the centrifugal, coriolis and friction matrix and $G(\theta) \in \Re^{5}$ represents the gravity matrix, expressed as follows:

$$V(\dot{\theta},\theta) = B_i \cdot I_{5\times5} \qquad G(\theta) = e_{g_i} \cdot I_{5\times5} \quad (9)$$

where $I_5 \in \Re^{5}$ is the identity matrix and e_{g_i} shows the elements of mass and gravity matrices, represented as follows:

$$e_{g_i} = (l_i \sum_{i=i+i}^{5} (m_i) + l_{c_i} m_i) g sin(\theta_i) \quad (10)$$

where g represents gravitational acceleration. Table 1 illustrates the physical features of the robotic arm's links.

Table 1. Physical features of the robotic arm.

link	$l_i(m)$	$l_{c_i}(m)$	$m_i(Kg)$	I_i	B_i
$i = 1$	0.3	0.15	0.748	0.0013	0.72
$i = 2$	0.19	0.095	0.8020	0.0043	0.83
$i = 3$	0.14	0.07	0.792	0.0023	0.95
$i = 4$	0.15	0.075	0.691	0.0015	1.88
$i = 5$	0.04	0.02	0.2562	0.00012	0.83

In the next stage, the forward kinematic based on modified D-H (mD-H) algorithms has been developed to establish the relative position of the 5DoF robotic arm end-effector to its reference frame O [35]. Table 2 represents the parameters of the mD-H.

In Table 2, α_{i-1}, θ_i, d_i and a_{i-1} represent the twist angle, joint angle, link offset and link length, respectively. The mD-H homogeneous transformation is expressed as follows:

$${}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\theta_{i-1} & cos\theta_i cos\theta_{i-1} & -sin\alpha_{i-1} & -d_i sin\alpha_{i-1} \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & d_i cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

The transformation matrix of the end-effector is the transformation matrix from the reference frame to the last frame, which is shown as follows:

$${}^{0}_{5}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \quad (12)$$

where ${}^{0}_{1}T$, ${}^{1}_{2}T$, ${}^{2}_{3}T$, ${}^{3}_{4}T$ and ${}^{4}_{5}T$ are the transformation matrices of each joint to its previous joint. The transformation matrix from the reference frame to end-effector is as follows:

$$ {}^{0}_{5}T = \begin{bmatrix} t_{1,1} & t_{1,2} & t_{1,3} & t_{1,4} \\ t_{2,1} & t_{2,2} & t_{2,3} & t_{2,4} \\ t_{3,1} & t_{3,2} & t_{3,3} & t_{3,4} \\ t_{4,1} & t_{4,2} & t_{4,3} & t_{4,4} \end{bmatrix} \quad (13) $$

where $t_{1,4}$, $t_{2,4}$ and $t_{3,4}$ express the end-effector position relative to the reference frame, which is shown as follows:

$$ x = t_{1,4} = -\frac{1}{2}(l_4 sin(\theta_4 + \theta_3 + \theta_2 + \theta_1) - l_4 \sin(\theta_4 + \theta_3 + \theta_2 - \theta_1) + l_3 cos(\theta_3 + \theta_2 + \theta_1) + l_3 cos(\theta_3 + \theta_2 - \theta_1) + l_2 cos(\theta_2 + \theta_1) + l_2 cos(\theta_2 - \theta_1)) \quad (14) $$

$$ y = t_{2,4} = \frac{1}{2}(l_4 cos(\theta_4 + \theta_3 + \theta_2 + \theta_1) - l_4 cos(\theta_4 + \theta_3 + \theta_2 - \theta_1) + l_3 sin(\theta_3 + \theta_2 + \theta_1) - l_3 sin(\theta_3 + \theta_2 - \theta_1) + l_2 sin(\theta_2 + \theta_1) - l_2 sin(\theta_2 - \theta_1)) \quad (15) $$

$$ z = t_{3,4} = l_4 cos(\theta_4 + \theta_3 + \theta_2) + l_3 sin(\theta_3 + \theta_2) + l_2 sin(\theta_2) \quad (16) $$

Table 2. mD-H parameters for the 5DoF robotic arm.

Joints	θ_i	d_i	α_{i-1}	a_{i-1}
One	θ_1	0	0	0
Two	θ_2	0	$\frac{\pi}{2}$	l_2
Three	θ_3	0	0	l_3
Four	θ_4	0	0	l_4
Five	θ_5	0	$-\frac{\pi}{2}$	l_5

3. Optimal Inverse Kinematic

Since the number of variables is greater than the number of equations and the end-effector position is non-linear, the usage of traditional methods such as Gaussian elimination are not practical [36]. Thus, in this paper, the IK is defined as a mono-objective optimization problem. The desired position of the end-effector is set to be achieved by minimizing the objective function. In this study, the hybrid version of the GA and PSO, named GSO, is adopted to solve the IK problem, because GA is developed initially by random values due to its reliability and robust performance [37] and PSO is sufficient to find accurate results in a few iterations with low computational time. Subsequently, PSO is initialized by the results of the GA. In the GSO algorithm, the GA provides searching space and initial values for PSO to avoid becoming trapped in local optima.

The summation of squared error (SSE), which is a well-known statistic in multiple regression analyses [38], is chosen as an objective function because it shows the squared sum of residuals, which is the error between the measured and desired trajectory of the end-effector, and illustrates how close a regression line is to a set of residuals. The squaring is necessary to remove any negative signs. The objective function is given as follows:

$$ f_{obj} = \sqrt{(e_x)^2 + (e_y)^2 + (e_z)^2} \quad (17) $$

where e_x, e_y and e_z are the errors, represented as follows:

$$ e_x = x - x_{des} \quad (18) $$

$$ e_y = y - y_{des} \quad (19) $$

$$ e_z = z - z_{des} \quad (20) $$

where x_{des}, y_{des} and z_{des} are the desired positions of the endpoint regarding the reference frame. x, y and z express the position of the endpoint, which are determined by the gene of the GA from Equations (18)–(20). The population structure of an iteration is represented in Figure 2.

$x_1(1)$	$x_2(1)$		$x_{i_{th}}(1)$
$x_1(2)$	$x_2(2)$		$x_{i_{th}}(2)$
$x_1(3)$	$x_2(3)$	....	$x_{i_{th}}(3)$
$x_1(4)$	$x_2(4)$		$x_{i_{th}}(4)$
$1st$	$2nd$		i_{th}

Figure 2. Structure of population for an iteration.

In each population, there is a gene which consists of each joint angle of the robotic arm, represented as x_1, x_2, x_3 and x_4, which are θ_1, θ_2, θ_3 and θ_4, respectively. In the robotic arm model, there are limitations for the angular trajectory of each joint, which create the searching space for the GA, which is as follows:

$$-3.02(rad) \leq \theta_1 \leq 2.89(rad) \tag{21}$$

$$-0.13(rad) \leq \theta_2 \leq 2.16(rad) \tag{22}$$

$$-2.22(rad) \leq \theta_3 \leq 2.05(rad) \tag{23}$$

$$-2.03(rad) \leq \theta_4 \leq 1.87(rad) \tag{24}$$

$$\theta_5 = 1.57(rad) \tag{25}$$

θ_5 is set as 90 degrees and is not included in the design variables because it is assumed that the gripper is located at last link point down to grab the objects. The first iterations of the GA are set randomly within the searching space, as demonstrated in Equations (21)–(24). After the initialization, the objective function is calculated for each gene of the population for evaluation and sorted in ascending order. The next iterations are created by crossover and mutation. The crossover enhances the possibilities of finding the most optimum results by blending the previous iterations as children and parents using the uniform crossover operator. In addition, mutation is performed to maintain the diversity of the GA [39–41]. The algorithm is continued by the evaluation of each gene by determining the objective function followed by sorting in ascending order. This trend continues until the maximum iterations are reached. In the last iteration, because of the ascending sorting, the first gene is the result of GA and represents the optimum angles of joints which are needed to lead the endpoint to reach the desired position.

