Engineering Applications of Artificial Intelligence 74 (2018) 104–120

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A soft computing approach for inverse kinematics of robot manipulators

Carlos Lopez-Franco *, Jesus Hernandez-Barragan, Alma Y. Alanis, Nancy Arana-Daniel [1]

University of Guadalajara, CUCEI, Computer Science Department, Mexico

ARTICLE INFO

Keywords:
Forward kinematics
Inverse kinematics
Soft computing
Path tracking

ABSTRACT

The solution of the inverse kinematics problem is an essential capability for robotic manipulators. This capability is used to solve tasks such as path planning, control of manipulators, object grasping, etc. In this paper, we present an approach for solving the inverse kinematics of robot arm manipulators using a soft computing approach. Given a desired end effector pose, the proposed approach is able to solve both the position and orientation for the inverse kinematic problem. In addition, the proposed approach avoids singularities configurations, since, it is based on the forward kinematics equations. We present simulations and experiments, where a comparative study among some selected soft computing algorithms is realized. The simulations and experiments illustrate the effectiveness of the proposed approach.

1. Introduction

The solution of the inverse kinematics of robot arm manipulators is a crucial task for solving path tracking and control problems. A robot arm can be consider a redundant manipulator when the robot exceeds the number of degrees of freedom (DOF) than strictly needed for executing a task. The manipulators with 6 or more DOF are able to reach any desired end effector pose, but they are always considered as redundant robots. The solution of the inverse kinematics of redundant manipulators may have as a result multiples joint configurations to reach the same end effector pose. These redundancy solutions may have singularities configurations and non linearities which become the inverse kinematic problem difficult to solve (Huang et al., 2012). In this paper, we propose an approach for solving the inverse kinematic problem of redundant robot manipulators.

Recently, soft computing techniques have become very popular, they have been successfully applied in many research and application areas. These algorithms have gained the attention of researchers in order to solve the inverse kinematics of redundant robots (Huang et al., 2012; Glumac and Kovacic, 2013; Nearchou, 1998; Yang et al., 2007; Li et al., 2012; Du and Wu, 2011; Pham et al., 2008; Tabandeh et al., 2006).

The inverse kinematic problem can be solved by minimizing an error function using an iterative process (Glumac and Kovacic, 2013; Sciavicco and Siciliano, 1996). However, conventional approaches suffers problems such as singularities configurations (Sciavicco and Siciliano, 1996). These singularities make the robot joint speeds necessary to achieve a configuration become excessive. In order to overcome such problems, we proposed the use of soft computing algorithms to solve the inverse kinematics, as an optimization problem. An objective function was formulated based on the kinematics of the redundant manipulator, where the position and orientation of the end effector are considered. During the iterative process, the proposed approach solves the inverse kinematics based on its forward kinematics equations, since the forward kinematics always have a solution to find a position and an orientation for the given joint settings. This approach avoids the problems of singularities configurations.

In this paper, we perform tests with different soft computing algorithms to solve the inverse kinematics problem for robot manipulators. The soft computing algorithms considered in this work are: Artificial Bee Colony (ABC), Bat Algorithm (BA), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Cuckoo Search (CS), Differential Evolution (DE), Differential Search (DS), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). A comparative study is presented, where all such method are compared solving the inverse kinematic and path tracking problems for some manipulators. We also compare this algorithms using nonparametric statistical tests such as The sign test and the Wilcoxon signed ranks test. Furthermore, we compare the proposed approach against a traditional inverse kinematics method. In addition, experimental results are presented, where the best soft computing algorithm is used for solving the path tracking problem in a real robot manipulator.

Then the contribution of this paper is an approach for the solution of the inverse kinematics problem for robot manipulators, including the

* Corresponding author.
E-mail address: carlos.lopez@cucei.udg.mx (C. Lopez-Franco).
[1] Since 1880.

https://doi.org/10.1016/j.engappai.2018.06.001
Received 10 June 2017; Received in revised form 4 May 2018; Accepted 4 June 2018

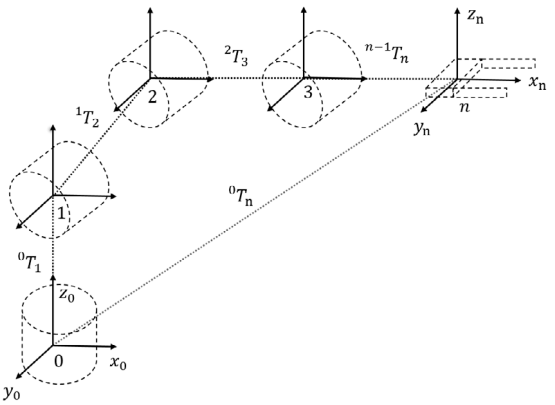


Fig. 1. Kinematic chain of a robot. $^{i-1}T_i(\theta_i)$ for $i \in 1, 2, 3, \ldots, n$ is the homogeneous matrix that represents the coordinate frame of the link i.

redundant robots arms. This approach solves the inverse kinematics problem for the given position and orientation of the end effector pose. Because the proposed approach does not require a Jacobian matrix and the forward kinematics always have a solution, there are not singular configurations as the result of the inverse kinematics. The simulation and experimental results prove the effectiveness of the proposed approach.

The rest of the paper is organized as follows: the next section will present a brief introduction to the forward and inverse kinematics. Then, in Section 3 the soft computing algorithms are described. In Section 4, the proposed approach is described. The results of simulations and real experiments are given in Section 5 and Section 6, respectively. Finally, in Section 7 we give the conclusions.

2. Problem statement

Robotic manipulators are composed of a series of rigid bodies (links) connected by joints, this structure forms a kinematic chain. The number of joints defines the degrees of freedom (DOF) of a robot manipulator. Each joint is associated with an articulation and it defines a joint variable. The computation of the end effector pose given the joint variables is a straight forward problem and it is called forward kinematics (Craig, 2005; Spong and Vidyasagar, 2008). On the other hand, the computation of the joint variables given the end effector pose is not straight forward and it is called inverse kinematics. The next subsection gives a brief introduction to the forward and inverse kinematics.

2.1. Forward and inverse kinematics

A robot system can be described from the kinematic model of its links and joint of the robot. Considering the robot's kinematic chain in Fig. 1, the forward kinematic can be calculated as

$$^0T_n = {}^0T_1(\theta_1)\ {}^1T_2(\theta_2)\ {}^2T_3(\theta_3)\ \cdots\ {}^{n-1}T_n(\theta_n) = \prod_{i=1}^{n} {}^{i-1}T_i(\theta_i) \tag{1}$$

where n is the total number of degrees of freedom, ^0T_n represents a homogeneous matrix that contains the position and orientation of the end effector pose and the parameters $\theta_i \in 1, 2, 3, \ldots, n$ represent the position for each joint of the robot.

The forward kinematics can be expressed as the product of the matrices that represents each link, from the beginning at the end. The kinematics of the robot can be described using the Denavit–Hartenberg (DH) model. In this model each link is represented by an homogeneous matrix $^{i-1}T_i(\theta_i)$ that transforms the frame attached to the link $i-1$ into the frame fixed to link i. This homogeneous matrix can be expressed as the product of four basic transformations

$$^{i-1}T_i(\theta_i) = T_{rot_z}(\theta)\ T_{trans_x}(a)\ T_{trans_z}(d)\ T_{rot_x}(\alpha) = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where the parameters θ_i, a_i, d_i and α_i are the parameters associated with the link and joint i. These parameters represent the joint angle, link length, link offset and link twist respectively. For brevity, we represent sin and cos operations with the letters s and c, respectively.

The homogeneous matrix $^{i-1}T_i(\theta_i)$ can be written as

$$^0T_n = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ & \mathbf{0} & & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

where R is a rotation matrix that represents the orientation of the respectively frame i and the vector $\mathbf{p}$ represent its position.

The inverse kinematics consist in the computation of the joint variables given the pose of the end effector (Sciavicco and Siciliano, 1996). This can be defined as the computation of the vector $\mathbf{q}$ for the function (1), where, the vector $\mathbf{q}$ is defined as

$$\mathbf{q} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \cdots & \theta_n \end{bmatrix}^T \tag{4}$$

For further information about forward and inverse kinematics, we encourage the reader to read (Sciavicco and Siciliano, 1996; Craig, 2005; Rafael Kelly Ph.D. and Davila Ph.D., 2005; Spong and Vidyasagar, 2008).

3. Soft computing algorithms

The solution of the inverse kinematics can be solve by minimizing an error function using some iteration process. This error function usually represents an euclidean distance between the desired and the actual end effector position. Given a 2 DOF manipulator with links length $L_1 = 0.5$ and $L_2 = 0.5$ meters, we can solve the inverse kinematics for a desired end effector position at the point $x = 0.5$ and $y = 0.5$ meters. In this case, there are two different arm configurations to reach the desired end effector position, see Fig. 2. One possible solution is given by $\theta_1 = 0$ and $\theta_2 = 1.5708$ and the other solution is given by $\theta_1 = 1.5708$ and $\theta_2 = -1.5708$ radians. Both solution are suitable if they are inside of the joints limits.

As we can see in Fig. 2, the inverse kinematics problem is given by a multimodal function. In this case, two local minimum represent the suitable joint configurations for the desired end effector position. However, many local minimum may be presented as the number of DOF increase, in consequence the optimization problem becomes challenging to solve. For this reason, it is necessary to search for no conventional methods to solve the inverse kinematics problem. In this work, we propose the use of soft computing algorithms in order to solve the inverse kinematics problem as a constrained optimization problem. The proposed approach is able to provide the optimal joints configuration for the given end effector pose. If there exist suitable multiples solutions for the inverse kinematics, the proposed approach provides whichever of these solutions. In the case of a path tracking problem, the solution of the inverse kinematics for every point in the path is computed taking into account the previous joints configurations. This last avoids abrupt changes in the joints configurations during the path tracking task.

Soft computing algorithms have been commonly used for solving many optimization problems such as classification, clustering, image processing and neural networks. Several soft computing algorithms have

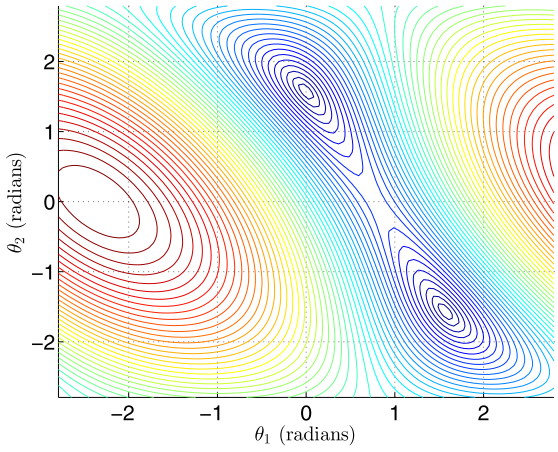


Fig. 2. Contour plot for an error function of a 2 DOF planar robot arm.

been proposed in the field of medicine as well (Selvaraj et al., 2014). Currently, there are many soft computing algorithms, the most common are Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007), Bat Algorithm (BA) (Yang, 2010), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2006), Cuckoo Search (CS) (Yang and Deb, 2009), Differential Evolution (DE) (Storn and Price, 0000), Differential Search (DS) (Civicioglu, 2012), Genetic Algorithm (GA) (Man et al., 1996) and Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995). In this work, we test all these algorithms to solve the inverse kinematics problems.

4. Description of the proposed approach

In this work we propose to use a soft computing approach to solve the inverse kinematic problem of robot manipulators. The inverse kinematics problem consist in computing a vector $\mathbf{q}$ that satisfy the position $\mathbf{p}$ and orientation R required for the homogeneous matrix ${}^0T_n$. One of the methods for solving the inverse kinematic problem is the geometric approach. However, the method is very specific and cannot be generalized. Another approach to solve the inverse kinematics of a robot consist in minimizing an error function using an iterative optimization technique. The iterative solution of the inverse kinematics using conventional methods may have singularities problems.

In this work we propose to solve the inverse kinematics problem using the forward kinematics equations and a soft computing algorithm. Our approach does not suffer from singularities since the forward kinematics always have a solution. The next subsections give a detailed explanation about the proposed approach for solving the inverse kinematics problems.

