International Journal of Pioneering Technology and Engineering

Research Article

Fast Convergent Ali Baba and Forty Thieves Algorithm for Inverse Kinematic Solution of 7-DOF Robotic Manipulator

Bilal Özak [a*]
[a] Graduate School of Natural and Applied Sciences, Erciyes University, Kayseri, Türkiye

Article Info

Article history
Received: 31/10/2023
Revised: 18/11/2023
Accepted: 05/12/2023

Keywords:
Inverse Kinematics,
Metaheuristics,
Robotics,
Optimization

ABSTRACT

Inverse kinematics is a crucial topic in robotics, enabling robots to calculate the joint angles required to achieve the desired end effector position and orientation. Solving the inverse kinematics problem quickly with high accuracy is vital for robot manipulators. If sufficient speed is provided, the real-time motion planning task of robot manipulators can be achieved. Real-time motion planning of complex robot manipulators is not possible with classical mathematical methods. Overcoming this problem will provide many benefits in the design and control of robot manipulators. In contemporary research, metaheuristic approaches have become widely employed for addressing the inverse kinematics problem. This investigation utilizes the efficient and simple Ali Baba and the Forty Thieves (AFT) algorithm to resolve the inverse kinematics problem. To increase convergence speed of AFT, a parameter has been used in early iterations of the algorithm to prevent thieves from randomly searching within the search area to find Ali Baba when they realized they had been deceived. Additionally, an approach has been proposed regarding the accuracy of the information brought by the Marjaneh. Finally, the inverse kinematics solution of the 7-DOF robot manipulator was carried out comparatively.

1. Introduction

In the rapidly evolving landscape of optimization problems, the quest for efficient solutions has fueled the development of innovative techniques. Among these, metaheuristic optimization algorithms have emerged as powerful tools for tackling complex, real-world optimization challenges [1]. These algorithms draw inspiration from natural processes, collective intelligence, and problem-solving heuristics to navigate intricate search spaces and uncover optimal or near-optimal solutions.

Lately, people have been using metaheuristic algorithms a lot to solve complex real-world problems that involve both many things happening at once and things that aren't straightforward. These algorithms give decent solutions fairly quickly, but there's no guarantee they'll find the absolute best solution for a specific problem [2]. Many nature-inspired meta-heuristic methods are documented in the research literature. The Genetic Algorithm (GA) draws inspiration from Darwinian evolution. Widely applied in various optimization problems, GA is recognized as one of the most effective algorithms, employing two crucial combination and mutation operators [3].

The Differential Evolution (DE) algorithm, conceived by Storn and Price in the mid-1990s, stands as a powerful optimization technique rooted in evolutionary strategies [4]. DE operates by evolving a population of candidate solutions through the differential mutation of vectors, crossover, and selection mechanisms. Known for its simplicity, robustness, and versatility, DE has demonstrated efficacy across various optimization problems, making it a noteworthy subject of research and application in diverse domains.

The Particle Swarm Optimization (PSO) algorithm, inspired by the collective behavior of bird flocks and fish schools, is a popular meta-heuristic approach for solving complex optimization problems. Introduced by Kennedy and Eberhart in 1995, PSO relies on the interaction and collaboration of particles in a search space to iteratively converge toward optimal solutions [5]. Its simplicity, effectiveness, and adaptability have led to its widespread application in diverse domains, making it a subject of extensive research and practical implementation.

In 2005, Karaboga et al. introduced the Artificial Bee Colony (ABC) algorithm, inspired by the collective behavior of bees [6]. The ABC algorithm simulates worker bees, precursor bees, and search bees, providing mathematical formulas for each step. Similar to other meta-heuristic algorithms, it has its drawbacks, leading to subsequent releases of improved versions.

An optimization algorithm called the Sine Cosine Algorithm (SCA) is introduced in to address problems involving multiple randomized candidate solutions [7]. Furthermore, a mathematical model utilizing sine and cosine functions in sinusoidal and cosinusoidal (up and down) space is proposed, aiming to approach the ideal solution.

Robotic systems can be defined as the combination of three central systems, namely mechanical, electronic, and computational system, working together to perform the required tasks [8]. In recent years, there has been a great increase in industrial applications of robot manipulators. This increase in robots has also brought with it a great challenge. Because each robot has a different structure, different solution approaches are needed to fulfill their duties. One of the most important tasks to be fulfilled is the robot inverse kinematics solution. Nowadays,

***Corresponding author:** Bilal Özak
*E-mail address: ozakbilal@gmail.com
https://doi.org/10.56158/jpte.2023.58.2.02

robot inverse kinematics solution and trajectory tracking attract the attention of researchers [9]. The inverse kinematic solution is to find the joint angles from the position and direction information of the manipulator [10].

Inverse kinematics solution is a much more complex and time-consuming solution compared to forward kinematics due to nonlinear equations. Inverse kinematics solution of robotic manipulators with complex structures is in the NP (nondeterministic polynomial) problem group [11]. The fact that the inverse kinematics solution is very complex and time-consuming plays a restrictive role in many aspects, from robot design to control. In the field of robotics, meta-heuristic methods have revolutionized the way we approach complex optimization problems such as inverse kinematics of robot manipulators, tuning PID parameters, trajectory planning, etc. [12, 13, 14].

