УДК 621.865.8(045)
DOI 10.22213/2413-1172-2018-4-28-34

OBTAINING THE KINEMATICS SOLUTION OF AN AERIAL MANIPULATOR USING THE SHUFFLED FROG-LEAPING ALGORITHM

I. N. Ibrahim, PhD Student, Kalashnikov ISTU, Izhevsk, Russia

This paper concentrates on deriving the real-time kinematics solution of a manipulator attached to an aerial vehicle, while the vehicle's movement itself is not analyzed. The manipulator kinematics solution using Denavit-Hartenberg model was introduced, too. The fundamental scope of this paper is to get a global online solution of the design configurations with a weighted objective function subject to some constraints. Adopting the resulted forward kinematics equations of the manipulator, the trajectory planning problem turns into an optimization task. Several and well-known computing methods are documented in the literature for solving constrained complicated nonlinear functions, where in this study a shuffled frog-leaping algorithm (SFLA) is suggested, which is one of the artificial intelligence techniques and regarded as a search method. It is a constrained metaheuristic and population-based approach. Moreover, it is able to solve the inverse kinematics problem considering the mobile platform, in addition to avoiding singularities, since it does not demand the inversion of a Jacobian matrix. Simulation experiments were carried out for the trajectory planning of a six degree-of-freedom (DOF) aerial manipulator, and the obtained results confirmed the feasibility and effectiveness of the suggested method.

Keywords: inverse kinematics, metaheuristics methods, evolutionary algorithms, techniques, shuffled frog-leaping algorithm.

Introduction

The inverse kinematics (IK) solver is a primary problem in robotic manipulation, particularly when demand real-time and precision in calculations. Mathematically, the numerical solution of kinematics is intricate because of the high degree of nonlinearity. furthermore, the Linear and dynamic programming techniques usually fail or reach local optimum in solving NP-hard problems with a large number of variables and nonlinear objective functions. moreover, Traditionally Jacobian-based solutions are identified to scale inadequately with the high number of degrees of freedom (DOF) [1] in addition to singularities existence. In contrast [2] presented a comparative study of several methods based on the Jacobian matrix, clarifying that the modified Levenberg-Marquardt method is much better for a quite large set of random configurations than others but may lose convergence compared to Jacobian transpose and Pseudocode inverse methods. recently many researchers [3] proposed a new method for solving real-time IK without using the Jacobian matrix based on the position of end-effector (ee), using numerical and analytical mathematical tools but not mentioned exactly the performance as the time consuming to get the solution, in [4] also applied alike method for $(2n + 1)$ DOF hyper-redundant manipulator arm. Authors in [5] combined two methods as a real-time IK solver for a human-like arm manipulator based on closed-form analytical equations for a given position while others [6] presented an online adaptive strategy based on the Lyapunov stability theory in addition to Radial Basis Function Network (RBFN) and quadratic programming which requires a complex hardware resources, the simulation was done for the position of ee in addition to avoid obstacles and was conducted on the 7-DOF PA-10 robot manipulator. In [7] a kinematic and time-optimal trajectory planning was considered for redundant robots, two approaches were presented, joint space decomposition and a numerical null-space method for a given pose. They were tested by 7-DOF industrial robots and demand high consuming time for resolving IK. Nowmetaheuristic optimization algorithms are an encouraging alternative approach to traditional IK techniques due to their strong performance on challenging and high-DOF problems in many various domains, the solution can be solved by minimizing an objective function. [8, 9] proposed an SFLA and MSFLA respectively, that for a high dimensional continuous function optimization. These methods yield a strong robustness and best convergence also presented a comparative study for PSO, SFLA, MSFLA, and MSFLA-EO. Which designated that MSFLA is better than others. In [10] a modified SFLA was assumed for obtaining the optimum preventive maintenance scheduling of generating units in power system. While [11] presented a comparative study

