Nigerian Journal of Technology (NIJOTECH)
Vol. 40, No. 4 July, 2021, pp.674 –683
www.nijotech.com

Print ISSN: 0331-8443
Electronic ISSN: 2467-8821
http://dx.doi.org/10.4314/njt.v40i4.14

Industrial Manipulator Dynamic Parameter Estimation Using Mutating Particle Swarm Optimization (Mupso)

A. Umar[1], Z. Shi[2,*], A. Khlil[3], and Z. I. B. Farouk[4]

[1,2,3] *School of Mechanical Engineering, Hebei University of Tech., 300401, Tianjin-China.*
[4]*School of Mechanical Engineering, Tianjin University, 300350, Tianjin-China.*

Abstract
This work aims at developing a dynamic model and estimating the unknown parameters of the first three joints (at the arm) of a 6 degree of freedom industrial robot manipulator, a finite Fourier series algorithm was used to design an excitation trajectory, a mutating particle swarm optimization algorithm was used to optimise the parameters of the Fourier series thereby minimizing the condition number of the observation matrix, and a linear least-squares methods was implemented for estimating the unknown dynamic parameters of the manipulator. A mutation function was implemented to break the algorithm out of stagnation. Out of the thirty unknown parameters at the industrial manipulator arm, twenty were identified independently, two were identifiable in linear combinations, and the remaining eight parameters were unidentifiable. The mutating particle swarm optimization algorithm dominated other algorithms and was found suitable for robot dynamic analysis.

Keywords: Industrial manipulators, Dynamic model, Parameter Estimation, Mutating PSO

1.0 INTRODUCTION

The fourth industrial revolution is expected to unfold over the twenty-first century and would build its momentum around artificial intelligence, robotics, the Internet of Things, and machine learning. Robotics is expected to be a significant driver of the fourth industrial revolution [1], [2]. An industrial robot manipulator is a device fixed to a stationary base with limited motion within its workspace, it is used to manipulate objects without direct human input for automating basic industrial processes [3] like welding, cutting, etc. Precision agriculture is another important application of robots, where robotic technologies, sensors, and artificial intelligence have been applied to maximize agricultural output [4], while minimizing the input cost of planting, fertilizing, weed control, harvesting, etc., thereby giving more consistent product yield and maximizing output. Other recent applications of robot technology include medical surgery, aircraft manufacture, dangerous/ hazardous material handling, hostile environment/space exploration, etc. [5]. The 6 degrees of freedom (DOF) articulated robot remains the most popular industrial robot configuration which is still relevant in real-world applications [6]. It is regarded as the configuration with optimum dexterity capable of completing most jobs in an industrial workspace [7], [8].

***Corresponding author (Tel:** +86-226-043-8217)
Email addresses: 201541201002@stu.hebut.edu.cn (A. Umar), z_shi@hebut.edu.cn, (Z. Shi), 201740000010@stu.hebut.edu.cn (A. Khlil) zulfiqarbibifarouk@yahoo.co.uk (Z. I. B. Farouk)

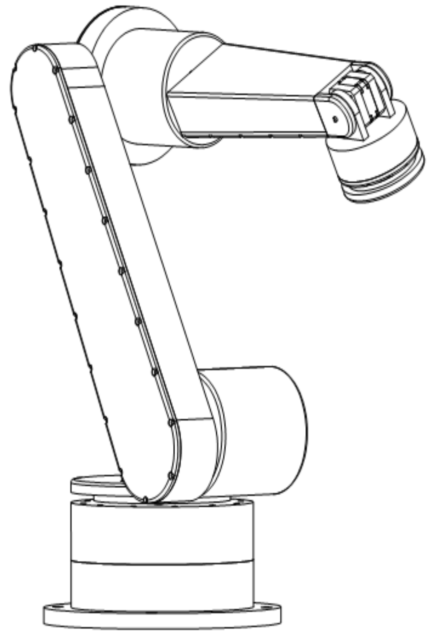
Figure 1: The 6 DOF articulated robot manipulator.

Figure 1 shows a 6DOF industrial manipulator and Table 1 presents its D-H parameters.

A manipulator's trajectory tracking efficiency is directly proportional to its performance and depends on the robot's dynamic model; the dynamic model is used to derive the energy required to actuate the joints [9]. Geometric errors, coupling, and non-linearity introduces uncertainties in the robotic system, causing the parameters to deviate from the nominal values. These errors degrade the robot’s performance; therefore, re-calibrating robot parameters are necessary for efficient robot analysis and control, which can be achieved through parameter estimation. Various techniques have been implemented for robot parameter identification including singular value decomposition [10] , maximum likelihood estimation [11] , Kalman filter [12] , and swarm-based algorithms [13], [14]. Generally, robot optimization analysis involves tedious and lengthy multistep mathematical calculations, and none of the existing methods of calculation has been proven without flaws. Therefore, it is of great theoretical significance and engineering value to study new parameter estimation methods.