In this study, Ali Baba and Forty Thieves Algorithm (AFT) was used to solve the inverse kinematics problem [15]. A compelling reason for the preference of the AFT algorithm is its rapid operation, facilitated by its straightforward and uncomplicated structure. AFT, drawing inspiration from the narrative of Ali Baba and the Forty Thieves, falls under the category of a human-based algorithm. In this algorithm, the thieves play the role of search agents, the environment serves as the search space, positions within the town represent potential solutions, Ali Baba's house acts as the objective function, and Ali Baba symbolizes the global solution. The story revolves around a search-based scenario featuring a group of 40 thieves relentlessly pursuing Ali Baba with the primary goal of seeking revenge and recovering their stolen treasure by apprehending him. This pursuit is conducted iteratively, spanning multiple rounds that build upon previous iterations' solutions. The proposed algorithm utilizes a population to mirror the collective behavior of the thieves throughout their search. Marjaneh, the central character in the tale, consistently employs countermeasures to impede the gang's search mission in each iteration, thereby introducing elements of complexity and challenge into the search process.

To increase the speed of convergence, a parameter has been added to AFT during its first iterations, intended to prevent thieves from making random searches within the search space once they became aware of their deception. Furthermore, an approach has been introduced concerning the precision of the data provided by the Marjaneh. As a result of these additions, the proposed algorithm is called Modified Ali Baba and Forty Thieves Algorithm (mAFT) for ease of use.

2. Proposed Algorithm

The AFT algorithm was developed by Malik et al., inspired by the well-known fairy tale Ali Baba and the Forty Thieves. The story involves a search-based narrative where the gang of thieves iteratively pursues Ali Baba, representing the algorithm's iterative nature. The search is based on the collective behavior of the thieves, akin to the algorithm's population representation. Marjaneh, the main character, repeatedly thwarts the gang's attempts, reflecting countermeasures in the algorithm. The town where Ali Baba lives serves as the search space. The tale's smart tricks and tactics inspire the algorithm's exploration efficiency. This storytelling approach serves as a foundation for the mathematical models that underpin the AFT algorithm's design and optimization.

In this work in order to expedite the convergence rate, a parameter has been employed during the initial stages of the algorithm to deter the thieves from haphazardly exploring the search region when they became aware of being misled in their pursuit of Ali Baba. Furthermore, an alternative method has been suggested concerning the precision of the data provided by the Marjaneh. This proposed algorithm is called the Modified Ali Baba and the Forty Thieves (mAFT) algorithm for ease of understanding. The mAFT algorithm consists of three main cases described below:

Case 1 The location of the thieves while chasing Ali Baba with guidance from an informant can be replicated as depicted in the following equation:

$$x_i^{t+1} = g\text{best}^t + \left[Td^t\left(\text{best}_i^t - y_i^t\right)r_1 + Td^t\left(y_i^t - m_{a(i)}^t\right)r_2\right]\text{sgn}\,(\text{rand} - 0.5)$$
$$p \geq 0.6 - \left(\frac{t}{T}\right)0.2, q > Pp^t \tag{1}$$

where x_i^{t+1} represents the current position of the ith thief, y_i^t corresponds to Ali Baba's position relative to thief i, best_i^t signifies the optimal position attained by thief i, $g\text{best}^t$ denotes the best global position reached by any thief thus far. $m_{a(i)}^t$ represents the level of Marjaneh's cleverness used to conceal thief i, Td^t denotes the tracking distance of the thieves defined by Eq. (2), Pp^t signifies the potential perceptual ability of the thieves regarding Ali Baba as defined in Eq. (3). r_1, r_2, rand, p, and q are random values generated within the range [0, 1], where p takes values of either 0 or 1, sgn (rand $-$ 0.5) can assume values of -1 or 1, and a can be determined according to the equation displayed in Eq. (4). By using the $p \geq 0.6 - \left(\frac{t}{T}\right)0.2$ inequality regarding the precision of the data provided by Marjaneh, the accuracy of the data was reduced depending on the iteration.

$$Td^t = \tau_0 e^{-\tau_1 (t/T)_1^{\tau}} \tag{2}$$

where λ_0 ($\lambda_0 = 0.1$) represents the ultimate estimate of the likelihood that the thieves will attain their objective at the conclusion of the search process, while λ_1 ($\lambda_1 = 2.0$) serves as a constant parameter utilized to govern the balance between exploration and exploitation.

$$Pp^t = \lambda_0 log\,(\lambda_1 (t/T)^{\lambda_0}) \tag{3}$$

where λ_0 (λ_0=0.1) represents the ultimate estimate of the likelihood that the thieves will attain their objective at the conclusion of the search process, while λ_1 (λ_1=2.0) serves as a constant parameter utilized to govern the balance between exploration and exploitation.

$$a = [(n-1) \cdot rand\,(n, 1) \tag{4}$$

where rand(n, 1) is a vector of random variables created in the range [0, 1].