among five evolutionary-based optimization algorithms as GA, MA, PSW, ACO, and SFLA, they presented the processing time for solving the F8 function and concluded that the SFLA is the best. This work is considered as an extension of work in [12-15]. The proposed algorithm isthe modified shuffled frog-leaping algorithm (MSFLA), which is characterized as accurate and fast converging in discovering the solution based on the previous literature study. Initially, we define an objective function to minimize the error between the desired and the actual end-effector pose. The objective function considers the minimal movement between the previous and the actual joint configurations. To overcome the constrained problems, we use a penalty function to penalize all those manipulator configurations that violate the allowed joint boundary. Hence, the proposed approach estimates the feasible manipulator configuration needed to reach the desired end-effector pose. The remainder of this paper is organized as follows: Section 2 present the architecture and kinematics of a robotic manipulator. Sections 3 and 4 introduce metaheuristic optimization algorithm MSFLA and the weighted objective function. Section 4 shows the simulation results of the proposed trajectory planning method applied to the manipulator.

Manipulator Kinematics

In order to determine the relationship between the coordinate frames, which are assigned to robots' links and joints, homogeneous transformations are required. Three parameters are employed to describe the rotation while another three parameters are used to define the translation. Accordingly, the Denavit-Hartenberg (DH) convention was used to describe kinematically the rigid motion by assigning the values of four quantities for each link, two describe the link itself, and two describe the link's connection to a neighboring link. Where θ, a, d and α are the joint angle, link length, link offset and link twist between joints. While T_i is the homogeneous transformation matrix between the frames that is a function of θ while the other three parameters are constant. The data in Table 1 represent link parameters of the arm-part based on DH strategy in two formulas: standard and modified DH. Whereas the standard simulation form of LabVIEW Robotics module was used, in order to validate the design. The position of all links of an arm-part manipulator can be specified with a set of 6 joint variables from the shoulder's joints till wrist's joints. This set of variables is often referred to as a 6×1 joint vector [12].

Table 1. **Link parameters of the manipulator's arm-part**

Modified denavit hartenberg				
α_{i-1}	a_{i-1} [cm]	d_i [cm]	θ_i	Initial value of θ_i
$-\pi/2$	l_0	0	θ_1	$\pi/2$
$\pi/2$	l_1	0	θ_2	$-\pi/2$
0	l_2	0	θ_3	$-\pi/2$
$-\pi/2$	0	l_3+l_4	θ_4	0
$\pi/2$	0	0	θ_5	$-\pi/2$
$-\pi/2$	l_5	0	θ_6	0

Standard denavit hartenberg					
α_i	a_i [cm]	d_i [cm]	θ_i	Initial value of θ_i	Joint offset
$-\pi/2$	6.4	0	θ_1	0	0
0	30.2	0	θ_2	$-\pi/2$	$-\pi/2$
$\pi/2$	0	0	θ_3	$\pi/2$	$\pi/2$
$\pi/2$	0	23.5	θ_4	0	0
$-\pi/2$	5.3	0	θ_5	$\pi/2$	$\pi/2$
0	5.6	–2	θ_6	0	0

The space of all joint variables is referred to as the joint-space $\Theta=[\theta_1,\theta_2,\ldots,\theta_6]^T$. Here we have been concerned with computing the Cartesian space representation from the knowledge of the joint-space information. Hence the homogeneous transformations of the links were used ${}^{i-1}_{i}T$. If the robot's joint-position sensors are estimated by servomechanisms, the Cartesian position and orientation of the hand-part can be computed by ${}^{0}_{7}T$ [12].

Proposed Optimization Techniques for solving kinematics

The evolutionary optimization algorithms can solve the complicated nonlinear equations completely and efficiently. The solution of the inverse kinematics for the manipulator is a very difficult problem to obtain by traditional approaches. Besides, the suggested strategies do not require the inversion of any Jacobian matrix, and then it avoids singularities configurations. In this paper, two algorithms are used to optimize this problem, the differential evolution and the modified shuffled frog-leaping algorithms. In general, this optimization technique is based on the forward kinematics equations, which always produces a solution in cooperation with an objective function. Hence, the general aspect of the problem can be expressed as minimizing $J(\Theta)$, constrained by $\Theta_{\min}\le\Theta\le\Theta_{\max}$. Furthermore, the objective function could be defined as the weighted sum of the errors as follows

$$J(\Theta) = \sigma P_{error}(\Theta) + \varepsilon O_{error}(\Theta) = \\ = \sigma \left\| P_G - P_E(\Theta) \right\| + \varepsilon \left\| O_G - O_E(\Theta) \right\|,$$