A dynamic model estimation procedure requires that firstly, a linearized model of the robot manipulator is developed, then a trajectory that excites the robot's rigid body dynamics is developed, afterward input torque and output joint configuration data are obtained then finally the dynamic parameters are estimated. The accuracy of the estimation method is largely dependent on the chosen trajectory and measuring accuracy. The actuator torque can be measured from input electric currents while optic encoders are incorporated for measuring output joint angles. The recent improvement in technology has made available resolution encoders with high accuracy and repeatability, reduction gears that can minimizing gearbox backlash. Therefore, carefully selecting an observation matrix is a major objective of this work. Observing that the Linear Least Square (LLS) estimation method is computationally efficient, designing an optimized excitation trajectory for monitoring the robot's joint motion and extracting experimental data is hereby emphasized, where the LLS would be implemented for dynamic model estimation. The particle swarm optimization (PSO) has proven to be efficient in analyzing complex optimization problems. A Mutating Particle swarm optimization algorithm with modified parameters was introduced by [15] and was found suitable for robot kinematic analysis.

This work would show that the algorithm is also suitable for robot dynamic analysis. This variant of PSO shall be implemented here for dynamic model estimation of a 6 DOF industrial manipulator, and the results compared with other state of the art algorithms. The rest of the paper is structured as follows; section 2, 3 and 4 introduces the PSO algorithm, the dynamic, and parameter estimations models respectively. Section 5 presents the simulation experiment and results, while section 6 concludes the findings.

Table 1: D-H Parameters for a 6DOF Industrial robot manipulator

Joint	Link Length a (mm)	Off-set Length d (mm)	Joint Displacement θ (rad)	Off-set Displacement α (rad)
1	0	290	Variable (-11π/22 : 11π/22)	-pi/2
2	270	0	Variable (-11pi/18 : 11π/18)	0
3	70	0	Variable (-11pi/18 : 7π/18)	-pi/2
4	0	302	Variable (-8pi/9 : 8π/9)	pi/2
5	0	0	Variable (-3pi/2 : 3π/2)	-pi/2
6	0	72	Variable (-121π/90 : 121π/90)	0

2.0 PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM

The PSO is a stochastic metaheuristic swarm-based algorithm, it mimics the behaviour of swarm animals (birds or fish) to solve complex mathematical problems. It is randomly populated with individuals or particles sharing information between each other while updating their positions and velocity according to (1) and (2) until the desired solution is attained. The particle with the solution that best minimizes the fitness function is referred to as the global best position (*Gbest*), while each particle's best solution is called its personal best position (*Pbest*). Each particle in the swarm is attracted towards the *Gbest* and its *Pbest*.

$$X_k^{\dim}(iter+1) = X_k^{\dim}(iter) + V_k^{\dim}(iter+1) \quad (1)$$

$$\begin{aligned} V_k^{\dim}(iter+1) = {} & wV_k^{\dim}(iter) + c_1r_1\left(Pbest_k^{\dim}(iter) - X_k^{\dim}(iter)\right) \\ & + c_2r_2\left(Gbest^{\dim}(iter) - X_k^{\dim}(iter)\right) \end{aligned} \quad (2)$$

Where k = 1, 2, 3…$nPop$ denotes the index of each particle, $nPop$ is the swarm size, and dim = 1,2…N denotes the dimensionality of the solution space, N is the number of harmonics of the Finite Fourier Series (FFS). V_k^{dim} and X_k^{dim} stand for the position and velocity vectors of the kth particle, respectively. $V_k^{dim} = [v_k^1, v_k^2, \ldots, v_k^N]$, $X_k^{dim} = [x_k^1, x_k^2, \ldots, x_k^N]$. w is the inertia weight. c_1 and c_2 are cognitive and social learning coefficients. r_1 and r_2 are two uniformly distributed random numbers within the range of [0,1].

It was established in [15] that the basic parameters of PSO are not capable of converging the solutions for robot optimization problems especially when the DOF is greater than three. The basic parameters of the mutating PSO were modified to satisfy the requirement for solving robot optimization problems. This was achieved by testing the performance of various PSO parameters on four popular robot configurations, and a relationship between the inertia weight and the social learning coefficient was derived. A non-linearly decreasing inertia weight was implemented with values between (2.1 – 0.6), the cognitive learning coefficient was set constantly at 2.24, while the social learning coefficient was non-linearly increasing between (1.8 – 3.9). The equations for updating w and c_2 are (3) and (4).

$$w_{it} = w_{initial} * n^{it}, \tag{3}$$

$$c_{2it} = c_{2initial} / m^{it}, \tag{4}$$

Where *it* is the iteration number, and *n* and *m* are coefficients. The coefficients *n* and *m* can be determined by setting *it* to the maximum value.

