Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 13 NO: 4 PAGE: 950-968 YEAR: 2024
DOI: 10.17798/bitlisfen.1482747

Q Learning Based PSO Algorithm Application for Inverse Kinematics of 7-DOF Robot Manipulator

Murat Erhan ÇİMEN[1*]

[1]*Sakarya University of Applied Sciences, Faculty of Technology, Department of Electrical and Electronic Engineering, Sakarya, Türkiye*
(ORCID: 0000-0002-1793-485X)

Keywords: Robot kinematics, Reinforcement learning, Q learning, PSO.

Abstract

Solving inverse kinematics problems is one of the fundamental challenges in serial robot manipulators. In this study, a learning-based algorithm was developed to minimize the complexity of solving the inverse kinematics problem for a 7-degree-of-freedom serial manipulator. The parameters of the Particle Swarm Optimization algorithm, modified with Q-learning, a reinforcement learning technique, are updated depending on the states. This approach aimed to increase the efficiency of the algorithm in finding solutions. In the simulation studies, two different end positions of the robot, measured in meters, were used to compare the performance of the proposed algorithm. The location error of the proposed algorithm was statistically compared, and meaningful results were obtained regarding the reliability of the outcomes through Wilcoxon analysis. The simulation results demonstrated that the reinforcement learning-based particle swarm optimization algorithm can be effectively used for inverse kinematics solutions in serial robot manipulators.

1. Introduction

Nowadays, robots have entered many different areas in various sectors and have provided many conveniences to our lives [1], [2]. Robots, in particular, have become an important factor in industrial systems due to reasons such as their adaptation to different places, their ability to perform different tasks like humans, and their widespread use [1], [2], [3], [4]. When looked at the industry, there are many types of robots, large and small, fixed, autonomous, serial or parallel [2], [5], [6], [7]. Sectors and companies using this technology can become the pioneers of the sector as they can increase their production capacity and profit margins. Therefore, issues such as position control, motion control, acceleration or thrust controls, and structural designs of robots are examined by many different researchers [4], [8].

In industry, kinematic equations of robots must be derived and designs must be made before robots can be used [9]. Therefore, it can be said that the derivation of advanced and inverse kinematic equations in the control of robots is the first stage [4]. Thanks to these equations, the angle between the robot's joints or axes or the distance between the linear distances can be used to determine the position and direction of the robot, thanks to advanced kinematic equations. Or, given the direction and the positions it needs to go, the angles or distances between the joints in the robot can be calculated using inverse kinematic equations [1].

Analytical and iterative methods could be used to solve the inverse kinematics problem in robots. Solutions made with analytical methods become difficult in robots with high degrees of freedom due to the increase in degrees of freedom depending on the motors used in the robot [1], [10]. It requires a good background, skill and ability, especially in deriving and solving the inverse kinematic equations of robotic arms with higher degrees of freedom. On the other hand, the inverse kinematics problem can be solved with iterative methods. However, since the computation burden is high, their solutions take time.

Reinforcement learning is one of the machine learning methods [11], [12], [13]. Reinforcement learning is a type of learning based on perceiving the environment in which any agent (individual or person) is located, interacting with it,

* Corresponding author: muratcimen@subu.edu.tr
Received: 12.05.2024, Accepted: 25.09.2024

regulating its behavior according to this interaction, and taking action in accordance with its goal [13], [14], [15], [16], [17], [18]. The learning process is reward-based work. Therefore, the agent adopts the behavior or action that provides the most reward in order to achieve his goal. He gets used to the action that gives the least reward or punishment and learns to do that action less. For example, while a player makes moves or follows strategies in which he earns more points in any game, he is careful not to make strategies in which he earns fewer points or loses the game. In terms of reinforcement learning, the player becomes an agent (individual or person) and learns the game by interacting with the environment. Then he makes the moves that will earn him the most points. Therefore, reinforcement learning can be applied to many optimization problems in the game industry, industrial control applications, image processing applications, path planning or industry.

Optimization is the process of determining variables to minimize or maximize a certain objective criterion [19], [20], [21], [22], [23], [24], [25]. Optimization is encountered in almost every field, from logistics to finance, from chemistry to health, from machinery to electrical and electronics, from health to tourism, from automotive to construction, from medicine to food and retail, from education to social sciences [19], [26], [27], [28], [29]. The reason is that in the problem to be applied, there is a goal criterion that is desired to be maximized or minimized, and there are variables that will ensure this. In addition to classical optimization methods, there are swarm-based optimization algorithms inspired by nature. These algorithms were developed inspired by the behavior of living creatures in nature and are more successful in swarm-based global search [19], [24]. These algorithms can be given as Genetic Algorithm [30], [31], Particle Swarm Optimization (PSO) [20], [32], [33], Firefly Optimization (FA) [34], [35], Cuckoo Search Optimization [36], Gray Wolf Optimization [37], Flower Pollination Optimization [38], Whale Optimization Algorithm [35], [39] and many more. For instance, PSO is one of the well-known swarm-based optimization algorithms inspired by nature. PSO was developed by taking inspiration from the natural behaviors of fish and birds, such as finding food and escaping from predators [20], [32], [33]. In PSO, swarm experience and each particle's (fish or bird) speed are repeatedly used to solve optimization problem. By this means, algorithm try to find best particle in swarm. Overall, PSO is a basic and easy to apply any scientific or engineering optimization problem. Moreover, setting only a few parameters is sufficient. On the contrast, this algorithm suffers both complexity with high dimension, is very sensitive to particles optimism and particles' speed [3], [33], [40]. Especially, there are many studies to improve the performance of PSO algorithm. These studies are usually about parameter tuning or control of parameters. Parameter tuning is usually performed before the optimization process [41], [42], [43], [44]. The parameter tuning approach produces more efficient and successful results in some problems compared to the simple PSO method. However, its disadvantage is that some adaptation features are lost when the algorithm with the adjusted parameters is run. The other method, parameter control, adapts the algorithm to the specified conditions while the algorithm is running and a more dynamic optimization process is performed. In general, historical experience [45], [46], small test period [47], fuzzy logic [48], [49] and reinforcement learning methods [18], [50] are used in control-based PSO. However, there's no longer a need to manually design rules and fine-tune parameters, significantly reducing the burden for users. While parameters in Reinforcement Learning (RL) are important, experiments have demonstrated that a single set of parameters can yield good results across various algorithms and test functions. Consequently, in practice, adjusting RL parameters is often unnecessary. Additionally, by leveraging past experiences, the algorithm's applicability becomes broader, and its effectiveness improves. For instance, Liu et al. proposed the reinforcement learning method in the PSO method and applied this method to some multi-objective problems. In this adaptive method, the Q table has four states. These states, which can be adjusted adaptively, are created from the best particle among the particles used by the PSO algorithm and the particle in the swarm. The output of the Q table is determined as the inertia and correlation coefficients, which are the parameters of PSO [18]. Xu and Pi similarly tried to increase the performance of Q learning method by associating it with PSO [51]. However, unlike [18], they used different topologies while creating the states of Q table and added the diversity within the swarm in addition to the performance of the algorithm while creating the reward function. Xu and his colleagues similarly applied the reinforcement learning method to PSO algorithm. Unlike [18], they developed the algorithm by training an artificial intelligence model via deep deterministic policy (DDP) method. They applied DDP method and set its inputs as iteration, diversity and reward value obtained in the previous iteration. They determined the output as the parameters of PSO, inertia and correlation coefficients [50].