Where $P_{error}(\Theta)$ and $O_{error}(\Theta)$ represent the position and orientation errors respectively and could be computed as a difference in distance between the target and current position, in this work we used an Euclidean formula as a representation of distance. While the parameters σ and ε are the weights of the position and the orientation, respectively. Let $G = (P_G, O_G)$ be a given target end-effector pose while $E(\Theta) = (P_E(\Theta), O_E(\Theta))$ is the current end-effector pose in the workspace corresponding to configuration $\Theta = [\theta_1, \theta_2 \dots \theta_6]^T$ which can be calculated using forward kinematics, where *P* refers to the 3D position vector of pose while *O* refers to the vector of Roll-Pitch-Yaw Euler angles of pose (in radians), respectively. Which the optimization algorithms are exploring directly in the configuration space of the manipulator. Hence, each individual $\Theta_i = [\theta_{i,1}, \theta_{i,2} \dots \theta_{i,j} \dots \theta_{i,6}]^T$ represents an *i*-th candidate set of joint angles. Henceforward, at each iteration, we evaluate each candidate configuration Θ_i by passing it through the forward kinematics module and measuring the position and orientation error between where the end-effector would be at configuration Θ_i and the target end-effector pose. In order to enforce joint limits, each dimension *j* of element Θ_i should be limited to searching in the range of valid joint angles $\Theta_i \in [\Theta_{\min}, \Theta_{\max}]$. This can be realized by clamping each dimension *j* within these bounds at each iteration immediately after it is updated.

Modified Shuffled Frog-Leaping Algorithm

The shuffled frog-leaping algorithm (SFLA) was developed by Eusuff and Lansey in 2003 [8]. It is a member of the Memetic algorithm family, a particular type of meta-heuristic optimization approaches and evolutionary algorithms, which is based on population. It is inspired by the memetic evolution of frogs exploring food in a lake, which consolidates the benefits of the genetic-based memetic algorithms (MAs) and by the social behavior-based particle swarm optimization [9]. generally, the SFLA incorporates two alternating processes: a local exploration in the sub-memeplex and a global information exchange among all memeplexes.The SFLA optimization achievement basically relies on two facts, the first one is the evolution process on each memeplex that embraces different cultures of frogs, where the culture stimulates a fitness value, and serves as a local search within memeplex analogous to PSO algorithm which imitates the social behavior of the leaping action of frogs searching for food. The second fact is an idea held within each frog which can be influenced by the ideas of other frogs from other memeplexes throughout the shuffling rule, this animates the cooperation process which it implies an adaptation idea and improves the success rate of discovering the solution in the optimization puzzle. In this process, a modification was applied to the frog-leaping action that enhances the exploration manner in the space [10, 11]. Moreover, the randomization strategy in the evolution process proffers the algorithm the ability to discover the local best solution within search space stochastically in addition to the communication process that possibly finds a global optimum solution in shorter time. The local search and the shuffling processes continue until the defined convergence criteria are satisfied. The pseudocode of the algorithm is presented in Algorithm 1.

The MSFLA meta-heuristic strategy is summarized in the following steps:

a. Initialization step, construct the population *NP* of frogs randomly similar to the first step in DE algorithm, then Select *m*, and *n*, where *m* is the number of memeplexes and *n* is the number of frogs in each memeplex. Therefore, the total amount of frogs *NP* can be calculated as $NP = m.n$, additionally, the i[th] frog is expressed as a vector with a dimension equal to the configuration space as follows $\Theta_i = (\theta_{i,1}, \theta_{i,2}, \dots, \theta_{i,6});\ i = 1, 2, \dots, NP.$

b. Rank step, compute the performance value f_i for each frog Θ_i. Sort the *NP* frogs in a descending order according to their fitness. Save them in an array $U = \{f_i, \Theta_i;\ i = 1, 2, \dots, NP\}$, so that $i = 1$ denotes the frog with the best performance value and could save it as a Θ_g in each iteration while the algorithm is running.