The mutation function is an artificial perturbation of the system used to push the algorithm out of stagnation. If the algorithm stagnates in a local minimal solution (false minimum), new swarm particles are generated through mutation to replace the previous swarm. Suppose any solution from the previous iteration should remain. In that case, that solution shall be the global best solution for the next iteration, causing the entire swarm to converge on that solution and a recurring stagnating cycle results; therefore, the mutation probability was set at 100%. Four variables and two end conditions were introduced to train the algorithm to identify a stagnating solution. The abandonment threshold (*E*) is the global minimum solution set at 1e-8. The Fitness error (*e*) is the difference between the current Fitness and the previous Fitness as elaborated in (5), the abandonment counter (*q*) monitors the second differential of Fitness error, and the abandonment limit Q is the upper limit for q. When the second differential of the Fitness error becomes smaller than *E* then the algorithm is assumed to have slowed down. Therefore, the condition in (6) states that when the difference in *e* is less than *E*, *q* begins to count consecutively through every iteration. If the condition in (6) is broken, then counter in *q* is reset to zero.

$$e = Fitness_{it-1} - Fitness_{it}, \tag{5}$$

$$q = \begin{cases} q+1, & if, (e_{it-1} - e_{it}) < E \\ 0, & Else \end{cases}, \tag{6}$$

$$f(MuPSO) = \begin{cases} end & if \quad q \geq Q \quad and \quad Fitness \leq E \\ mutate & if \quad q \geq Q \quad and \quad Fitness > 1e^{-3} \end{cases}, \tag{7}$$

$$X_{mutate} =\sim X_{it} = random[size(X_{it})], \tag{8}$$

The two end conditions in (7) state that when *q* is equal to or greater than the abandonment limit (*Q*) and Fitness is less than *E*, then the algorithm has found the global minimum solution and should be terminated. Still, when the first condition is met and Fitness is greater than 1e-3, then this signifies that the algorithm has run into stagnation and should be mutated according to (8), where new particles are generated. *Q* should be large enough so as not to prematurely terminate a promising solution, allowing the algorithm to break out of stagnation without mutation but must also not be too large to allow a failed solution to continue.

3.0 ROBOT DYNAMICS

Implementing the Linear-least-square (LLS) method for dynamic parameter estimation of the robot manipulator requires that the Newton-Euler equations are linear with respect to dynamic parameters and it is sensitive to noise in the measured data. [16] observed that the actuator forces of an industrial manipulator are linear functions of dynamic parameters, [17] inferred that reformulating the Newton-Euler dynamic model such that the link inertia tensors are expressed about the link coordinate frames instead of the center of the links' mass, would result in a linearized Newton-Euler formulation. It involves two sets of recursive computations; the forward and the backward computations. As shown by equations (9) – (14), the forward recursive computation transforms the kinematic variables into individual joint forces and moments acting on the links, starting at base through the end effector such that the index *i=0,1,2,…,dof-1*. Where the variable *R* is the rotation matrix, the angular velocity and acceleration at the base is zero (i.e. $\omega_0=\dot{\omega}_0=0$), the unit vector $\hat{Z}=[0,\ 0,\ 1]^T$, θ and its integrals signify joint position, velocity and acceleration; *Pc* is the position of center of mass, *P* is the position of the end-effector, the linear acceleration of the base is $\dot{v}_0=[0, 0, gr]$, and gravity $gr=9.8ms^{-1}$. The backward recursive computations presented in (15) and (16) transforms the forces and moments generated in the forward computations into the net joint forces and torques, starting at the end-effector back to the base such that *i=dof,dof-1,…,1*. Where *I* is the inertia tensor, *Fr* is the force acting on the link, *Nm* is the moment acting on the link, *fr* and *nm* are the net force, and net moments (torque) acting on the joint due to the motion of the links.

$$\omega_{i+1} = R_i^{i+1}\left(\omega_i + \hat{Z} * \dot{\theta}_{i+1}\right) \tag{9}$$

$$\dot{\omega}_{i+1} = R_i^{i+1}\left(\dot{\omega}_i + \hat{Z} * \ddot{\theta}_{i+1} + \omega_i \times \hat{Z} * \dot{\theta}_{i+1}\right) \tag{10}$$

$$\dot{v}_{i+1} = R_i^{i+1}\left(\dot{\omega}_i \times P_{i+1} + \omega_i \times \left(\omega_i \times P_{i+1}\right) + \dot{v}_i\right) \tag{11}$$

$$\dot{v}_{c,i} = \dot{v}_i + \dot{\omega}_i \times P_{c,i} + \omega_i \times \left(\omega_i \times P_{c,i}\right) \tag{12}$$

$$Fr_i = m_i \dot{v}_{c,i} \tag{13}$$

$$Nm_i = I_i \dot{\omega}_i + \omega_i \times I_i \omega_i \tag{14}$$

$$fr_i = R_{i+1}^i fr_{i+1} + Fr_i \tag{15}$$

$$nm_i = R_{i+1}^i nm_{i+1} + P_i \times fr_i + P_{c,i} \times Fr_i + Nm_i \tag{16}$$