In the literature, inverse kinematic problems were usually solved by conventional methods [52]. It is a much more complicated and time-consuming solution because of non-linear equations [1]. These studies are still ongoing. Düzgün proposed new methods for solving the kinematics of robots in his doctoral thesis [2]. In addition to these methods, especially in recent years, artificial intelligence, fuzzy logic, metaheuristic optimization algorithms or reinforcement learning methods have been used to solve this problem. Koker et al. presented the inverse kinematic solution of a 3-joint robot using artificial intelligence [53]. Özüdoğru, in his master's thesis, used constructive artificial neural network to determine the joint angles and trajectory of industrial robots according to the target position [1]. Alamdar et al. presented an alternative solution to the inverse kinematic problem with the found logic-based ANFIS method [54]. The basic logic of the studies [53], [1], [54] is to collect data and determine the joint angles or joint distances of the robot using machine intelligence according to the collected data. In the study in [53], constructive neural networks, which are more successful in training, were used in [1], unlike the study. This proposed structure was implemented in real time on an industrial robot and successful results were obtained. On the other hand, in [54], they used ANFIS to be an alternative solution to solving kinematic problems. The superiority of ANFIS; It can produce successful results especially in situations containing uncertainty, as well as other artificial intelligence models. The reason for this is that it fuzzifies the fuzzy logic inputs by means of membership functions and then performs the clarification process according to the rules. The disadvantage is that many rules and membership functions need to be selected appropriately. Segoto et al. have collected synthetic data consisting of angles, speeds, torque from an industrial robot and trained it on an artificial neural network model. Then, they compared the performance of the trained model with machine learning logic [8]. Jin et al. have modeled a robot with six degrees of freedom. They used interpolation methods to determine joint angles [55]. However, Dereli transformed the inverse kinematic problem of a 7-axis robot with more joints into an optimization problem and then used the Artificial Bee Algorithm (ABC) algorithm to solve this problem [3]. In addition to this study, in [7], Quantum based PSO algorithm was applied to the same problem and more efficient and successful results were obtained. Quantum based PSO algorithm is the development of particles by affecting the quantum physics. Due to this feature, it has produced more successful results in finding the global search optimal than PSO algorithm. In addition, Quantum based PSO algorithm was compared to ABC algorithm in solving the inverse kinematic problem of a 7-DOF robot. Advantages of Quantum based PSO algorithm are the shorter computation time, fewer iterations and the number of particles. In the study conducted in [56], a similar process was performed with PSO for the target position determined by using the kinematic equations of a 7-DOF robot. However, the difference of this study is that the processing time and trajectories of the manipulator were optimized. In [57], an optimization study was performed to reduce the energy cost of a robot with a different objective criterion. Another machine learning method used in the control of robots is the reinforcement learning method. The advantage of reinforcement learning is that the robot is controlled by learning the behavior of the robot according to a reward function from a model determined by the researcher without needing the exact kinematic equation of the robot. In this direction, Avery et al. in their study, they carried out a study on the movement, speed and path planning of a robot with the incremental reinforcement learning method [6] In their study, Hou and Li proposed a reinforcement learning method for a 6-axis robot to grasp objects used in daily life that it recognizes in the image, and they achieved successful results [58].

In this study, a reinforcement learning-based particle swarm optimization algorithm was developed to solve a seven-axis robot inverse kinematics problem, and the problem was solved with particle swarm optimization and Q learning-based Particle Swarm optimization algorithm. Differences of the proposed model from [18], [51] are parameters and golden ratio has been inserted. The performance of the proposed reinforcement learning based particle swarm optimization algorithm was statistically compared with the mean value and standard deviation over different swarm sizes, iteration numbers and parameters. It has been observed that the proposed Q learning based Particle Swarm Optimization Algorithm produces successful results. Briefly, main contributions of the study as follows;

1. Q learning PSO algorithm (PSO-RL-Q) was developed using golden ratio that is used while calculating states,
2. PSO-RL-Q was applied to solve inverse kinematics of 7-DOF Robot Manipulator,
3. Statistical analysis and Wilcoxon test for results of proposed PSO-RL-Q were made to evaluate the performance.

2. Material and Method

2.1. Partical Swarm Optimization

Particle Swarm Optimization, proposed by Kennedy and Eberhart in 1995, is an algorithm inspired by the behavior of swarm of birds and fish [33]. It is an algorithm developed by modeling the behavior of each individual in the swarm, such as finding food and avoiding predators. Each individual in the swarm has a position (x_t) and speed (v_t) [19], [33], [59], [60]. Each individual interacts with other individuals in the swarm and iteratively updates its position and speed. This process takes place as in Equation 1.

$$v_{t+1} = \mathrm{w}v_t + c_1 r_1(p_{best} - x_t) + c_2 r_2(g_{best} - x_t) \quad (1)$$

$$x_{t+1} = x_t + v_t$$

As seen in Equation 1, the speeds of individuals are affected by g_{best}, which is in the best position in the swarm, and p_{best}, which is in the best position in the iteration. These effects are weighted with correlation coefficients (c_1, c_2) and random values between 0-1 in each iteration (r_1, r_2). In addition, the speed (x_t) of each individual is multiplied by a weight factor (w) and reflected in the next speed. The next location information of the individuals (x_{t+1}) is also updated by collecting their current location (x_t) and speed (v_t) [19], [33], [59], [60].

2.2. Q Learning

Q learning algorithm, one of the machine learning methods, dates back to Bellman's studies on optimal control theory, that is, in the 1950s [11], [61]. Bellman's work, which tried to solve the dynamic optimization problem to find the control signal in discrete systems, formed the basis of the Q learning algorithm in later years. Q learning algorithm basically requires an agent to interact with the environment according to its own knowledge and experience. After this interaction, according to the data collected from the environment, the agent learns the environment according to a reward/punishment value and decides its next move according to the current situations[12], [62]. As shown in Figure 1, the Agent generates a movement signal from the environment according to the situation. According to this signal, the agent moves within the environment and moves to the next state. The next state also produces a reward signal [14].

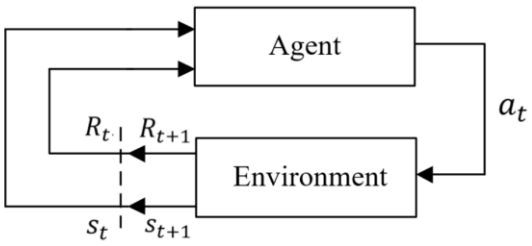

Figure 1. Environment and Agent Interaction.

Q learning method and State-Action-Reward-State-Action (SARSA) methods are model-independent or model-free reinforcement learning methods [13], [17]. Reinforcement learning also has a feature that learns mostly from behavior. It achieves this during the interaction of an agent with the environment. The determined agent interacts with the environment in a certain way and produces an output called reward by evaluating the outputs of the environment. Depending on the current state of the agent, its interaction with the environment, and the reward for the agent's next state, the agent that learns the environment begins to act in a way that will earn a higher reward at each step. They can be preferred especially in applications that are difficult to model, as they can learn by experiencing the results of actions rather than using a model [14]. In particular, reinforcement learning is provided by the Bellman equation suggested by Richard Bellman and used by Watkin in reinforcement learning, and the actions and outcomes depending on the situations are learned. In this learning method, the system is learned in terms of situations and action and reward value rather than a specific model. The simplest version of the Bellman equation used in Q learning is given in Equation 2. s_t used in the equation is for the instantaneous state at time t, a_t is for the instantaneous action at time t, s_{t+1} is for the instantaneous state at time t+1, $R(s_t, a_t)$ is for the reward value of s_t and a_t, $Q(s_t, a_t)$ is for the value of s_t and a_t. state of training value, α is called the learning factor, and γ is called the discount factor. $\max_a Q(s_{t+1}, a)$ is the value at which the maximum Q value is produced according to the α action in case s_{t+1}. By performing this process in each iteration, Q values are updated [11], [12], [14], [17].

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (2)$$

Additionally, the epsilon greedy method is applied to determine Q values. This method allows the Agent to visit all possible states during learning.

Algorithm 1. Q learning Pseudocode

Input:
State (s)
Action (a_t)
Learning rate (α)
Discount factor (γ)
Reward $R(s_t, a_t)$
Updated table $Q(s_t, a_t)$
Output:
Selected action according to updating table $Q(s_t, a_t)$
For iter=1, Max_iteration do
  Initialise state s_t
  For t=1, Max_iteration do
    Choose a_t with ϵ greedy probability
    Execute a_t and observe state s_{t+1} and reward r_t
    Update table $Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$
  End for
End for

Thus, the q values that will maximize the reward by learning the environment better are determined. The pseudo code of the Q learning algorithm is given in Figure 1. This method is an off policy and the learning agent learns the value function based on the current action derived from the policy currently in use. The pseudocode of the Q learning algorithm is given in Algorithm 1.