c. Partition Step, partition array *U* into *m* memeplexes $Y_1, Y_2, \dots, Y_m$, each including *n* frogs, such that

$$Y^k = \left[\Theta_i^{\,k}, f_i^k \mid \Theta_i^{\,k} = \Theta_{(k+m(i-1))}, \right. \\ \left. f_i^k = f_{(k+m(i-1))},\ i = 1, \dots, n \right];\ k = 1, \dots, m.$$

In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog *m* goes to the *m*[th] memeplex, and frog *m* + 1 goes back to the first memeplex, etc.

d. Memetic Evaluation step, evolve each memeplex $Y^k; k = 1, \dots, m$ according to the frog-leaping

algorithm as follow. Within each memeplex, the frogs with the best and the worst fitness values are defined as Θ_b and Θ_w, respectively. Furthermore, the frog with the global best fitness is defined as Θ_g. Next, an improvement process is applied to only the frog with the worst fitness in each cycle. Hence, the position of the frog with the worst fitness is modified which emulates the leaping process as follows: leaping distance $D = C_L rand(0,1)[\Theta_b - \Theta_w]$, then new position $\Theta_w = \Theta_w + D;\ D \in]-D_{\max}, D_{\max}[$. Where, $rand(0,1)$ is a random number between 0 and 1, $D_{\max}$ is the maximum allowed change in a frog's position and C_L is the modification of the algorithm which it is a constant indicates the amount of frog-leaping in each memeplex. The evaluation process, for all memeplexes, is repeated by an adaptable number of iterations iM, until no improvement becomes possible.

e. Shuffle Memeplexes Step, shuffle frogs and replace all memeplexes $Y^k; k = 1, \ldots, m$ into U, such that $U = \{f_i, \Theta_i;\ i = 1, 2, \ldots, NP\}$ similar to the initialization phase, afterwards sort U in order to decrease the performance value, update the population best frog's position Θ_g.

f. Check convergence, check the convergence criteria if satisfied then stop otherwise return to the partition step and continue for a specific quantity of iterations iN, finally after each iteration the first frog in the sorted list represents a global solution. The number of iterations iM specifies the depth of search within memeplexes while iN governs the solution producing process.

Algorithm 1. **The pseudo-code of the Shuffled Frog-Leaping Algorithm**

Initialization:
$Population \leftarrow \{\Theta_1, \Theta_2, \ldots, \Theta_i, \ldots, \Theta_{NP}\}$;
$m \leftarrow$ *number ofmemeplexes*;
$n \leftarrow$ *quantity of frogs in each memeplex*;
$l \leftarrow 1, iN$
while (*convergence criteria is satisfied Or until met iN*) *do*
  Rank Step: Evaluate each frog Θ_i using a fitness function;
  Partition Step:
  Construct an array U of frogs and their fitness's values;
  Sort the array U in descending order based on the fitness column;
  Construct $(Y^k; k = 1, \ldots, m)$ memeplexes each including n frogs;
  Evaluation Step:
  for $\ell \leftarrow 1, iM$ *do*
    for $k \leftarrow 1, m$ *do*
    Determine the worst and best frogs position based on their fitness's values;
    Improve the worst frog position using a leaping distance;
    end for
  end for
  Shuffle Memeplexes Step: combine the evolved memeplexes;
  Check Convergence: Update the population best frog's position Θ_g;
  $l \leftarrow l + 1$;
end while

Simulation Results and Discussions

In this work, we solved the inverse kinematics of a redundant manipulator with six joints to follow a destination pose. The manipulator's joints correspond to the variable $\theta_j : j = 1, 2, \ldots, 6$ are constrained. The DH table is presented in Table 1. In the inverse kinematics experiments, the desired end-effector pose for the arm-part of the manipulator was determined by this vector

$$G = (P_G, O_G) = (x, y, z, roll, pitch, yaw) = \\ = (-20, 3, 40, 0, 10, 15).$$

Moreover, the parameters of the objective function were adjusted as follows $\varepsilon = 1 - \sigma = 0.7$, so there is a balance between position and orientation to be optimized. In case of MSFLA, the parameters of the algorithm were introduced in Table 2, and a summary of the results of utilizing the algorithm for multiple scenarios was introduced in Table 3.