From the torque model presented in (16), it would be observed that the Newton-Euler model is linear in inertia tensors *I* and nonlinear in center of mass vectors *Pc* and kinematic parameter vectors *P*. Therefore, making the formulation linear in dynamic parameter *Pc,* would result in the entire dynamic model become linear in all dynamic parameters. Therefore, linearized Newton-Euler formulation, the equation (17) presents the parallel axis theorem, transforming the classical inertia tensor from the center of mass of link *i* to the origin of the link, where *eye* is a 3x3 identity matrix. Equation (18) is obtained by substituting (12) – (14) into (16), then implementing the parallel axis theorem in (17) gives the expression in (19). The vector identities in (20) and (21) are introduced to further simplify the dynamic formulation, (22) is obtained by substituting the vector identities in (20) and (21) into (19), where *viA* and *viB* are 3x1 vectors. Eliminating the similar terms in (22) results in the simplified NE dynamic model that is linear in dynamic parameters as expressed in (23).

$$\hat{I}_i = I_i + m_i\left(P_i^T P_i * eye - P_i P_i^T\right) \tag{17}$$

$$nm_i = R_{i+1}^i n_{i+1} + {}^iP_i \times fr_i + m_i\,{}^iP_{c,i}\left(\dot{v}_i + \dot{\omega}_i \times {}^iP_i + \omega_i \times \left(\omega_{i+1} \times {}^iP_{c,i}\right)\right) + I_i\dot{\omega}_i + \omega_i \times I_i\omega_i \tag{18}$$

$$\begin{aligned} nm_i = {} & R_{i+1}^i nm_{i+1} + {}^iP_i \times fr_i + m_i\left({}^iP_{c,i} \times \dot{v}_i\right) + m_i\left({}^iP_{c,i} \times \left(\dot{\omega}_i \times {}^iP_{c,i}\right)\right)\ldots \\ & + m_i\left({}^iP_{c,i} \times \left(\omega_i \times \left(\omega_i \times {}^iP_{c,i}\right)\right)\right) + \hat{I}_i\dot{\omega}_i + \omega_i \times \hat{I}_i\omega_i \ldots \\ & - m_i\left(P_i^T P_i * eye - P_i P_i^T\right)\dot{\omega}_i - \omega_i \times m_i\left(P_i^T P_i * eye - P_i P_i^T\right)\omega_i \end{aligned} \tag{19}$$

$$viA \times \left(viB \times \left(viB \times viA\right)\right) = viB \times \left[viA^T \cdot viA * eye - viA \cdot viA^T\right]viB \tag{20}$$

$$viA \times \left(viB \times viA\right) = \left[viA^T \cdot viA * eye - viA \cdot viA^T\right]viB \tag{21}$$

$$\begin{aligned} nm_i = {} & R_{i+1}^i nm_{i+1} + {}^iP_i \times fr_i + m_i\left({}^iP_{c,i} \times \dot{v}_i\right) + m_i\left(P_i^T P_i * eye - P_i P_i^T\right)\dot{\omega}_i \ldots \\ & + m_i\left(\omega_i \times \left(P_i^T P_i * eye - P_i P_i^T\right)\right)\omega_i + \hat{I}_i\dot{\omega}_i + \omega_i \times \hat{I}_i\omega_i \ldots \\ & - m_i\left(P_i^T P_i * eye - P_i P_i^T\right)\dot{\omega}_i - \omega_i \times m_i\left(P_i^T P_i * eye - P_i P_i^T\right)\omega_i \end{aligned} \tag{22}$$

$$nm_i = R_{i+1}^i nm_{i+1} + {}^iP_i \times fr_i + m_i\left({}^iP_{c,i} \times \dot{v}_i\right) + \hat{I}_i\dot{\omega}_i + \omega_i \times \hat{I}_i\omega_i \tag{23}$$

The inertia tensor matrix is a 3x3 symmetric matrix, therefore, only six out of the nine values are unique as shown in (24). The notation in (25) and (26) presents a simplification of the cross and dot multiplications of kinematic parameters as demonstrated in (27) and (28). They are introduced to simplify the implementation of the dynamic model algorithm, especially for higher DOF robots.

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \approx \begin{bmatrix} I_{xx} & I_{yy} & I_{zz} & I_{xy} & I_{xz} & I_{yz} \end{bmatrix}^T, \tag{24}$$

$$\left[\omega \times\right] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{25}$$

$$\left[\bullet\,\omega_i\right] = \begin{bmatrix} \omega_x & \omega_y & \omega_z & 0 & 0 & 0 \\ 0 & \omega_x & 0 & \omega_y & \omega_z & 0 \\ 0 & 0 & \omega_x & 0 & \omega_y & \omega_z \end{bmatrix}, \tag{26}$$

$$\left[\omega \times\right] \cdot c = \left(\omega_y c_z - \omega_x c_y\right) - \left(\omega_x c_z - \omega_z c_x\right) + \left(\omega_x c_y - \omega_y c_x\right) \tag{27}$$

$$\left[\bullet\,\omega_i\right] \cdot \left[I_{xx} \quad I_{yy} \quad I_{zz} \quad I_{xy} \quad I_{xz} \quad I_{yz}\right]^T = \begin{bmatrix} \omega_x I_{xx} + \omega_y I_{xy} + \omega_z I_{xz} \\ \omega_x I_{xy} + \omega_y I_{yy} + \omega_z I_{yz} \\ \omega_x I_{xz} + \omega_y I_{yz} + \omega_z I_{zz} \end{bmatrix} \tag{28}$$