4.1. Objective function

An error function can be defined as the error between the desired pose and the current pose. We have the homogeneous matrix ${}^0T_n$ which contains the desired pose of the end effector and the homogeneous matrix ${}^0\widehat{T_n}$ which represents the pose of the end effector estimated by the soft computing algorithm. The estimated pose is defined as

$$ {}^0\widehat{T}_n = \begin{bmatrix} \hat{n}_x & \hat{o}_x & \hat{a}_x & \hat{p}_x \\ \hat{n}_y & \hat{o}_y & \hat{a}_y & \hat{p}_y \\ \hat{n}_z & \hat{o}_z & \hat{a}_z & \hat{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{n}} & \hat{\mathbf{o}} & \hat{\mathbf{a}} & \hat{\mathbf{p}} \\ & \mathbf{0} & & 1 \end{bmatrix} = \begin{bmatrix} \hat{R} & \hat{\mathbf{p}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5} $$

We can define a position error function between the positions $\hat{\mathbf{p}}$ and $\mathbf{p}$. We also can define an orientation error function between the orientations $\hat{R}$ and R. However, we have to define two different ways to compute the orientation error, when we are working with robot manipulators with 6 or more DOF and when we use a manipulator with less than 6 DOF. The next paragraphs gives a detailed explanation for the position and orientation error functions.

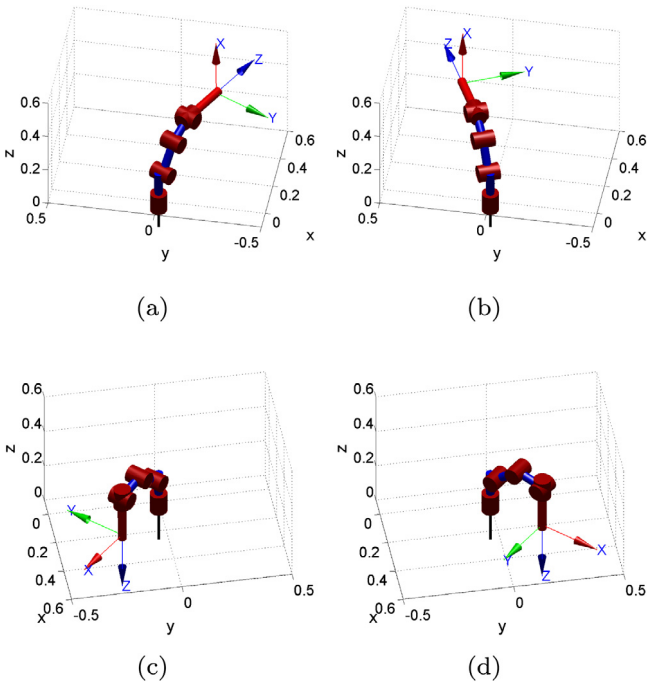


Fig. 3. Different joint configurations for an arm manipulator with less than 6 DOF.

Position error. To measure the position error p_{error}, we propose to use

$$ p_{error} = \frac{||\hat{\mathbf{p}} - \mathbf{p}||}{||\mathbf{p}||} \tag{6} $$

where $\hat{\mathbf{p}}$ and $\mathbf{p}$ correspond to the position of the end effector for the estimated and desired, respectively. The p_{error} is a relative error measure.

Orientation error of manipulator with 6 or more DOF. An arm manipulator with 6 or more DOF is able to reach the desired orientation R at any position inside of its work space. In the case of a robot manipulator with 6 DOF or more, the error function R_{error} is defined as

$$ R_{error} = \frac{||\hat{R} - R||}{||R||} \tag{7} $$

where $\hat{R}$ and R are the rotation matrix for the estimated and desired orientation, respectively.

Orientation error of manipulator with less than 6 DOF. For a manipulator with less than 6 DOF, it is not possible to reach some desired positions $\mathbf{p}$ given a desired orientation R. However, the position $\mathbf{p}$ can be reached, if we let the end effector pose rotates about a fixed axis in R. As we can see in Fig. 3(a) and Fig. 3(b), these two desired positions have been reached by rotating about the x axis. Similarly, in Fig. 3(c) and Fig. 3(d), these two positions now have been reached by rotating about the z axis.

It is known that a desired rotation matrix can be computed by $R = R_x(\theta_x) R_y(\theta_y) R_z(\theta_z)$, where R_x, R_y and R_z are basic rotations matrices that rotate about the x, y and z axes by the angles θ_x, θ_y and θ_z, respectively. Thus, to define the desired orientation in Fig. 3(a) and Fig. 3(b), we need to rotate $\theta_x = \pi$ and $\theta_y = \frac{\pi}{2}$, which results in the following rotation matrix $R_T = R_x(\pi) R_y\left(\frac{\pi}{2}\right) R_z(0)$. Then, we can fixed the x axis as the product of the rotation matrix R_T followed by a rotation

matrix $R_x(\theta)$, which results in

$$R = R_T\ R_x(\theta) = \begin{bmatrix} 0 & \sin(\theta) & \cos(\theta) \\ 0 & -\cos(\theta) & \sin(\theta) \\ 1 & 0 & 0 \end{bmatrix} \tag{8}$$

On the other hand, to define the desired orientation in Fig. 3(c) and Fig. 3(d), we need to rotate $\theta_x = \pi$ which results in the rotation matrix $R_T = R_x(\pi)\ R_y(0)\ R_z(0)$. Then, we fixed the z axis as follows

$$R = R_T\ R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ -\sin(\theta) & -\cos(\theta) & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{9}$$

As we can see in (8), the vector $\mathbf{n} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ in the rotation matrix remains constant. Similarly, in (9) the vector $\mathbf{a} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T$ now remains constant. To conclude, it is necessary to identify which axis in the rotation matrix R needs to be fixed. Then, to measure the orientation error υ_{error} for a robot arm with less than 6 DOF, we propose to use

$$\upsilon_{error} = \frac{||\hat{\mathbf{v}} - \mathbf{v}||}{||\mathbf{v}||} \tag{10}$$

where $\hat{\mathbf{v}}$ is the estimated orientation and $\mathbf{v}$ is the desired orientation. The vector $\mathbf{v}$ is assigned as the vector $\mathbf{v} = \mathbf{n}$, $\mathbf{v} = \mathbf{o}$ or $\mathbf{v} = \mathbf{a}$, where it depends on the required fixed axis.

Objective function. For manipulators with 6 or more DOF, the objective function $f({}^0\hat{T}_n, {}^0T_n)$ is defined as the summation of the position error (6) and the orientation error (7) as follows

$$f({}^0\hat{T}_n, {}^0T_n) = w\ p_{error} + (1 - w)\ R_{error} \tag{11}$$

When a manipulator has less than 6 DOF, then the objective function is defined as the summation of the position error (6) and the orientation error (10) as

$$f({}^0\hat{T}_n, {}^0T_n) = w\ p_{error} + (1 - w)\ \upsilon_{error} \tag{12}$$

where w is a weighting factors that scale the contribution of each term. Then (11) is used for arm manipulators with 6 or more DOF and (12) is used for manipulators with less than 6 DOF. The term w can be computed as

$$w = \alpha\ e^{-\delta} + \beta \tag{13}$$

where $\delta = R_{error}$, $\alpha = 0.7$ and $\beta = 0.3$. These two values are selected by experimentation. If the value of R_{error} is big ($w \approx 0.3$) then the weighting factor takes priority for the R_{error} term. On the other hand, if the value of R_{error} is small ($w \approx 1$), then the weighting factor takes priority for the p_{error} term. Finally, the objective function f is used in the proposed approach method.

4.2. How to solve the inverse kinematic problem

In the soft computing algorithm, every individual contains a set of joint variables $\hat{\mathbf{q}}_j = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \cdots & \theta_n \end{bmatrix}^T$ with $j \in 1, 2, 3, \ldots, m$, where m is the total number of individuals and n is the total number of DOF. Each individual represents an estimated solution ${}^0\hat{T}_n$ which contains the estimated position $\hat{\mathbf{p}}_j$ and the estimated orientation $\hat{R}_j$. The best solution $\hat{\mathbf{q}}_{best}$ is found using a soft computing algorithm to minimize the objective function f in each iteration, until it finds the optimal configuration $\hat{\mathbf{q}}_{best}$ for the desired pose ${}^0T_n$.

To solve the inverse kinematic problem we proceed as follows:

Initialization. Every individual $\hat{\mathbf{q}}_j$ is initialized randomly according to

$$\hat{\mathbf{q}}_j = \mathbf{l_b} + (\mathbf{u_b} - \mathbf{l_b})\ \mathbf{r} \tag{14}$$

where $\mathbf{r} = \begin{bmatrix} rand_1 & rand_2 & rand_3 & \cdots & rand_n \end{bmatrix}^T$ is a random vector with $\mathbf{r} \in [0, 1]$. The vectors $\mathbf{l_b}$ and $\mathbf{u_b}$ are the lower bound and upper bound, respectively. This boundary is defined as

$$\mathbf{l_b} = \begin{bmatrix} \theta_{1_{min}} & \theta_{2_{min}} & \theta_{3_{min}} & \cdots & \theta_{n_{min}} \end{bmatrix}^T \tag{15}$$

$$\mathbf{u_b} = \begin{bmatrix} \theta_{1_{max}} & \theta_{2_{max}} & \theta_{3_{max}} & \cdots & \theta_{n_{max}} \end{bmatrix}^T \tag{16}$$

where $\theta_{i_{min}}$ and $\theta_{i_{max}}$, with $i \in 1, 2, 3, \ldots, n$, are the minimum and maximum limits for each joint respectively.

Fitness value calculation. For every $\hat{\mathbf{q}}_j$, we compute its forward kinematic chain with (1), this equation defines an estimated homogeneous matrix ${}^0\hat{T}_n$ that contains the estimated position $\hat{\mathbf{p}}_j$ and orientation $\hat{R}_j$. We use these vectors to compute the fitness value using (12).

The best solution $\hat{\mathbf{q}}_{best}$ is found by solving an optimization problem defined as

$$\text{minimize } f({}^0\hat{T}_n, {}^0T_n), \text{ subject to } \mathbf{l_b} \leq \hat{\mathbf{q}}_j \leq \mathbf{u_b}.$$

Stop criteria. The stop criteria occurs when the algorithm achieves the total number of iterations, or the fitness value reaches a value of tolerance.

4.3. How to solve a path tracking problem

To solve a path tracking problem, we divide the path in m points. For each point $k \in 1, 2, 3, \ldots, m$ the algorithm computes its inverse kinematic. At the end, the algorithms provides an optimal configuration $\hat{\mathbf{q}}_{best}(k)$ for the desired position $\mathbf{p}(k)$ and orientation $R(k)$ for all points in the path.

The inverse kinematic for the first point is computed with individual initialized randomly. The initialization of the first point is defined as

$$\mathbf{q}_j(1) = \mathbf{l_b} + (\mathbf{u_b} - \mathbf{l_b})\ \mathbf{r} \tag{17}$$

On the other hand, the initialization of the rest of the points is defines as

$$\mathbf{q}_j(k + 1) = \mathbf{q}_j(k) \tag{18}$$

From the previous equation we can observe that the initialization of the individual is defined as the previous joint solution, with exception of the first point. This initialization is based on the fact that the joint configuration of the next point in a path is close to the joint configuration of the current point in the path.

To test the proposed approach we performed several simulations and real experiments. The results are used to examine the effectiveness and performance of the proposed approach to solve the inverse kinematic for a path tracking problem. These results are presented in the next sections.

5. Simulation results

In this section, we present a comparative study among the ABC, BA, CMA-ES, CS, DE, DS, GA and PSO algorithms. These soft computing algorithms solved the inverse kinematic problems and their results were compared. During the next subsections, we give a detailed descriptions for this simulations.

5.1. Robot arm manipulator settings

We proposed to use four different robot manipulators for simulation experiments. The first robot is an Anthropomorphic RRR robot based on the kinematic model presented in Table 1. The second robot is the 5 DOF Youbot manipulator presented in Table 2. The third robot is based on the kinematic model of the 6 DOF Puma 560 arm manipulator from Table 3. The fourth robot is a 7 DOF manipulator based on the kinematic model of the Baxter arm manipulator shown in Table 4. The simulations were implemented in Matlab using the robotics toolbox (Corke, 2011).