Table 2. **Setting of the MSFL Algorithm**

m	Number of memeplexes	3
n	Number of frogs within memeplexes	NP/m
C_L	Amount of Leaping	1.3

Hereafter, Fig. 1 displays the values of the objective function, while Fig. 2 and Fig. 3 represent the position and orientation of the manipulator's end-effector after applying the solutions to validate IK solver.

Table 3. **Inverse Kinematics Results of MSFL Algorithm**

Tests	Population	Iterations		$J(\Theta)$	Total Error	Execution Time [ms]	Reaching Target ($x, y, z, roll, pitch, yaw$)
		iN	iM				
1	20	30	10	11.618	29.71	729	(–15.7365, 5.43, 52.57, 6.63, 12.66, 16.164)
2	30	30	10	7.6614	12.08	1045	(–21.183, 2.915, 50.77, –0.201, 10.01, 14.85)
3	40	30	15	10.5382	19.21	1685	(–25.08, 8.56, 46.818, –6.2, 9.6, 5.4251)
4	40	40	30	18.4625	18.46	4526	(–25.23, 8.34, 47.59, –2.53, 8.62, 14.13)
5	60	40	30	8.2925	8.292	6645	(–24.46, 0.0421, 44.59, 1.65, 11.116, 14.05)
6	80	50	40	11.024	11.02	13540	(–26.998, 3.594, 42.87, –0.068, 9.81, 15.67)
7	100	60	60	29.774	29.77	24191	(–20.03, 30.039, 39.971, –7.71, 3.679, –0.72)
8	130	70	60	0.1511	0.649	46282	(–20.09, 2.99, 40.004, 0.208, 9.64, 14.89)
9	170	60	50	0.6168	2.168	40459	(–20.151, 2.84, 40.09, 0.89, 10.36, 16.15)
10	200	90	40	0.1139	0.298	57362	(–19.927, 3.0072, 39.98, –0.134, 10.105, 14.89)
11	200	100	60	0.0729	0.378	92779	(–20.002, 2.998, 39.99, –0.137, 10.151, 14.92)
12	200	120	80	2.7672	5.8164	150246	(–20.48, 4.153, 40.49, 0.807, 4.087, 13.787)
13	200	200	100	2.6713	1.9339	318481	(–19.27, 2.235, 41.94, –0.979, 10.88, 11.48)
14	250	90	40	0.003	0.016	69818	(–19.99, 3.00023, 39.99, 0.0049, 10.006, 14.99)
15	250	140	80	1.266	6.553	215027	(–20, 3, 40, –7.01976e –10, 10, 15)
16	250	140	100	4.647e-9	1.05e-8	260325	(–20.09, 2.97, 40.0008, –1.687, 6.689, 13.57)
17	300	140	80	1.01e-9	3.33e-9	255989	(–20, 3, 40, 1.16487e –9, 10, 15)
18	500	90	40	5.49e-10	9.9e-10	136888	(–20, 3, 40, –1.66261e –11, 10, 15)
19	500	200	100	3.02e-15	1.56e-14	681646	(–20, 3, 40, 3.22962e –15, 10, 15)
20	1000	30	45	0.0968	0.025	95197	(–20.031, 3.04, 40.04, –0.0037, 10.96, 14.876)