2.3. Kinematic Model of the Robot

The homogeneous transformation matrix is expressed by Equation 3. Transformation matrices of the serial robot manipulator are obtained by the David Hartembert method (DH) [4], [9]. The DH parameters listed in Table 1 represent the connection length (l_i), connection angle (α_i), connection offset (d_i) and joint angles (θ_i). Transformation matrices of adjacent link coordinate frames are obtained according to the DH parameters of the robotic manipulator given in Figure 2 and Equation 1. Equation 5 is obtained by multiplying the six transformation matrices in Equation 4. Lengths are in meters and angles are in degrees [3].

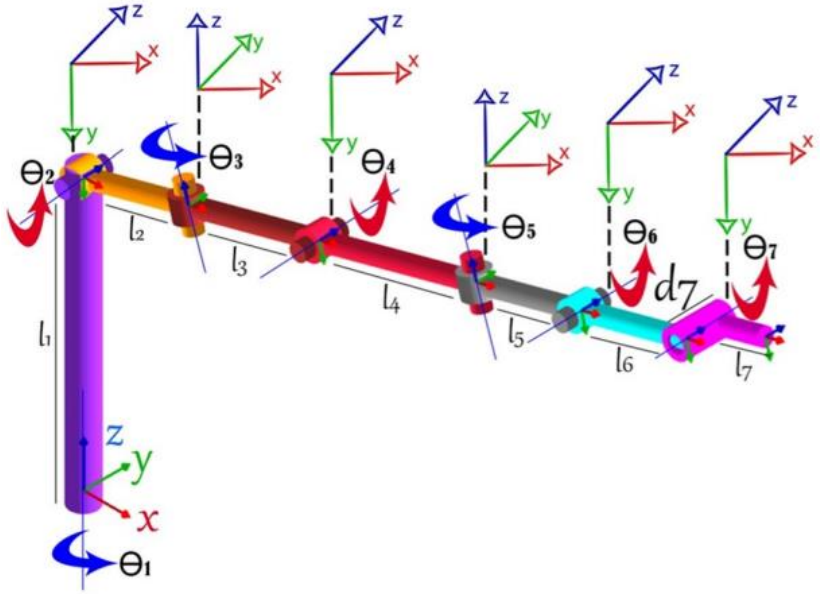

Figure 2. 7-DOF Robot

$$ {}^{i-1}_{i}T\big(cos(\theta_i)\big) = \begin{bmatrix} cos(\theta_i) & -cos(\alpha_i)sin(\theta_i) & sin(\alpha_i)sin(\theta_i) & l_i cos(\theta_i) \\ sin(\theta_i) & cos(\alpha_i)cos(\theta_i) & -cos(\theta_i)sin(\alpha_i) & l_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^{1}_{0}T = \begin{bmatrix} cos(\theta_1) & 0 & -sin(\theta_1) & 0 \\ sin(\theta_1) & 0 & cos(\theta_1) & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{1}T = \begin{bmatrix} cos(\theta_2) & 0 & sin(\theta_2) & l_2 cos(\theta_2) \\ sin(\theta_2) & 0 & -cos(\theta_2) & l_2 sin(\theta_2) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{2}T = \begin{bmatrix} cos(\theta_3) & 0 & sin(\theta_3) & l_2 cos(\theta_3) \\ sin(\theta_3) & 0 & -cos(\theta_3) & l_2 sin(\theta_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{4}_{3}T = \begin{bmatrix} cos(\theta_4) & 0 & sin(\theta_4) & l_4 cos(\theta_4) \\ sin(\theta_4) & 0 & -cos(\theta_4) & l_4 sin(\theta_4) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{4}T = \begin{bmatrix} cos(\theta_5) & 0 & -sin(\theta_5) & l_5 cos(\theta_5) \\ sin(\theta_5) & 0 & cos(\theta_5) & l_5 sin(\theta_5) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{6}_{5}T = \begin{bmatrix} cos(\theta_6) & -sin(\theta_6) & 0 & l_6 cos(\theta_6) \\ sin(\theta_6) & cos(\theta_6) & 0 & l_6 sin(\theta_6) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

$$ {}^{7}_{6}T = \begin{bmatrix} cos(\theta_7) & -sin(\theta_7) & 0 & l_7 cos(\theta_7) \\ sin(\theta_7) & cos(\theta_7) & 0 & l_7 sin(\theta_7) \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{7}T = {}^{0}_{1}T(\theta_1){}^{2}_{1}T(\theta_2){}^{4}_{3}T(\theta_3){}^{5}_{4}T(\theta_4){}^{6}_{5}T(\theta_5){}^{7}_{6}T(\theta_6) = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

2.3. Objective Function

In the application of reinforcement learning based Particle swarm optimization to inverse kinematics equations, each individual (x_t) is the joint variables $(\theta_1^o, \theta_2^o, \theta_3^o, \theta_4^o, \theta_5^o, \theta_6^o, \theta_7^o)$ of the 7-axis serial robot manipulator [3]. In order to provide an optimal solution, the end effector reaching the target position is achieved through optimal adjustments. The robot arm has many destination paths from its starting point to its destination. The important issue at this point is that the manipulator reaches the target with minimum error with the fitness function.

$$ E_r = \sqrt{(P_x - P_x')^2 + \left(P_y - P_y'\right)^2 + (P_z - P_z')^2} \tag{6} $$

The positions to be calculated P_x', P_y', P_z' given in Equation 6 represent the target position of the end effector and E_r represents the error between P and P'. The aim of this study is to minimize the E_r error value.

2.3. Q Learning Based Particle Swarm Optimization

Q learning method is one of the Reinforcement Learning algorithms [13], [14]. By integrating Q learning into particle swarm optimization, it is aimed to update the parameters depending on the situation and thus increase the performance of the Particle Swarm optimization method. In this direction, situations were determined to be used within the Q learning method. In order to determine these states, $f_{min}, f_{max}, v_{min}, v_{max}, d_{min}, d_{max}$ values to be used in the states are determined. The pseudocode for performing this operation is given in Algorithm 2.

Algorithm 2. Pseudocode for calculation of $f_{min}, f_{max}, v_{min}, v_{max}, d_{min}, d_{max}$ values

$f_{min} = func(x(t)), f_{max} = func(x(t)), v_{min} = \|v(t)\|, v_{max} = \|v(t)\|,$
$d_{min} = \|x(t) - p\|, d_{max} = \|x(t) - p\|$
For t=1, maximum generation **do**
  If $f_{min} > func(x(t))$
    $f_{min} > func(x(t))$
  End If
  If $f_{max} < func(x(t))$
    $f_{max} = func(x(t))$
  End If
  If $v_{min} > \|v(t)\|$
    $v_{min} = \|v(t)\|$
  End If
  If $v_{max} < \|v(t)\|$
    $v_{max} = \|v(t)\|$
  End If
  If $d_{min} > \|x(t) - p\|$
    $d_{min} = \|x(t) - p\|$
  End If
  If $d_{max} < \|x(t) - p\|$
    $d_{min} = \|x(t) - p\|$
  End If
End for

Values have been generated by comparing the $f_{min}, f_{max}, v_{min}, v_{max}, d_{min}, d_{max}$ produced with Psoudecode with the x, d, v variables used in PSO, and corresponding states would be created. This part is inspired by golden section search. The golden ratio is a ratio that can be found in the shape and structure of countless living and non-living entities in nature. The golden ratio is a numerical ratio that was discovered by the ancient Egyptian and Greek civilizations and has been applied to works of art such as sculpture, painting, and architecture for centuries. The value of the golden ratio is $\emptyset = \frac{1+\sqrt{5}}{2} \cong 1.61803$. As seen in Figure 3, there is a large piece |AC| (L) and a small piece |BC| (S) between points A and B. As seen in Figure 3 and in Equality 7, the ratio of the large piece to the small piece is a situation where it is equal to a fixed value ratio. When Equation 7 is arranged and Equation 8 is obtained, the ratio of the small piece |BC| to |AC| is calculated as 0.61803. In this study, the value $\emptyset - 1 = 0.61803$ was used from this value.