$$x_{ga} = [\theta_1, \theta_2, \theta_3, \theta_4] \tag{26}$$

x_{ga} is the output of GA which is used to create the range for the initial population of the PSO, which is shown as follows:

$$x_{1,j} = rand[x_{min}, x_{max}] \tag{27}$$

where j stands for the number of particles in the first population and $rand$ is the function used to generate a random value between x_{min} and x_{max}, which are the lower and higher bounds, given as follows [42]:

$$x_{min} = x_{ga} - r \tag{28}$$

$$x_{max} = x_{ga} + r \tag{29}$$

where $r \in \Re^{1\times 4}$ is a random vector between zero and one. After creating the particles of the first population, the objective function is determined for each particle to evaluate and sort in descending order. The particles of population for the next iteration are created as follows:

$$x_{i+1,j} = x_{i,j} + v_{i+1,j} \tag{30}$$

where $x_{i+1,j}$ is the particle of the next iteration. $v_{i+1,j} \in \Re^{1\times 4}$ is a vector that represents the velocity and direction of each particle through the particle of the next iteration, which is shown as follows:

$$v_{i+1,j} = \omega_i v_{i,j} + c_1 r(p_{best} - x_{i,j}) - c_2 r(g_{best} - x_{i,j}) \tag{31}$$

where $p_{best,i}$ is called the best position, containing the particles that have the minimum objective function. g_{best} is the global best, including the particles which are the minimum of the p_{best}, which is shown as follows:

$$g_{best} = min\{p_{best_i}\} \qquad i = 1, 2, \ldots, i_{max} \tag{32}$$

where i and i_{max} are the current and maximum number of iterations, respectively. In the first iteration, after evaluation, the minimum particle is saved as p_{best} and g_{best}, and the velocity is a zero vector.

$$v_{1,j} = [0, 0, 0, 0] \tag{33}$$

In Equation (31), ω_i is the inertia weight, where its adjustable value for each iteration is given by the following equation:

$$\omega_{i+1} = \omega_{damp}\omega_i \tag{34}$$

where ω_{damp} is the damping value for ω, set as 0.05, and c_1 and c_2 are coefficients of self and social recognition, respectively. The value of c_1 is greater than c_2, and their summation should remain at 4 in all iterations [43].

$$c_1 = 1.8b + 2.1 \tag{35}$$

$$c_2 = 1.8a + 0.1 \tag{36}$$

where, a and b are the ascending and descending gains between zero and one, represented as follows:

$$a = \frac{i}{i_{max}} \qquad i = 1, 2, \ldots, i_{max} \tag{37}$$

$$b = 1 - a \tag{38}$$

Figure 3 represents the changes of parameters of PSO during all iterations. The initial values for c_1, c_2 and ω are 3.9, 0.1 and 1.2, respectively [44].

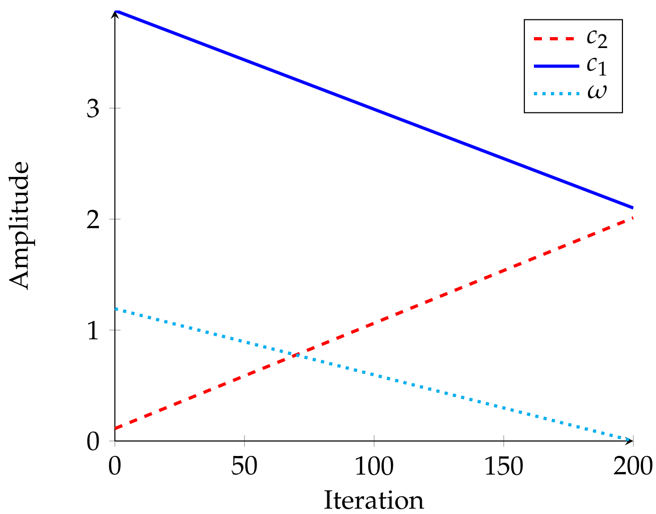


Figure 3. Changes in parameters of modified PSO.

After generating each population, the evaluation and sorting of its particles is developed. This trend is followed until the number of iterations is equal to i_{max}. In this paper, the size of the population for GA and PSO is 40 and i_{max} is 200 for each. Algorithm 1 and Figure 4 show the pseudo-code and flow chart of GSO.

Algorithm 1 Pseudo code of GSO

Start;
Set the target position of the endpoint;
Start GA;
Initialize the first population randomly;
Evaluate initial population;
while Number of iterations equal to maximum iteration of GA **do**;
    Create new iteration using crossover and mutation;
    Evaluate the population by determining the objective function;
    Sort the genes in ascending order;
end while
Select the first gene of the last iteration as the result;
End GA;
Start PSO;
Initialize particles of the first population of PSO based on GA results;
Evaluate the first population;
while Number of iterations equal to maximum iteration of PSO **do**;
    Create new population;
    Evaluate the particles of population;
    Set the minimum particle as the P_{best};
    Set the minimum P_{best} as g_{best};
end while
Establish particle of the g_{best} as the results;
End.

