Istraživanja i projektovanja za privredu
Journal of Applied Engineering Science

Original Scientific Paper

doi:10.5937/jaes0-30557

Paper number: 19(2021)3, 832, 592-599

PARTICLE SWARM OPTIMIZATION FOR INVERSE KINEMATICS SOLUTION AND TRAJECTORY PLANNING OF 7-DOF AND 8-DOF ROBOT MANIPULATORS BASED ON UNIT QUATERNION REPRESENTATION

Javier Alexis Abdor-Sierra[1]*, Emmanuel Alejandro Merchán-Cruz[4], Flavio Arturo Sánchez-Garfias[2], Ricardo Gustavo Rodríguez-Cañizo[1], Edgar Alfredo Portilla-Flores[3], Valentin Vázquez-Castillo[3]

[1]Instituto Politécnico Nacional, ESIME UA, México, México
[2]Instituto Politécnico Nacional, ESCOM, México, México
[3]Instituto Politécnico Nacional, CIDETEC, México, México
[4]Transport and Telecommunication Institute, Riga, Latvia

Finding the inverse kinematic solution of a serial manipulator has always attracted the attention of optimization enthusiasts, as the solution space is highly nonlinear and, depending on the number of degrees of freedom, has multiple solutions. In the literature, one can find several proposed solutions using heuristic techniques; however, for highly redundant manipulators, e.g., seven or more, the discussions focused on minimizing the positional error. In this paper, a metaheuristic approach is presented to solve not only the inverse kinematics of a 7 and 8 DOF manipulators but the proposed algorithm is used to find the robot's poses for trajectory planning where the robot is required to meet the desired position and orientation based on quaternion representation of each point along the path. The metaheuristic approach used in this paper is particle swarm optimization (PSO), where the unit quaternion is used in the objective function to find the orientation error. The results prove that the use of the unit quaternion representation improved the performance of the algorithm and that our approach can be used not only for individual poses but for trajectory planning.

Key words: particle swarm optimization, 7-DOF, 8-DOF, inverse kinematics, trajectory planning, quaternion

INTRODUCTION

The inverse kinematics (IK) problem is one of the most studied topics in robotics research. The problem consists of finding the joint angles for a desired position and orientation for the end-effector. Geometric, numerical, and algebraic methods have been used to solve the IK, however, these get more complex as the degrees of freedom (DOF) increase. Also, the emergence of redundant manipulators with seven or more DOF further complicates the problem since there are multiple solutions and postures that the robot can have for a specific pose of the end-effector; therefore, paving the way for researchers to develop and implement new solutions. Metaheuristics approaches, or in other words, search algorithms have gained traction and interest to solve the IK for redundant manipulators.

[1], introduced a quantum-behaved particle swarm algorithm to minimize the position error of a seven DOF manipulator. This algorithm guarantees global convergence by utilizing a wave function instead of position and velocity variables. On the other hand, [2] proposed a combination of (PSO) and real coded genetic algorithm (RCGA) to solve the IK for a seven DOF manipulator. The hybrid algorithm was composed of all the basic operators of a genetic algorithm, such as mutation and crossover. [3], also proposed the (ABC) algorithm to minimize the position error for a seven DOF robot. ABC was able to minimize the position error better than the standard PSO but had the same execution time. [4], introduced a PSO algorithm for a six DOF manipulator that solved the IK for position and orientation in less than 16 iterations. [5], introduced the chaotic and parallelized artificial bee colony algorithm (CPABC), a modified version of the ABC algorithm to solve the IK for position and orientation for a seven DOF manipulator. CPABC outperformed the standard ABC algorithm by providing more stable solutions. [6], for a seven DOF robot, used joint parameterization to turn the problem into a one-dimensional problem. To find the solution for the lone joint angles, a combination of GA and PSO was used. To avoid local minima, [7], introduced a mutating PSO algorithm for a six DOF robot that detects if the algorithm is stuck in local minima with four new variables. Also, [8] utilized the firefly algorithm to solve for the position of a seven DOF robot, which had better performance in terms of error but not execution time. [9], developed an adaptive PSO algorithm that adapts the acceleration and inertia parameters to avoid getting stuck in a local optimum. Lastly, [10] utilizes PSO to find the desired position and orientation of a 6 DOF robot using a weighted RMSE objective function.

Based on previous related work, the IK solution for 7 DOF robots focused on minimizing the position error but not the orientation error. For real applications, having a

*jabdors1900@alumno.ipn.mx

highly redundant robot reach the desired position with any orientation is not feasible and does not utilize its full potential. Also, most of the works besides [2] focus on reaching a certain pose or random poses but do not show if the approach is limited to a couple of poses or if it can be applied for trajectory planning. Furthermore, works based on minimizing the position and orientation error is studied on robot manipulators with 6 or less DOF using the rotation matrix in the objective function to minimize the rotation error. Therefore, the objective of this paper is to present a novel approach based on PSO and unit quaternions to solve the IK for 7 and 8 DOF robot manipulators. This approach will minimize not only the position error but the orientation error for a desired position and orientation of the end-effector. The unit quaternion plays a key role since the number of parameters for the orientation is reduced from nine to four, which has not been proposed in previous works along with metaheuristic approaches. In addition, our approach extends its application to not only solving for individual poses but for trajectory planning where the robot must describe a smooth trajectory along or between waypoints.