Marjaneh adjusts her clever strategies if the quality of the newly proposed solution by the thieves surpasses the quality of their previous solutions, as determined by the following equation:

$$m_{a(i)}^t = \begin{cases} x_i^t & if \;\; f(x_i^t) \geq f(m_{a(i)}^t) \\ m_{a(i)}^t & if \;\; f(x_i^t) < f(m_{a(i)}^t) \end{cases} \tag{5}$$

where $f(\cdot)$ represents the fitness function's score.

Case 2 Thieves might become aware of being deceived and, consequently, they will explore fresh, randomly chosen regions, as specified in the manner outlined below:

$$x_i^{t+1} = \left(\frac{t}{T}\right)Td^t\left[(u_j - l_j)r + l_j\right]; p \geq 0.5, q \leq Pp^t \tag{6}$$

where r represents a random value generated within the interval [0, 1], while u_j and l_j indicate the upper and lower boundaries of the search area along dimension j, respectively.

In this case tracking distance of the thieves Td^t used by multiplying $\left(\frac{t}{T}\right)$. As a result, early convergence was achieved by transforming the tracking distance of thieves into a structure that first increased and then decreased, as shown in Figure 1.

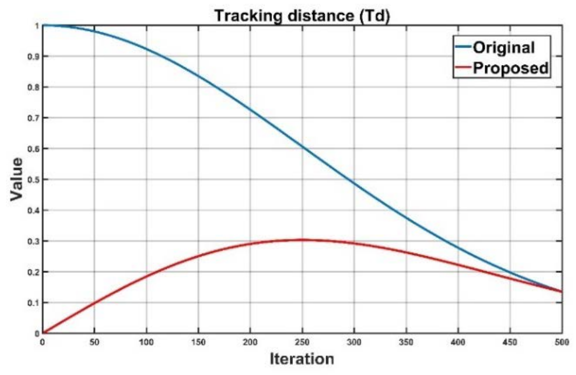

Fig. 1. Original Td^t and proposed iterative function for $Td^t\left(\frac{t}{T}\right)$

Case 3 Thieves have the potential to explore positions beyond those determined by Eq. (1), thereby enhancing both the exploration and exploitation aspects of the mAFT algorithm. This situation can be described as follows:

$$x_i^{t+1} = ggest^t - [Td^t(\text{best}_i^t - y_i^t)r_1 + Td^t(y_i^t - m_{a(i)}^t)r_2]\text{sgn}\,(\text{rand} - 0.5) \tag{7}$$

3. Kinematics Analysis of Motoman SIA20D Robot Manipulator

The Yaskawa Motoman SIA20D robot manipulator, a Japanese brand, used in the study, is shown in Figure 2. To solve the inverse kinematics problem with a metaheuristic optimization algorithm, it is necessary to obtain the forward kinematic solution. The forward kinematics were used to calculate the end-effector's final position using the joint angle set computed by the algorithm. The difference between this calculated position and the desired position, in other words, the positional error, was used as the fitness function.

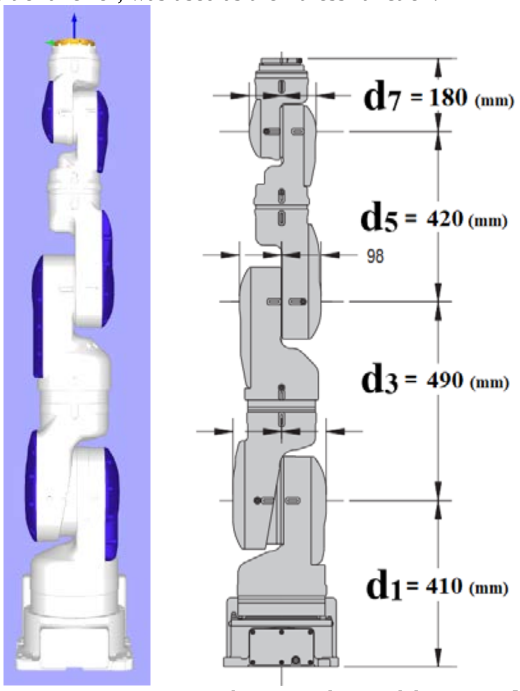

Fig. 2. Motoman SIA20D Robot manipulator and dimensions [16]

Various methods are available for solving forward kinematics, and in this study, the most commonly used method, the Denavit-Hartenberg method, has been employed. The relationship between two joints is expressed by Denavit and Hartenberg using four parameters. The DH parameters of the robot manipulator are presented in Table 1, where i, a, d, α, and θ represent the joint number, link lengths, joint offset, twist angles, and joint angles, respectively. The lengths are given in millimeters, and the angles are provided in degrees.