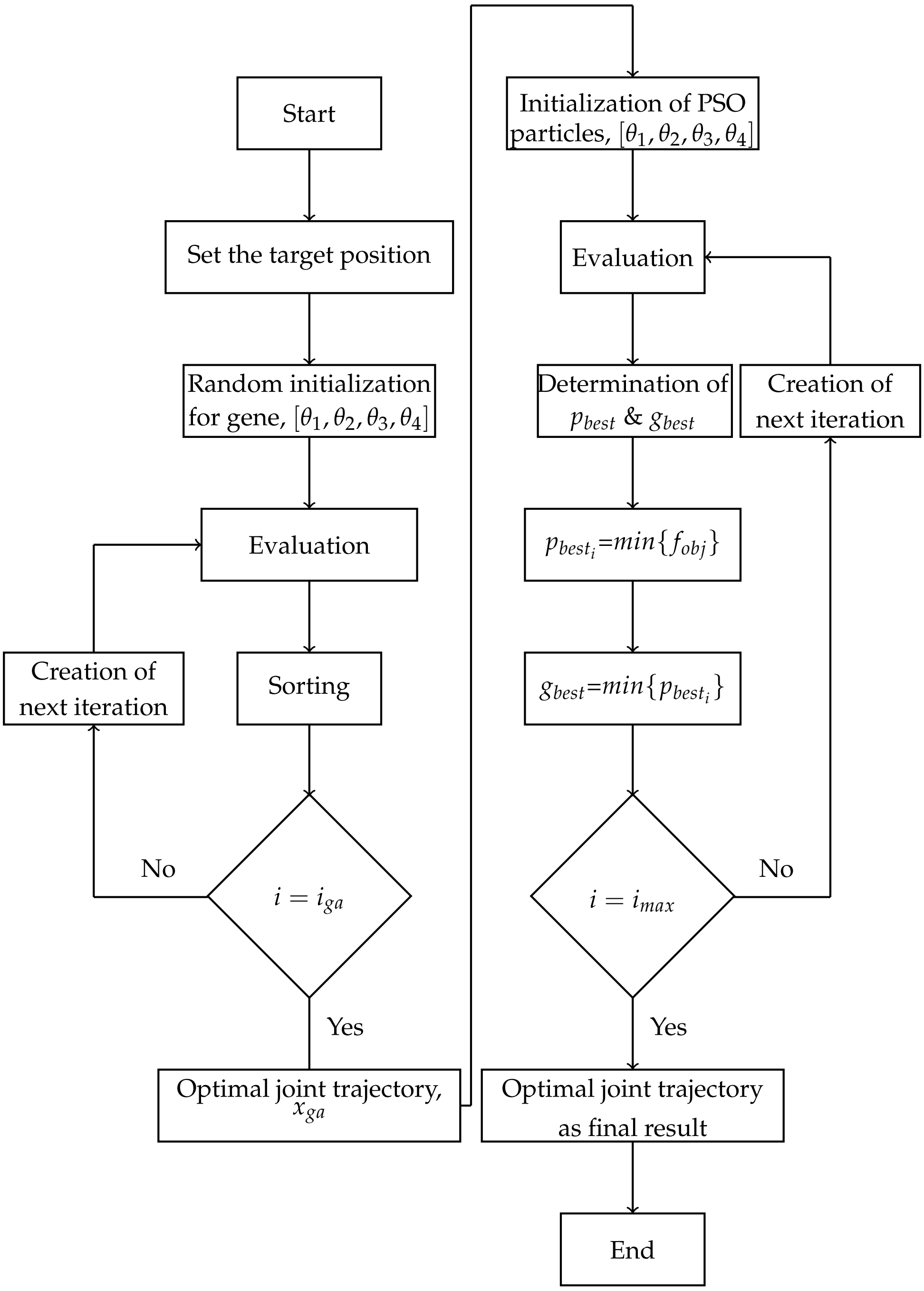


Figure 4. Flow chart of GSO.

4. Control System and Tuning

A closed-loop control system is developed for each joint, and its parameters are adjusted by GSO to converge by adjusting the required torque toward each joint. The desired angular trajectory is determined by the IK. Figure 5 demonstrates the control system of the robotic arm.

In Figure 5, $J_1(t)$, $J_2(t)$, $J_3(t)$ and $J_4(t)$ are plants of each joint; (x_t, y_t, z_t) is the desired position; θ_{d_1}, θ_{d_2}, θ_{d_3} and θ_{d_4} are the desired angular trajectory determined by the IK; θ_{a_1}, θ_{a_2}, θ_{a_3} and θ_{a_4} are the actual angular trajectory for each joint; and e_1, e_2, e_3 and e_4 are the trajectory errors that are the difference between the desired and actual angular trajectory, given as follows:

$$e_i = \theta_{d_i} - \theta_{a_i} \qquad i = 1,2,3,4 \tag{39}$$

The PID controllers $C_1(s)$-$C_4(s)$ for each joint are given as follows:

$$C_i(t) = K_P e_i(t) + K_I \int e_i dt + K_D \frac{de_i}{dt} \quad (40)$$

The tuning of the PID controller is assumed to be an optimization problem, and its parameters are defined as design variables. The tuning processes are carried out by the GSO algorithm, in which the GA starts to optimize the design variables based on random initial parameters, and subsequently the algorithm is continued by PSO based on the output of the GA. Initial parameters of the first population are randomly chosen between 0 and 1, given as follows:

$$x_{1,j} = rand[0,1] \qquad j = 1, ..., j_{max} \quad (41)$$

where x is the particle of the PSO and gene of the GA in each population and j and j_{max} are the current and maximum number of populations. After setting the initial values for particles, an evaluation is carried out based on the objective function of tuning, which is the absolute steady-state error:

$$f = |\theta_{act} - \theta_{des}| \quad (42)$$

where θ_{act} and θ_{des} are the actual and desired joint trajectory, respectively. θ_{act} is measured from a simulation model in real-time, and θ_{des} is developed by the optimal IK. After evaluation and sorting in descending order, the particles of the population for next iteration are generated by mutation and crossover. Whenever the number of iterations meets half of the maximum iterations, the algorithm is switched to PSO. The searching space of PSO is limited around the results of the GA to lead the algorithm toward global optima, as follows:

$$x_{i,j} = [x_{ga} - \frac{x_{ga}}{2}, x_{ga} + \frac{x_{ga}}{2}] \quad (43)$$

The next populations of PSO are created by the particles of the next iteration. The algorithm continues until the number of iterations reaches the maximum. The output is an optimal set for PID parameters.

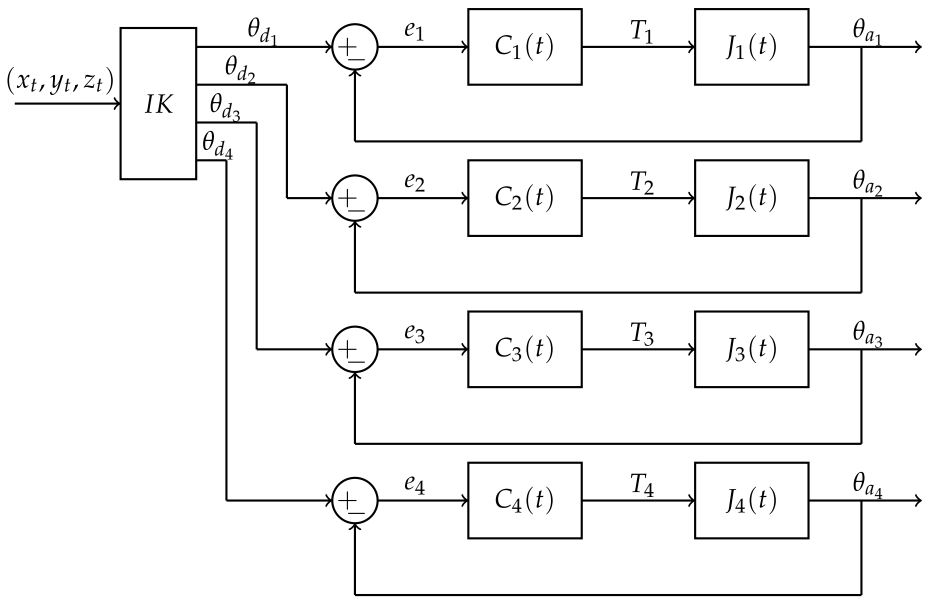


Figure 5. Block diagram of control system for each joint.

5. Results and Discussion

The optimal IK and PSO tuning of controllers was applied in the 3D simulation of a robotic arm in a 3D environment to simulate robots integrated with ROS [45]. A GUI was programmed by using Python to run the algorithms and communicate with the simulation model. In addition, by providing a camera in the Gazebo environment, it was possible to monitor the results visually and numerically, as shown in Figure 6.