PARTICLE SWARM OPTIMIZATION

The particle swarm optimization (PSO) is a metaheuristic algorithm presented by [11], that is inspired by the social behavior of animals, most notably bird flocks and fish shoaling. In the wild, swarms of these animals work together to find resources for survival by communicating with each other and sharing information. This same principle is also applied to solving non-linear engineering problems, where a set of candidate solutions look through a search space, and with each iteration, a best new solution is obtained. Some advantages of PSO over other metaheuristic approaches are that it has few parameters to adjust and it has a fast convergence rate. To illustrate PSO, a swarm with n number of particles is initialized, where each particle is a possible solution to the optimization problem. Each particle in the swarm has a position and velocity vector that is updated every iteration with the following equations.

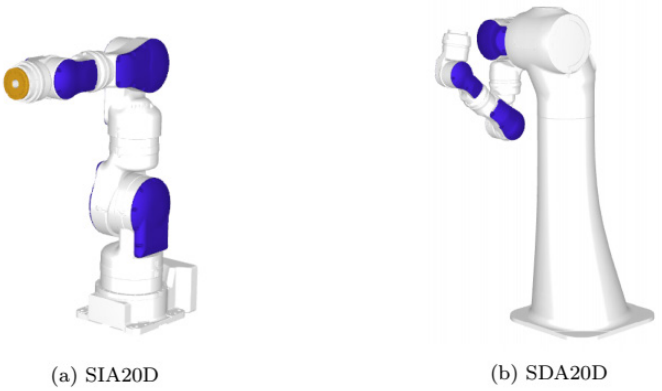
$$V_{ij}(t+1)=wV_{ij}(t)+r_1C_1(P_{ij}(t)-X_{ij}(t)+r_2C_2(g_j(t)-X_{ij}(t)) \quad (1)$$

$$X_{ij}(t+1)=X_{ij}+V_{ij}(t+1) \quad (2)$$

As shown in equation (1), the velocity vector is made up of parameters w, r_1, r_2, C_1, and C_2. Parameter w is the inertia constant that affects the particle's exploration, it is usually initialized at 1 and it decreases as the iterations increase to reduce exploration in the search space while the algorithm gets close to the optimal solution. On the other hand, parameters r_1 and r_2 are random uniform values from [0,1] that helps the algorithm avoid premature convergence. Lastly, C_1 and C_2 are acceleration coefficients that determine the influence of the local best position and the global best position, these are usually set to two [12]. Furthermore, as seen in equation (1), the difference between the particles best position P_{ij} and the current position X_{ij} is calculated and helps the particle stay close to its best position if it gets distant from it. Also, the difference between the global best position g_j and the current position X_{ij} is calculated, which attracts all the particles to that position.

KINEMATICS ANALYSIS

The robots used for this study are the Motoman SIA20D and Motoman SDA20D, both from the Japanese manufacturer Yaskawa. On one hand, the Motoman SIA20D is a seven DOF robot and the SDA20D is a fifteen DOF robot that has two SIA20D arms with a turntable. For this study, we will be working with the right arm and the turntable to have an eight DOF robot, both robots are shown in figure 1.



(a) SIA20D (b) SDA20D

Figure 1: Robots used in this study

To solve the IK, the forward kinematics (FK) solution needs to be obtained. FK is the problem of finding the position and orientation of the end-effector with respect to the base given the joint angles. There are various methods to solve the FK, the most common being the Denavit-Hartenberg convention (DH). The DH convention relates two consecutive joints with the transformation matrix given in (3).

$$^{i-1}A_i=\begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & \alpha_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Tables 1 and 2 show the DH parameters for the seven and eight DOF robots respectively. The lengths are in millimeters (mm) and the minimum and maximum angles

Table 1: DH parameters SIA20D

Joint	θ [deg]	d [mm]	a [mm]	α [deg]
1	$-180 < \theta_1 < 180$	410	0	-90
2	$-110 < \theta_2 < 110$	0	0	90
3	$-170 < \theta_3 < 170$	490	0	-90
4	$-130 < \theta_4 < 130$	0	0	90
5	$-180 < \theta_5 < 180$	420	0	-90
6	$-110 < \theta_6 < 110$	0	0	90
7	$-180 < \theta_7 < 180$	180	0	0

Table 2: DH parameters SDA20D

Joint	θ [deg]	d [mm]	a [mm]	α [deg]
1	$-180 < \theta_1 < 180$	1200	0	90
2	$-180 < \theta_2 < 180$	410	0	-90
3	$-110 < \theta_3 < 110$	0	0	90
4	$-170 < \theta_4 < 170$	490	0	-90
5	$-130 < \theta_5 < 130$	0	0	90
6	$-180 < \theta_6 < 180$	420	0	-90
7	$-110 < \theta_7 < 110$	0	0	90
8	$-180 < \theta_8 < 180$	180	0	0

for each joint are in degrees. While all the joint limits are the same for the robot arm, we consider θ_1, the turntable angle for the SDA20D.