The formulation for the forward recursive computations can be summarized as (29) – (31). where *wr* is a 6-element wrench vector, ζ is a 6x10 observation matrix of joint parameters, and ϕ is a 10-element vector of unknown dynamic parameters for each link that needs to be identified, it includes link mass *m*, first moments $[mc_x\ mc_y\ mc_z]^T$ along the *x-y-z* axis, and six inertia tensor I_{ij}. Suppose wr_{ii} represents the wrench acting on ith joint due to the net forces and torques generated by the ith link only, then wr_{ij} represents the net forces and torques acting on ith joint due to the net forces and torques generated by the jth link only, and a wrench transformation matrix *Rw* would be required to transform the wrech from the jth frame to the ith frame as described in (32). Equation (33) presents a 6x6 pseudo-rotation matrix for transforming the wrench along frames, where the *R* is a rotation matrix, and [*Px*] is the cross-product matrix of the joint position *P* as derived from (25).

Cascading the linearized dynamic expressions in (34) – (37) results to (38), where *U* is also a 6x10 matrix. Each *wr* in the left-hand side of (38) is a full force-torque vector of a joint. Since only the torque about the joint axis is usually measured, (38) is reduced to (39) where τ_{ij}=[0,0,0,0,0,1]wr_{ij}, and K_{ij}=[0,0,0,0,0,1]U_{ij}. In the new expression, each τ_{ij} is a single variable; each K_{ij} would be a 1x10 vector; therefore, the compound observation matrix would be a 6x10*DOF matrix, this results to an ill conditioned matrix due to loss of rank.

$$wr_{ii} = \zeta_i \cdot \phi_i \tag{29}$$

(24)

$$\begin{bmatrix} fr_{ii} \\ nm_{ii} \end{bmatrix} = \begin{bmatrix} \dot{v}_i & \left[\dot{\omega}_i \times\right] + \left[\omega_i \times\right] \cdot \left[\omega_i \times\right] & 0 \\ \left[P_i x\right]\dot{v}_i & \left[-\dot{v}_i x\right] + \left[P_i x\right]\left[\dot{\omega}_i \times\right] + \left[P_i x\right]\left[\omega_i \times\right]\left[\omega_i \times\right] & \left[\bullet\,\dot{\omega}_i\right] + \left[\omega_i \times\right] \cdot \left[\bullet\,\omega_i\right] \end{bmatrix} \cdot \phi_i \tag{30}$$

(25)

$$\phi_i = \begin{bmatrix} m & mc_x & mc_y & mc_z & I_{XX} & I_{YY} & I_{ZZ} & I_{XY} & I_{XZ} & I_{YZ} \end{bmatrix}^T \tag{31}$$

$$wr_{ij} = {}^iRw_j \cdot wr_{jj} \tag{32}$$

$$^{i}Rw_{j}=\begin{bmatrix} ^{i}R_{j} & 0 \\ [P_{i}\times]\,^{i}R_{j} & ^{i}R_{j} \end{bmatrix} \tag{33}$$

$$wr_{ij}={}^{i}Rw_{i+1}\cdot{}^{i+1}Rw_{i+2}\cdot\ldots\cdot{}^{j-1}Rw_{j}\cdot wr_{jj} \tag{34}$$

$$Let,\quad U_{ij}={}^{i}Rw_{i+1}\cdot{}^{i+1}Rw_{i+2}\cdot\ldots\cdot{}^{j-1}Rw_{j}\cdot\zeta_{j} \tag{35}$$

$$And,\quad U_{ii}=\zeta_{i} \tag{36}$$

$$wr_{ij}=U_{ij}\cdot\phi_{j} \tag{37}$$

$$\begin{bmatrix} wr_{1} \\ wr_{2} \\ wr_{3} \end{bmatrix}=\begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}\cdot\begin{bmatrix} \phi_{1} \\ \phi_{2} \\ \phi_{3} \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} \tau_{1} \\ \tau_{2} \\ \tau_{3} \end{bmatrix}=\begin{bmatrix} K_{11} & K_{12} & K_{13} \\ 0 & K_{22} & K_{23} \\ 0 & 0 & K_{33} \end{bmatrix}\cdot\begin{bmatrix} \phi_{1} \\ \phi_{2} \\ \phi_{3} \end{bmatrix} \tag{39}$$

4.0 PARAMETER ESTIMATION MODEL

4.1 Fitness Function

Let the compound torque (*Γ*) for the manipulator be a vector of individual joint torques *τ*, the compound observation matrix (Ω) comprise of individual joint vectors $K(q,\ \dot{q},\ \ddot{q}\ )$, while the compound vector of unknown parameters *Φ* comprise of individual vector of unknown robot parameters *ϕ* in every link as shown in (40). Substituting (40) in (39) gives (41).