Table 1 DH parameters of Motoman SIA20D Robot manipulator

i	ai-1 (mm)	αi-1 (°)	di (mm)	θ_i (°) (Range)
1	0	-90	d1 = 410	-180< θ_1 <180
2	0	90	0	-110< θ_2<110
3	0	-90	d3 = 490	-170< θ_3<170
4	0	90	0	-130< θ_4 <130
5	0	-90	d5 = 420	-180< θ_5<180
6	0	90	0	-110< θ_6<110
7	0	0	d7 = 180	-180< θ_7<180

Forward kinematics enables us to find the position and orientation of the robot's end-effector using the given joint angles, by utilizing the robot's kinematic equations. Essentially, its mathematical representation is as shown in Eq. (8).

$$F_{forward\ kinematics}(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = (p_x,p_y,p_z,r_x,r_y,r_z) \tag{8}$$

The forward kinematics of the robot manipulator can be determined by solving the forward kinematic equations obtained using the homogeneous transformation matrix seen in Eq. (9).

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_{i-1} & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

The transformation matrices obtained for each joint using the DH parameters and the homogeneous transformation matrix are shown in Eq. (10).

$$ {}^0_1T = \begin{bmatrix} cos\theta_1 & 0 & -sin\theta_1 & 0 \\ sin\theta_1 & 0 & cos\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1_2T = \begin{bmatrix} cos\theta_2 & 0 & sin\theta_2 & 0 \\ sin\theta_2 & 0 & -cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$ {}^2_3T = \begin{bmatrix} cos\theta_3 & 0 & -sin\theta_3 & 0 \\ sin\theta_3 & 0 & cos\theta_3 & 0 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^3_4T = \begin{bmatrix} cos\theta_4 & 0 & sin\theta_4 & 0 \\ sin\theta_4 & 0 & -cos\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$
$$ {}^4_5T = \begin{bmatrix} cos\theta_5 & 0 & -sin\theta_5 & 0 \\ sin\theta_5 & 0 & cos\theta_5 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^5_6T = \begin{bmatrix} cos\theta_6 & 0 & sin\theta_6 & 0 \\ sin\theta_6 & 0 & -cos\theta_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$ {}^6_7T = \begin{bmatrix} cos\theta_7 & -sin\theta_7 & 0 & 0 \\ sin\theta_7 & cos\theta_7 & 0 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The general transformation matrix is obtained by multiplying these obtained matrices in a forward direction, as shown in Eq. (11).

$$ {}^0_7T = {}^0_1T\ {}^1_2T\ {}^2_3T\ {}^3_4T\ {}^4_5T\ {}^5_6T\ {}^6_7T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

The matrix ${}^0_7T$, as shown in Eq. (11), describes the position and orientation of the robot manipulator. In this equation p_x, p_y and p_z represent the position values of the robot's end-effector, while r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33} represent the rotation values of the robot's end-effector.

The position error, or fitness value, is obtained by calculating the difference between the computed position and the desired position.

$$Fitness = \sqrt{(p_{x_d} - p_{x_c})^2 + (p_{y_d} - p_{y_c})^2 + (p_{z_d} - p_{z_c})^2} \tag{12}$$

where $p_{x_c}, p_{y_c}, p_{z_c}$represent the positions calculated by the algorithms, and p_{x_d}, p_{y_d}, p_{z_d} represent the desired positions.

To explain the working principle of swarm intelligence algorithms for this problem in writing: each individual in the algorithm, equal to the population size, represents a solution set comprising, for instance, seven angles for each joint that allows a 7-degree-of-freedom robot's end-effector to reach its position. Through competition among these individuals, the set of angles that achieves the desired final position while minimizing the position error evolves. This described working principle is visually represented in the form of a flowchart in Fig. 3.

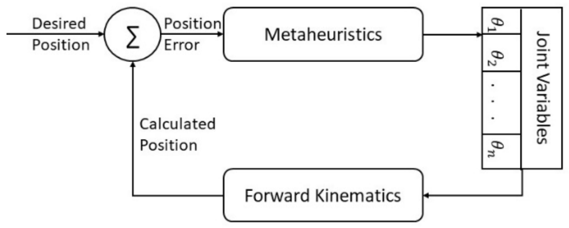

Fig. 3. Inverse kinematics solution with metaheuristic algorithms

4. Experimental Setup

In this section, experimental studies comparing the proposed mAFT with widely used optimization techniques in the literature are presented. The proposed mAFT algorithm is compared with SCA, DE, PSO, ABC, and AFT algorithms. Parameter settings of the algorithms are given in Table 2. After running 30 separate runs for each algorithm, best, mean, and standard deviation values were calculated. Firstly, the convergence capability of the proposed algorithm was analyzed using 20 well-known benchmark functions to demonstrate its effectiveness. These benchmark functions can be categorized as unimodal and multimodal.