Table 1
DH table for the Anthropomorphic RRR arm manipulator.

Joint	a [mm]	α [rad]	d [mm]	θ [rad]
1	0	$\pi/2$	500	θ_1
2	700	0	0	θ_2
3	700	0	0	θ_3

Table 2
DH table for the KUKA Youbot arm manipulator. DH parameters obtained from Falco and Natale (2014).

Joint	a [mm]	α [rad]	d [mm]	θ [rad]
1	33	$\pi/2$	147	θ_1
2	155	0	0	θ_2
3	135	0	0	θ_3
4	0	$\pi/2$	0	θ_4
5	0	0	217.5	θ_5

Table 3
DH table for the Puma 560 arm manipulator. DH parameters obtained from Corke (2011).

Joint	a [mm]	α [rad]	d [mm]	θ [rad]
1	0	$\pi/2$	0	θ_1
2	431.8	0	0	θ_2
3	20.3	$-\pi/2$	150	θ_3
4	0	$\pi/2$	431.8	θ_4
5	0	$-\pi/2$	0	θ_5
6	0	0	0	θ_6

Table 4
DH table for the Baxter arm manipulator. DH parameters obtained from Corke (2011).

Joint	a [mm]	α [rad]	d [mm]	θ [rad]
1	69	$-\pi/2$	270.35	θ_1
2	0	$\pi/2$	0	θ_2
3	69	$-\pi/2$	364.35	θ_3
4	0	$\pi/2$	0	θ_4
5	10	$-\pi/2$	374.29	θ_5
6	10	$\pi/2$	0	θ_6
7	10	0	229.525	θ_7

The boundaries of each joint of the Anthropomorphic RRR robot are set to

$$\mathbf{l_b} = \begin{bmatrix} -160 & 150 & -135 \end{bmatrix}^T$$
$$\mathbf{u_b} = \begin{bmatrix} 160 & 150 & 135 \end{bmatrix}^T$$

The boundaries of each joint of the Youbot arm are fixed as follows

$$\mathbf{l_b} = \begin{bmatrix} -169 & 0 & -151 & -12.5 & -167.5 \end{bmatrix}^T$$
$$\mathbf{u_b} = \begin{bmatrix} 169 & 155 & 146 & 192.5 & 167.5 \end{bmatrix}^T$$

The boundaries for the Puma 560 manipulator are selected as

$$\mathbf{l_b} = \begin{bmatrix} -160 & -125 & -135 & -150 & -100 & -266 \end{bmatrix}^T$$
$$\mathbf{u_b} = \begin{bmatrix} 160 & 125 & 135 & 150 & 100 & 266 \end{bmatrix}^T$$

Finally, the boundary for the Baxter manipulator are fixed as follows

$$\mathbf{l_b} = \begin{bmatrix} -97.5 & -123 & -175 & -3 & -175 & -90 & -175 \end{bmatrix}^T$$
$$\mathbf{u_b} = \begin{bmatrix} 97.5 & 60 & 175 & 150 & 175 & 120 & 175 \end{bmatrix}^T$$

5.2. Soft computing algorithms settings

The parameters settings that are common for all the soft computing algorithms used to test the proposed approach are the population size and the number of evaluations. We decided to use a population size of 30 individuals and a maximum of 350 evaluations for each algorithm. The particular parameters setting for each algorithm defines as follows. The parameters settings for ABC are $n_o = 50\%$ of the colony, $n_e = 50\%$ of the colony, $n_s = 1$ and $limit = n_e * D$. We used the standard ABC algorithm with these setting values base on Karaboga and Basturk (2008). The parameters setting for the BA are defined accordingly to the results obtained in Yang (2010). We selected the following setting for the standard BA algorithm. The values are $\alpha = \gamma = 0.9$, $f_{min} = 0$ and $f_{max} = 2$. We also use the initial loudness A_i^0 as a random value between $[1, 2]$ and r_i^0 a random value $\in [0, 1]$. For the CMA-ES algorithm, we use the default parameters reported in Hansen (2006). They suggest to keep that configuration, except for the population size λ. We increased the population size to $\lambda = 30$. We used the standard CMA-ES algorithm. The parameters of the standard CS algorithm were set to $p_a = 0.25$ and $\alpha = 1.5$. We based these setting as recommended in Civicioglu and Besdok (2011). Based on Gämperle et al. (2002), we fixed the parameters F and CR to 0.6 and 0.9, respectively. We use the standard DE with the mutation strategy $DE/best/2/bin$. As reported by Civicioglu (2012), the most appropriate parameters setting for the DS algorithm are $p_1 = 0.3 * rand$ and $p_2 = 0.3 * rand$. The standard version of DS is used with the Subjective method. We use the standard version of GA with the parameters selection p_c and p_m fixed to 0.9 and 0.01, respectably. We based this setting on Man et al. (1996), as they suggest. The PSO version used in this test includes the inertia weight modification. Here, the inertia weight is set to $w = 0.8$ as recommended in Shi and Eberhart (1998) and both the parameters C_1 and C_2 are fixed to 2.

5.3. Simulation tests

The simulation tests were conducted as follows: First, each soft computing algorithm (ABC, BA, CMA-ES, CS, DE, DS, GA, PSO) computes the inverse kinematic problem for the considered arms manipulators, the second part consist in solving a line and a circle path tracking problems. The soft computing algorithms run 500 times and their results are shown using box plot to represent the data distribution. The intention of the use of box plots is to display graphically statistical variation of the proposed soft computing algorithms results. In the simulations, the execution time, position error, orientation error and the displacement results are presented into box plots, where we are looking for the algorithm with the smaller data distribution with the lower value results and the less quantities of outliers. The position error is a Euclidean distance between the desired and the estimated position of the end effector, the orientation and the displacement results are relative error measures. In the comparisons, the values of each point in the path where averaged, thus, we show the data distribution which correspond to all averaged point in the path.

5.3.1. Solution of the line path tracking

For the Anthropomorphic RRR arm manipulator, the line is given by a start point $\mathbf{p}_{start} = \begin{bmatrix} 1.2 & -0.2 & 1.0 \end{bmatrix}^T$ and an end point $\mathbf{p}_{end} = \begin{bmatrix} 1.2 & 0.2 & 1.0 \end{bmatrix}^T$. With these points we compute $P_{total} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \cdots & \mathbf{p}_k \end{bmatrix}^T$ which contains the k-points that belong to the line. In this case, the orientation problem was not considered due to the manipulator kinematics limitations. For the KUKA Youbot arm manipulator, the line is given by a start point $\mathbf{p}_{start} = \begin{bmatrix} 0.4 & -0.2 & 0.1 \end{bmatrix}^T$ and an end point $\mathbf{p}_{end} = \begin{bmatrix} 0.4 & 0.2 & 0.1 \end{bmatrix}^T$. The desired orientation for each point is fixed to $\mathbf{v} = \mathbf{n} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. In the case of the Puma 560 arm manipulator, the line is given by a start point $\mathbf{p}_{start} = \begin{bmatrix} 0.5 & -0.2 & 0.5 \end{bmatrix}^T$ and an end point $\mathbf{p}_{end} = \begin{bmatrix} 0.5 & 0.2 & 0.5 \end{bmatrix}^T$. Respect to the Baxter arm manipulator, the line is given by a start point $\mathbf{p}_{start} = \begin{bmatrix} 0.8 & -0.2 & 0.6 \end{bmatrix}^T$ and an end point $\mathbf{p}_{end} = \begin{bmatrix} 0.8 & 0.2 & 0.6 \end{bmatrix}^T$. The desired orientation for the Puma 560 and Baxter manipulators are fixed to

$$R = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

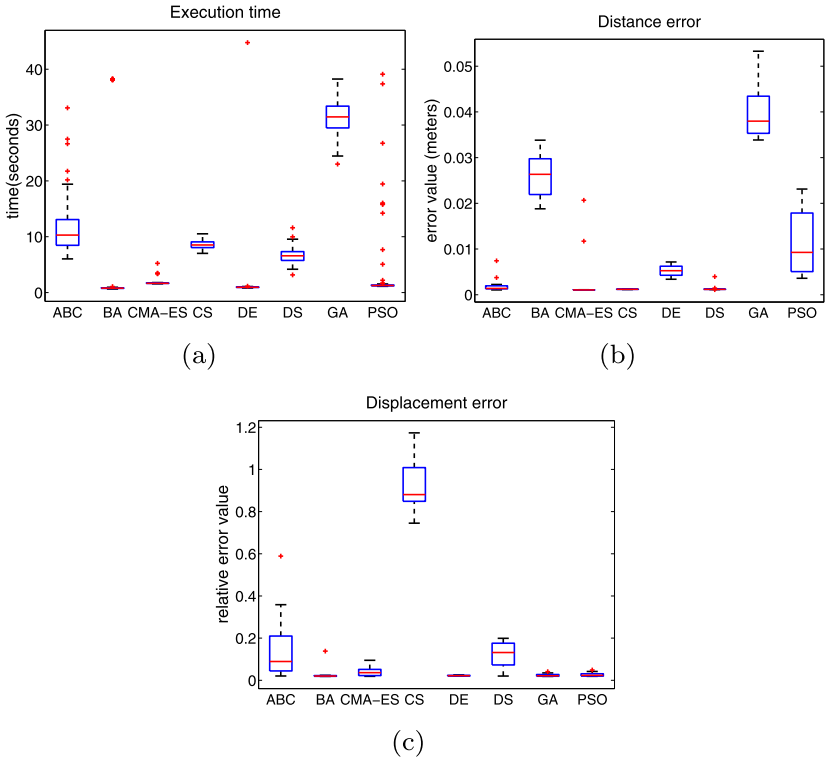


Fig. 4. Line tracking results for the Anthropomorphic RRR arm manipulator.

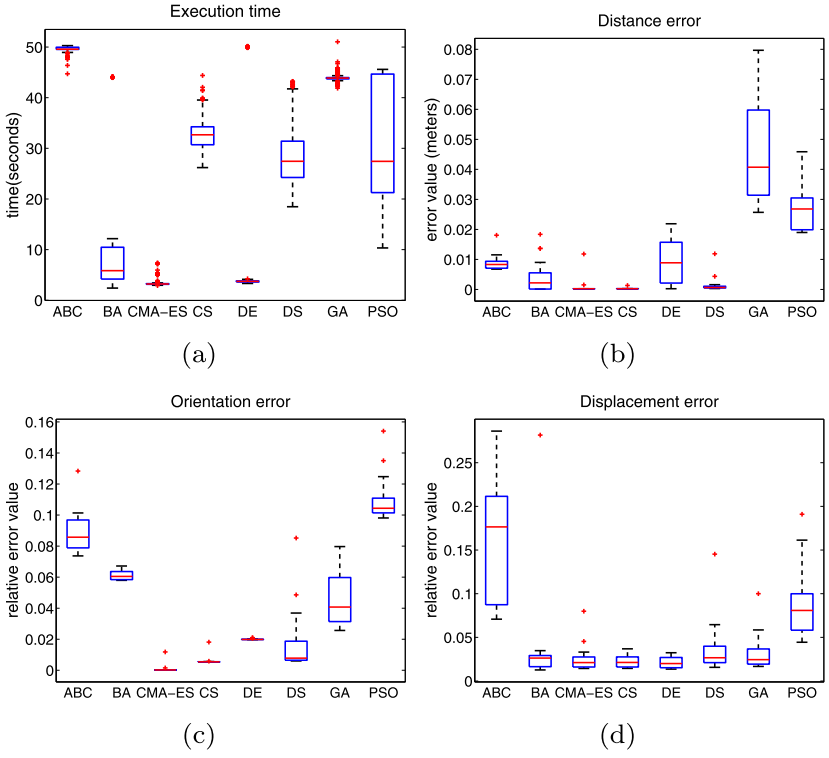


Fig. 5. Line tracking results for the KUKA Youbot arm manipulator.

Finally, the allowed tolerance for the considered manipulator are set to $tol = 0.001$.