The observation matrix is a function of the joint position, velocity, and acceleration only and its components are computed directly from the measured kinematic parameters, and together with the known torque vector, the unknown dynamic parameters can be estimated using the LLS method as elaborated in (42) and (43). The expression in (43) has six equations and ten unknowns, therefore, the wrench has to be sampled in at least two locations to produce a system of more equations than unknowns. More sampling data points are desirable to account for noise, also implementing the Finite Fourier series (FFS) as an excitation trajectory increases the signal-to-noise ratio. Unfortunately, the observation matrix, cannot be inverted due to loss of rank, this problem can be resolved by removing the unidentifiable parameters from the dynamic model.

$$Let\quad \Gamma=\begin{vmatrix} \tau_{i} \\ \vdots \\ \tau_{dof} \end{vmatrix},\qquad \Phi=\begin{vmatrix} \phi_{i} \\ \vdots \\ \phi_{dof} \end{vmatrix},\quad and\qquad \Omega=\begin{vmatrix} K_{i}(q,\dot{q},\ddot{q}) \\ \vdots \\ K_{dof}(q,\dot{q},\ddot{q}) \end{vmatrix} \tag{40}$$

$$\Gamma=\begin{vmatrix} K_{i}(q,\dot{q},\ddot{q}) \\ \vdots \\ K_{dof}(q,\dot{q},\ddot{q}) \end{vmatrix} * \begin{vmatrix} \phi_{i} \\ \vdots \\ \phi_{dof} \end{vmatrix} \tag{41}$$

$$\Gamma=\left(\left(\Omega^{T}\Omega\right)^{-1}\Omega^{T}\right)^{inv}\Phi \tag{42}$$

$$\Phi=\left(\Omega^{T}\Omega\right)^{-1}\Omega^{T}*\Gamma \tag{43}$$

$$fitness=cond(\Omega)+\frac{1}{\varphi_{\min}(y)}+\varphi_{\max}y \tag{44}$$

The fitness function would be as described by the expression in (44). The intelligent swarm-based algorithm would be required to optimize the parameters of the FFS, and also minimize the condition number of the observation matrix while maximizing its smallest singular value and minimizing its largest singular value. Where φ_{max} and φ_{min} represent the maximum and minimum singular values of the observation matrix respectively, and *y* is a weighting factor.

4.2 Excitation Trajectory

It was observed by [18] that the excitation trajectory can be optimized by minimizing the condition number of the observation matrix, an optimized excitation trajectory improves the estimation accuracy. The condition number of the observation matrix is a measure of noise immunity, a value closer to unity results in a better signal-to-noise ratio (SNR).

The FFS has computational advantages when implemented for designing an excitation trajectory, it inhibits noise interference. If the identification experiment is repeated several times, averaging the measured data in the time domain improves SNR. Reference [19], used the periodic FFS to formulate an excitation trajectory at the joint space of the manipulator. Reference [20] proposed an improved Fourier series and [21] implemented the FFS on a circular trajectory in the Cartesian space. The equation of motion is given in (45) – (47), the MuPSO algorithm was used to optimize the parameters of the Fourier series thereby minimizing the condition number of the observation matrix.

$$Q_{x}(it)=\sum_{k=1}^{N}\left(\cos(P_{it})*a0_{k}^{it}+\frac{b0_{k}^{it}}{freq*k}\sin\left(freq*k*tym\right)-\frac{c0_{k}^{it}}{freq*k}\cos\left(freq*k*tym\right)\right) \tag{45}$$

$$Q_y(it)=\sum_{k=1}^{N}\left(\sin(P_{it})*a0_k^{it}+\frac{b0_k^{it}}{freq*k}\sin\left(freq*k*tym\right)-\frac{c0_k^{it}}{freq*k}\cos\left(freq*k*tym\right)\right) \quad (46)$$

$$Q_z(it)=\sum_{k=1}^{N}\left(\frac{b0_k^{it}}{freq*k}\sin\left(freq*k*tym\right)-\frac{c0_k^{it}}{freq*k}\cos\left(freq*k*tym\right)\right) \quad (47)$$

$$\left.\begin{array}{l} \min \ \ cond(\Omega) \quad such \ \ that; \\ \quad if \quad Q=\left[Q_x,Q_y,Q_z\right] \\ \qquad Q_{coordinate}=\left[q_1,q_2,\Lambda\ q_n\right]^T \\ \qquad \forall Q,q\in WorkSpace \\ \qquad 0\le q(it)\le 2T \\ \qquad \dot{q}_{initial}=\dot{q}_{final}=0 \\ \qquad \ddot{q}_{initial}=\ddot{q}_{final}=0 \end{array}\right\} \quad (48)$$

The variable Q describes the coordinates of the excitation trajectory in Cartesian space, the variable P is the polynomial, the variables c_0 and b_0 are the coefficients of the FFS, a_0 is the radius. *freq* is the fundamental frequency and *tym* is the period. To achieve continuity along the tool-path, the boundary conditions for this analysis is such that for every coordinate of Q in the Cartesian coordinate frame, $Q=[q_1,q_2,...,q_n]^T$, and q must not exceed the robot's work space. Where the initial and final values for q are 0 and 2π; likewise, the initial and final values for the velocity and acceleration of joints is zero as detailed in (48).