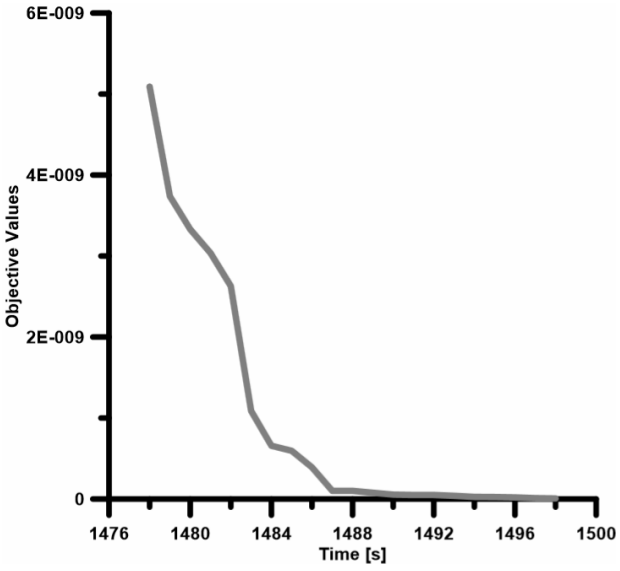

Fig. 1. The objective function values after applying IK-MSFLA solver

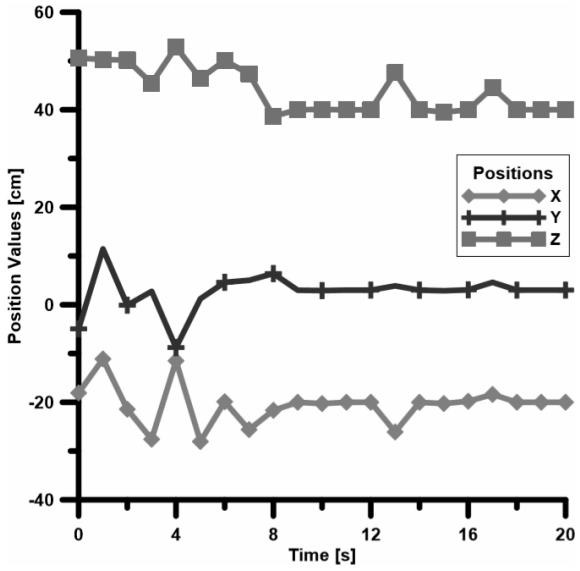

Fig. 2. The end-effector position of the manipulator after applying the solutions to validate IK-MSFLA solver

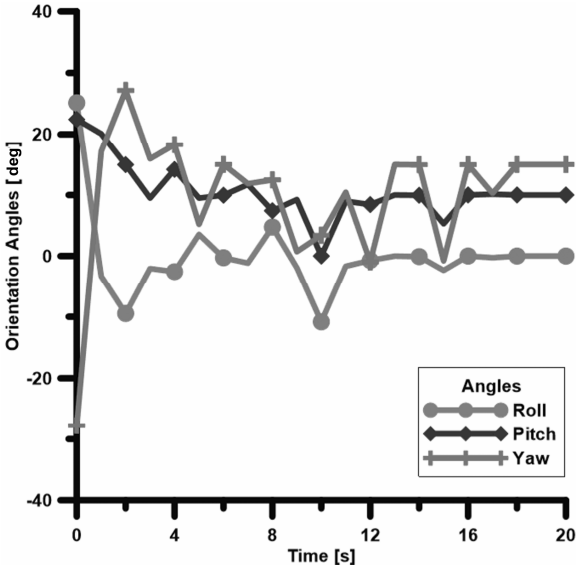

Fig. 3. The end-effector orientation of the manipulator after applying the solutions to validate IK-MSFLA solver

References

1. Buss S. R. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 2004, 17(1-19), 16.

2. Dulęba I., Opałka M. A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators. *International Journal of Applied Mathematics and Computer Science*, 2013, 23(2), 373-382.

3. Wang X., Zhang D., Zhao C. The inverse kinematics of a 7R 6-degree-of-freedom robot with non-spherical wrist. *Advances in Mechanical Engineering*, 2017, 9(8), 1687814017714985.

4. Ananthanarayanan H., Ordóñez R. Real-time Inverse Kinematics of (2*n* + 1) DOF hyper-redundant manipulator arm via a combined numerical and analytical approach. *Mechanism and Machine Theory*, 2015, 91, 209-226.

5. Tolani D., Badler N.I. Real-time inverse kinematics of the human arm. *Presence: Teleoperators & Virtual Environments*, 1996, 5(4), 393-401.