For each joint, the DH parameters are substituted in the matrix (3), resulting in seven and eight matrices for each respective robot. These matrices are multiplied in order by joints to obtain ${}^0A_7$ and ${}^0A_8$ which provides the position and orientation of the end-effector relative to the base of the robot. The resulting matrix is in the form:

$$A=\begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where n_x, n_y, n_z, s_x, s_y, s_z, a_x, a_y, a_z denotes the orientation of the end-effector with respect to the base of the robot and p_x, p_y, p_z denotes the position of the end-effector with respect to the base of the robot.

In addition, since our approach makes use of unit quaternions, we convert the rotation parameters of the matrix (4) to a unit quaternion. A unit quaternion is another form to represent the orientation of a rigid body and provides an advantage over rotation matrices by having only four parameters. A quaternion is represented with a 4-tuple (q_0, q_1, q_2, q_3) as:

$$Q = [q_0 \; q_1 \; q_2 \; q_3] = [\eta \in] \tag{5}$$

where η is the scalar part and $\epsilon=[\epsilon_x \; \epsilon_y \; \epsilon_z]^T$ is the vector part. Since the orientation is represented by a 3x3 matrix, we extract 6 from 4 and compute a unit quaternion from a matrix as follows:

$$R=\begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix} \tag{6}$$

$$\eta=\frac{1}{2}\sqrt{n_x+s_y+a_z+1} \tag{7}$$

$$\in=\frac{1}{2}\begin{bmatrix} sgn(s_z-a_y)\sqrt{n_x-s_y-a_z+1} \\ sgn(s_z-a_y)\sqrt{n_x-s_y-a_z+1} \\ sgn(s_z-a_y)\sqrt{n_x-s_y-a_z+1} \end{bmatrix} \tag{8}$$

where

$$sgn(x)=\begin{cases} 1, & x \geq 0 \\ -1 & <0 \end{cases} \tag{9}$$

Finally, by finding the homogenous transformation matrix (4) and converting the rotation matrix to a unit quaternion, we will be able to obtain the position and orientation of the end-effector given the set of the joint angles.

OPTIMIZATION APPROACH

As previously mentioned, the objective of this paper is to find the IK solution for 7 or 8 DOF robots, that is, find the joint angles for a desired position and orientation of the end-effector. The joint angles are obtained from the optimization algorithm which are then used to find the position and orientation using the FK. Therefore, we get two matrices, the desired matrix and the matrix that is being evaluated. Both matrices are expressed in the form of matrix (4) and are used to find the position and orientation error. To start, we first compute the position error by finding the Euclidean distance between the two positions using the following equation:

$$Position\ error = \sqrt{(x_d-x_c)^2+(y_d-y_c)^2+(z_d-z_c)^2} \tag{10}$$

where subscript d is the desired position and subscript c is the current position that is being evaluated. Next, we obtain the orientation error by converting both rotation matrices to a unit quaternion. Notice that by doing this conversion, the rotation is now represented by four parameters instead of nine which simplifies the problem significantly. Once we obtain the unit quaternion, we separate the orientation error into two parts, the scalar part, and the vector part. For the scalar part, we take the absolute difference of the scalar part of the desired orientation with the scalar part of the current orientation. As for the vector part, we compute the magnitude of the vector part of the desired orientation with the vector part of the current orientation as follows:

$$s=|\,q_{0d}-q_{0c}\,| \tag{11}$$

$$v=\sqrt{(q_{1d}-q_{1d})^2+(q_{2d}-q_{2d})^2+(q_{3d}-q_{3d})^2} \tag{12}$$

Next, we add the scalar and vector error and multiply it by 200 to give it weight in the cost function with respect to the position error, this weight was chosen experimentally. Finally, the total orientation error can be obtained with equation (13).

$$Orientation\ error=(s+v)*200 \tag{13}$$

Lastly, the objective function to minimize the position and orientation error is expressed in equation (14) and the optimization problem can be described as:

$$min\ f(A_d, A_c)=position\ error+orientation\ error*w \tag{14}$$

where, A_d is the desired homogenous matrix and A_c is the homogenous matrix that is being evaluated, obtained with the FK and output joint angles given by the algorithm.

PSO implementation for solving the inverse kinematics

The PSO algorithm was implemented in Matlab© on a personal laptop with an Intel Core i5, 2.40 GHz processor, and 8 GB of RAM. For the IK solution, each particle

in the swarm is made up of a seven or eight-dimensional vector, which is initialized randomly within the joint limits shown in tables 1 and 2. The parameters set for the algorithm are the following: w=1, C_1=C_2=2, max generations=300, swarm size=50, and velocity limiter=0.009. For the parameters r_1 and r_2, these change every generation to a number from 0 to 1. Also, while the generations increase in the algorithm, we decrease w by 0.01 to minimize the exploration of each particle. Even though the values for w, C_1, and C_2 are close to what is recommended in the literature, other values were manually tuned until achieving the desired performance.