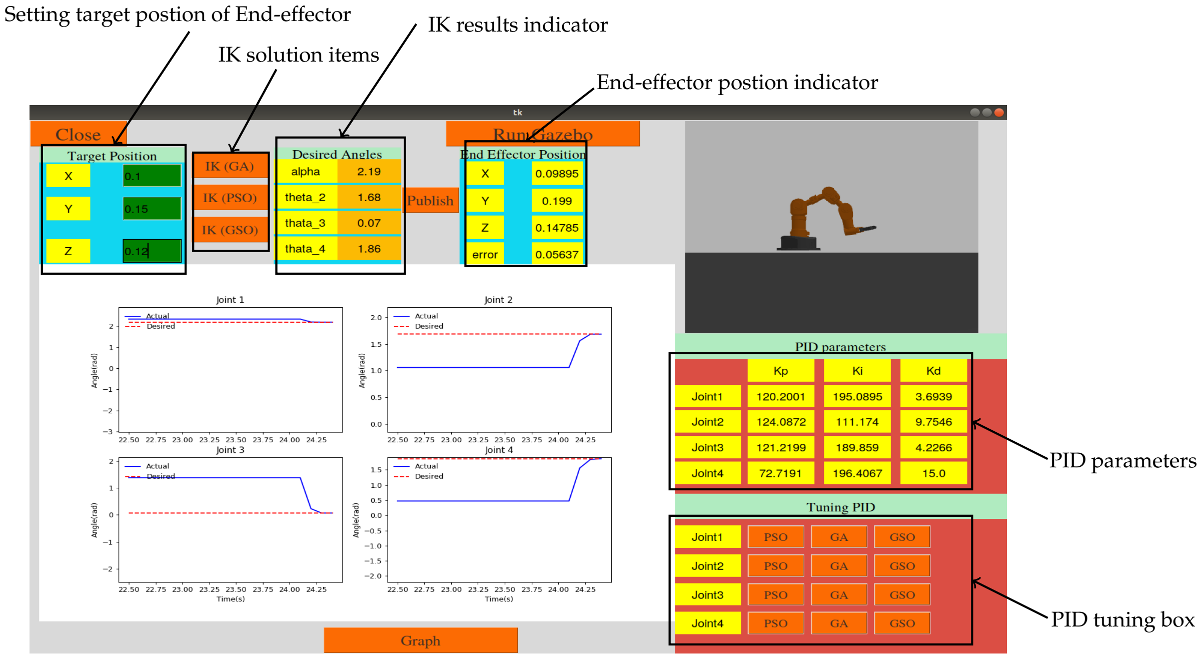


Figure 6. GUI for simulation model.

The desired position for the three different algorithms—i.e., GA, PSO and GSO—could be selected according to the IK method, and the actual position of end-effector, error of the actual trajectory and desired trajectory of each joint could be monitored. In addition, the controller parameters could be tuned in real time and observed. Table 3 compares the optimal results for GA, PSO and GSO for the IK solution, while f_{obj} is the SSE for various sets of optimization parameters.

Various sets of parameters were defined to observe the influences of changes in parameters on the optimization algorithms.

- For GA, set^1_{ga}: crossover = 0.9, mutation = 0.1, population = 40 and generation = 400; set^2_{ga}: crossover = 0.8, mutation = 0.2, population = 40, and generation = 400; set^3_{ga}: crossover = 0.7, mutation = 0.3, population = 40 and generation = 400;
- For PSO, set^1_{pso}: particles = 20, and generation = 200; set^2_{pso}: particles = 30 and generation = 300; set^3_{pso}: particles = 40 and generation = 400;
- For GSO, set^1_{gso}: crossover = 0.9, mutation = 0.1, population of GA and particles of PSO = 40, generation of GA = 300 iteration of PSO = 100, set^2_{gso}: crossover = 0.8, mutation = 0.2, population of GA and particles of PSO = 40, generation of GA = 200 iteration of PSO = 200, set^3_{gso}: crossover = 0.7, mutation = 0.3, population of GA and particles of PSO = 40, generation of GA = 100 and iteration of PSO = 300.

The mean of f_{obj} for GSO in set^3_{gso} has the lowest f_{obj} of 7.9×10^{-15}, and the maximum of f_{obj} is 4.99×10^{-14}, which is the nearest value to its mean compared to other results. This causes the lowest variance of all tests. The mean of the PSO results is lower than GA, while GSO shows the minimum results, which represents a significant improvement for the results obtained by the GSO algorithm. This is due to the hybrid of the GA and PSO algorithms in series; creating the initial values of PSO based on results of the GA increases accuracy compared to using each algorithm individually. In addition, by increasing the number of iterations and particles of PSO, GSO and PSO algorithms show improvements in their results.

Table 3. Numerical analysis for the f_{obj} of GA, PSO and GSO for various sets of parameters.

	Runs	set_{ga}^{1}	set_{ga}^{2}	set_{ga}^{3}
GA	1	4.33×10^{-5}	1.87×10^{-5}	6.12×10^{-7}
	2	0.0013	4.43×10^{-5}	1.6×10^{-5}
	3	1.24×10^{-5}	1.99×10^{-4}	5.23×10^{-8}
	4	1.7×10^{-5}	2.05×10^{-5}	2.96×10^{-7}
	5	4.25×10^{-5}	2.0×10^{-5}	1.76×10^{-5}
	6	2.43×10^{-5}	6.38×10^{-4}	3.6×10^{-5}
	7	3.913×10^{-5}	8.19×10^{-5}	7.5×10^{-5}
	8	3.74×10^{-5}	6.27×10^{-5}	6.65×10^{-5}
	9	3.95×10^{-5}	3.95×10^{-5}	1.75×10^{-6}
	10	1.13×10^{-5}	1.13×10^{-5}	6.96×10^{-5}
Mean		1.54×10^{-4}	3.83×10^{-5}	2.83×10^{-5}
Max		1.3×10^{-3}	8.19×10^{-5}	7.5×10^{-5}
variance		1.61×10^{-7}	5.87×10^{-10}	9.69×10^{-10}
H-value			0.03	
	Runs	set_{pso}^{1}	set_{pso}^{2}	set_{pso}^{3}
PSO	1	1.14×10^{-6}	1.45×10^{-11}	6.20×10^{-17}
	2	5.43×10^{-8}	2.2×10^{-6}	9.41×10^{-14}
	3	0.14×10^{-3}	6.79×10^{-17}	6.79×10^{-17}
	4	8.57×10^{-5}	2.44×10^{-14}	1.99×10^{-14}
	5	2.63×10^{-3}	6.2×10^{-17}	6.2×10^{-17}
	6	2.5×10^{-5}	1.01×10^{-11}	6.2×10^{-17}
	7	3.33×10^{-6}	9.9×10^{-12}	1.0×10^{-16}
	8	7.28×10^{-13}	6.24×10^{-10}	5.66×10^{-13}
	9	3.79×10^{-7}	5.42×10^{-16}	6.2×10^{-17}
	10	3.68×10^{-8}	2.81×10^{-11}	6.2×10^{-17}
Mean		2.89×10^{-4}	2.2×10^{-7}	6.8×10^{-14}
Max		2.63×10^{-3}	2.2×10^{-6}	5.66×10^{-13}
Variance		6.79×10^{-7}	4.83×10^{-13}	3.14×10^{-26}
H-value			16.07	
	Runs	set_{gso}^{1}	set_{gso}^{2}	set_{gso}^{3}
GSO	1	8.34×10^{-8}	6.2×10^{-17}	6.2×10^{-17}
	2	1.74×10^{-7}	3.03×10^{-13}	2.37×10^{-16}
	3	2.7×10^{-13}	2.45×10^{-11}	1.54×10^{-15}
	4	1.71×10^{-5}	1×10^{-16}	6.2×10^{-17}
	5	7.59×10^{-11}	7.85×10^{-17}	2.02×10^{-16}
	6	7.19×10^{-6}	6.79×10^{-17}	6.2×10^{-17}
	7	8.1×10^{-4}	5.43×10^{-15}	2.02×10^{-16}
	8	6.2×10^{-5}	7.76×10^{-16}	5.49×10^{-17}
	9	3.14×10^{-7}	6.2×10^{-17}	2.67×10^{-14}
	10	2.87×10^{-12}	7.63×10^{-14}	4.99×10^{-14}
Mean		9.97×10^{-5}	2.94×10^{-12}	7.9×10^{-15}
Max		8.1×10^{-4}	2.45×10^{-11}	4.99×10^{-14}
Variance		7.13×10^{-8}	5.98×10^{-23}	2.86×10^{-28}
H-value			15.84	

The H-values were measured by the Kruskal–Wallis method and were 0.03, 16.07 and 15.84 for the GA, PSO and GSO respectively. The test was calculated with the assumption of $\alpha = 0.05$; therefore, the critical value for this test was 5.99. Since PSO and GSO had greater H-values than critical points, there were significant differences among the groups of f_{obg} calculated by PSO and GSO.