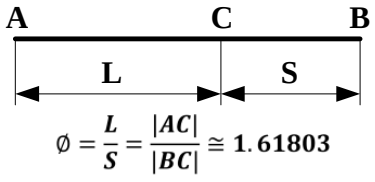

Figure 3. Golden ratio

$$\emptyset = \frac{L}{S} = \frac{|AC|}{|BC|} \cong \frac{1+\sqrt{5}}{2} \cong 1.61803 \quad (7)$$

$$\frac{1}{\emptyset} = \frac{S}{L} = \frac{2}{1+\sqrt{5}} = \emptyset - 1 \cong 0.61803 \quad (8)$$

When environment and nature are observed, the golden ratio is encountered in many places. One of the first places where the golden ratio is used in architecture is seen in Figure 4 and Figure 5. Or, as a few examples from nature itself, the proportions of snail shells, plants and human limbs are also places that contain the golden ratio, as seen in Figure 6 [63].

The golden ratio, known for centuries, is a method that has been applied to optimization problems before [64]. The golden ratio is aimed to be developed as a new and highly efficient method by adding it to the Q learning algorithm used with PSO. Therefore, in this study, the state assignment was carried out by comparing the objective value, speed and distance to the local best position of the particles or individuals used in the states produced for the swarm element, with the golden ratio. For example, according to the objective criterion to be produced for the herd element at index t, the value of $state_1$ is calculated using Equation 9. The ε value used in the equation was chosen as a small value of 10^{-3} so that the expression does not become infinite.

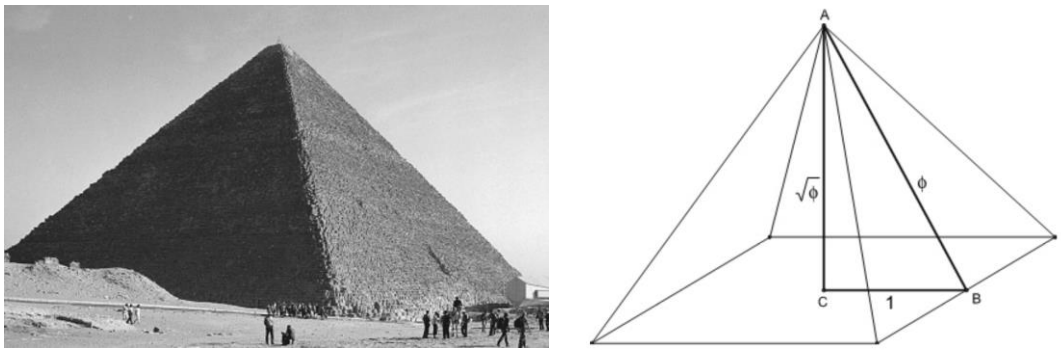

Figure 4. Pyramids in Egypt

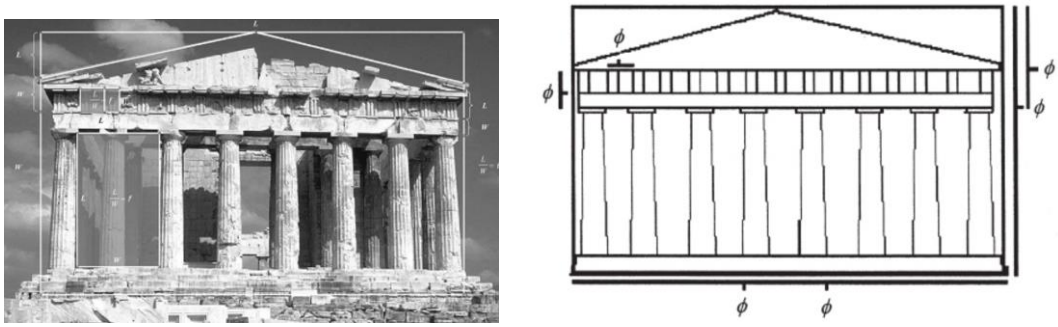

Figure 5. Parthenon in Athens, Greece

Figure 6. Snail shell, plant and human

Then, $state_2$ for the speeds of the particles was created as in Equation 10. Additionally, the situation obtained according to the distance of the particles to the best position is given in Equation 11.

$$state_1(\mathrm{t}) = \begin{cases} 1 & \dfrac{f(x_t) - f_{min}}{f_{max} - f_{min} + \varepsilon} < \emptyset - 1 \\ 0 & else \end{cases} \quad (9)$$

$$state_2(\mathrm{t}) = \begin{cases} 1 & \dfrac{\|v(t)\| - v_{min}}{v_{max} - v_{min} + \varepsilon} < \emptyset - 1 \\ 0 & else \end{cases} \quad (10)$$

$$state_3(\mathrm{t}) = \begin{cases} 1 & \dfrac{\|d(t)\| - d_{min}}{d_{max} - d_{min} + \varepsilon} < \emptyset - 1 \\ 0 & else \end{cases} \quad (11)$$

The parameters of the PSO algorithm will be updated according to $state_1$, $state_2$ and $state_3$ used to create the states of the Q learning algorithm. Algorithm 3 is given to better express this process. The main working logic of the algorithm is that while PSO is running, it creates actions using the Q learning table and tries to find the optimum point by updating the parameters of the PSO algorithm according to the situations.

Algorithm 3. PSO-RL-Q learning Psoude Code

1: Initialize population and reset Q table, determine f
2: **For** t= 1,maximum generation **do**
3: $f_{min}, f_{max}, v_{min}, v_{max}, d_{min}, d_{max}$ calculate
4: $state_1 = []$, $state_2 = []$, $state_3 = []$ ve $states = []$
5: **For** i= 1, population size **do**
6: $state_1$(i), $state_2$(i), $state_3$(i) calculate
$states(i) = [state_1, state_2, state_3]$
7: **If** rand()>0.5
8: Action(i)=rand_int(0-2)
9: **Else**
10: Specify $Action(i) = \arg\left(\underset{a}{max} Q(s_t, a), Q(s_t, a)\right)$
11: **If End**
Specify w, c_1 and c_2 with respect to Action(i)
12:
$$\omega = \begin{cases} 0.05 & Action(i) = 0 \\ 0.50 & Action(i) = 1 \\ 0.95 & Action(i) = 2 \end{cases}, c_1 = \begin{cases} 0.95 & Action(i) = 0 \\ 0.50 & Action(i) = 1, \\ 2.00 & Action(i) = 2 \end{cases}$$
$$c_2 = \begin{cases} 0.95 & Action(i) = 0 \\ 0.50 & Action(i) = 1 \\ 2.00 & Action(i) = 2 \end{cases}$$
13: $v_{i.d}(t+1) = wv_{i.d}(t) + c_1 r_1 \left(p_i(t) - x_{i.d}(t)\right) + c_2 r_2 \left(p_a(t) - x_{i.d}(t)\right)$
14: $x_{i.d}(t+1) = x_{i.d}(t) + v_{i.d}(t)$
15: **End For**
16: $Reward = [\ \ ], next\_state = [\ \ ]$
17: **For** t=1, population size **do**
18: $f_{min}, f_{max}, v_{min}, v_{max}, d_{min}, d_{max}$ update
19: $Reward$ (i)$= -f(x_t)$
20: $state_1$(i), $state_2$(i), $state_3$(i) calculate
$next\_states(i) = [state_1, state_2, state_3]$
21: Apply Equation 2' and update $Q(states(i), Action(i))$ Table
22: **End For**
23: **End For**

In this context, when the psoudecode of the PSO-RL-Q algorithm is examined, the population to be used in PSO is first created. Then, the function f is determined and initial values are assigned to the Q table. In 2nd line, a for loop was created to repeat the operations in the algorithm for the maximum number of generations. In order to be used in calculating the states within the loop, $f_{min}, f_{max}, v_{min}, v_{max}, d_{min}, d_{max}$ values and variables to be used in the algorithm are assigned. Then, the states are calculated within the loop. After that, the control signal was determined depending on the rand() probability. According to this process, $Action$s are determined for each particle. According to the determined action, the parameters (ω, c_1, c_2) to be used in PSO were determined and the positions and speeds of each individual were updated. Afterwards, variables were created for Reward and next_state states. These variables are used to determine reward values according to the positions of the individuals in the swarm, to determine the status of the individuals according to their new positions, and to update the Q table. Therefore, a loop was created at the 17th line to update each individual. In this cycle, the negative value of each individual's goal criterion value is determined as reward. This is due to minimization. Subsequently, the states of each individual were calculated according to their x, d, v values, and next_states were created. Then, the value in the Q Table is updated according to the state and action of the relevant individual.