Table 2 Parameters settings of algorithms

Algorithm	Parameters settings
Sine Cosine Algorithm (SCA)	p = 30, a = 2, [1, r2, r3, r4]
Differential Evolution (DE)	p = 30, CR = 0.5, F = 0.5
Particle Swarm Optimization (PSO)	p = 30, w = 0.6, c1 = 1.8, c2 = 1.8
Artificial Bee Colony (ABC)	p = 30, a = [2, 0]
Ali Baba and the Forty Thieves (AFT)	p = 30, $\tau_0 = 1$, $\tau_1 = 2$, $\lambda_0 = 0.1$, $\lambda_1 = 2$
Modified Ali Baba and the Forty Thieves (mAFT)	p = 30, $\tau_0 = 1$, $\tau_1 = 2$, $\lambda_0 = 0.1$, $\lambda_1 = 2$

The optimization performance of the algorithms was evaluated by examining whether they converged to a global or local solution for both unimodal and multimodal functions. To test the proposed algorithm, a comprehensive set of 20 benchmark functions, including 10 unimodal (F1-F10) and 10 multimodal (F11-F20) functions, as seen in Tables 3 and 4, was utilized. Range defines the boundary of the search space, dimension indicates the dimensionality of the search space, and f_{min} represents the global optimum in Tables 3 and 4.

Table 3 Unimodal benchmark functions

Function	Range	Dim.	f_{min}
F1− Sphere	[-100, 100]	30	0
F2− Quartic Noise	[-1.28, 1.28]	30	0
F3− Schwefel's 2.20	[-100, 100]	30	0
F4− Schwefel's 2.21	[-100, 100]	30	0
F5− Step	[-100, 100]	30	0
F6− Schwefel's 1.20	[-100, 100]	30	0
F7− Rosenbrock	[-30, 30]	30	0
F8− Brown	[-1, 4]	30	0
F9− Dixon and Price	[-10, 10]	30	0
F10− Powell Singular	[-4, 5]	30	0

All algorithms were executed for 500 iterations on a set of 20 benchmark functions, which include 10 unimodal and 10 multimodal functions. All experiments has been run using MATLAB R2019A on a computer with an Intel(R) Core (TM) i5-7500 CPU @ 3.40GHz and 8 GB of RAM.

Table 4 Multimodal benchmark function (n: dimension of the search space)

Function	Range	Dim.	f_{min}
F11− Qing	[-500,500]	30	0
F12− Alpine N. 1	[-10,10]	30	0
F13− Xin-She Yang	[-5,5]	30	0
F14− Ackley	[-32, 32]	30	0
F15− Trignometric 2	[-500, 500]	30	1
F16− Salomon	[-100, 100]	30	0
F17− Styblinski-Tang	[-5, 5]	30	-39.16599×n
F18− Griewank	[-100, 100]	30	0
F19− Xin-She Yang N.4	[-10, 10]	30	-1
F20− Xin-She Yang N.2	[-2π, 2π]	30	0

5. Experimental Results

Unimodal functions serve as a valuable means to evaluate the exploitation potential of optimization algorithms. The average fitness (mean), best fitness (best), and standard deviation (std) outcomes for both mAFT and the other algorithms are presented in Table 5. These results were obtained through 30 running of each algorithm. As can be seen from Table 5, mAFT showed a much superior performance than other algorithms.

Table 5 Results of the algorithms in unimodal benchmark test functions

		mAFT	AFT	DE	PSO	ABC	SCA
F1	best	3,79E-08	3,74E-05	4,30E-04	9,25E-07	21,6474	0,1366
	mean	**1,21E-06**	4,27E-04	1,53E-03	4,66E-03	83,7481	43,6784
	std	1,01E-06	4,42E-04	6,90E-04	0,0134	43,7348	102,2682
F2	best	6,29E-05	0,0229	0,0210	6,05E-03	0,4434	5,71E-03
	mean	**3,71E-03**	0,0634	0,0519	0,0250	1,4745	0,1440
	std	3,02E-03	0,0277	0,0158	0,0104	0,7713	0,1642
F3	best	1,99E-03	0,0908	0,0224	0,2808	3,6089	7,47E-03
	mean	**0,0278**	0,9411	0,0427	2,0754	9,2695	0,1868
	std	0,0350	0,9476	0,0157	1,8043	3,0471	0,4735
F4	best	0,1332	6,2865	6,9158	2,6692	54,5809	10,3866
	mean	**0,3072**	13,04	11,9308	4,5335	63,2262	32,8396
	std	0,1424	2,8352	2,8618	1,4421	4,2686	9,6772
F5	best	3,54E-07	3,19E-05	3,58E-04	0,0000	33,3892	4,1687
	mean	**2,84E-06**	4,43E-04	1,14E-03	0,0127	123,3942	12,3487
	std	3,96E-06	5,04E-04	5,99E-04	0,0598	82,5217	13,6954
F6	best	0,0154	62,1335	2,88E+04	42,1739	4,84E+04	835,5361
	mean	**0,2709**	283,5652	3,90E+04	225,5739	7,16E+04	9,79E+03
	std	0,3485	112,9652	5,13E+03	211,3200	1,05E+04	5,69E+03
F7	best	24,1941	10,7364	28,9944	14,1078	9,66E+05	30,6923