Table 4 represents the computational time in seconds for GA, PSO and GSO, while the parameters are determined as set^3_{ga}, set^3_{pso} and set^3_{gso}, respectively.

Table 4. Computational time for GA, PSO and GSO in seconds.

Runs	GA	PSO	GSO
1	8.35 (s)	2.46 (s)	3.87 (s)
2	8.18 (s)	2.41 (s)	3.81 (s)
3	8.01 (s)	2.35 (s)	3.83 (s)
4	8.11 (s)	2.40 (s)	3.77 (s)
5	8.09(s)	2.41 (s)	3.45 (s)
6	8.33 (s)	2.36 (s)	3.80 (s)
7	8.26 (s)	2.39 (s)	3.86 (s)
8	8.26 (s)	2.35 (s)	3.54 (s)
9	8.13 (s)	2.37 (s)	3.88 (s)
10	8.38 (s)	2.43 (s)	3.84 (s)
Mean	8.21 (s)	2.39 (s)	3.76 (s)

The mean computational time of PSO was less than GA and GSO by 5.82 s and 1.37 s, respectively. Although the computational time of PSO was the lowest, the combination of GA and PSO reduced the computational time consumption significantly compared to GA by 4.45 s. By considering the value of f_{obj} of GSO in Table 3 and the computational time, the usage of GSO for IK solution can be seen to have resulted in significant improvements in accuracy and computational time consumption.

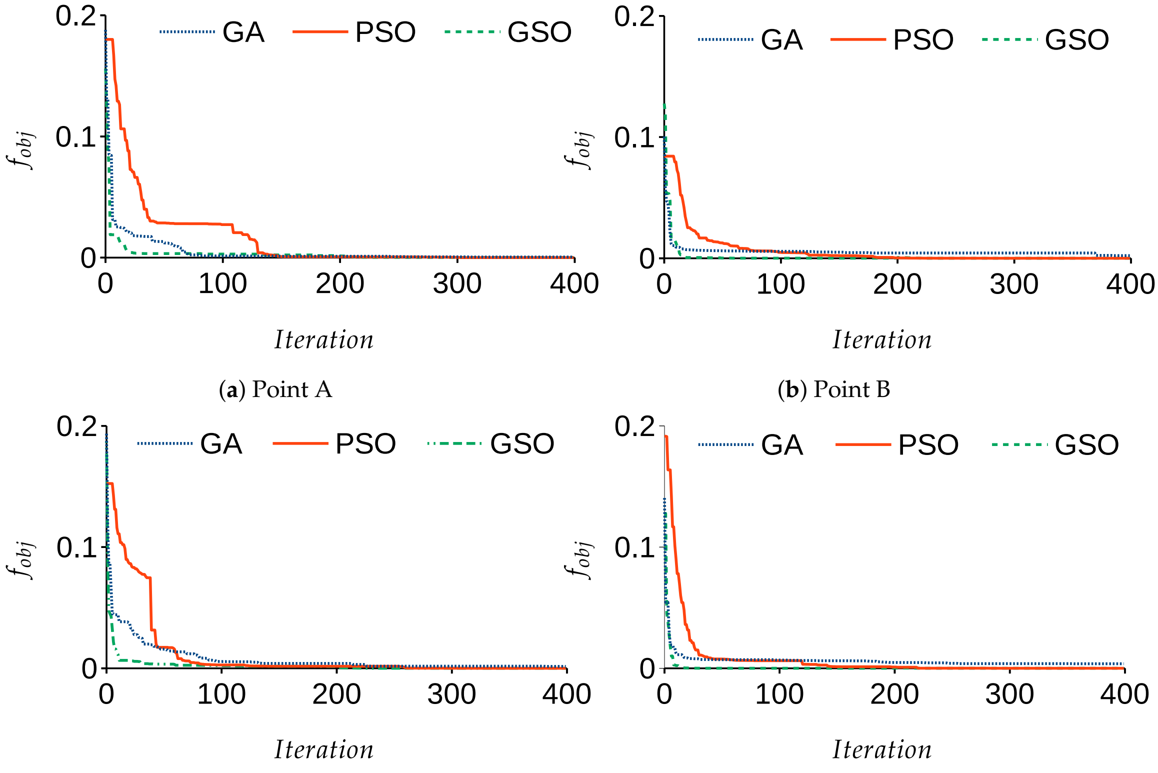
In order to test the IK results solved by GSO in the robotic arm model in the Gazebo environment, nine desired position coordinates were expressed. Table 5 illustrates the coordinates of the nine target positions and the angles for each joint.

Table 5. Coordinates and angles of the target points.

Positions		Angles			
Points	**Coordinates**	θ	θ_2	θ_3	θ_4
A	(0.11,0.25,0.14)	2.41	1.26	1.705	−0.83
B	(0.21,0.32,0.22)	2.35	1.37	0.63	−0.029
C	(0.12,0.14,0.12)	2.11	1.17	1.21	0.75
D	(0.19,0.14,0.05)	2.08	1.58	1.36	−0.95
E	(0.19,−0.1,0.5)	1.25	0.3	0.59	1.04
F	(0.21,−0.16,0.7)	1.13	0.72	0.07	−1.17
G	(0.15,0.1,0.3)	1.94	0.45	1.35	0.93
H	(0.14,−0.11,0.14)	1.16	1.49	0.34	1.67
I	(0.12,0.15,0.05)	2.15	1.44	1.49	−0.17

Figure 7 shows the objective function f_{obj} for three ways of tuning PID parameters with 400 iterations, and the parameters of GSO were the same as set^3_{gso} in Table 3.

From the results, it can be observed that GSO converges faster than GA and PSO, because by establishing the angle of joints for the desired points, the angular trajectories are exported to the control system and then tuned by GSO. The performance of the closed-loop control system is validated for each joint, in which the angular trajectories solved by the IK are set as desired (θ_{d_i}). Table 6 represents the tuned PID parameters.



(**a**) Point A (**b**) Point B (**c**) Point C (**d**) Point D

Figure 7. The objective functions over iterations.

Table 6. PID parameters tuned by PSO.

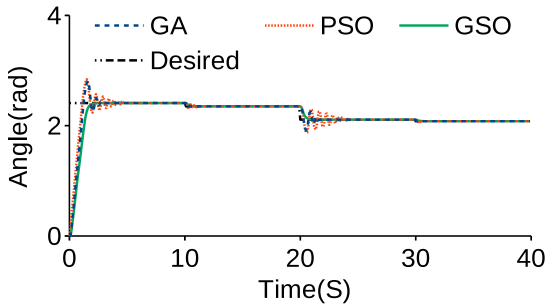
	PSO			GA			GSO		
	K_p	K_i	K_d	K_p	K_i	K_d	K_p	K_i	K_d
Joint 1	15.35	34.3233	2.8305	51.2684	175.3676	0.2106	36.2147	295.5165	0.2556
Joint 2	26.8257	25.2366	7.8637	59.9833	173.4063	8.6907	39.5278	242.1184	8.4769
Joint 3	13.3082	15.4017	3.3305	76.2230	160.6402	3.0397	83.4758	175.8795	3.3379
Joint 4	5.4971	6.4876	3.4602	83.6052	189.7420	7.7188	68.0942	183.7764	4.1200

Figure 8 and Table 7 compare the angular trajectories and average errors tuned by the GA, PSO and GSO while the end-effector moved to the desired positions and its PID parameters were tuned by the GA, PSO and GSO. Figures 9 and 10 show the error and angular velocity for each joint while the end-effector tracked points A, B, C, and D.