A Q table must be created to run the algorithm. This created Q table depicted the states, actions and rewards. The sample Q table obtained after running the algorithm is given in Table 1. This

table contains states and actions. The values of the states obtained depending on the states and actions are given in the table.

3. Results and Discussion

In order to test the performance of the PSO algorithm and PSO- RL-Q algorithm in different parameters, the inverse kinematics problem of a 7-axis serial robot manipulator was used. In this study, the position of the end effector was chosen randomly. In this direction, Dereli et al. used the DH method to derive the kinematic equations of this robot and used the parameters in Table 2.

The parameters used to compare the performance of the proposed PSO- RL-Q algorithm with different criteria such as number of iterations and swarm size are given in Table 3. As can be seen, while the control parameters ω , c_1, c_2 for PSO_par_1 were $0.05, 0.95, 0.95$, the tests were carried out using the iteration numbers as 100, 1000 and the Swarm size as 50 and 100. Similarly, it has been implemented in PSO_par_2, PSO_par_3, and PSO-RL-Q algorithm. To test performances of algorithms, two different points were considered for the final position of the manipulator. According to these points, each algorithm was run independently 30 times. Then, the results obtained were compared statistically. Additionally, their statistical reliability was compared to test whether there were significant results.

Table 1. A Q table results for population size 50 and maximum iteration 1000 for PSO-RL-Q

$States$	$Action = 0$ $\omega = 0.05$ $c_1 = 0.50$ $c_2 = 0.95$	$Action = 1$ $\omega = 0.50$ $c_1 = 0.50$ $c_2 = 0.95$	$Action = 2$ $\omega = 0.95$ $c_1 = 2.00$ $c_2 = 2.00$
(0,0,0)	-2.2268×10^{-17}	-2.0845×10^{-17}	-3.7241×10^{-17}
(0,0,1)	-1.0336×10^{-16}	-1.0861×10^{-6}	-3.2223×10^{-7}
(0,1,0)	-7.3675×10^{-17}	-2.7739×10^{-6}	-4.7972×10^{-6}
(0,1,1)	-3.1769×10^{-11}	-1.5064×10^{-16}	-1.6349×10^{-9}
(1,0,0)	-0.4615	-0.3071	-0.3664
(1,0,1)	-0.9919	-0.87698	-1.1141
(1,1,0)	-0.2333	-0.33376	-0.3182
(1,1,1)	-1.6299	-2.3279	-2.2797

Table 2. DH parameters of 7-DOF Robot

i	$l_i(m)$	$\alpha_i(°)$	$d_i(m)$	$\theta_i(°)$
1	$l_1 = 0.5$	-90	0	$-180 < \theta_1 < 180$
2	$l_2 = 0.2$	90	0	$-90 < \theta_2 < 30$
3	$l_3 = 0.25$	-90	0	$-90 < \theta_3 < 120$
4	$l_4 = 0.3$	90	0	$-90 < \theta_4 < 90$
5	$l_5 = 0.2$	-90	0	$-90 < \theta_5 < 90$
6	$l_6 = 0.2$	0	0	$-90 < \theta_4 < 60$
7	$l_7 = 0.2$	0	$d_7 = 0.05$	$-90 < \theta_4 < 90$

Table 3. Parameters of Algorithms

Method	Parameters	Iteration Numbers	Swarm Size
PSO_par_1	$\omega = 0.05,\quad c_1 = 0.95, c_2 = 0.95$	100, 1000	50,100
PSO_par_2	$\omega = 0.50,\quad c_1 = 0.50, c_2 = 0.50$	100, 1000	50,100
PSO_par_3	$\omega = 0.95,\quad c_1 = 2.00, c_2 = 2.00$	100, 1000	50,100
PSO-RL-Q	$\begin{cases} \omega = 0.05, & c_1 = 0.95, c_2 = 0.95 \quad a_t = 0 \\ \omega = 0.50, & c_1 = 0.50, c_2 = 0.50 \quad a_t = 1 \\ \omega = 0.95, & c_1 = 2.00, c_2 = 2.00 \quad a_t = 2 \end{cases}$	100, 1000	50, 100

Table 4. Optimal joint angle values produced by PSO_par_1, PSO_par_2, PSO_par_3, PSO_Q algorithms according to target positions by swarm size 50, iteration number 100.

Position	Method	θ_1^o	θ_2^o	θ_3^o	θ_4^o	θ_5^o	θ_6^o
[-25.00 100.00 50.00]	PSO_par_1	-39.9982	-40.3681	54.3798	18.5928	7.3264	45.6443
	PSO_par_2	-0.07114	35.5679	3.5509	-78.1808	-21.9988	51.0210
	PSO_par_3	-73.2728	-90.0	13.9466	-20.2021	12.5424	-15.0000
	PSO-RL-Q	-16.0189	-10.2971	-1.4765	-33.4392	72.7653	22.1168
[-30.00 20.00 80.00]	PSO_par_1	-77.7382	-23.6192	68.1264	32.7416	110.5017	56.5476
	PSO_par_2	-30.6456	14.0067	-90.0	89.9859	-52.3304	39.1834
	PSO_par_3	-17.6197	-90.0	-90.0	90.0	119.9999	69.9999
	PSO-RL-Q	9.8517	-31.9813	-85.8088	69.2948	-78.6991	-12.5035

The joint angles obtained for the inverse kinematic analysis results in the case of swarm size 50, iteration number 100, according to the determined target positions [-25.00 100.00 50.00] and [-30.00 20.00 80.00] are listed in Table 4. In general, it has been observed that some algorithms produce joint angles at different angles. The reason for this is that Equation 6 can be minimized, at different angles of joint if the robot has many joints of robot.

At Table 5, Px, Py and Pz positions produced by the algorithms for 2 different locations, location errors of the algorithms and processing times are given. Although the proposed PSO-RL-Q method produced the lowest error in terms of position error, it showed lower performance than other methods in terms of computation time. In addition, the best values produced by the algorithms according to the iterations are shown in Figure 7. The best results of these algorithms, which contain random parameters, were close during iterations. However, when the iteration continued, the dynamics of these algorithms differentiated the results. The part where this will be noticed best is the part where statistical analysis will be performed. In for that, the minimum and maximum mean and standard deviation values produced by the algorithms as a result of 30 different simulation studies where the number of swarm is 50 and the iteration is 100 are given in Table 6. The results are written with the best values in bold. However, when examined in terms of mean value, the proposed PSO-RL-Q algorithm produced more successful results than others.

Table 5. Optimal joint angle values produced by PSO_par_1, PSO_par_2, PSO_par_3, PSO_Q algorithms according to target positions by swarm size 50, iteration number 100.