The simulations results for the Anthropomorphic manipulator are given in Fig. 4. The execution time results are illustrated in Fig. 4(a). We notice that BA, CMA-ES, and CS algorithms had the best performance, their execution time performed below to 5 s. The PSO algorithm presented good results, but it has larger quantities outliers. Respect to the position error in Fig. 4(b), CMA-ES, CS and DS performed below to 0.1 mm, they also had the smaller data distribution. The others algorithms reported bigger results with a larger data distribution. Finally, Fig. 4(c)

shows the displacement error which represents the relative movement error of each joint of the robot arm for each point in the path. A small data distribution here represents a minimal movement error of the joints for each point. In this case, BA, CMA-ES, DE, GA and PSO presented results below to 15% of error. The BA and DE algorithms reported the smaller data distribution.

In Fig. 5, the simulations results for the Youbot manipulator are presented. The results of the execution time are shown in Fig. 5(a). We can observe that the algorithms CMA-ES and DE have better performance among the others. The rest of the algorithms have a large data distribution or their execution time is greater. Fig. 5(b) shows the position error case. In this case the algorithms CMA-ES and CS have the best performance, their results performed below to 0.0005 which represents 0.5 mm. The DS algorithm has good results below to 0.002 with a data distribution larger than CMA-ES and CS. Fig. 5(c) presents the orientation error case. The relative error for CMA-ES and CS performed below to 0.01 which means that the error is less than 1%, the other algorithms present a bigger error. A small data distribution represents small changes in the orientation of the end effector for the points in the path, which is the case for CMA-ES, CS and DE. The data distribution is shown in Fig. 5(d). In this case the algorithms BA, CMA-ES, CS, and DE have similar data distribution. These four algorithms performed below to 0.04 which means that the minimal movement error is below to 4%.

The simulations results for the Puma manipulator are shown in Fig. 6. Then, Fig. 6(a) illustrates the execution time results. In this case, it is shown that CMA-ES outperformed the other algorithms with results under 10 s. The position error is reported in Fig. 6(b), where ABC, CS and DS performed better than the others with the smaller data distribution and results below to 2.5 mm. The orientations results are shown in Fig. 6(c), where ABC and CS reported results under 1% of error, they also reported a small data distribution. Finally, the displacement results are given in Fig. 6(d). In this case, the BA, DE and GA, performed better than the others algorithms. They performed below to 15% of error. We notice that, BA, DE, CS, DS and PSO reported a large data distribution in general.

The simulations results for the Baxter manipulator are illustrated in Fig. 7. The execution time results are reported in Fig. 7(a). In this case, the CMA-ES and DE algorithms performed below to 10 s, the other algorithms reported bigger execution time. Then, in 7(b), CMA-ES and CS presented a small data distribution with results under 2 mm. DE and DS also presented results under 5 mm with a larger data distribution. Respect to the orientation case in 7(c), the CMA-ES algorithm outperformed the others with the smaller data distribution and results below to 1% of error. Finally, in 7(d) the results for the displacement error are given. Here, the BA, CMA-ES, DE, GA and PSO algorithms reported results under an error of 20%, where BA, GA and PSO had the smaller data distribution.

According to the results presented previously, CMA-ES and CS appears to be the best algorithms for the line path tracking problem without taking into account the execution time. However, the algorithm CMA-ES is consider the algorithm with the best results in general. The other algorithm reported bigger error results with larger data distributions.

5.3.2. Solution of the circle path tracking

For the Anthropomorphic RRR arm manipulator, we defined a circle with center $\mathbf{p}_{center} = \begin{bmatrix}1.2 & 0.0 & 1.0\end{bmatrix}^T$ and radio $r = 0.05$ m. For this manipulator, the orientation problem was not considered due to the manipulator kinematics limitations. For the KUKA Youbot arm manipulator, we defined a circle with center $\mathbf{p}_{center} = \begin{bmatrix}0.4 & 0.0 & 0.3\end{bmatrix}^T$ and radio $r = 0.05$ m. The desired orientation for each point is fixed to $\mathbf{v} = \mathbf{n} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. In the case of the Puma 560 arm manipulator, we defined a circle with center $\mathbf{p}_{center} = \begin{bmatrix}0.5 & 0.2 & 0.3\end{bmatrix}^T$ and radio $r = 0.05$ m. Respect to the Baxter arm manipulator, we defined a circle with center $\mathbf{p}_{center} = \begin{bmatrix}0.8 & 0.0 & 0.6\end{bmatrix}^T$ and radio $r = 0.05$ m. The desired orientation for the Puma 560 and Baxter manipulators are fixed to

$$R = \begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

and the allowed tolerance for the considered manipulator are set to $tol = 0.001$.

Fig. 8 presents the simulation results for the Anthropomorphic RRR arm manipulator. Respect to the execution time results reported in Fig. 8(a), the BA, CMA-ES, DE and PSO algorithms performed better than the others. Their results are presented below to 5 s with a small variation in the data distribution. In the case of the position error in Fig. 8(b), ABC, CMA-ES, CS, DE and DS had results under 2.5 mm and a small data distribution. BA also presents a small data distribution with an error of 5 mm. Finally, the displacement error results are given in Fig. 8(c). In this case, BA, DE, GA and PSO had results under 20% of error and the smaller data distribution. The CMA-ES and DS algorithms also had results under 20% with a bigger data variation.

The simulations results for the Youbot manipulator are presented in Fig. 9. In Fig. 9(a) we can observe that CMA-ES is the only algorithm with good results, it performs constantly below to 6 s. In contrast, all the others algorithms take either to much execution time or they have a large data distribution. The position error shown in Fig. 9(b). In this case, CMA-ES and CS perform below to 0.0004 m which is a 0.4 millimeters. The algorithms DE and DS have good results as well, but they have a larger data distribution. Then, Fig. 9(c) shows the results of the orientation error. The algorithms CMA-ES and CS have the smaller data distribution. Their relative error performed below to 0.006 which is an error of 0.6%. The other algorithms have a bigger error with a larger distribution. Finally in Fig. 9(d) we can observe that the relative displacement error is similar to BA, DE and GA. They performed better than the others algorithms with a relative error below to 0.05 which is an error of 5%.

Fig. 10 illustrates the simulation results for the Puma 560 arm manipulator. The execution time results are shown in Fig. 10(a), where CMA-ES and DE outperformed the other algorithms with results below to 10 s. The other algorithms required a bigger execution time and a larger data distribution. In the case of the distance error in Fig. 10(b), the ABC, CMA-ES and CS algorithms performed below to 1 millimeters while BA, DE, GA and PSO performed above 20 millimeters. In Fig. 10(c), the orientation error results are presented. Here, the CMA-ES algorithm outperformed the others with an error measure under 1%. Finally, the displacement results are shown in Fig. 10(d). As we can see, BA, CMA-ES, DE, GA and PSO reported a minimal joint movement below to 10% of error.

Fig. 11 illustrates the simulation results for the Baxter arm manipulator. The execution time results are given in Fig. 11(a), where CMA-ES and DE outperformed the other algorithms with results below to 10 s. The execution time results for the other algorithms are given adobe 50 s. Then, in Fig. 11(b) the distance error results are presented. In this case, CMA-ES, CS, DE and DS reported results below to 2 mm and a small distribution variation. The orientation results are presented in Fig. 11(c), where CMA-ES outperformed the other algorithms with values below to 1% of error. The other algorithms presented bigger errors. Finally, the displacement errors are given in Fig. 11(d). In this case, BA, GA and PSO reported the best results with a small data distribution and results below to 10% of error. Additionally, CMA-ES, DE and CS reported error results below to 20%.

According to the results presented so far, the algorithms CMA-ES and CS have better performance in almost all the results, but if we consider the results of the execution time, then CMA-ES is the best algorithm for the circle path tracking.

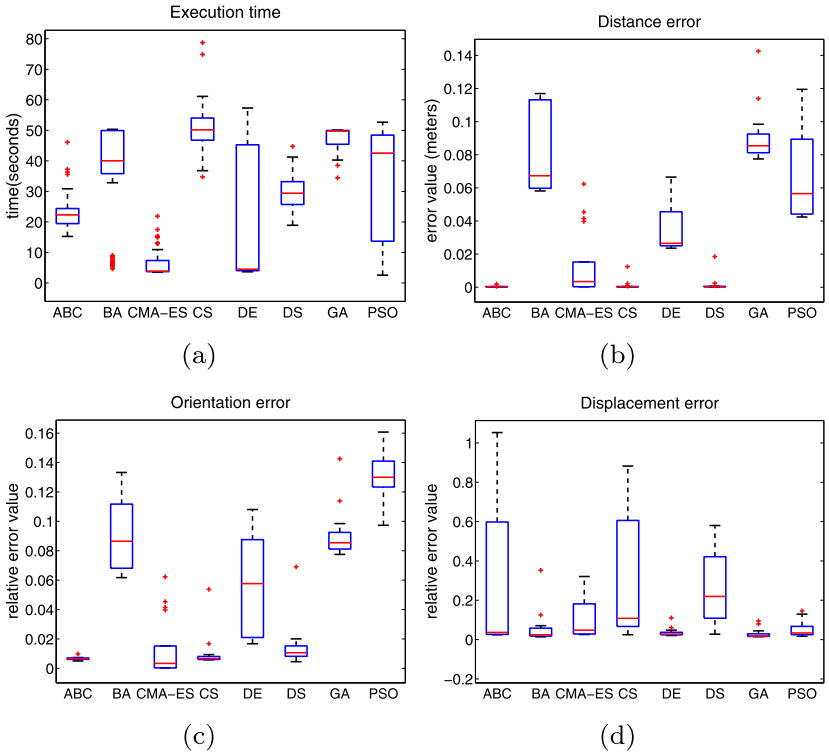


Fig. 6. Line tracking results for the Puma 560 arm manipulator.

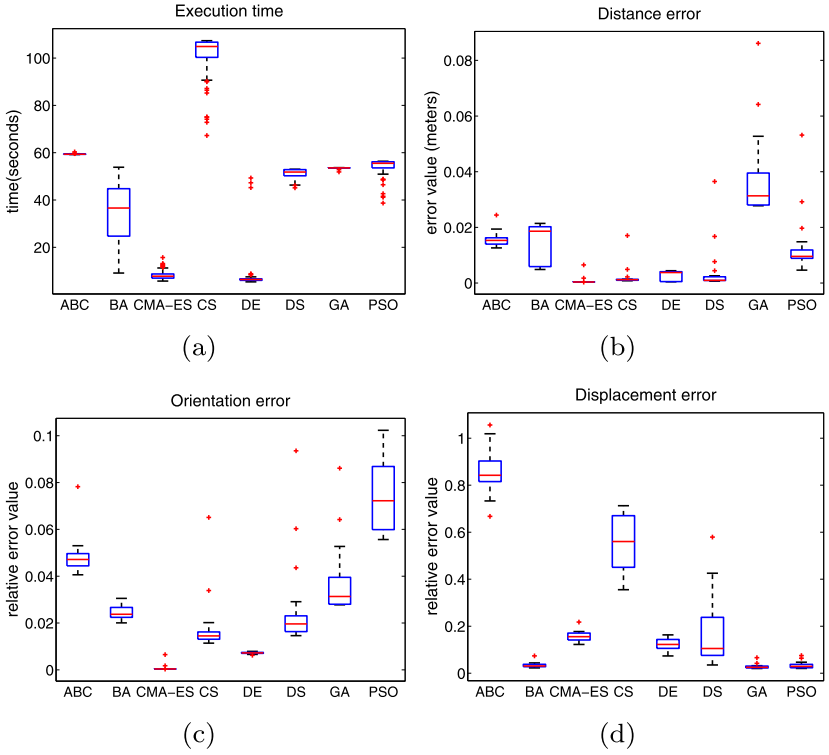


Fig. 7. Line tracking results for the Baxter arm manipulator.

5.4. Nonparametric statistical tests

Nonparametric tests are used to compare the soft computing algorithms. These tests can be used to prove if a soft computing algorithm has a significant improvement against the others algorithms. We use the sign test and the Wilcoxon signed ranks test for pairwise comparison. The results of these two comparison tests were compared to the results obtained from simulations based on their data distribution.