Furthermore, a trajectory from point A to point B is made up of intermediate poses of the tool plate, therefore, in the algorithm, we take the solution from the previous pose and insert it to the new swarm for the new pose, similar to the elitism operation from a genetic algorithm. This will help maintain the robot in the same posture throughout the trajectory and decrease the execution time. In addition, we limit the velocity vector by taking the difference between the maximum and the minimum angles for each joint and multiply it by 0.009, this will also help the algorithm stay close to the previous solution during the trajectory while improving the execution time. Finally, after updating the new position, we apply lower and upper bound limits to make sure that the solution stays within the robot's joint limits, the pseudo-code is shown in algorithm 1.

SIMULATION AND RESULTS

To validate the performance of the proposed algorithm, two square trajectories for each robot were generated. For the 7 DOF robot, each side of the square measures 250 mm with 25 intermediate points, meaning the whole trajectory is made up of 100 poses, the same type of trajectory is applied for the 8 DOF robot. The trajectory and initial configuration of the 7 DOF robot are shown in figure 2 and for the 8 DOF robot, it is shown in figure 3.

As seen in figure 2 the robot needs to complete the trajectory outlined in red while maintaining its initial orientation. The results of the simulation for the 7 DOF robot can be seen in figure 4, where we selected ten poses from the trajectory, and it shows the robot maintaining the orientation of its end-effector and its posture throughout the trajectory. The same procedure was done with the 8 DOF robot, as seen in figure 5, ten poses were selected from the trajectory and it kept its initial orientation and posture for the whole trajectory.

To validate the precision of the algorithm, 30 separate trajectory runs were made for both robots to obtain the execution time for the whole trajectory, the average orientation and position error of all 100 poses in the trajectory, and the average generations it takes to solve the IK for one pose, this is presented in tables 3 and 5. Moreover, tables 4 and 6 list the statistical analysis for each robot where we show the best and worst run, the average of all 30 runs combined, and the standard deviation.

Algorithm 1: Inverse kinematics PSO algorithms

Initialize PSO Parameters
Generate trajectory poses
for *each pose* **do**
  P_n ←Initialize the swarm with n number of particles
  C_n ←Evaluate each particle with the cost function (14)
  BC_n ←Update the best cost for each particle
  BP_n ←Update the best position for each particle
  V_n ←Initialize all particles velocity to 0
  $GlobalP_{best}$ ←Update the global best position
  $GlobalC_{best}$ ←Update the global best cost
  for $1 : MaxGen$ **do**
    for *each* P_n **do**
      V_n ←Calculate the velocity with equation (1)
      V_n ←Apply velocity limiter
      P_n ← Update the position with equation (2)
      P_n ←Apply lower and upper bound joint limits
      C_n ←Evaluate the particle with the cost function (14)
      if $C_n < BC_n$ **then**
        $BC_n \leftarrow C_n$
        $BP_n \leftarrow P_n$
      end if
      if $BC_n < GlobalC_{best}$ **then**
        $GlobalC_{best} \leftarrow BC_n$
        $GlobalP_{best} \leftarrow BP_n$
      end if
      if $GlobalC_{best} < Desired\ Error$ **then**
        Return $GlobalP_{best}$
        break
      end if
      $w \leftarrow$ Lower the inertia constant by 0.01
    end for
  end for
end for