Target Position (cm)	Method	Px (cm)	Py (cm)	Pz (cm)	Position Error	Computation Time (sec)
[-25.00 100.00 50.00]	PSO_par_1	-24.99	99.99	50.00	9.4954×10^{-5}	1.40824
	PSO_par_2	-25.00	100.00	49.99	2.2551×10^{-14}	1.87351
	PSO_par_3	-24.05	99.92	50.82	1.2634×10^{-2}	**1.33919**
	PSO-RL-Q	-24.99	100.00	49.99	$\mathbf{8.0251\times 10^{-16}}$	4.51963
[-30.00 20.00 80.00]	PSO_par_1	-29.48	31.23	79.25	1.5306×10^{-2}	1.11302
	PSO_par_2	-30.00	29.99	80.00	3.7180×10^{-15}	1.13652
	PSO_par_3	-29.97	29.82	79.92	1.9437×10^{-3}	**1.05694**
	PSO-RL-Q	-30.00	30.00	80.00	$\mathbf{1.7554\times 10^{-15}}$	3.70225

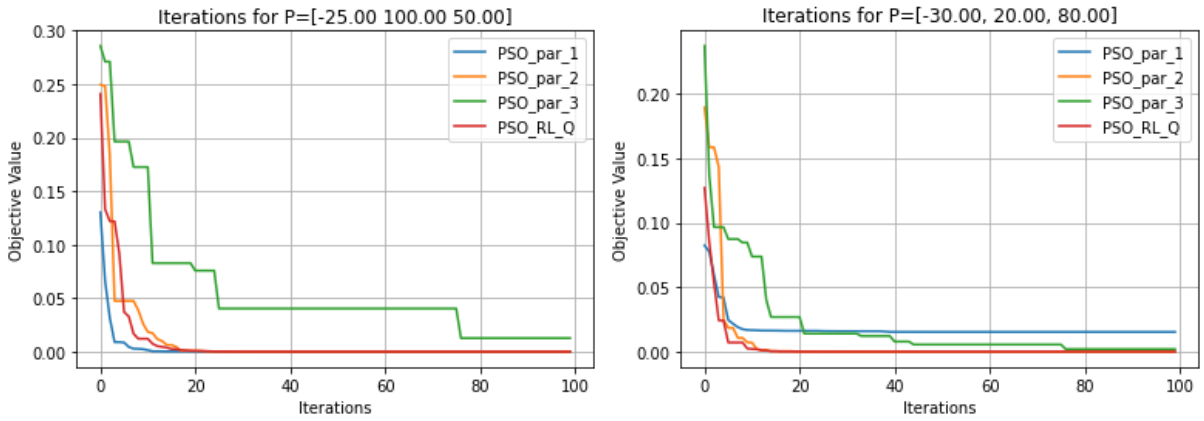

Figure 7. Objective values along the iterations for Swarm Size 50, iteration number 100

For statistical reliability, the Wilxocon test was applied to the trials. The results obtained are given for different locations in Table 7 and Table 8. From the pairwise comparison of the results, it is seen that, since the significance value is lower than 0.05, there is a significant difference in the results except for PSO_par_1/PSO_par_1, PSO_par_2/PSO_par_2, PSO_par_4/PSO_par_4 and PSO-RL-Q/PSO_par_3 results.

It has been observed that the PSO algorithm is successful in solving the inverse kinematics problem of the 7-DOF robot. At the same time, it has been observed that changing the parameters of PSO or adding reinforcement learning into PSO increases the performance of the algorithm. However, when the number of iterations and swarm sizes of the algorithms are increased, the processing time becomes longer because the search takes longer and with more particles. At the same time, the potential to find its global minimum is higher. For this case, only for the [-25.00 100.00 50.00] position, the algorithms were applied to the solution of the inverse kinematic problem with the swarm size being 100 and the number of iterations being 1000, and the results were analyzed. The results obtained according to the iterations are depicted in Figure 4. As can be seen from these results, since the number of swarms and the number of iterations are high, all algorithms converge to the same result, which is the global result.

Table 6. Statistical results for swarm size 50, iteration number 100.

Target Position (cm)	Method	Minimum	Max	Mean	Standard deviation
[-25.00 100.00 50.00]	PSO_par_1	9.4954×10^{-5}	0.2327	4.8891×10^{-2}	3.0914×10^{-2}
	PSO_par_2	2.2484×10^{-14}	0.0332	8.7174×10^{-3}	3.2777×10^{-3}
	PSO_par_3	1.2634×10^{-2}	0.1744	3.5619×10^{-2}	4.4413×10^{-3}
	PSO-RL-Q	$\mathbf{7.8504\times 10^{-17}}$	$\mathbf{1.2567\times 10^{-7}}$	$\mathbf{2.2654\times 10^{-8}}$	4.7879×10^{-9}
[-30.00 20.00 80.00]	PSO_par_1	1.5306×10^{-2}	0.3715	9.5057×10^{-2}	1.7430×10^{-1}
	PSO_par_2	3.8298×10^{-15}	$\mathbf{9.5388\times 10^{-2}}$	2.3801×10^{-2}	1.0253×10^{-2}
	PSO_par_3	1.9437×10^{-3}	0.1011	2.4124×10^{-2}	2.3077×10^{-2}
	PSO-RL-Q	**0.0**	0.1162	$\mathbf{2.1863\times 10^{-2}}$	1.1943×10^{-2}

Table 7. Wilxocon test results for [-25.00 100.00 50.00], swarm size 50, iteration number 100.

	PSO_par_1	PSO_par_2	PSO_par_3	PSO-RL-Q
PSO_par_1	1.00	9.01×10^{-10}	8.58×10^{-33}	9.03×10^{-10}
PSO_par_2	9.01×10^{-10}	1.00	1.30×10^{-25}	0.0775
PSO_par_3	8.58×10^{-33}	1.30×10^{-25}	1.00	2.58×10^{-28}
PSO-RL-Q	9.03×10^{-10}	0.0775	2.58×10^{-28}	1.00

Table 8. Wilxocon test results for [-30.00 20.00 80.00], swarm size 50, iteration number 100.

	PSO_par_1	PSO_par_2	PSO_par_3	PSO-RL-Q
PSO_par_1	1.00	2.57×10^{-26}	3.92×10^{-13}	1.39×10^{-28}
PSO_par_2	2.57×10^{-26}	1.00	7.14×10^{-24}	0.0227
PSO_par_3	3.92×10^{-13}	7.14×10^{-24}	1.00	6.92×10^{-26}
PSO-RL-Q	1.39×10^{-28}	0.0227	6.92×10^{-26}	1.00

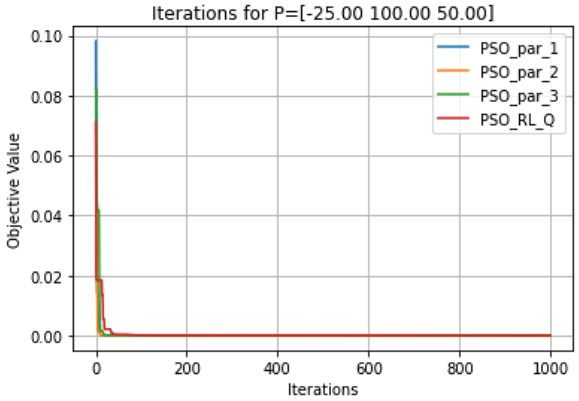

Figure 8. Objective values along the iterations for [-25.00 100.00 50.00] Position ve Swarm Size 100, iteration number 1000.

Furthermore, for a numerical comparison of the results obtained, the positions reached, position errors and processing times are demonstrated at Table 9. As can be seen, as the processing load increases, processing times vary between 9 and 14 seconds. Sürü sayısı ve iterasyon sayısı arttığında, all algorithms found the same global optimal solution within 30 different trials. However, the calculation times have changed due to differences in calculations and operations. Although the results are the same, as can be seen, the PSO_par_1 algorithm produced the best result, while the PSO-RL-Q algorithm was the slowest. As a result, it is more correct to choose the faster

method among the methods that show the same performance.

Statistical analysis results are demonstrated at Table 10. Since the results of 30 independent experiments reached the desired global value, the minimum, maximum, average value and standard deviation values were obtained as 0. The reason why all the results are 0 in Table 10, is because when the swarm numbers and iteration numbers of the algorithms were increased, all algorithms found the same global optimal solution within 30 different trials. Therefore, when the optimal solution was one, the error function, the mean value, minimum value, maximum value and standard deviation of the error function were found to be 0.

Table 9. Optimal joint angle values produced by PSO_par_1, PSO_par_2, PSO_par_3, PSO_Q algorithms according to target positions by swarm size 100, iteration number 1000.

Target Position (cm)	Method	Px (cm)	Py (cm)	Pz (cm)	Position Error	Computation Time (sec)
[-25.00 100.00 50.00]	PSO_par_1	-25.00	100.00	50.00	**0.0**	**9.47613**
	PSO_par_2	-25.00	100.00	50.00	**0.0**	11.7058
	PSO_par_3	-25.00	100.00	50.00	**0.0**	12.7068
	PSO-RL-Q	-25.00	100.00	50.00	**0.0**	13.2338

Table 10. Statistical results for swarm size 100, iteration number 1000.