6. Toshani H., Farrokhi M. Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach. *Robotics and Autonomous Systems*, 2014, 62(6), 766-781.

7. Reiter A., Müller, A. Gattringer H. Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators. Proc. *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE* (2016, October), pp. 6873-6878. IEEE.

8. Eusuff M., Lansey K., Pasha F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering optimization*, 2006, 38(2), 129-154.

9. Li X., Luo J., Chen M. R., Wang N. An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimisation. *Information Sciences*, 2012, 192, 143-151.

10. Samuel G. G., Rajan C. C. A. A modified shuffled frog leaping algorithm for long-term generation maintenance scheduling. Proc. *Third International Conference on Soft Computing for Problem Solving*, 2014, pp. 11-24. Springer, New Delhi.

11. Afzalan E., Taghikhani M. A., Sedighizadeh M. Optimal placement and sizing of DG in radial distribution networks using SFLA. *International Journal of Energy Engineering*, 2012, 2(3), 73-77.

12. Ibrahim I. N. Ultra-Light Weight Robotic Manipulator. *Vestnik IzhGTU imeni M. T. Kalashnikova*, 2018, vol. 21, no. 1, pp. 12-18. Doi: 10.22213/2413-1172-2018-1-12-18 (in Rus.).

13. Ibrahim I. N., Al Akkad M. A. Studying the Disturbances of Robotic Arm Movement in Space Using the Compound-Pendulum Method. *Vestnik IzhGTU imeni M. T. Kalashnikova*, 2017, vol. 20, no. 2, pp. 156-159.

14. Ibrahim I. N., Al Akkad M. A., Abramov I. V. Attitude and altitude stabilization of a microcopter equipped with a robotic arm. Proc. *International Siberian Conference "Control and Communications (SIBCON)"*, 2017, June, pp. 1-8. IEEE.

15. Ibrahim I. N., Al Akkad M. A., Abramov I. V. Exploring Ackermann and LQR stability control of stochastic state-space model of hexacopter equipped with robotic arm. *Journal of Physics: Conference Series*, 2018, vol. 1015, no. 3, p. 032160.

16. Ibrahim I. N., Al Akkad M. A., Abramov I. V. UAV efficient PID and LQR controllers design based on its stochastic state space dynamics model including disturbances. Proc. *Electronic and Networking Technologies (MWENT)*. Moscow, 2018, March, pp. 1-9. IEEE.

Кинематическое решение для манипулятора беспилотного летательного аппарата на основе модифицированного алгоритма прыжка лягушки

И. Н. Ибрахим, аспирант, ИжГТУ имени М. Т. Калашникова, Ижевск, Россия

Рассмотрено кинематическое решение в реальном времени для манипулятора, прикрепленного к беспилотному летательному аппарату; движение самого транспортного средства в данном исследовании не анализируется. Представленное кинематическое решение для манипулятора основано на модели Денавита – Хартенберга. Основной целью исследования является получение глобального решения в реальном времени для конфигурации и проектирования с взвешенной целевой функцией с наложением некоторых ограничений. Применение уравнений прямой кинематики манипулятора, полученных в результате исследования, позволяет превратить задачу планирования траектории в задачу оптимизации. Хорошо известны несколько типов вычислительных методов для решения ограниченных сложных нелинейных функций.

В данном исследовании предлагается модифицированный алгоритм прыжка лягушки (SFLA), который является одним из методов искусственного интеллекта и рассматривается как метод поиска. Это ограниченный метаэвристический и популяционный подход. С его помощью представляется возможным решение обратной кинематической задачи с учетом мобильности платформы. Кроме того, данный метод предотвращает появление сингулярных точек, поскольку он не требует инверсии матрицы Якоби. Результаты экспериментального моделирования для планирования траектории манипулятора с шестью степенями свободы подтвердили целесообразность и эффективность предлагаемого метода.

Ключевые слова: манипулятор, обратная кинематика, метаэвристические методы, эволюционный алгоритм, методы оптимизации, алгоритм прыжка лягушки.

Получено 12.09.2018