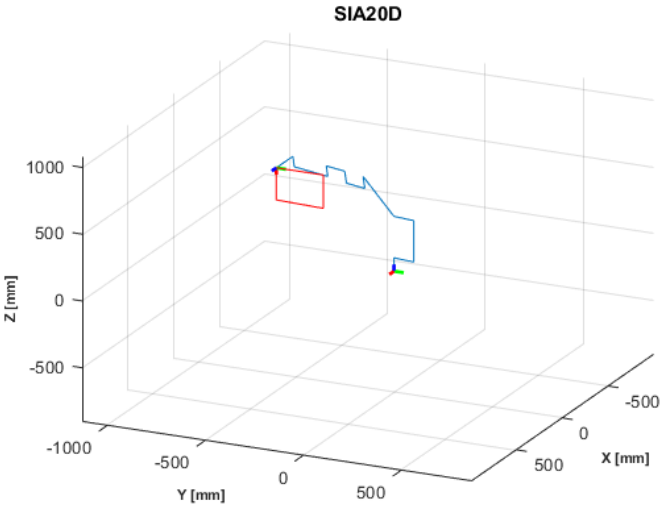


Figure 2: SIA20D initial pose and desired trajectory

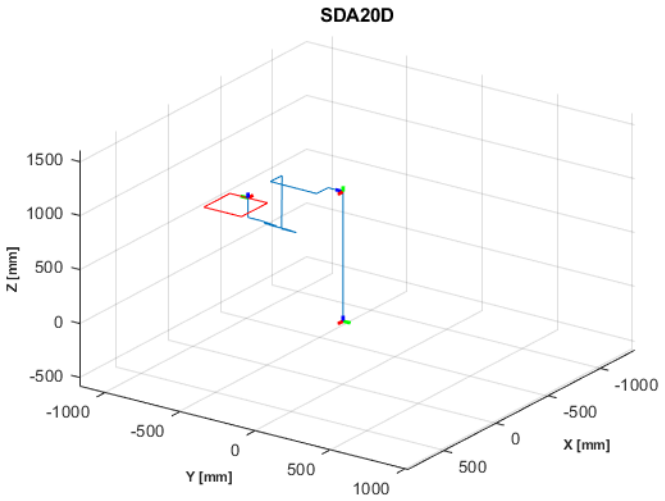


Figure 3: SDA20D initial pose and desired trajectory

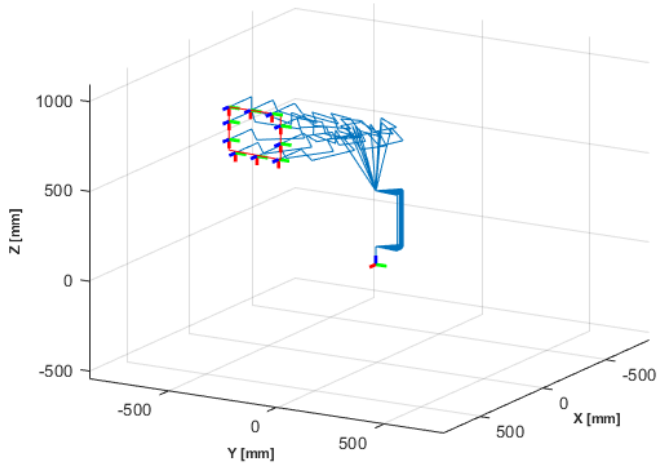


Figure 4: SIA20D trajectory

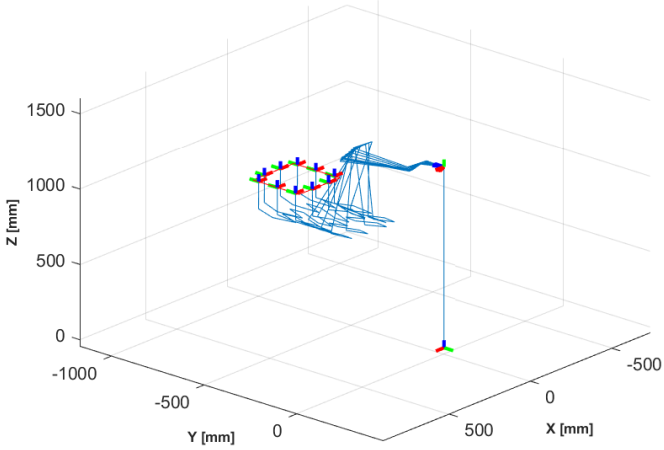


Figure 5: SDA20D trajectory

For the 7 DOF robot, the average execution time for the PSO algorithm to find the IK solutions for 100 poses is 39.51 seconds, thus for each pose, it takes an average of 0.3951 seconds to find a solution. On the other hand, for the 8 DOF robot, the algorithm takes an average of 0.4756 seconds for each pose, an increase of 0.08 seconds compared to the 7 DOF robot due to the extra degree of freedom.

In the case of the orientation error, both robots have an average error of 1.88E-03 and 2.61E-03 which can be considered zero. Additionally, the average position error for the 7 DOF robot is 8.79E-03 mm and the 8 DOF robot is 8.40E-03 mm which can be minimized even further by sacrificing execution time, however, since the repeatability factor given by the manufacturer is 0.1 mm, our solution is already well below that value. Moreover, the standard deviation data from tables 4 and 6 proves that the algorithm is finely tuned since the variation in the position and orientation error of all 30 runs is close to zero. In the case of the execution time, for both robots, the variation is around two seconds which is a good indication that the algorithm is consistent.

Finally, to verify that the robot joints move smoothly throughout the trajectory and that there are not any large movements between each pose, figures 6 and 7 show the graph of the position of the joints during the trajectory. As it can be observed, the joints move from pose to pose smoothly not generating any large movements between poses that can affect a real robot.