The basic concepts of nonparametric statistics are the null hypothesis and the alternative hypothesis. The null hypothesis is assumed when

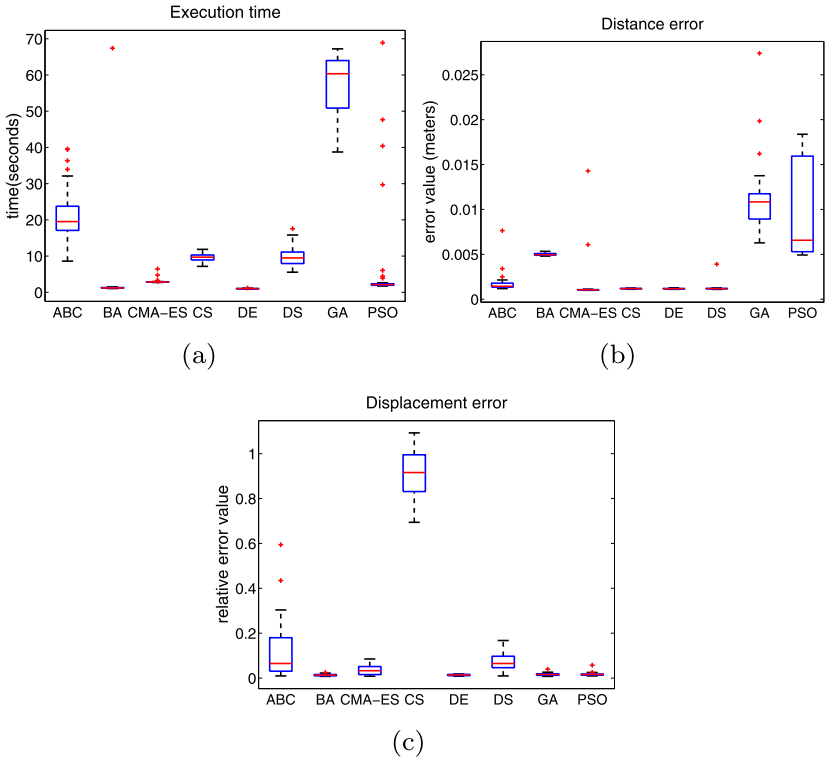


Fig. 8. Circle tracking results for the Anthropomorphic RRR arm manipulator.

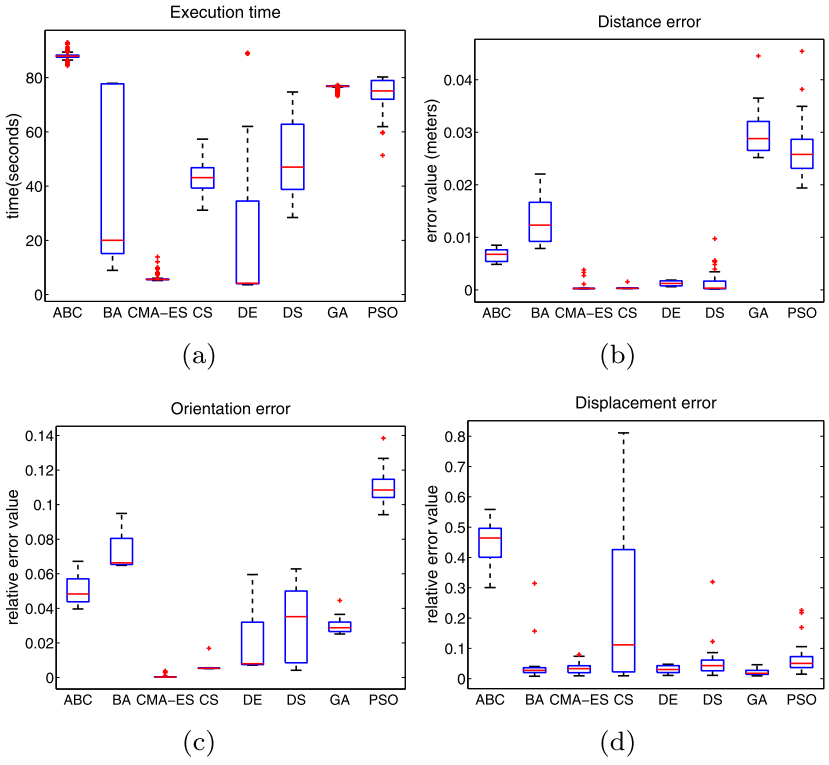


Fig. 9. Circle tracking results for the KUKA Youbot arm manipulator.

the compared algorithms are equivalent, and the alternative hypotheses represents significant difference between them. A level of significance α is used to determine at which level a hypothesis may be rejected, and a p-value provides information about whether a statistical hypothesis test is significant or not. In Derrac et al. (2011), the authors give a detailed information about nonparametric statistical tests.

For the sign test and the Wilcoxon signed ranks test, we use the KUKA Youbot arm based on the DH parameters presented in Table 2. We also use 25 different end effector poses, where the desired orientation

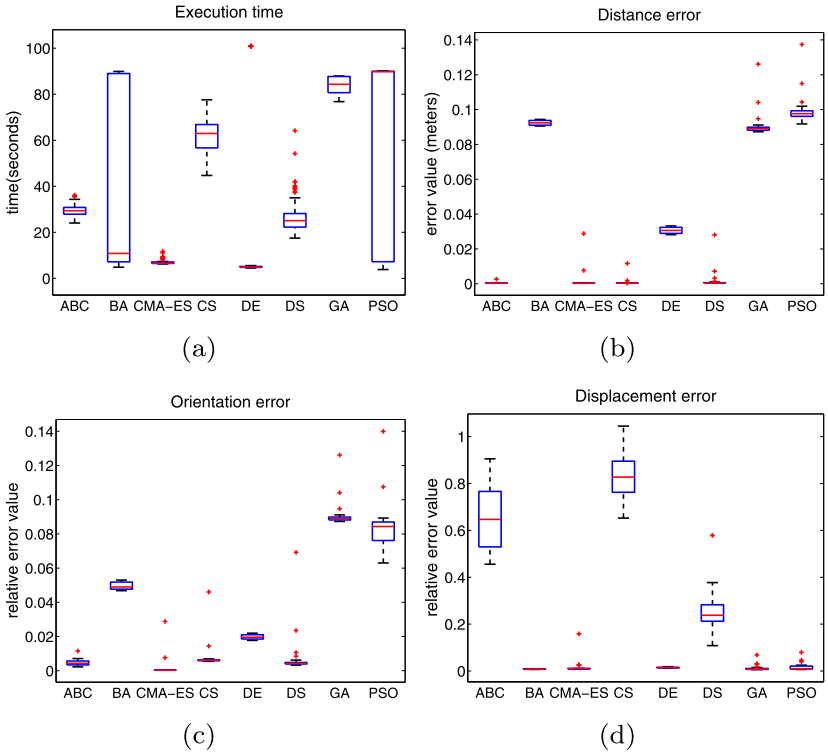


Fig. 10. Circle tracking results for the Puma 560 arm manipulator.

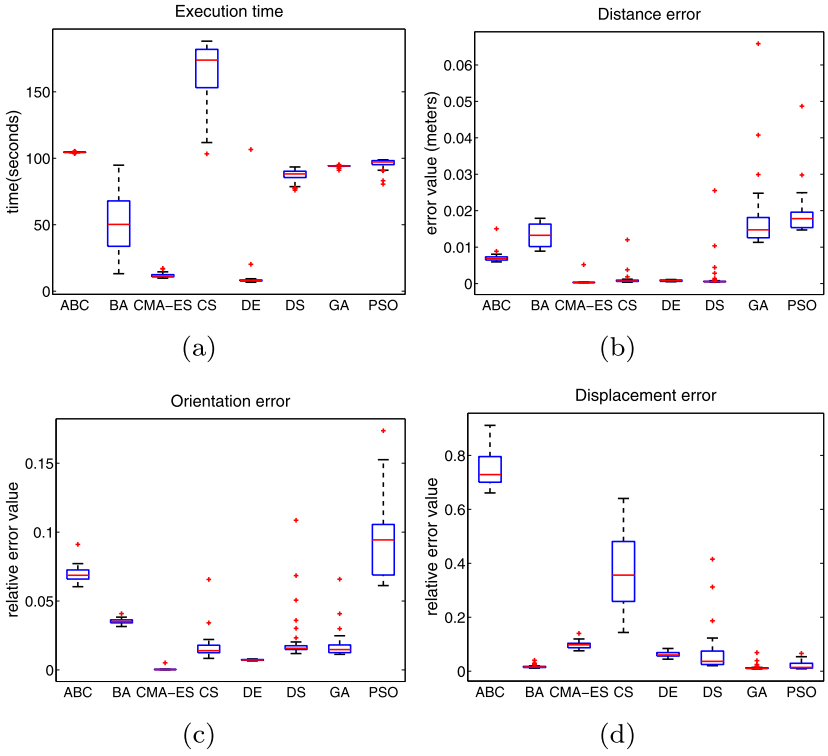


Fig. 11. Circle tracking results for the Baxter arm manipulator.

$\mathbf{v} = \mathbf{n} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is the same for all poses. For each end effector pose, all the soft computing algorithms compute the inverse kinematic 500 times and their RMS were computed. The RMS error is shown in Table 5 for the best fitness values and Table 6 shows the results of the execution time. The tolerance allowed is fixed to $tol = 0.01$.

5.4.1. The sign test

From the simulations results we conclude that the CMA-ES performed better in general than the others algorithms. For this reason, the CMA-ES algorithm is compared using the sign test against the others. To perform the sign test, we count the number of wins achieved either

Table 5
RMS values for the best fitness values obtained from the solution of the inverse kinematic for the proposed end effector poses.

$\mathbf{p}(k)$	ABC	BA	CMA-ES	CS	DE	DS	GA	PSO
1	0.0236	0.0827	0.0079	0.0089	0.0209	0.0306	0.1167	0.1453
2	0.0247	0.0776	0.0079	0.0090	0.0139	0.0359	0.1240	0.1452
3	0.0257	0.0645	0.0078	0.0089	0.0164	0.0336	0.1203	0.1295
4	0.0297	0.0580	0.0077	0.0090	0.0151	0.0355	0.1169	0.1373
5	0.0316	0.0550	0.0078	0.0088	0.0141	0.0366	0.1140	0.1341
6	0.0336	0.0487	0.0076	0.0090	0.0190	0.0389	0.1228	0.1355
7	0.0347	0.0528	0.0078	0.0088	0.0127	0.0439	0.1224	0.1386
8	0.0392	0.0545	0.0077	0.0087	0.0129	0.0427	0.1310	0.1461
9	0.0418	0.0607	0.0083	0.0088	0.0145	0.0458	0.1361	0.1381
10	0.0440	0.0620	0.3021	0.0095	0.0178	0.0486	0.1425	0.1490
11	0.0471	0.0696	0.2924	0.0099	0.0198	0.0489	0.1500	0.1599
12	0.0503	0.0855	0.3112	0.0100	0.0226	0.0490	0.1423	0.1648
13	0.0495	0.0936	0.2823	0.0102	0.0256	0.0469	0.1499	0.1780
14	0.0492	0.0878	0.2834	0.0100	0.0205	0.0474	0.1519	0.1637
15	0.0477	0.0745	0.2876	0.0097	0.0156	0.0459	0.1509	0.1518
16	0.0458	0.0675	0.2991	0.0093	0.0169	0.0529	0.1437	0.1420
17	0.0406	0.0561	0.0083	0.0090	0.0181	0.0457	0.1407	0.1351
18	0.0367	0.0528	0.0078	0.0087	0.0186	0.0442	0.1418	0.1417
19	0.0357	0.0487	0.0079	0.0088	0.0177	0.0430	0.1203	0.1312
20	0.0330	0.0528	0.0077	0.0090	0.0143	0.0421	0.1269	0.1251
21	0.0309	0.0543	0.0078	0.0089	0.0141	0.0363	0.1195	0.1379
22	0.0268	0.0585	0.0077	0.0092	0.0167	0.0370	0.1168	0.1446
23	0.0270	0.0638	0.0078	0.0089	0.0163	0.0362	0.1128	0.1374
24	0.0246	0.0734	0.0078	0.0089	0.0140	0.0341	0.1196	0.1410
25	0.0232	0.0811	0.0078	0.0088	0.0151	0.0312	0.1296	0.1396

Table 6
Average values for execution time obtained from the solution of the inverse kinematic for the proposed end effector poses.