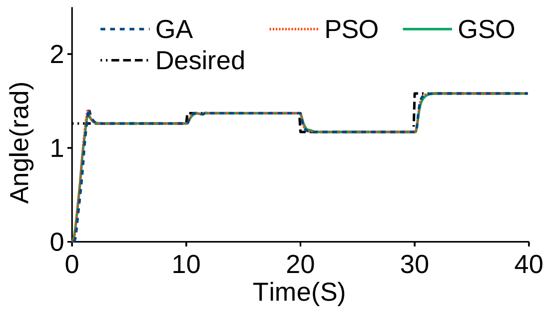
Table 7. Angular trajectory average error for optimal tuned controllers.

	AE_{GA}	AE_{PSO}	AE_{GSO}
Joint 1	0.0561	0.05871	0.5407
Joint 2	0.0348	0.0313	0.0304
Joint 3	0.0728	0.0794	0.0675
Joint 4	0.0621	0.0605	0.0488

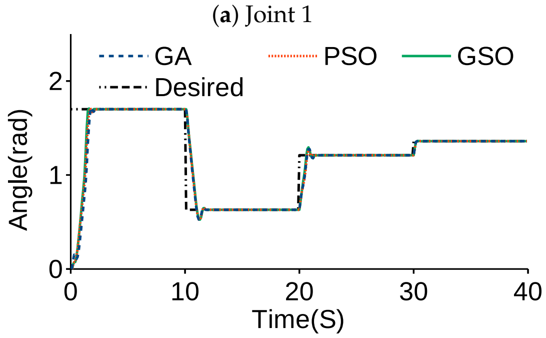
In Figures 8 and 9, overshoot can be observed when the joints are changing trajectories. In Figure 10, there are fluctuations when there is a change in position of the joints and they are tracking each point. The rest of the graph levels off at zero. This shows the stability of the control system, because while joints do not move and are in a stable situation, the joints do not shake. In Table 7, AE_{GA}, AE_{PSO} and AE_{GSO} are averages of the SSE for GA, PSO and GSO. The actual trajectory of each joint shows the significant effects of GSO compared to PSO and the GA due to the initialization of PSO by results of the GA for GSO, which allows the algorithm to find the global optima more accurately. From AE_{GA}, AE_{PSO} and AE_{GSO}, it can be concluded that the GSO resulted in a lower value because of its efficiency in finding precise results and lower chance of becoming trapped in local optima.



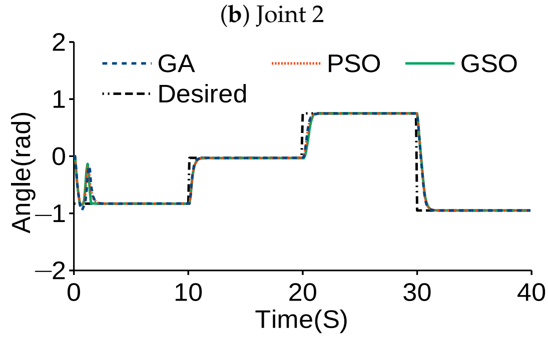
(**a**) Joint 1



(**b**) Joint 2

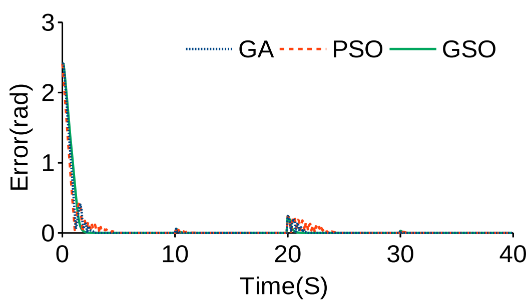


(**c**) Joint 3

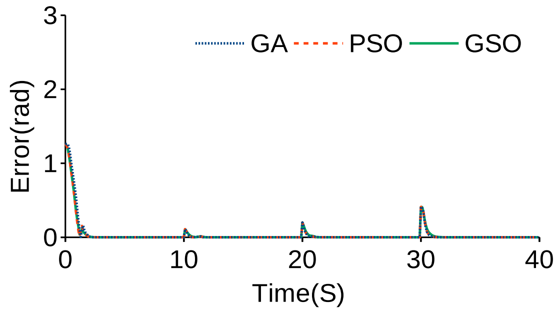


(**d**) Joint 4

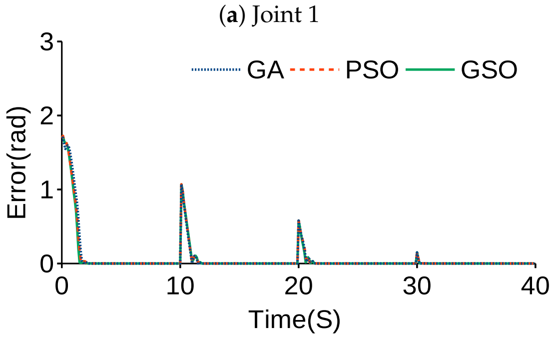
Figure 8. Angular trajectory of each joint.



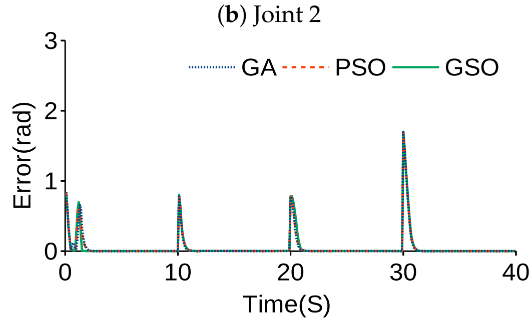
(**a**) Joint 1



(**b**) Joint 2

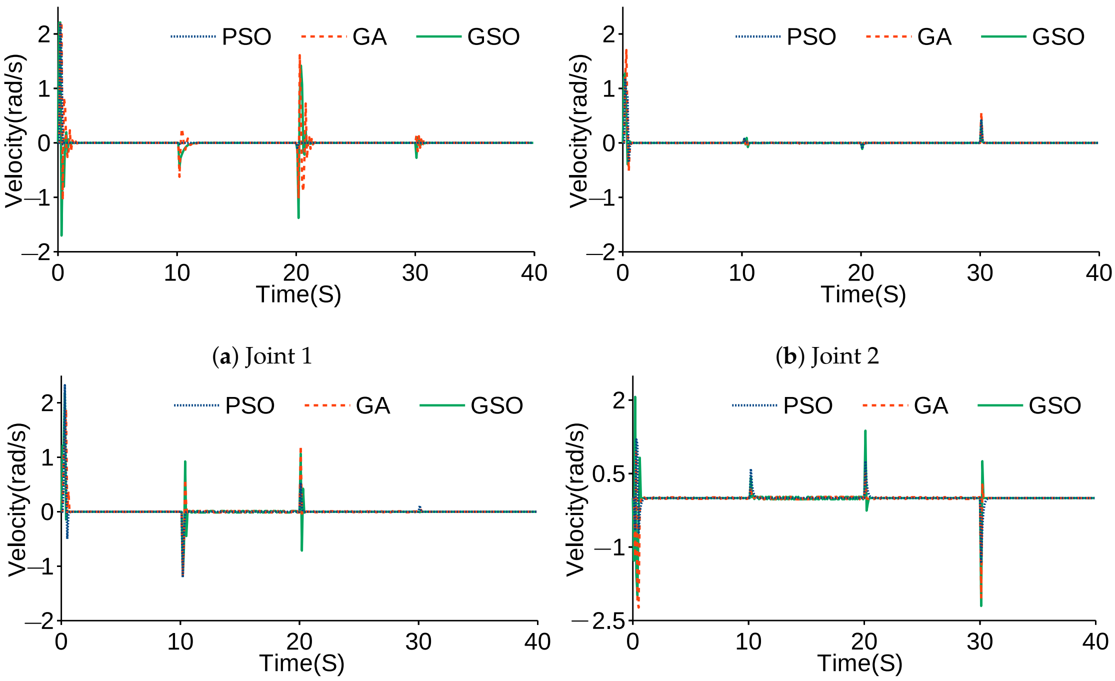


(**c**) Joint 3



(**d**) Joint 4

Figure 9. Angular trajectory error of each joint.