Table 3: Trajectory runs of PSO algorithm for the SIA20D

Run	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
1	35.26	2.04E-03	1.00E-02	133
2	40.68	1.97E-03	9.67E-03	156
3	38.22	1.68E-03	8.84E-03	141
4	39.17	2.13E-03	9.87E-03	150
5	38.68	1.94E-03	7.51E-03	156
6	41.25	2.00E-03	8.29E-03	157
7	39.69	1.99E-03	1.09E-02	152
8	42.12	1.89E-03	8.23E-03	157
9	38.79	1.64E-03	8.89E-03	143
10	40.74	1.92E-03	9.06E-03	151
11	39.98	1.65E-03	8.16E-03	138
12	36.24	1.51E-03	7.52E-03	135
13	39.72	1.89E-03	8.23E-03	151
14	43.52	2.64E-03	1.03E-02	165
15	37.72	2.24E-03	7.91E-03	146
16	38.82	1.65E-03	8.38E-03	148
17	40.50	1.62E-03	9.43E-03	151
18	41.74	1.78E-03	8.35E-03	154
19	35.17	1.89E-03	8.80E-03	142
20	40.11	1.76E-03	8.86E-03	147
21	39.71	2.11E-03	8.49E-03	163
22	38.57	2.12E-03	9.10E-03	158
23	42.80	1.78E-03	9.12E-03	157
24	40.66	1.84E-03	8.29E-03	149
25	40.20	2.01E-03	8.00E-03	147
26	37.62	1.88E-03	7.18E-03	152
27	38.01	1.61E-03	9.47E-03	136
28	40.14	1.71E-03	8..81E-03	148
29	42.26	1.78E-03	1.00E-02	153
30	37.39	1.61E-03	8.12E-03	138

Table 4: Statistical Analysis for the SIA20D

	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
Best	35.17	1.89E-03	8.80E-03	142
Worst	43.52	2.64E-03	1.03E-02	165
Average	39.52	1.88E-03	8.79E-03	149
Standard Deviation	2.01	2.31E-04	8.63E-04	7.94

Table 5: Trajectory runs of PSO algorithm for the SDA20D

Run	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
1	43.37	3.05E-03	1.01E-02	141
2	47.72	3.50E-03	7.14E-03	138
3	47.63	2.36E-03	7.43E-03	128
4	48.98	1.95E-03	7.23E-03	135
5	45.19	2.72E-03	7.17E-03	137
6	47.73	2.72E-03	7.15E-03	130
7	47.87	2.73E-03	6.91E-03	128
8	47.76	2.71E-03	6.42E-03	135
9	44.85	2.47E-03	6.81E-03	125
10	44.14	2.05E-03	6.63E-03	125
11	49.16	2.64E-03	6.16E-03	136
12	46.39	2.39E-03	8.52E-03	135
13	48.76	2.38E-03	7.11E-03	127
14	51.41	3.02E-03	7.07E-03	135
15	47.11	1.77E-03	7.03E-03	122
16	44.98	2.09E-03	7.53E-03	125
17	45.18	2.54E-03	6.73E-03	128
18	50.87	3.19E-03	7.53E-03	140
19	45.34	2.16E-03	6.73E-03	124
20	51.52	2.16E-03	6.93E-03	143
21	48.48	2.53E-03	7.01E-03	133
22	47.74	3.59E-03	4.38E-02	131
23	49.49	2.90E-03	7.18E-03	139
24	48.52	2.94E-03	6.80E-03	135
25	45.06	2.52E-03	6.94E-03	127
26	47.93	2.47E-03	7.17E-03	135
27	50.18	2.70E-03	7.57E-03	143
28	48.25	2.87E-03	7.60E-03	135
29	49.02	2.85E-03	6.80E-03	135
30	46.03	2.17E-03	6.66E-03	127

Table 6: Statistical Analysis for the SDA20D

	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
Best	44.98	2.09E-03	7.07E-03	125
Worst	51.52	2.16E-03	6.93E-03	143
Average	47.56	2.61E-03	8.40E-03	133
Standard Deviation	2.10	4.23E-04	6.60E-03	5.86

On the other hand, to validate the effectiveness of using the unit quaternion, we tested the same algorithm 30 times for both robots but using the rotation matrix in the objective function. That is, to calculate the rotation error we take the absolute difference between every parameter of the desired matrix and the matrix that is being evaluated. Then we take the sum of all nine errors and multiply it by 60, this weight was also obtained experimentally. The results of all 30 runs are shown in table 7

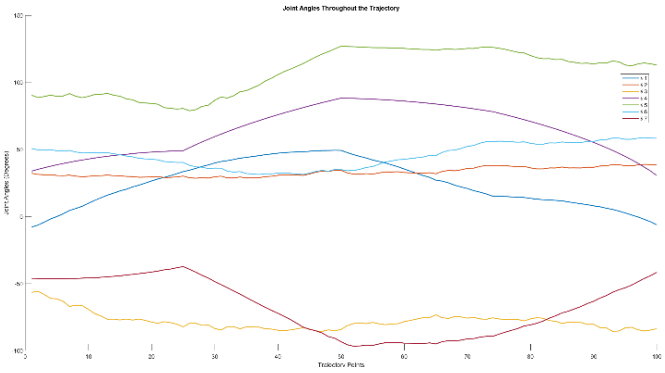


Figure 6: SIA20D joint angles throughout the trajectory

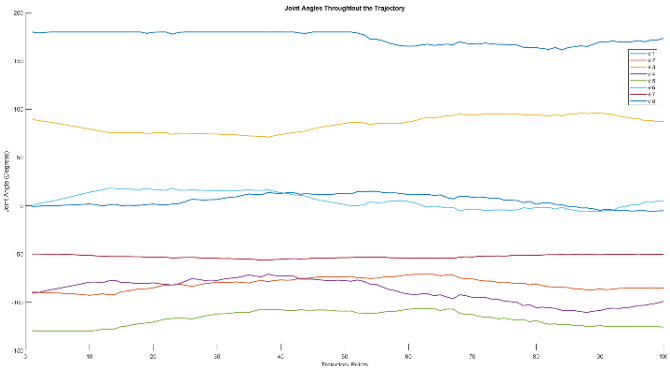


Figure 7: SDA20D joint angles throughout the trajectory

for the 7 DOF robot and table 9 for the 8 DOF robot, the statistical analysis is presented in tables 8 and 10 for each respective robot.