5.0 Simulation Experiments and Results

An experiment was first performed to find the optimal swarm size for minimizing the condition number of the observation matrix using PSO. the algorithm was run for 200 iterations and 30 generations. The experiment was performed with MATLAB R2016a installed on a Windows 7 (64bit) operating system, i5-7400 CPU running at 3GHz and 8GB of ram. Figure 2.0 shows the best solutions obtained for various swarm sizes and the required computational time. The *Best-cost* of 4.6 was achieved with a swarm size of 70, and a B*est-Cost* of 5.2 was obtained when the swarm size was 50, while the increase in swarm size from 50 to 70 resulted to an increase in the average computational time per generation from 1.12e^4 seconds to 1.57e^4 seconds. It can be summarized that a 13% decrease in best cost was achieved at 70 particles with a 40% increase in computation time therefore, the optimal swarm size was taken as 50 particles owing to its computational advantage.

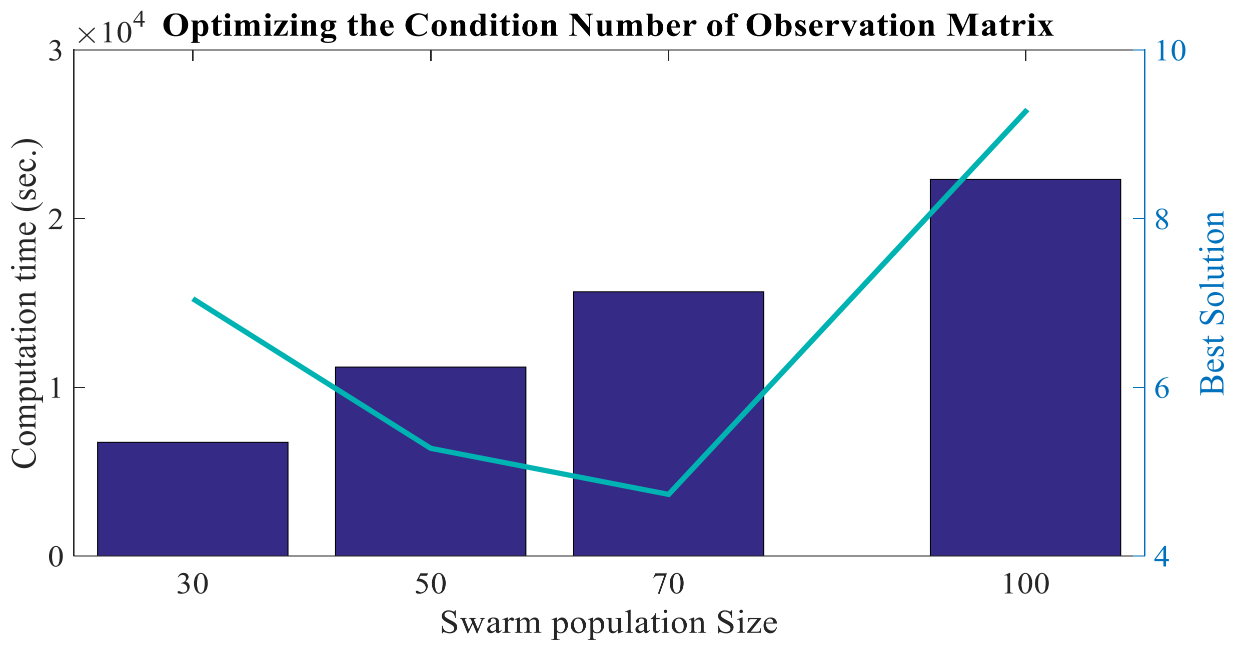

Figure 2: Comparing the computation time and minimum solution for various populations

Table 2: Initial parameters for metaheuristic algorithms.

Algorithm	*Parameter*	*Value*
SPSO	Inertia weight/Velocity coefficient(c_2 and c_2)	0.72/1.45
WOA	Velocity coefficient (a)	2 - 0
GWO	Velocity coefficient (a)	2 - 0
GOA	Exploration Coefficient (c_{Max})/Exploitation Coefficient (c_{Min})	1/4e-5
DE	Inertia weight (F)/Cross-over Probability (CR)	0.85/1

$$\mathrm{Mm} = \begin{bmatrix} \mathrm{m1} \\ \mathrm{m2} \\ \mathrm{m3} \end{bmatrix} = \begin{bmatrix} 5.029912009 \\ 4.530794148 \\ 3.056783430 \end{bmatrix}, \tag{49}$$

$$\mathrm{Mc} = \begin{bmatrix} \mathrm{mcx1} & \mathrm{mcy1} & \mathrm{mcz1} \\ \mathrm{mcx2} & \mathrm{mcy2} & \mathrm{mcz2} \\ \mathrm{mcx3} & \mathrm{mcy3} & \mathrm{mcz3} \end{bmatrix} = \begin{bmatrix} 4.3956 & -15.8576 & 10.1804 \\ -44.1094 & 93.9465 & 183.7506 \\ 0.37299 & 0.001086 & 100.6463 \end{bmatrix}, \tag{50}$$