(**a**) Joint 1 (**b**) Joint 2

(**c**) Joint 3 (**d**) Joint 4

Figure 10. Angular velocity of each joint.

6. Conclusions

This paper presented a hybrid optimal IK solution of a 5DoF robotic arm to determine the joint trajectories based on its end-effector position. The Denavit–Hartenberg method was used to establish the kinematic and was solved by GSO, which is a combination of the GA and PSO. The trajectories were implemented with PID control as desired trajectories for each joint. The tuning of PID parameters was presented as optimization problem and carried out by GSO. A GUI was created to operate and visualize the performance of the robotic arm in a virtual environment. This method addresses the issue of finding one-to-many possible solutions of IK for a 5DoF robotic arm to control each joint efficiently and precisely to determine end-effector positions in Cartesian space.

The results show that GSO has a lower average error for each joint than PSO and GA. For instance, for joint 4, the average error for one specific path for GSO is 19.33% and 22.7% less than PSO and the GA, respectively. These results show that initialization particles for PSO by using the GA can give more accurate results and avoid algorithms becoming trapped in local optima. In addition, the mean computational time for GSO is lower than the GA by 4.45 s and higher than PSO by 1.37 s. Therefore, GSO is a sufficient algorithm for IK solution for robotic arms.

This method can be used for any robotic arms to control their end-effector. The limitation of this work is that we did not apply the hybrid proposed method to a real robotic arm. In addition, the target position of the end-effector was chosen by the user. Therefore, in future work, this position can be issued by sensors such as depth camera or tag marker measurement algorithms. Furthermore, the proposed IK and control system developed by the GSO algorithm was validated in the 3D simulation environment of Gazebo, and the effect of sensor noises was not considered; this can be covered in the future work. In addition, the proposed method can be tested for applications in which the accurate position of end-effectors is needed, such as welding, material handling and thermal spraying or any other industrial applications.

Author Contributions: Conceptualization, M.S.A. and R.R.; methodology, M.S.A. and R.R.; software, M.S.A.; validation, M.S.A. and R.R.; formal analysis, M.S.A. and R.R.; investigation, M.S.A. and

R.R.; resources, M.S.A.; data curation, M.S.A. and R.R.; writing—original draft preparation, M.S.A.; writing—review and editing, M.S.A. and R.R.; visualization, M.S.A.; supervision, R.R.; project administration, R.R.; funding acquisition, R.R.; All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank Universiti Kebangsaan Malaysia (UKM) and the Ministry of Higher Education Malaysia for the financial support received under research grants KK-2020-014 and DPK-2020-015.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Universiti Kebangsaan Malaysia (UKM) and the Ministry of Higher Education Malaysia for the financial support received under research grants KK-2020-014 and DPK-2020-015.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jahnavi, K.; Sivraj, P. Teaching and Learning Robotic Arm Model. In Proceedings of the International Conference on Intelligent Computing, Instrumentation and Control Technologies, Kerala, India, 6–7 July 2017; pp. 1570–1575. [CrossRef]
2. Quiros, A.R.F.; Abad, A.C.; Dadios, E.P. Object Locator and Collector Robotic Arm using Artificial Neural Networks. In Proceedings of the International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM, Cebu, Philippines, 9–12 December 2016; pp. 200–203. [CrossRef]
3. Pavlovčič, U.; Arko, P.; Jezeršek, M. Simultaneous hand–eye and intrinsic calibration of a laser profilometer mounted on a robot arm. *Sensors* **2021**, *21*, 1037. [CrossRef]
4. Shah, S.K.; Mishra, R.; Ray, L.S. Solution and Validation of Inverse Kinematics using Deep Artificial Neural Network. *Mater. Today Proc.* **2020**, *26*, 1250–1254. [CrossRef]
5. Shirafuji, S.; Ota, J. Kinematic Synthesis of a Serial Robotic Manipulator by Using Generalized Differential Inverse Kinematics. *IEEE Trans. Robot.* **2019**, *35*, 1047–1054. [CrossRef]
6. Sun, P.; Li, Y.B.; Wang, Z.; Chen, K.; Chen, B.; Zeng, X.; Zhao, J.; Yue, Y. Inverse displacement analysis of a novel hybrid humanoid robotic arm. *Mech. Mach. Theory* **2020**, *147*, 103743. [CrossRef]
7. Megalingam, R.K.; Boddupalli, S.; Apuroop, K.G.S. Robotic Arm Control through Mimicking of Miniature Robotic Arm. In Proceedings of the 4th International Conference on Advanced Computing and Communication Systems, ICACCS, Coimbatore, India, 6–7 January 2017; pp. 4–10. [CrossRef]
8. Xu, Y.; Ding, C.; Shu, X.; Gui, K.; Bezsudnova, Y.; Sheng, X.; Zhang, D. Shared control of a robotic arm using non-invasive brain–computer interface and computer vision guidance. *Robot. Auton. Syst.* **2019**, *115*, 121–129. [CrossRef]
9. Fang, B.; Ma, X.; Wang, J.; Sun, F. Vision-based posture-consistent teleoperation of robotic arm using multi-stage deep neural network. *Robot. Auton. Syst.* **2020**, *131*, 103592. [CrossRef]
10. Narayan, J.; Mishra, S.; Jaiswal, G.; Dwivedy, S.K. Novel design and kinematic analysis of a 5-DOFs robotic arm with three-fingered gripper for physical therapy. *Mater. Today Proc.* **2020**, *28*, 2121–2132. [CrossRef]
11. Ye, H.; Wang, D.; Wu, J.; Yue, Y.; Zhou, Y. Forward and inverse kinematics of a 5-DOF hybrid robot for composite material machining. *Robot. Comput. Integr. Manuf.* **2020**, *65*, 101961. [CrossRef]
12. Wei, H.; Bu, Y.; Zhu, Z. Robotic arm controlling based on a spiking neural circuit and synaptic plasticity. *Biomed. Signal Process. Control* **2020**, *55*, 101640. [CrossRef]
13. Ren, H.; Ben-Tzvi, P. Learning Inverse Kinematics and Dynamics of a Robotic Manipulator using Generative Adversarial Networks. *Robot. Auton. Syst.* **2020**, *124*, 103386. [CrossRef]
14. Kang, M.; Shin, H.; Kim, D.; Yoon, S.E. TORM: Fast and accurate trajectory optimization of redundant manipulator given an end-effector path. *IEEE Int. Conf. Intell. Robot. Syst.* **2020**. [CrossRef]
15. Jin, M.; Liu, Q.; Wang, B.; Liu, H. An Efficient and Accurate Inverse Kinematics for 7-DOF Redundant Manipulators Based on a Hybrid of Analytical and Numerical Method. *IEEE Access* **2020**, *8*, 16316–16330. [CrossRef]
16. Cao, Y.; Gan, Y.; Dai, X. Kinematic Optimization of Redundant Manipulators to Improve the Fault-Tolerant Control. In Proceedings of the 31st Chinese Control and Decision Conference, Nanchang, China, 3–5 June 2019, [CrossRef]
17. Starke, S.; Hendrich, N.; Zhang, J. Memetic Evolution for Generic Full-Body Inverse Kinematics in Robotics and Animation. *IEEE Trans. Evol. Comput.* **2019**, *23*, 406–420. [CrossRef]
18. Jamali, B.; Rasekh, M.; Jamadi, F.; Gandomkar, R.; Makiabadi, F. Using PSO-GA algorithm for training artificial neural network to forecast solar space heating system parameters. *Appl. Therm. Eng.* **2019**, *147*, 647–660. [CrossRef]
19. Chen, Q.; Chen, Y.; Jiang, W. Genetic particle swarm optimization–based feature selection for very-high-resolution remotely sensed imagery object change detection. *Sensors* **2016**, *16*, 1024. [CrossRef]