$\mathbf{p}(k)$	ABC	BA	CMA-ES	CS	DE	DS	GA	PSO
1	2.3123	1.0777	0.1545	3.0231	0.4290	1.9038	2.0613	1.8700
2	2.3085	1.1152	0.1593	2.9935	0.3701	1.9253	2.0530	1.8462
3	2.3168	1.0644	0.1574	2.9580	0.4194	1.9366	2.0637	1.8400
4	2.3085	1.0590	0.1567	3.0839	0.4193	1.9487	2.0637	1.8913
5	2.3157	1.1070	0.1583	2.9878	0.4176	1.9683	2.0788	1.8742
6	2.3321	1.0351	0.1575	2.9586	0.5290	1.9690	2.0736	1.8789
7	2.3232	1.1217	0.1582	2.9809	0.4041	1.9649	2.0725	1.8459
8	2.2978	1.1126	0.1578	2.9658	0.4077	1.9556	2.0750	1.8326
9	2.3146	1.1358	0.1689	3.0176	0.4219	1.9641	2.0786	1.9258
10	2.3331	1.3485	1.7089	3.2618	1.0227	1.9851	2.0748	1.9209
11	2.3467	1.3211	1.7151	3.3035	0.9574	1.9953	2.0841	1.8860
12	2.3453	1.3383	1.7327	3.3326	0.9197	2.0007	2.0815	1.9243
13	2.3423	1.3177	1.6744	3.3455	1.0186	1.9900	2.0850	1.8948
14	2.3394	1.3826	1.6880	3.3031	0.9277	1.9854	2.0882	1.8896
15	2.3552	1.3502	1.6725	3.2601	0.9386	1.9829	2.0822	1.9207
16	2.3451	1.3544	1.7245	3.2121	0.9441	1.9936	2.0857	1.8855
17	2.3006	1.0538	0.1739	3.0652	0.4859	1.9811	2.0804	1.8928
18	2.3219	1.0926	0.1632	2.9780	0.5116	1.9470	2.0754	1.8224
19	2.3329	1.0414	0.1615	2.9411	0.5070	1.9748	2.0696	1.8597
20	2.3310	1.1164	0.1621	2.9861	0.4263	2.0072	2.0704	1.8578
21	2.3063	1.0942	0.1614	3.0099	0.4131	1.9714	2.0703	1.8575
22	2.2956	1.0666	0.1590	3.0566	0.4391	2.0014	2.0605	1.8499
23	2.2889	1.0413	0.1583	3.0108	0.4192	1.9644	2.0670	1.8602
24	2.2671	1.0662	0.1570	3.0262	0.3748	1.9380	2.0725	1.9043
25	2.2696	1.0604	0.1520	2.9821	0.3714	1.9306	2.0495	1.8678

Table 7
Sign test results for the best fitness values. CMA-ES shows a significant improvement over all soft computing algorithms with a level of significance $\alpha = 0.05$.

CMA-ES	ABC	BA	CS	DE	DS	GA	PSO
Wins(+)	18	18	18	18	18	18	18
Loses(−)	7	7	7	7	7	7	7
p-value	0.0500	0.0500	0.0500	0.0500	0.0500	0.0500	0.0500

by CMA-ES or by the compared algorithm. In Derrac et al. (2011), the authors reported that in the case of 25 different functions, an algorithm is significantly better than other if it performs better on at least 18 times for a level of significance $\alpha = 0.05$, and at least 17 for a level of $\alpha = 0.1$.

Table 7 shows the results of the sign test for the best fitness values obtained from Table 5. Here, CMA-ES performed over all the others algorithms with a level of significance $\alpha = 0.05$. In Table 8 we show the results of the execution time obtained from Table 6. Here, CMA-ES outperformed ABC, CS, DS, GA and PSO, and it shows a significant improvement over BA and DE with a level of significance $\alpha = 0.05$.

Table 8
Sign test results for execution time. CMA-ES shows a significant improvement over all soft computing algorithms with a level of significance $\alpha \leq 0.05$.

CMA-ES	ABC	BA	CS	DE	DS	GA	PSO
Wins(+)	25	18	25	18	25	25	25
Loses(−)	0	7	0	7	0	0	0
p-value	–	0.0500	–	0.0500	–	–	–

The CMA-ES algorithm then, is considered better than the others soft computing algorithms according to the sign test.

5.4.2. The Wilcoxon signed ranks test

The CMA-ES algorithm is compared using the Wilcoxon signed ranks test against the others algorithms. To perform the Wilcoxon test, we first calculate the values R^+ and R^- that correspond to the comparison between CMA-ES and the others algorithms. Then, their associated p-values can be estimated from (Zar, 2010)(Table B.12) or any other table with critical values of the Wilcoxon distribution.

The results of the Wilcoxon test for the best fitness values are shown in Table 9. Here, CMA-ES performed better than ABC, BA and GA with a level of significance $\alpha < 0.025$, better than CS with $\alpha < 0.05$ and better than DE, DS and PSO with $\alpha < 0.025$.

From Table 10, we notice the results of the Wilcoxon test for execution time. Here, CMA-ES outperformed ABC, CS, DS, GA and PSO and it performed better than DE with a level of significance $\alpha < 0.025$. For this case, CMA-ES is not consider better than BA with a p-value > 0.2.

According to the results for the Wilcoxon test obtained so far, we may consider that CMA-ES performed better than the others algorithms in general.

As we can observe in the simulations, the algorithm CMA-ES performed better in general than the ABC, BA, CS, DE, DS, GA and PSO. Taking into account these results we choose the algorithm CMA-ES for the real experiments.

5.5. Comparative study of inverse kinematic approaches

In this experiment we performed two tests. The first test solves a circle path tracking problem using the four robot arm manipulators (RRR arm, Youbot arm, Puma arm, Baxter arm) using the proposed approach and the traditional approach. In the second test, we solved the inverse kinematics of a given set of end effectors poses. These end effector poses may have a singular configuration with the traditional inverse kinematic method. We used the CMA-ES algorithm to compare the proposed approach against the traditional approach. We notice that joint limits are not considered in the robotic toolbox (Corke, 2011). For that reason, we did not use joint limits in the proposed method as well. In the case of the Anthropomorphic RRR robot manipulator, we did not solved the orientation problem due to its kinematics limitations.

5.5.1. Circle path tracking test

In this test, we solved a circle path tracking problem for each robot arm manipulator. For the Anthropomorphic RRR arm manipulator, we define a circle with center point $\mathbf{p}_{center} = \begin{bmatrix}1.1950 & 0.0 & 0.990\end{bmatrix}^T$ and radius $r = 0.05$ m. In the case of the Youbot arm manipulator, we define a circle with center point $\mathbf{p}_{center} = \begin{bmatrix}0.4 & 0.0 & 0.3\end{bmatrix}^T$ and radius $r = 0.05$ m. In this manipulator, the desired orientation is fixed to $\mathbf{v} = \mathbf{n} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. For the Puma 560 arm manipulator, we define a circle with center point $\mathbf{p}_{center} = \begin{bmatrix}0.5068 & -0.1500 & 0.4049\end{bmatrix}^T$ and radius $r = 0.05$ m. In the case of the Baxter arm manipulator, we define a circle with center point $\mathbf{p}_{center} = \begin{bmatrix}0.8040 & 0.0 & 0.6072\end{bmatrix}^T$ and a radius $r = 0.05$ m. The desired orientation for the Puma 560 and Baxter manipulators are fixed to

$$R = \begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

For both approaches, we run 500 times each path tracking test. The tolerance allowed for the proposed method was set to $tol = 0.0001$.

Table 11 presents the results for the Anthropomorphic RRR arm manipulator. The traditional approach (robotic toolbox method) performed better than the proposed approach in all measurement presented about their mean and standard deviation. The position error is measured using the Euclidean distance between the estimated position $\hat{\mathbf{p}}$ and the desired position $\mathbf{p}$. The percent relative error is computed between the point k and the point $k+1$ in the path.

In the experiments with the robots Yubot, Puma and Baxter the position error is measured using the Euclidean distance between the estimated position $\hat{\mathbf{p}}$ and the desired position $\mathbf{p}$. The orientation error is a percent relative error between the estimated rotation matrix $\hat{R}$ and the desired matrix R. The percent relative error is computed between the point k and the point $k+1$ in the path.

The data reported in Table 12 shows the results for the Youbot arm manipulator. The proposed approach shows an improvement over the traditional approach in the position error, orientation error and displacement error. The huge results errors in orientation and displacement measure indicates that the traditional approach did not solve the orientation problem and it presents a large joint displacement during the circle tracking.

Table 10
Wilcoxon signed ranks test results for execution time. CMA-ES shows a significant improvement over DE with a level of significance $\alpha < 0.025$, and it outperformed ABC, CS, DS, GA and PSO for all cases.

CMA-ES	ABC	BA	CS	DE	DS	GA	PSO
R^+	325	207	325	239	325	325	325
R^-	0	118	0	86	0	0	0
p-value	–	<0.2500	–	<0.0250	–	–	–

Table 11
Comparison results for the Anthropomorphic RRR arm (3 DOF)

Measurement	*Mean*		*Standard deviation*	
	Proposed approach	Traditional approach	Proposed approach	Traditional approach
Time (s)	3.8659	**0.19996**	0.57026	**0.003795**
Position error (mm)	1.1347	**8.571e−05**	4.0004	**9.3089e−06**
Displacement error (%)	2.6555	**1.6448**	2.3207	**0.54631**

Table 12
Comparison results for the Youbot arm (5 DOF)

Measurement	*Mean*		*Standard deviation*	
	Proposed approach	Traditional approach	Proposed approach	Traditional approach
Time (s)	7.969	**1.717**	0.81404	**0.039059**
Position error (mm)	**0.50832**	6.7713	**2.1779**	41.1881
Orientation error (%)	**0.32288**	30.648	**1.0055**	11.7732
Displacement error (%)	**3.9981**	70.0212	**1.5325**	387.5064

Table 9
Wilcoxon signed ranks test results for the best fitness values. CMA-ES shows a significant improvement over ABC, BA and GA with a level of significance $\alpha < 0.025$, over CS with $\alpha < 0.05$ and over DE, DS and PSO with $\alpha < 0.1$.

CMA-ES	ABC	BA	CS	DE	DS	GA	PSO
R^+	242	238	228	219	225	248	224
R^-	83	87	97	106	100	77	101
p-value	<0.0250	<0.0250	<0.0500	<0.1000	<0.1000	<0.0250	<0.1000

Table 13
Comparison results for the Puma 560 arm (6 DOF)

Measurement	*Mean*		*Standard deviation*	
	Proposed approach	Traditional approach	Proposed approach	Traditional approach
Time (s)	**11.224**	56.6028	1.8542	**0.51097**
Position error (mm)	1.0373	**3.6415e−11**	2.4604	**2.1572e−10**
Orientation error (%)	**0.33211**	150.5966	**0.4802**	36.5004
Displacement error (%)	**3.0546**	3387.0487	**3.2438**	9152.0016

Table 14
Comparison results for the Baxter arm (7 DOF)

Measurement	*Mean*		*Standard deviation*	
	Proposed approach	Traditional approach	Proposed approach	Traditional approach
Time (s)	**18.3866**	62.392	5.2475	**0.061361**
Position error (mm)	**1.1618**	897.2703	**3.1322**	362.135
Orientation error (%)	**0.28609**	165.9518	**0.40164**	40.9133
Displacement error (%)	**11.8585**	735.114	**1.6714**	1458.1312

In Table 13 we present the results for the Puma 560 arm manipulator. The proposed approach shows significant results over the traditional approach in the time, orientation and displacement measurements. The traditional approach takes more time to solve the tracking problem than the proposed approach. The traditional approach has larger errors in the orientation and the displacement. Furthermore, the errors can be very large due to singularities, or when the traditional approach did not find a solution.

The comparison results for the Baxter arm manipulator are shown in Table 14. The proposed approach presents a significant improvement over the traditional approach in all measurement respect to their mean. The huge results for the traditional approach indicates that the path tracking problem was not solve.