Target Position (cm)	Method	Minimum	Max	Mean	Standard deviation
[-25.00 100.00 50.00]	PSO_par_1	**0**	**0**	**0**	**0**
	PSO_par_2	**0**	**0**	**0**	**0**
	PSO_par_3	**0**	**0**	**0**	**0**
	PSO-RL-Q	**0**	**0**	**0**	**0**

In addition, the Wilxocon test was performed to test whether there was a significant difference in the results. To perform this test, the results obtained by the algorithms in 30 different experiments were used. The algorithms always found the same error value with a high number of iterations and a high number of swarm. Therefore, there was no difference in the results obtained when high number of iterations and swarm sizes was used in the algorithms. The results obtained when the Wilxocon test is applied for the reliability of this statistical result are given in Table 11. In the results given in Table 11, it was seen that there was no significant difference in the results since all values were greater than 0.05. The reason why all the results are 1.00 in Table 11, is because when the swarm numbers and iteration numbers.

As the results are examined, changing the parameters of the PSO algorithm affects the performance of the algorithm, and the solution performance of the algorithm can be increased when a reinforcement learning-based method is added, as in this study. Ultimately, 7-DOF robot used in industry was turned into an inverse kinematic problem with optimization using advanced kinematic equations with DH to determine its angles and distances using optimization algorithms. So that, PSO-RL-Q algorithm distinctly showed successful results in finding the joint angles and distances of the robot within short iteration number than plain PSO algorithm.

Table 11. Wilxocon test results for [-25.00 100.00 50.00], swarm size 100, iteration number 100.

	PSO_par_1	PSO_par_2	PSO_par_3	PSO-RL-Q
PSO_par_1	1.00	1.00	1.00	1.00
PSO_par_2	1.00	1.00	1.00	1.00
PSO_par_3	1.00	1.00	1.00	1.00
PSO-RL-Q	1.00	1.00	1.00	1.00

4. Conclusion and Suggestions

In this study, a new optimization method was applied to find the joint angles of a 7-axis robot at the desired position. In this direction, first the kinematic equation of the robot was obtained, and then the PSO-based PSO_par_1, PSO_par_2, PSO_par_3 and reinforcement learning-based proposed PSO-RL-Q optimization method using golden section have been used to find the manipulator angle and distances. In order to compare the performances of the algorithms, 30 independent studies have been carried out and the results were compared statistically. Statistically, it has been observed that the proposed PSO-RL-Q algorithm produces more successful results than standard PSO algorithms when the number of iterations is 100 and swarm size is 50. From the results, while the swarm size of the algorithms is 50 and the number of iterations is 100, the PSO-RL-Q algorithm performed the best result by producing 2.2654×10^{-8} mean value for the position [-25.00 100.00 50.00]. Also, it produced more successful results than other algorithms by producing a mean value of 2.1863×10^{-2} for the position [-30.00 20.00 80.00]. In addition, the test results obtained were subjected to the Wilxocon test and it was observed that there was a significant difference in the results when the number of iterations was 50 and iteration number are 100. Since reinforcement learning works on a reward basis, it appears to be a method that can be easily applied to many different algorithms. In future studies, it is planned to apply these proposed methods and their variations to different optimization problems and systems.

Statement of Research and Publication Ethics

The study is complied with research and publication ethics.

References

[1] F. Özüdoğru, "Endüstriyel Robot Kolu Modelinin Hedef Konum Eklem Açilarinin Yapıcı Sinir Ağı Ile Kestirimi Ve Kontrollü Yörünge Uygulamasi," Yüksek Lisans, Elektrik Elektronik Mühendisliği, Tokat, 2020.

[2] E. Düzgün, "Paralel ve Hibrit Manipülatörlerin Ileri Kinematik Çözümü Için Yeni Metotlar Geliştirilmesi," Doktora, Fen Bilimleri Enstitüsü, Bursa, 2023.

[3] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm," *Artif Intell Rev*, vol. 53, pp. 949–964, 2020.

[4] F. Aysal, İ. Çelik, E. Cengiz, and Y. Oğuz, "A comparison of multi-layer perceptron and inverse kinematic for RRR robotic arm," *Politeknik Dergisi*, vol. 27, no. 1, pp. 121–131, 2023.

[5] S. Hwang, H. Kim, Y. Choi, K. Shin, and C. Han, "Design Optimization Method for 7 DOF Robot Manipulator Using Performance Indices," *International Journal of Precision Engineering and Manufacturing*, vol. 18, no. 3, pp. 293–299, 2017.

[6] A. Avaei, L. van der Spaa, L. Peternel, and J. Kober, "An incremental inverse reinforcement learning approach for motion planning with separated path and velocity preferences," *Robotics*, vol. 12, no. 2, 2023.

[7] S. Dereli and R. Köker, "Simulation based calculation of the inverse kinematics solution of 7-DOF robot manipulator using artificial bee colony algorithm," *SN Appl Sci*, vol. 2, no. 1, p. 27, 2020.

[8] S. Baressi Šegota, N. Anđelić, M. Šercer, and H. Meštrić, "Dynamics Modeling of Industrial Robotic Manipulators: A Machine Learning Approach Based on Synthetic Data," *Mathematics*, vol. 10, no. 7, p. 1174, 2022.

[9] Z. Bingül and S. Küçük, *Robot Kinematiği*. Umuttepe Yayınları, 2019.

[10] H. Danaci, L. A. Nguyen, T. L. Harman, and M. Pagan, "Inverse Kinematics for Serial Robot Manipulators by Particle Swarm Optimization and POSIX Threads Implementation," *Applied Sciences*, vol. 13, 2023.

[11] C. J. Watkins and P. Dayan, "Q-learning," *Mach Learn*, vol. 8, pp. 279–292, 1992.

[12] J. Peng and R. J. Williams, "Incremental Multi-Step Q-Learning," 1996.

[13] İ. Tunç and M. Söylemez, "Fuzzy logic and deep Q learning based control for traffic lights," *Alexandria Engineering Journal*, vol. 67, pp. 343–359, 2023.

[14] M. E. Çimen, Z. Garip, Y. Yalçın, M. Kutlu, and A. F. Boz, "Self Adaptive Methods for Learning Rate Parameter of Q-Learning Algorithm," *Journal of Intelligent Systems: Theory and Applications*, vol. 6, no. 2, pp. 191–198, 2023.

[15] A. O. Köroğlu, A. E. Edem, S. N. Akmeşe, Ö. Elmas, I. Tunc, and M. T. Soylemez, "Agent-Based Route Planning with Deep Q Learning," in *13th International Conference on Electrical and Electronics Engineering (ELECO)*, 2021, pp. 403–407.

[16] A. Wang, H., Emmerich, M., & Plaat, "Monte Carlo Q-learning for General Game Playing," *arXiv preprint arXiv:1802.05944*.

[17] F. Candan, S. Emir, M. Doğan, and T. Kumbasar, "Takviyeli Q-Öğrenme Yöntemiyle Labirent Problemi Çözümü Labyrinth Problem Solution with Reinforcement Q-Learning Method," in *TOK2018 Otomatik Kontrol Ulusal Toplantısı*, 2048.

[18] Y. Liu, H. Lu, S. Cheng, and Y. Shi, "An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning," in *IEEE congress on evolutionary computation (CEC)*, 2019, pp. 815–822.

[19] M. Çimen, "Hibrit ve Kaotik Metasezgisel Arama Algoritmalari Kullanarak Model Öngörülü Kontrol Yapıları Tasarımı," Doktora, Sakarya Uygulamalı Bilimler Üniversitesi, 2022.

[20] A. F. Boz and M. E. Çimen, "An interface design for controlling dead time systems using PSO, CS and FA algorithms," in *8th International Advanced Technologies Symposium (IATS'17), 19-22 October*, 2017.

[21] A. F. Boz and M. E. Çimen, "PID Controller Design Using Improved FireFly Algorithm," in *8th International Advanced Technologies Symposium (IATS'17), 19-22 October*, 2017.

[22] M. E. Çimen and A. F. Boz, "Parameter identification of a non-minimum phase second order system with time delay using relay test and PSO, CS, FA algorithms," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 34, no. 1, pp. 461–477, 2019, doi: 10.17341/gazimmfd.416507.