20. Ajmi, N.; Helali, A.; Lorenz, P.; Mghaieth, R. MWCSGA—Multi Weight Chicken Swarm Based Genetic Algorithm for Energy Efficient Clustered Wireless Sensor Network. *Sensors* **2021**, *21*, 791. [CrossRef]
21. Dziwinski, P.; Bartczuk, L. A New Hybrid Particle Swarm Optimization and Genetic Algorithm Method Controlled by Fuzzy Logic. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 1140–1154. [CrossRef]
22. Farnad, B.; Jafarian, A.; Baleanu, D. A new hybrid algorithm for continuous optimization problem. *Appl. Math. Model.* **2018**, *55*, 652–673. [CrossRef]
23. Mahmud, M.; Motakabber, S.; Zahirul Alam, A.H.; Nordin, A.N. Adaptive PID Controller Using for Speed Control of the BLDC Motor. In Proceedings of the IEEE International Conference on Semiconductor Electronics, Kuala Lumpur, Malaysia, 28–29 July 2020; pp. 168–171. [CrossRef]
24. Misaghi, M.; Yaghoobi, M. Improved invasive weed optimization algorithm (IWO) based on chaos theory for optimal design of PID controller. *J. Comput. Des. Eng.* **2019**, *6*, 284–295. [CrossRef]
25. Amiri, M.S.; Ramli, R.; Ibrahim, M.F. Initialized Model Reference Adaptive Control for Lower Limb Exoskeleton. *IEEE Access* **2019**, *7*, 167210–167220. [CrossRef]
26. Castillo-Zamora, J.J.; Camarillo-GóMez, K.A.; Perez-Soto, G.I.; Rodriguez-Resendiz, J. Comparison of PD, PID and Sliding-Mode Position Controllers for V-Tail Quadcopter Stability. *IEEE Access* **2018**, *6*, 38086–38096. [CrossRef]
27. Demirel, B.; Ghadimi, E.; Quevedo, D.Q.; Johansson, M. Optimal Control of Linear Systems with Limited Control Actions: Threshold-Based Event-Triggered Control. *IEEE Trans. Control. Netw. Syst.* **2018**, *5*, 1275–1286. [CrossRef]
28. Ma, Z.; Yan, Z.; Shaltout, M.L.; Chen, D. Optimal Real-Time Control of Wind Turbine During Partial Load Operation. *IEEE Trans. Control. Syst. Technol.* **2015**, *23*, 2216–2226. [CrossRef]
29. Belkadi, A.; Oulhadj, H.; Touati, Y.; Khan, S.A.; Daachi, B. On the robust PID adaptive controller for exoskeletons: A particle swarm optimization based approach. *Appl. Soft Comput.* **2017**, *60*, 87–100. [CrossRef]
30. Phu, D.X.; Mien, V.; Tu, P.H.T.; Nguyen, N.P.; Choi, S.B. A New Optimal Sliding Mode Controller with Adjustable Gains based on Bolza-Meyer Criterion for Vibration Control. *J. Sound Vib.* **2020**, *485*, 115542. [CrossRef]
31. Suhaimin, S.C.; Azmi, N.L.; Rahman, M.M.; Yusof, H.M. Analysis of Point-to-Point Robotic Arm Control using PID Controller. In Proceedings of the 7th International Conference on Mechatronics Engineering, ICOM, Putrajaya, Malaysia, 30–31 October 2019; pp. 5–10. [CrossRef]
32. Kang, S.; Chou, W. Kinematic Analysis, Simulation and Manipulating of a 5-DOF Robotic Manipulator for Service Robot. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation, Tianjin, China, 4–7 August 2019; pp. 643–649. [CrossRef]
33. Mukherjee, S.; Goswami, D.; Chatterjee, S. A Lagrangian Approach to Modeling and Analysis of a Crowd Dynamics. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *45*, 865–876. [CrossRef]
34. Wu, J.; Gao, J.; Song, R.; Li, R.; Li, Y.; Jiang, L. The design and control of a 3DOF lower limb rehabilitation robot. *Mechatronics* **2016**, *33*, 13–22. [CrossRef]
35. Sun, J.D.; Cao, G.Z.; Li, W.B.; Liang, Y.X.; Huang, S.D. Analytical inverse kinematic solution using the D-H method for a 6-DOF robot. In Proceedings of the 14th International Conference on Ubiquitous Robots and Ambient Intelligence, Jeju, Korea, 28 June–1 July 2017; pp. 714–716. [CrossRef]
36. Al-Oqaily, A.T.; Shakah, G. Solving Non-Linear Optimization Problems Using Parallel Genetic Algorithm. In Proceedings of the 8th International Conference on Computer Science and Information Technology, Amman, Jordan, 11–12 July 2018; pp. 103–106. [CrossRef]
37. Amiri, M.S.; Ramli, R.; Ibrahim, M.F. Optimal parameter estimation for a DC motor using genetic algorithm. *Int. J. Power Electron. Drive Syst. (IJPEDS)* **2020**, *11*, 1047–1054. [CrossRef]
38. Xu, S. Predicted Residual Error Sum of Squares of Mixed Models: An Application for Genomic Prediction. *G3 Genes Genomes Genet.* **2017**, *7*, 895–909. [CrossRef]
39. Rayno, J.; Iskander, M.F.; Kobayashi, M.H. Hybrid Genetic Programming with Accelerating Genetic Algorithm Optimizer for 3-D Metamaterial Design. *IEEE Antennas Wirel. Propag. Lett.* **2016**, *15*, 1743–1746. [CrossRef]
40. Cheng, Y.F.; Shao, W.; Zhang, S.J.; Li, Y.P. An Improved Multi-Objective Genetic Algorithm for Large Planar Array Thinning. *IEEE Trans. Magn.* **2016**, *52*, 1–4. [CrossRef]
41. Amiri, M.S.; Ramli, R.; Ibrahim, M.F. Genetically optimized parameter estimation of mathematical model for multi-joints hip–knee exoskeleton. *Robot. Auton. Syst.* **2020**, *125*, 103425. [CrossRef]
42. Amiri, M.S.; Ramli, R.; Ibrahim, M.F. Hybrid design of PID controller for four DoF lower limb exoskeleton. *Appl. Math. Model.* **2019**, *72*, 17–27. [CrossRef]
43. Sumathi, S.; Paneerselvam, S. *Computational Intelligence Paradigms Theory and Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2010. [CrossRef]
44. Amiri, M.S.; Ramli, R.; Ibrahim, M.F.; Wahab, D.A.; Aliman, N. Adaptive Particle Swarm Optimization of PID Gain Tuning for Lower-Limb Human Exoskeleton in Virtual Environment. *Mathematics* **2020**, *8*, 2040. [CrossRef]
45. Amiri, M.S.; Ramli, R.; Tarmizi, M.A.A.; Ibrahim, M.F.; Danesh Narooei, K. Simulation and Control of a Six Degree of Freedom Lower Limb Exoskeleton. *J. Kejuruter.* **2020**, *32*, 197–204. [CrossRef]