Figs. 12–15 illustrate the joint displacement results of the path tracking problem for the Anthropomorphic RRR arm, Youbot arm, Puma arm and the Baxter arm manipulators, respectively. In Fig. 12, we can observe that the proposed and the traditional approaches have a similar behavior. In Fig. 13.b, 14.b and 15.b we can notice that the traditional approach presents abrupt values for some joint positions. In the case of the joint displacement for the Baxter manipulator, the traditional approach did not solve the path tracking problem. Fig. 12.a, 13.a, and 14.a illustrate the effectiveness of the proposed approach.

According to the results for this test, we consider that the proposed approach performed better than the traditional approach when we use a robotic manipulator of 5, 6 and 7 DOF.

5.5.2. Singularity configuration test

The aim of this simulation is to test the proposed approach and the traditional approach against the possibility of singularities configurations. These singularities may occur as the result of the inverse kinematics problem. In Craig (2005), the authors give two examples of singularities than may occur for the Puma 560 arm manipulator. A singularity may occur when θ_3 is near $---90.0$ degrees. In this case, the links 2 and 3 are stretched out. The other singularity occurs whenever $\theta_5 = 0.0$ degrees. In this configuration the joint axes 4 and 6 line up. Given these two singularities configuration, we choose the next four end effector poses for the Puma 560 arm manipulator

$$R_2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{p}_2 = \begin{bmatrix} 0.50678 \\ -0.15005 \\ 0.40494 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} -0.28017 & 0.19617 & 0.93969 \\ 0.57358 & 0.81915 & 0 \\ -0.76975 & 0.53899 & -0.34202 \end{bmatrix} \mathbf{p}_2 = \begin{bmatrix} 0.32383 \\ -0.15005 \\ 0.25848 \end{bmatrix}$$

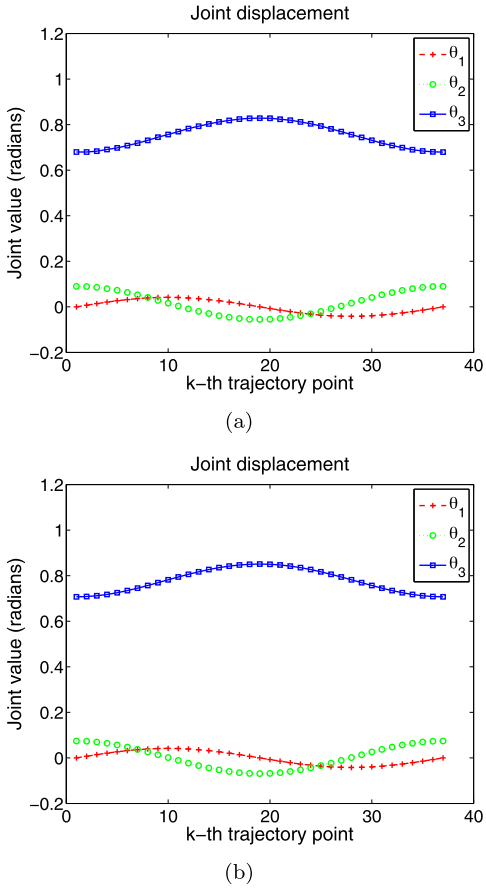


Fig. 12. Joint displacement results for the Anthropomorphic RRR arm manipulator. Figure (a) shows the results of the proposed approach, Figure (b) shows the results of the traditional approach.

Table 15
Results for the R_1 and $\mathbf{p}_1$ poses.

Measurement	Proposed approach	Traditional approach
Time (s)	**0.29545**	1.5899
Position error (mm)	0.0037501	**1.0007e−13**
Orientation error (%)	**0.078643**	184.0284

$$R_2 = \begin{bmatrix} 0 & -0.80902 & 0.58779 \\ 1 & 0 & 0 \\ 0 & 0.58779 & 0.80902 \end{bmatrix} \mathbf{p}_2 = \begin{bmatrix} 0.4372 \\ -0.15005 \\ 0.47923 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 0.57358 & 0 & 0.81915 \\ 0 & 1 & 0 \\ -0.81915 & 0 & 0.57358 \end{bmatrix} \mathbf{p}_2 = \begin{bmatrix} 0.64467 \\ -0.15005 \\ 0.42894 \end{bmatrix}$$

As a result of the inverse kinematic of these points, a singularity configuration for the Puma manipulator can occur. This test consist in solving the inverse kinematic problem for the four given end effector poses shown previously, using the proposed approach and the traditional approach. The position error is measured using the Euclidean distance between the estimated position $\hat{\mathbf{p}}_1$ and the desired position $\mathbf{p}_1$. The orientation error is a percent relative error between the estimated rotation matrix $\hat{R}_1$ and the desired matrix R_1.

Tables 15–18 present the results for the four end effector poses, respectively. As we can see, the proposed approach shows an improvement over the traditional approach under the time and orientation measurements. The traditional approach shows huge results in the

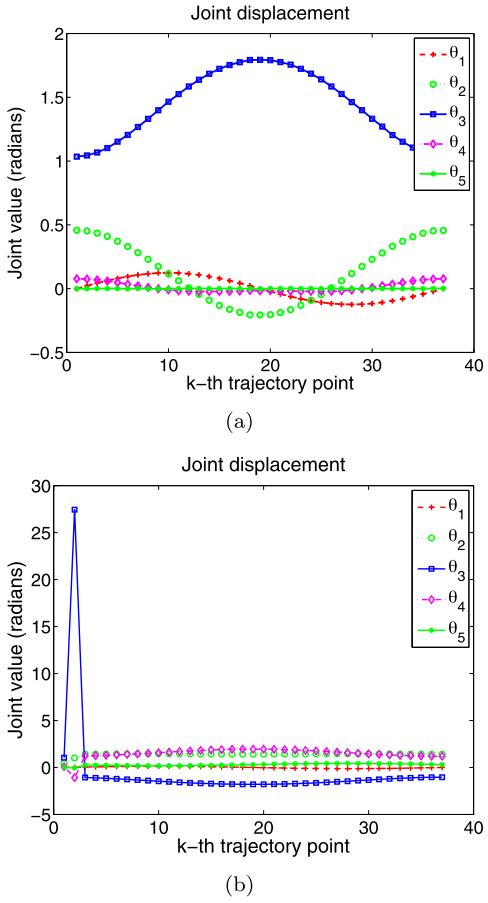


Fig. 13. Joint displacement results for the Youbot arm manipulator. Figure (a) shows the results of the proposed approach, Figure (b) shows the results of the traditional approach.

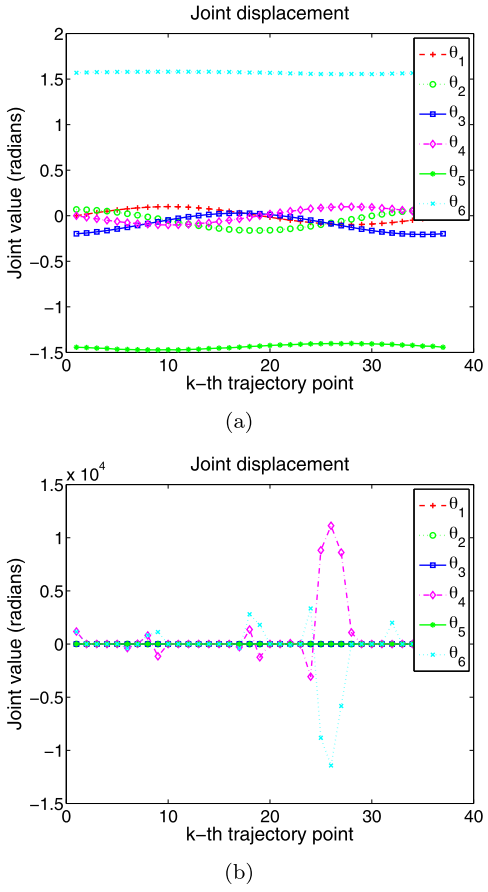


Fig. 14. Joint displacement results for the Puma 560 arm manipulator. Figure (a) shows the results of the proposed approach, Figure (b) shows the results of the traditional approach.

Table 16
Results for the R_2 and $\mathbf{p}_2$ poses.

Measurement	Proposed approach	Traditional approach
Time (s)	**0.47063**	1.513
Position error (mm)	0.013021	**9.6148e−14**
Orientation error (%)	**0.12865**	19.7277

Table 17
Results for the R_3 and $\mathbf{p}_3$ poses.

Measurement	Proposed approach	Traditional approach
Time (s)	**0.28732**	1.5156
Position error (mm)	0.036998	**2.2888e−13**
Orientation error (%)	**0.038278**	172.1685

Table 18
Results for the R_4 and $\mathbf{p}_4$ poses.

Measurement	Proposed approach	Traditional approach
Time (s)	**0.29336**	1.5174
Position error (mm)	0.029496	**1.0284e−11**
Orientation error (%)	**0.1385**	131.1084

orientation errors for the four end effector poses. This mean that the traditional approach did not solve the orientation problem.

The results for this test indicate that the proposed method is able to avoid singularities configurations because it is based on the forward kinematic equations. The proposed approach solve the position an orientation problem for the end effector pose. In contrast, the traditional approach finds a joint solution to avoid the singular configuration solving the position error, but this joint configuration does not solve the orientation problem.

6. Experimental results

Given the results of the simulations experiments, we chose the CMA-ES algorithm for the real experiments. The experiments were conducted as follows: In the first part, the proposed approach is used to solve the inverse kinematics problem. Then, in the second part the proposed approach is used to solve a path tracking problem.

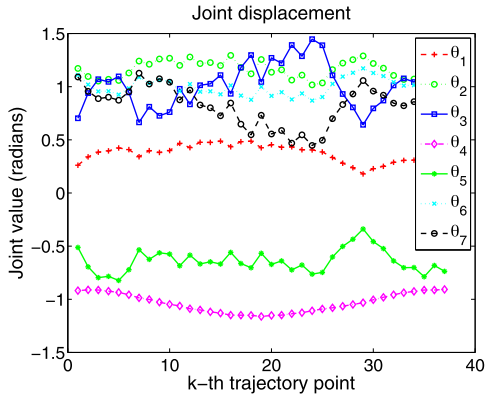
We use a 5 DOF KUKA Youbot manipulator, see Fig. 16. The DH parameters of the Youbot are defined in Table 2. We add an offset $\mathbf{q}_{offset}$ in order to use the robot arm controller integrated in the Youbot arm. The $\mathbf{q}_{offset}$ is defined as

$$\mathbf{q}_{offset} = \begin{bmatrix} 169 & (65+90) & -146 & (102.5+90) & 167.5 \end{bmatrix}^T$$

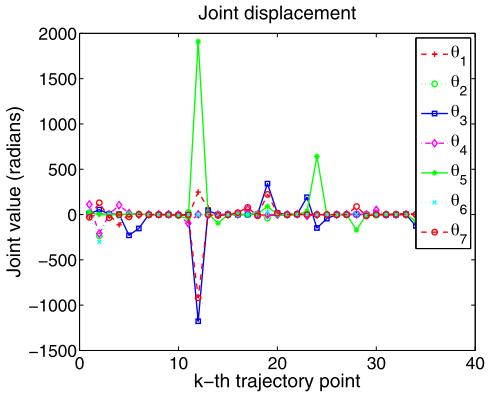
and we compute this offset with the $\hat{\mathbf{q}}_{best}$ obtained by the proposed approach. This operation is show below

$$\mathbf{q}_{ctrl} = -\hat{\mathbf{q}}_{best} + \mathbf{q}_{offset} \tag{19}$$

where $\mathbf{q}_{ctrl}$ is the joint configuration used int the Youbot arm controller. The proposed algorithm was implemented using C++ and ROS environment. There is an existing ROS component for the KUKA Youbot. We use this component to access to the Youbot arm hardware.



(a)



(b)

Fig. 15. Joint displacement results for the Baxter arm manipulator. Figure (a) shows the results of the proposed approach, Figure (b) shows the results of the traditional approach.

Fig. 16. KUKA Youbot omni-directional mobile platform with a 5 DOF manipulator.