[23] Z. B. Garip, M. E. Cimen, D. Karayel, and A. L. I. F. Boz, "The chaos-based whale optimization algorithms global optimization," vol. 0, no. 1, pp. 51–63, 2019.

[24] A. Akgül, Y. Karaca, Pala MA, M. Çimen, A. Boz, and M. Yıldız, "Chaos Theory, Advanced Metaheuristic Algorithms and Their Newfangled Deep Learning Architecture Optimization Applications: A Review," *Fractals*, vol. 32, no. 3, 2024.

[25] A. Hossain and Z. Yılmaz Acar, "Comparison of New and Old Optimization Algorithms for Traveling Salesman Problem on Small, Medium, and Large-scale Benchmark Instances," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 13, no. 1, pp. 216–231, 2024.

[26] M. E. Cimen, Z. Garip, A. F. Boz, and D. Karayel, "Firefly Algorithm and Particle Swarm Optimization for photovoltaic parameters identification based on single model," *ISMSIT 2018 - 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies, Proceedings*, 2018, doi: 10.1109/ISMSIT.2018.8567288.

[27] A. Karthikeyan, M. E. Cimen, A. Akgul, A. F. Boz, and K. Rajagopal, "Persistence and coexistence of infinite attractors in a fractal Josephson junction resonator with unharmonic current phase relation considering feedback flux effect," *Nonlinear Dyn*, vol. 103, no. 2, pp. 1979–1998, 2021, doi: 10.1007/s11071-020-06159-4.

[28] K. Rajagopal *et al.*, "A family of circulant megastable chaotic oscillators, its application for the detection of a feeble signal and PID controller for time-delay systems by using chaotic SCA algorithm," *Chaos Solitons Fractals*, vol. 148, no. May, p. 110992, 2021, doi: 10.1016/j.chaos.2021.110992.

[29] S. A. Celtek and S. Kul, "Parameter Extraction of PV Solar Cells Using Metaheuristic Methods," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 12, no. 4, pp. 1041–1053, 2023.

[30] M. Çimen, Z. Garip, E. M, and A. Boz, "Fuzzy Logic PID Design using Genetic Algorithm under Overshoot Constrained Conditions for Heat Exchanger Control," *Journal of the Institute of Science and Technology*, vol. 12, no. 1, pp. 164–181, 2022.

[31] H. Geçmez and H. Deveci, "Optimization of Hybrid Composite Laminates with Various Materials using the GA/GPSA Hybrid Algorithm for Maximum Dimensional Stability," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 13, no. 1, pp. 107–133, 2024.

[32] M. E. Çimen and A. F. Boz, "PSO, CS ve FA Algoritmalarıyla Ortak Emiterli BJT'li Yükselteç Tasarımı," *Cumhuriyet Üniversitesi Fen Edebiyat Fakültesi Fen Bilimleri Dergisi*, vol. 38, no. 1, pp. 119–130, 2017.

[33] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, 1995, pp. 1942–1948.

[34] X. S. Yang, "Firefly algorithms for multimodal optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5792 LNCS, pp. 169–178, 2009, doi: 10.1007/978-3-642-04944-6_14.

[35] M. Cimen and Y. Yalçın, "A novel hybrid firefly–whale optimization algorithm and its application to optimization of MPC parameters," *Soft comput*, vol. 26, no. 4, pp. 1845–1872, 2022.

[36] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, pp. 210–214, 2009, doi: 10.1109/NABIC.2009.5393690.

[37] S. Mirjalili, S. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[38] M. Çimen, Z. Garip, and A. Boz, "Chaotic flower pollination algorithm based optimal PID controller design for a buck converter," *Analog Integr Circuits Signal Process*, 2021.

[39] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[40] M. Juneja and S. K. Nagar, "Particle swarm optimization algorithm and its parameters: A review," in *International Conference on Control, Computing, Communication and Materials (ICCCCM)*, 2016.

[41] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 congress on evolutionary computation. CEC00*, 2000, pp. 84–88.

[42] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 240–255, 2004.

[43] H. M. Cui and Q. B. Zhu, "Convergence analysis and parameter selection in particle swarm optimization," *Jisuanji Gongcheng yu Yingyong (Computer Engineering and Applications)*, vol. 42, no. 23, pp. 89–91, 2007.

[44] C. Guimin, J. Jianyuan, and H. Qi, "Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm," *Journal-Xian Jiaotong University*, vol. 40, no. 1, p. 53, 2006.

[45] Tanabe R and Fukunaga A, "Success-history based parameter adaptation for differential evolution," in *IEEE Congress on Evolutionary Computation. IEEE*, 2013, pp. 71–78.

[46] Z. Liu and T. Nishi, "Multipopulation ensemble particle swarm optimizer for engineering design problems," *Math Probl Eng*, 2020.

[47] Tatsis VA and Parsopoulos KE, "Grid-based parameter adaptation in particle swarm optimization," in *2th Metaheuristics International Conference (MIC 2017)*, 2017, pp. 217–226.

[48] F. Olivas, F. Valdez, O. Castillo, and P. Melin, "Dynamic parameter adaptation in particle swarm optimization using interval type2 fuzzy logic," *Soft Computing*, vol. 20, no. 3, pp. 1057–1070, 2016.

[49] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, "Optimal design of fuzzy classification systems using pso with dynamic parameter adaptation through fuzzy logic," *Expert Syst Appl*, vol. 40, no. 8, pp. 3196–3206, 2013.

[50] S. Yin *et al.*, "Reinforcement-learning-based parameter adaptation method for particle swarm optimization," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5585–5609, 2023.

[51] Y. Xu and D. Pi, "A reinforcement learning-based communication topology in particle swarm optimization," *Neural Comput Appl*, pp. 10007–10032, 2020.

[52] C. Lee and M. Ziegler, "Geometric approach in solving inverse kinematics of PUMA robots," *IEEE Trans Aerosp Electron Syst*, vol. 6, pp. 695–706, 1984.

[53] R. Köker, C. Öz, T. Çakar, and H. Ekiz, "A study of neural network based inverse kinematics solution for a three-joint robot," *Rob Auton Syst*, vol. 49, no. 3–4, pp. 227–234, 2004.

[54] S. Alavandar and M. J. Nigam, "Neuro-fuzzy based approach for inverse kinematics solution of industrial robot manipulators," *International Journal of Computers Communications & Contro*, vol. 3, no. 3, pp. 224–234, 2008.

[55] G. Jin, S. Ma, and Z. Li, "Dynamic simulation modeling of industrial robot kinematics in industry 4.0," *Discrete Dyn Nat Soc*, pp. 1–11, 2022.

[56] Y. Chen, X. Zhang, Y. Huang, Y. Wu, and J. Ota, "Kinematics optimization of a novel 7-DOF redundant manipulator," *Rob Auton Syst*, vol. 163, p. 104377, 2023.

[57] S. Baressi Šegota, N. Anđelić, I. Lorencin, M. Saga, and Z. Car, "Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms," *International journal of advanced robotic systems*, vol. 17, no. 2, 2020.

[58] Y. Hou and J. Li, "Learning 6-DoF grasping with dual-agent deep reinforcement learning," *Rob Auton Syst*, vol. 166, 2023.

[59] S. Müftü and B. Gökçe, "Design and Implementation of an Optimized PID Controller for Two-Limb Robot Arm Control," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 13, no. 1, pp. 192–204, 2024.

[60] M. Çimen and A. Boz, "Parameter identification of a non-minimum phase second order system with time delay using relay test and PSO, CS, FA algorithms," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 34, no. 1, pp. 461–477, 2019.

[61] A. Angiuli, J. P. Fouque, and M. Laurière, "Unified reinforcement Q-learning for mean field game and control problems," *Mathematics of Control, Signals, and Systems*, vol. 34, no. 2, pp. 217–271, 2022.

[62] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D., Dissertation, King's College UK, 1989.

[63] A. S. Posamentier and I. Lehmann, *The Glorious Golden Ratio*. Prometheus Books, 2011.

[64] S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.