	mean	**26,1201**	146,0278	109,7393	60,2431	2,99E+06	5,68E+04
	std	0,9518	117,4616	91,3852	31,2730	1,27E+06	1,26E+05
F8	best	2,15E-10	2,02E-07	8,81E-07	8,31E-09	5,6500	2,21E-06
	mean	**4,96E-09**	4,00E-06	2,08E-06	2,25E-05	21,5444	0,0055
	std	9,23E-09	7,81E-06	1,13E-06	5,61E-05	12,9719	0,0104
F9	best	0,6667	0,6914	0,6796	0,6668	4,29E+03	0,8096
	mean	**0,6667**	2,5615	0,8717	0,9614	1,53E+04	1,14E+03
	std	6,63E-06	1,6872	0,2026	0,5457	6,64E+03	4,17E+03
F10	best	8,01E-06	0,0134	2,8017	2,48E-04	513,6474	0,0173
	mean	**2,91E-04**	0,1436	81,0033	0,0191	1,12E+03	9,1024
	std	3,35E-04	0,1577	80,5274	0,0157	359,2356	19,5399

To assess the performance of optimization algorithms in terms of avoiding local optima and their ability to explore, the literature often utilizes multimodal functions as benchmark tests. The results, including the average (mean), best, and standard deviation (std) values, over 30 separate runs, are presented in Tables 6 to evaluate the algorithms in the context of multimodal functions. The outcomes presented in Table 6 indicate that mAFT demonstrates highly effective performance when applied to address multimodal challenges as well. Among 10 multimodal comparison functions, mAFT gave the best results in 8 functions.

Table 6 Results of the algorithms in multimodal benchmark test functions

		mAFT	**AFT**	**DE**	**PSO**	**ABC**	**SCA**
F11	best	8,01E-04	0,5074	854,6543	0,0222	1,12E+09	3,38E+04
	mean	**0,0398**	9,5336	1,42E+03	4,0655	3,15E+09	3,16E+07
	std	0,1346	20,8414	248,9869	8,2497	1,17E+09	8,62E+07
F12	best	6,87E-05	8,59E-03	0,0317	1,37E-03	20,2762	8,51E-04
	mean	**1,16E-03**	0,1071	0,0771	0,0289	31,6088	0,3319
	std	1,05E-03	0,1123	0,0730	0,0650	4,8271	0,5056
F13	best	1,08E-09	0,0110	1,11E-04	4,84E-08	1,45E+05	7,19E-06
	mean	**3,37E-06**	1,2410	7,61E-03	0,0334	3,48E+06	4,7867
	std	8,05E-06	1,8387	0,0262	0,0508	5,63E+06	16,9073
F14	best	2,32E-05	0,9313	6,02E-03	1,78E-03	5,3457	0,0573
	mean	**2,27E-04**	2,9115	0,0122	1,5968	7,5231	15,1899
	std	1,68E-04	1,1217	3,81E-03	0,7917	1,1747	8,0826
F15	best	40,2358	89,7321	34,1135	55,3950	916,4713	94,8101
	mean	77,7058	196,3360	**50,5950**	202,0776	2,89E+03	431,3503
	std	29,1048	80,3187	8,1162	129,9851	1,27E+03	418,8828
F16	best	0,0999	0,8999	0,7028	0,4999	3,4400	0,2999
	mean	**0,3932**	1,4165	0,9097	0,9132	4,1855	1,0336
	std	0,1413	0,3075	0,0981	0,2921	0,4590	0,5976
F17	best	-1,05E+03	-1,08E+03	-1,17E+03	-1,09E+03	-769,4062	-637,2641
	mean	-992,1500	-996,8623	**-1,17E+03**	-994,0341	-703,6430	-582,9090
	std	32,7738	48,3479	6,0728	35,8785	34,6580	39,1947
F18	best	1,58E-08	4,30E-06	3,78E-05	8,40E-06	0,9516	8,53E-04
	mean	**2,43E-07**	0,0143	4,40E-03	9,97E-03	1,0358	0,4855
	std	2,64E-07	0,0202	0,0136	0,0137	0,0347	0,3176
F19	best	-0,9891	1,50E-19	2,45E-12	2,88E-20	9,45E-11	8,91E-11
	mean	**-0,9465**	7,97E-18	4,53E-12	1,25E-16	4,21E-10	3,50E-10
	std	0,0514	1,92E-17	1,39E-12	4,72E-16	2,41E-10	3,37E-10
F20	best	3,51E-12	4,18E-12	2,27E-11	3,80E-12	1,48E-06	4,91E-11
	mean	**4,78E-12**	8,19E-12	2,74E-11	5,00E-12	9,09E-06	9,30E-10
	std	3,58E-12	3,89E-12	2,66E-12	7,40E-13	7,95E-06	6,24E-10