6.1. Inverse kinematic problem

In this experiment, the desired position for the end effector was fixed to $\mathbf{p} = \begin{bmatrix}0.4 & 0.1 & 0.35\end{bmatrix}^T$, the desired orientation $\mathbf{v} = \mathbf{n} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$ and the value of the allowed tolerance is $tol = 0.0001$. The position error is measured using the Euclidean distance between the $\mathbf{p}_{youbot}$ achieved by the Youbot arm and the $\mathbf{p}$ desired. The orientation error is a percent relative error between the $\mathbf{n}_{youbot}$ obtained by the Youbot arm and the $\mathbf{n}$ desired.

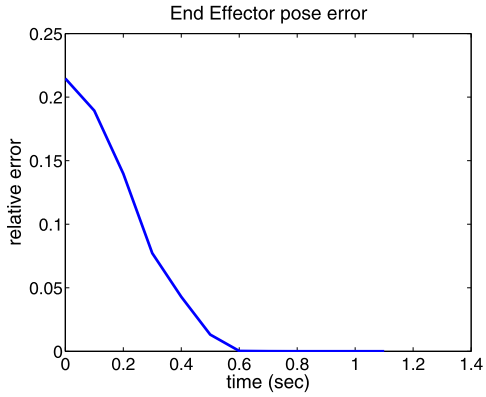


Fig. 17. End effector pose error. The relative error represents the error between the $\mathbf{q}_{ctrl}$ obtained by the proposed approach and the $\mathbf{q}_{youbot}$ achieved by the KUKA Youbot.

Table 19
Inverse kinematic results for the Youbot robot.

Measurement	Value
Best fitness	8.1531e−05
Best end effector pose error (%)	0.071752
Position error (mm)	0.1441
Orientation relative error (%)	0.1848
Time (s)	0.00304

Table 20
Path tracking results for the Youbot robot.

Measurement	Mean	Std
Position error (mm)	0.0716	0.0436
Orientation relative error (%)	0.0899	0.0449

Fig. 17 shows the convergence error between the $\mathbf{q}_{ctrl}$ obtained by the proposed approach and the $\mathbf{q}_{youbot}$ by the Youbot arm manipulator. This last means that the robot arm reached the estimated arm configuration successfully.

In Table 19, the results show a significant improvement over the results achieved from simulations. The position error performed below to 0.2 mm and the orientation error below to 1%. The end effector pose error is below to 0.01%, so we consider that the estimated configuration of the robot arm is the same as the configuration achieved by the Youbot arm manipulator. The time performed below to 0.00304 s, which is good enough to be applied for a real-time path tracking problem.

6.2. Path tracking problem

In this experiment the robot solves a path tracking problem. The path used for this test was a circle with a center point $\mathbf{p}_{center} = \begin{bmatrix}0.4 & 0.0 & 0.3\end{bmatrix}^T$ and radius $r = 0.05$ m. The desired orientation for each point are fixed to $\mathbf{v} = \mathbf{n} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$ and the tolerance allowed to $tol = 0.0001$. The position error is measured using the Euclidean distance between the $\mathbf{p}_{youbot}$ achieved by the Youbot arm and the $\mathbf{p}$ desired. The orientation error is a percent relative error between the $\mathbf{n}_{youbot}$ obtained by the Youbot arm and the $\mathbf{n}$ desired.

The graph in Fig. 18 represents the movement error of each joint of the robot arm for each point in the path. As we can see, the relative error is below to 1% which means that the algorithm has a minimal movement in the joints configuration. The graph illustrated in Fig. 19 shows the joint displacement during the real path tracking problem for each joint of the robot arm.

The results reported in Table 20 show the effectiveness of the proposed approach for solving a real time path tracking problem. Here, the mean value of the position error is performed below to 0.1 mm and

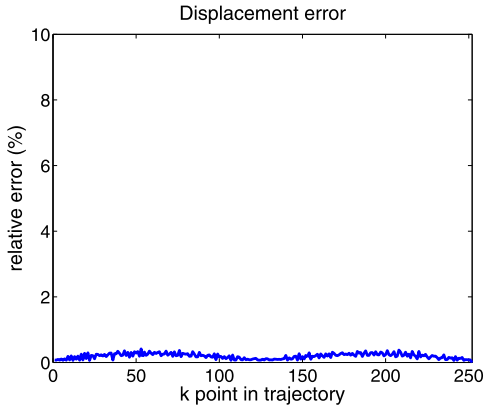


Fig. 18. Displacement error. The percent relative error is computed between the point k and the point $k+1$ in the path.

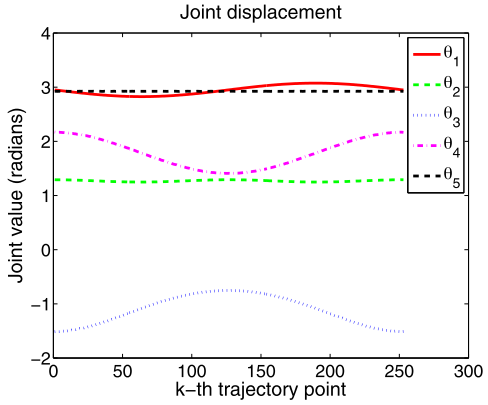


Fig. 19. Joint displacement of the KUKA Youbot arm manipulator.

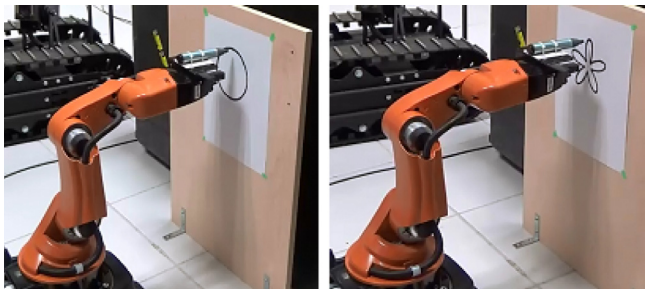


Fig. 20. KUKA Youbot arm solving different path tracking problems.

the orientation error below to 0.1%. The standard deviation indicates that the results for both measurements have no a significant data distribution.

According to the experimental results reported so far, we consider that the soft computing algorithm CMA-ES is able to solve the inverse kinematic and path tracking problems successfully. The proposed approach can be applied for solving different path tracking problem as we illustrate in Fig. 20.

7. Conclusions

In this paper, we have introduced an approach for solving the inverse kinematics of manipulator robots based on soft computing algorithms. From the simulations and experimental results, we found that the proposed method is able to solve the problem of orientation and position of a desired end effector pose. Moreover, according to the comparative study results, the CMA-ES algorithm obtained the best performance in general. According to its data distribution, CMA-ES has good results in position error, orientation error and execution time. In addition, the performance of CS is as good as the CMA-ES, but CS requires much more execution time to solve the proposed problems. On the other hand, as reported by the proposed nonparametric statistical tests, CMA-ES shows a significant improvement against the others soft computing algorithms. Therefore, the proposed approach is an interesting method that open new ways for solving the inverse kinematic and path tracking problems for redundant robot manipulators.

Acknowledgments

The authors thank the support of CONACYT México, through Projects CB256769 and CB258068 (Project supported by *Fondo de Investigación Sectorial para la Educación* FOINS 241246).

References

Civicioglu, P., 2012. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput. Geosci. 46, 229–247. http://dx.doi.org/10.1016/j.cageo.2011.12.011.

Civicioglu, P., Besdok, E., 2011. A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif. Intell. Rev. 39 (4), 315–346. http://dx.doi.org/10.1007/s10462-011-9276-0.

Corke, P.I., 2011. Robotics, Vision & Control: Fundamental Algorithms in Matlab. Springer.

Derrac, J., Garca, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1 (1), 3–18. http://dx.doi.org/10.1016/j.swevo.2011.02.002.

Du, Y., Wu, Y., 2011. Application of IPSO algorithm to inverse kinematics solution of reconfigurable modular robots. In: Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on. pp. 1313–1316. http://dx.doi.org/10.1109/MEC.2011.6025711.

Falco, P., Natale, C., 2014. Low-level flexible planning for mobile manipulators: a distributed perception approach. Adv. Robot. 28 (21), 1431–1444. http://dx.doi.org/10.1080/01691864.2014.946446.

Gämperle, R., Müller, S.D., Koumoutsakos, P., 2002. A parameter study for differential evolution. Adv. Intell. Syst. Fuzzy Syst. Evol. Comput. 10, 293–298.

Glumac, S., Kovacic, Z., 2013. Microimmune algorithm for solving inverse kinematics of redundant robots. In: Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on. pp. 201–207. http://dx.doi.org/10.1109/SMC.2013.41.

Hansen, N., 2006. In: Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (Eds.), Towards a new evolutionary computation: advances in the estimation of distribution algorithms. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 75–102. http://dx.doi.org/10.1007/3-540-32494-1_4.

Huang, H.C., Chen, C.P., Wang, P.R., 2012. Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators. In: Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on. pp. 3105–3110. http://dx.doi.org/10.1109/ICSMC.2012.6378268.

Craig, J.J., 2005. Introduction to Robotics Mechanics and Control, third ed. Pearson Education, Inc.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. 39 (3), 459–471. http://dx.doi.org/10.1007/s10898-007-9149-x.

Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. 8 (1), 687–697. http://dx.doi.org/10.1016/j.asoc.2007.05.007.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4. IEEE, Bur. of Labor Stat., Washington, DC, USA, pp. 1942–1948. http://dx.doi.org/10.1109/icnn.1995.488968.

Li, Z.M., Li, C.G., Lv, S.J., 2012. A method for solving inverse kinematics of PUMA560 manipulator based on PSO-RBF network. In: Natural Computation (ICNC), 2012 Eighth International Conference on. pp. 298–301. http://dx.doi.org/10.1109/ICNC.2012.6234507.

Man, K.F., Tang, K.S., Kwong, S., 1996. Genetic algorithms: concepts and applications [in engineering design]. IEEE Trans. Ind. Electron. 43 (5), 519–534. http://dx.doi.org/10.1109/41.538609.

Nearchou, A.C., 1998. Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. Mech. Mach. Theory 33 (3), 273–292. http://dx.doi.org/10.1016/S0094-114X(97)00034-7.

Pham, D.T., Castellani, M., Fahmy, A.A., 2008. Learning the inverse kinematics of a robot manipulator using the Bees Algorithm. In: Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on. pp. 493–498. http://dx.doi.org/10.1109/INDIN.2008.4618151.

Rafael Kelly Ph.D., A.L.P.a., Davila Ph.D., Victor Santibáñez, 2005. Control of robot manipulators in joint space. Advanced Textbooks in Control and Signal Processing, first ed. Springer-Verlag London.

Sciavicco, L., Siciliano, B., 1996. Modeling And Control of Robot Manipulators, vol. 8. McGraw-Hill New York.

Selvaraj, C., Siva Kumar, R., Karnan, M., 2014. A survey on application of bio-inspired algorithms. Internat. J. Comput. Sci. Inform. Technol. 5 (1), 366–370.

Shi, Y., Eberhart, R.C., 1998. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (Eds.), Evolutionary Programming VII: 7th International Conference, EP98 San Diego, California, USA, March 25–27, 1998 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 591–600. http://dx.doi.org/10.1007/BFb0040810.

Spong, M.W., Vidyasagar, M., 2008. Robot Dynamics and Control. John Wiley & Sons.

Storn, R., Price, K., 1997. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359. http://dx.doi.org/10.1023/A:1008202821328.

Tabandeh, S., Clark, C., Melek, W., 2006. A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. pp. 1815–1822. http://dx.doi.org/10.1109/CEC.2006.1688527.

Yang, X.-S., 2010. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (Eds.), Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 65–74. http://dx.doi.org/10.1007/978-3-642-12538-6_6.

Yang, X.S., Deb, S., 2009. Cuckoo search via Lévy flights. In: Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. pp. 210–214. http://dx.doi.org/10.1109/NABIC.2009.5393690.

Yang, Y., Peng, G., Wang, Y., Zhang, H., 2007. A new solution for inverse kinematics of 7-dof manipulator based on genetic algorithm. In: Automation and Logistics, 2007 IEEE International Conference on. pp. 1947–1951. http://dx.doi.org/10.1109/ICAL.2007.4338892.

Zar, J.H., 2010. Biostatistical Analysis, fifth ed. Prentice Hall.