As it can be seen, for both robots the rotation and position error are similar to the unit quaternion approach. However, considering the average execution time, it significantly increased with the use of the rotation matrix. For the 7 DOF robot, the average execution time went up by 35 seconds which is an 89% increase. On the other hand, the average execution went up by 54 seconds for the 8 DOF robot, a 113% increase. With these results, it is obvious that the use of the unit quaternion in the objective function significantly decreases the execution time compared to the results given using the rotation matrix while also minimizing the desired orientation and position error.

CONCLUSIONS

In this paper, we presented a novel approach by employing PSO and unit quaternions to minimize both, the position and orientation errors to iteratively solve the IK problem of 7 and 8 DOF robot manipulators to describe the desired trajectory. First, the D-H parameters were obtained for each robot to solve the forward kinematics

Table 7: Trajectory runs of PSO algorithm for the SIA20D using the rotation matrix

Run	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
1	63.50	5.81E-03	3.89E-03	237
2	83.69	6.55E-03	7.20E-03	242
3	70.82	2.64E-03	4.17E-03	212
4	71.49	2.20E-03	4.00E-03	215
5	90.08	7.03E-03	7.84E-03	257
6	86.55	1.30E-02	3.62E-03	255
7	84.11	2.28E-03	3.37E-03	245
8	87.68	6.15E-03	7.13E-03	239
9	83.75	1.45E-02	6.14E-03	246
10	75.22	1.92E-02	6.38E-03	252
11	77.02	8.47E-03	3.71E-03	252
12	75.85	1.09E-02	3.77E-03	247
13	73.83	5.17E-03	3.67E-03	235
14	74.15	6.17E-03	3.80E-03	243
15	58.32	1.10E-03	5.21E-03	199
16	69.03	4.87E-03	3.57E-03	228
17	75.64	1.35E-02	3.79E-03	248
18	77.56	7.26E-03	3.86E-03	252
19	70.94	3.52E-03	4.76E-03	233
20	77.01	1.40E-02	5.82E-03	247
21	73.37	1.45E-02	3.85E-03	237
22	78.37	1.34E-02	4.01E-03	256
23	71.51	5.98E-03	3.76E-03	235
24	67.49	3.13E-03	9.88E-03	223
25	71.63	6.29E-03	3.96E-03	235
26	64.80	5.69E-03	4.80E-03	210
27	69.47	3.24E-03	3.70E-03	226
28	66.33	3.02E-03	3.95E-03	223
29	75.09	7.97E-03	7.56E-03	241
30	73.34	5.79E-03	4.55E-03	241

Table 8: Statistical Analysis for the SIA20D using the rotation matrix

	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
Best	58.32	1.10E-03	5.21E-03	199
Worst	90.08	7.03E-03	7.84E-03	257
Average	74.60	7.44E-03	4.86E-03	237
Standard Deviation	7.23	4.52E-03	1.61E-03	14.36

Table 9: Trajectory runs of PSO algorithm for the SDA20D using the rotation matrix

Run	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
1	102.69	1.42E-02	4.04E-03	276
2	102.56	6.29E-03	1.32E-03	274
3	96.90	8.84E-03	2.32E-03	259
4	101.28	7.18E-03	4.98E-03	278
5	102.38	6.36E-03	2.64E-03	275
6	105.38	7.72E-03	7.11E-03	280
7	103.10	7.06E-03	4.15E-03	269
8	101.02	7.81E-03	3.06E-03	266
9	108.83	8.62E-03	2.37E-03	286
10	100.15	6.99E-03	2.90E-03	262
11	106.17	7.28E-03	1.33E-03	277
12	97.28	6.62E-03	5.44E-03	260
13	99.84	5.42E-03	1.33E-03	267
14	100.52	6.98E-03	3.05E-03	269
15	100.26	5.79E-03	1.31E-03	264
16	100.66	5.64E-03	1.72E-03	266
17	101.11	7.78E-03	2.40E-03	268
18	102.45	7.26E-03	2.27E-03	276
19	94.08	5.98E-03	1.61E-03	253
20	96.31	7.09E-03	1.95E-03	260
21	97.13	9.77E-03	2.37E-03	262
22	97.52	6.16E-03	1.42E-03	264
23	99.79	9.35E-03	2.41E-03	265
24	100.19	7.70E-03	2.84E-03	266
25	99.46	7.85E-03	3.02E-03	264
26	103.31	6.52E-03	2.79E-03	275
27	95.97	6.36E-03	3.91E-03	259
28	106.72	9.68E-03	1.08E-03	280
29	103.95	8.42E-03	3.33E-03	272
30	106.74	9.12E-03	7.37E-03	285