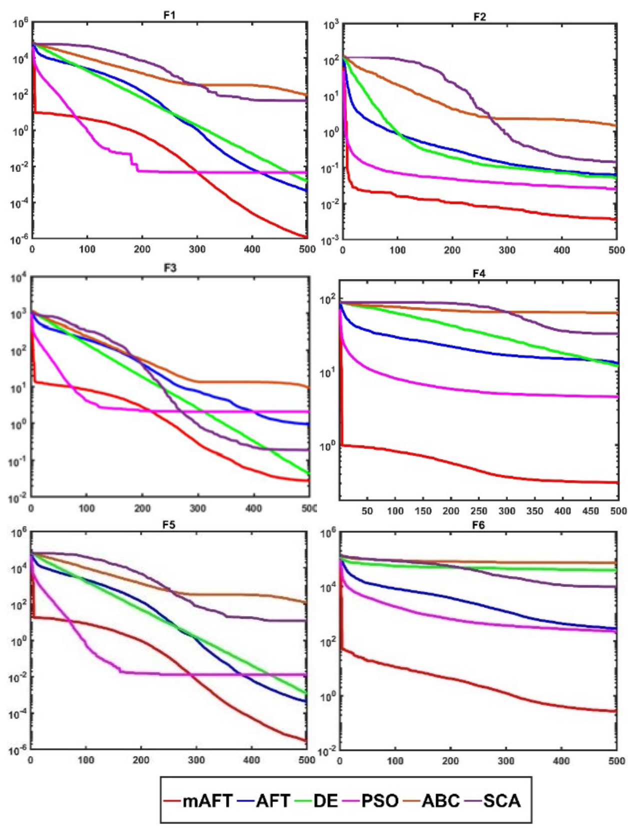

Fig. 4. Convergence curves of all algorithms for unimodal test functions F1, F2, F3, F4, F5 and F6

The convergence curves, concerning the mAFT algorithm's performance in solving a designated array of unimodal and multimodal test functions, are depicted in Fig. 4 and Fig. 5 for a maximum of 500 iterations.

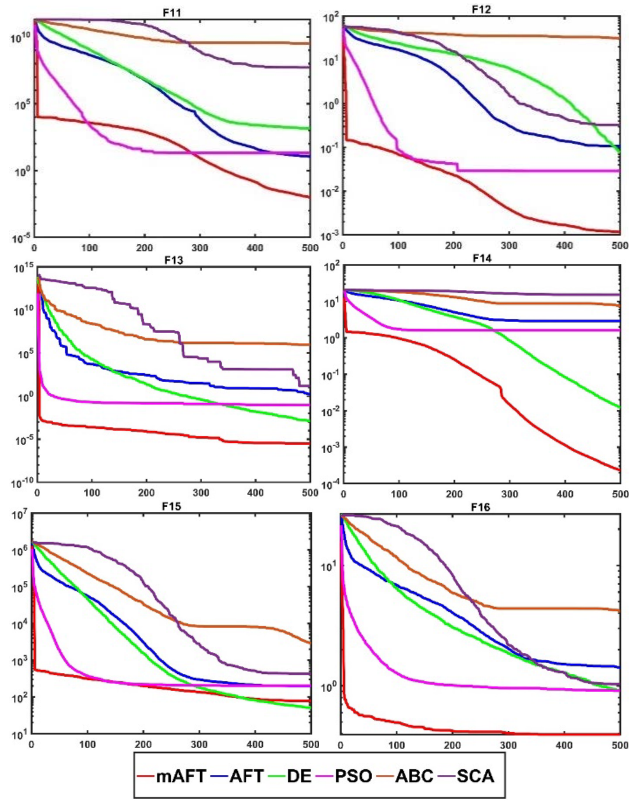

Fig. 5. Convergence curves of all algorithms for multimodal test functions F11, F12, F13, F14, F15 and F16

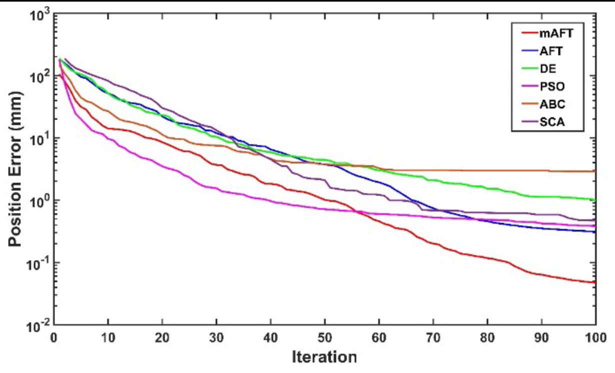

Fig. 6. Position error (mm) graph depending on iteration

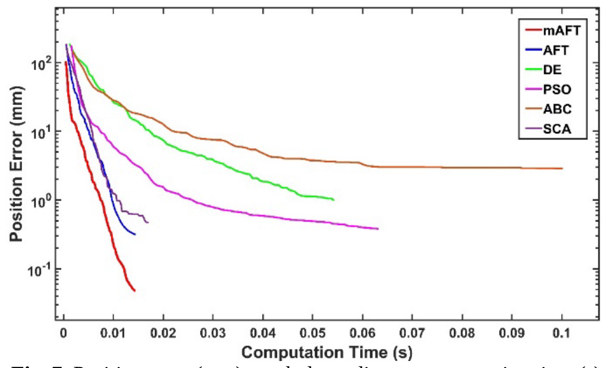

Fig. 7. Position error (mm) graph depending on computation time (s)

As can be clearly seen from Figures 6 and 7, mAFT algorithm is the algorithm that obtains the lowest value of position error depending on both iteration and calculation time.

Results, including average (mean), best (min), and standard deviation (std) values from more than 30 separate runs, are presented in Table 7 to evaluate the algorithms in the context of the inverse kinematics problem. The results presented in Table 7 show that mAFT performs best.

Table 7 Comparison of position error (mm) and computation time (s)

		mAFT	**AFT**	**DE**
Position Error (mm)	min	2,13E-05	0,0016	0,2014
	mean	**0,0475**	0,3144	0,9862
	std	0,0583	0,8245	0,6001
Compuation Time (s)	min	0,0141	0,0141	0,0534
	mean	**0,0144**	0,0145	0,0543
	std	5,76E-04	7,58E-04	9,77E-04
		PSO	**ABC**	**SCA**
Position Error (mm)	min	6,66E-03	0,4933	3,50E-05
	mean	0,3782	2,8860	0,4734
	std	0,3585	2,3721	0,8768
Compuation Time (s)	min	0,0615	0,0982	0,0166
	mean	0,0632	0,1002	0,0171
	std	1,51E-03	1,94E-03	7,01E-04

6. Conclusion

This research involved conducting simulations to validate the precision and effectiveness of the mAFT algorithm in the inverse kinematics computation for a 7-degree of freedom serial robot manipulator. The intricacy and the formidable nature of the inverse kinematics procedure make it highly conducive to the application of metaheuristics. The results obtained by leveraging the fast and straightforward structure of the AFT algorithm are clearly evident from the outcomes of the conducted experimental studies, which are highly satisfying. It has been observed that the proposed methods to enhance the convergence speed of the AFT algorithm have yielded successful results. As a result, this has enabled the rapid resolution of the inverse kinematics problem for inverse robot manipulators. In future studies, exploring potential extensions or hybrid approaches to improve the algorithm's efficiency and robustness could contribute to advancing the field of inverse kinematics for robot manipulators.

Declaration of conflicting interests

The authors declare no competing interests.

Funding

The author received no financial support for the research and/or authorship of this article.

References

[1] Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., ... & Herrera, F. "A tutorial on the design, experimentation, and application of metaheuristic algorithms to real-world optimization problems" *Swarm and Evolutionary Computation*, 64, 100888,2021

[2] Zaman, H. R. R., & Gharehchopogh, F. S. "An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems" *Engineering with Computers*, 38(Suppl 4), 2797-2831,2022.

[3] Holland, J. H. "Genetic algorithms" *Scientific american*, 267(1), 66-73,1992.

[4] Storn, R., & Price, K. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." *Journal of global optimization*, 11, 341-359,1997.

[5] Eberhart, R., & Kennedy, J. "Particle swarm optimization." *In Proceedings of the IEEE international conference on neural networks* Vol. 4, pp. 1942-1948,1995

[6] Karaboga, D. "An idea based on honeybee swarm for numerical optimization." *Technical report-tr06, Erciyes university, engineering faculty, computer engineering department*, Vol. 200, pp. 1-10,2005.

[7] Mirjalili, S. "SCA: a sine cosine algorithm for solving optimization problems". *Knowledge-based systems*, 96, 120-133,2016.

[8] Kurdila J. and Ben-Tzvi P., "Dynamics and control of robotic systems." *John Wiley and Sons*,2019.

[9] Sahu, V. S. D. M., Samal, P., and Panigrahi, C. K., 2022. *Modelling, and control techniques of robotic manipulators: A review.* Materials Today: Proceedings, 56:2758-2766.

[10] Köker, R., Öz, C., Çakar T, and Ekiz H., "A study of neural network based inverse kinematics solution for a three-joint robot" *Rob Auton Syst*, 49:227–234,2004.

[11] Dereli, S., and Köker, R., "A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm". *Artificial Intelligence Review*, 53:949-964,2020.

[12] Abdor-Sierra, J. A., Merchán-Cruz, E. A., and Rodríguez-Cañizo, R. G., "A comparative analysis of metaheuristic algorithms for solving the inverse kinematics of robot manipulators*" Results in Engineering*, 16:100597,2022.

[13] Kucuk, S., 2017. "Optimal trajectory generation algorithm for serial and parallel manipulators," *Robot. Comput. Integrated Manuf.* 48:219–232,2017.

[14] Souza, D. A., Batista, J. G., dos Reis, L. L., and Júnior, A. B., "PID controller with novel PSO applied to a joint of a robotic manipulator". *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 43:1-14,2021.

[15] Braik, M., Ryalat, M. H., & Al-Zoubi, H. "A novel meta-heuristic algorithm for solving numerical optimization problems": Ali Baba and the forty thieves.*" Neural Computing and Applications*, 34(1), 409-455,2021.

[16] Yaskawa America, Inc., "*SIA20D Compact 7-Axis Robot Arm*", Jan. 2018.