Table 10: Statistical Analysis for the SDA20D using the rotation matrix

	Time [s]	Ori. Error	Pos. Error [mm]	Avg Gen
Best	94.08	5.98E-03	1.61E-03	253
Worst	101.28	7.18E-03	4.98E-03	270
Average	101.12	7.59E-03	2.93E-03	269
Standard Deviation	3.47	1.70E-03	1.57E-03	7.96

problem. Then, the objective function for the PSO was designed to minimize the position and orientation error for the desired pose, where the position error is obtained using the Euclidian distance function and the orientation

error with the unit quaternion. The advantage of the objective function is that the unit quaternion simplifies the problem by having only four parameters to represent the orientation as opposed to the rotation matrix which has nine. To test the PSO algorithm, a simulation of two square trajectories were generated for each robot with 100 desired poses. The results show great performance as the average execution time for the trajectory for the 7 DOF robot was 39.52 seconds or 0.39 seconds per pose. Furthermore, the average rotation error was 1.88 E-03 which can be considered zero and the average position error is 8.79E-03 mm. On the other hand, the average execution time for the 8 DOF robot for a trajectory is 47.56 seconds or 0.47 seconds per pose, with an average rotation and position error of 2.61E-03 and 8.40E-03 mm for the latter. Also, based on the statistical analysis for both robots it shows that the algorithm is stable and well optimized. To show the advantage of using the unit quaternion representation, the same trajectories were generated but using the rotation matrix in the objective function. The results showed a similar average position and orientation error as with the quaternion approach, but the average execution time increased by 89% for the 7 DOF robot and 113% for the 8 DOF robot. Hence, making our approach feasible for applications like trajectory planning with redundant robots, which due to their dexterity provides flexibility in avoiding workspace obstacles.

ACKNOWLEDGEMENTS

The authors would like to thank the Instituto Politécnico Nacional for the funding of this work through project SIP20200631.

REFERENCES

1. Dereli, S., Köker, R. A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. Artif Intell Rev 53, 949–964 (2020). https://doi.org/10.1007/s10462-019-09683-x
2. C. Tsai, C. Hung and C. Chang, "Trajectory planning and control of a 7-DOF robotic manipulator," 2014 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, 2014, pp. 78-84, doi: 10.1109/ARIS.2014.6871496.
3. Dereli, S., Köker, R. Simulation based calculation of the inverse kinematics solution of 7-DOF robot manipulator using artificial bee colony algorithm. SN Appl. Sci. 2, 27 (2020). https://doi.org/10.1007/s42452-019-1791-7
4. M. Alkayyali and T. A. Tutunji, "PSO-based Algorithm for Inverse Kinematics Solution of Robotic Arm Manipulators," 2019 20th International Conference on Research and Education in Mechatronics (REM), Wels, Austria, 2019, pp. 1-6, doi: 10.1109/REM.2019.8744103.
5. Zhang, L., Xiao, N. A novel artificial bee colony algorithm for inverse kinematics calculation of 7-DOF serial manipulators. Soft Comput 23, 3269–3277 (2019). https://doi.org/10.1007/s00500-017-2975-y
6. Z. Zeng, Z. Chen, G. Shu and Q. Chen, "Optimization of analytical inverse kinematic solution for redundant manipulators using GA-PSO algorithm," 2018 IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, 2018, pp. 446-451, doi: 10.1109/ICPHYS.2018.8390746.
7. A. Umar, Z. Shi, W. Wang and Z. I. B. Farouk, "A Novel Mutating PSO Based Solution For Inverse Kinematic Analysis Of Multi Degree-Of-Freedom Robot Manipulators," 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 2019, pp. 459-463, doi: 10.1109/ICAICA.2019.8873449.
8. Serkan Dereli & Raşit Köker (2020) Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy, Inverse Problems in Science and Engineering, 28:5, 601-613, DOI: 10.1080/17415977.2019.1602124
9. S. V. Reyes and S. P. Gardini, "Inverse kinematics of Manipulator Robot using a PSO Metaheuristic with Adaptively Exploration," 2019 IEEE XXVI International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Lima, Peru, 2019, pp. 1-4, doi: 10.1109/INTERCON.2019.8853568.
10. Nguyen M.T., Yuan C., Huang J.H. (2019) Kinematic Analysis of A 6-DOF Robotic Arm. In: Uhl T. (eds) Advances in Mechanism and Machine Science. IFToMM WC 2019. Mechanisms and Machine Science, vol 73. Springer, Cham. https://doi.org/10.1007/978-3-030-20131-9_292
11. R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39-43, doi: 10.1109/MHS.1995.494215.
12. Burke, Edmund K., and Graham Kendall, eds. “Search Methodologies” (2014). doi:10.1007/978-1-4614-6940-7.

Paper submitted: 27.01.2021.
Paper accepted: 01.04.2021.