$$\mathrm{I}_1 = \begin{bmatrix} \mathrm{Ixx1} & Ixy1 & Ixz1 \\ Ixy1 & Iyy1 & Iyz1 \\ Ixz1 & Iyz1 & Izz1 \end{bmatrix} = \begin{bmatrix} 20975.4835 & 0 & 0 \\ 0 & 13651.6206 & 0 \\ 0 & 0 & 19154.2091 \end{bmatrix}, \tag{51}$$

$$\mathrm{I}_2 = \begin{bmatrix} \mathrm{Ixx2} & Ixy2 & Ixz2 \\ Ixy2 & Iyy2 & Iyz2 \\ Ixz2 & Iyz2 & Izz2 \end{bmatrix} = \begin{bmatrix} 103901.5055 & 0 & 0 \\ 0 & 79173.3684 & 0 \\ 0 & 0 & 30809.7396 \end{bmatrix}, \tag{52}$$

$$\mathrm{I}_3 = \begin{bmatrix} \mathrm{Ixx3} & Ixy3 & Ixz3 \\ Ixy3 & Iyy3 & Iyz3 \\ Ixz3 & Iyz3 & Izz3 \end{bmatrix} = \begin{bmatrix} 9855.9631 & 0 & 0 \\ 0 & 9759.3916 & 0 \\ 0 & 0 & 7360.3570 \end{bmatrix}, \tag{53}$$

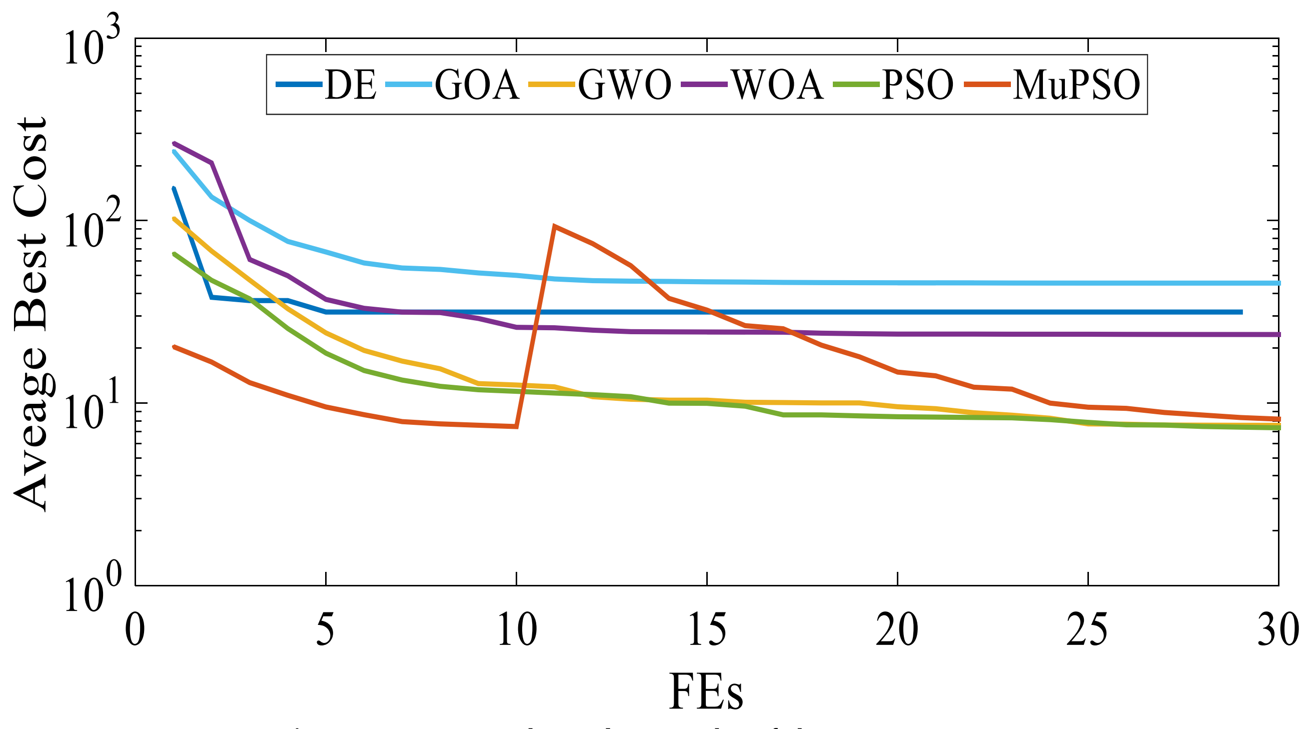

Figure 3: Minimizing the condition number of observation matrix

Table 3: Minimized condition number of optimization matrix for various metaheuristics.

Function		**Evolutionary Algorithms**					
		MuPSO	***SPSO***	***WOA***	***GWO***	***GOA***	***DE***
Condition Number for Excitation trajectory	*Minimum*	4.73	6.27	6.05	5.19	6.72	10.0
	Average	7.16	7.33	23.8	7.56	45.5	31.6
	St. Dev.	1.17	0.911	12.5	3.09	63.6	17.2
	Time (s)	7.15e3	7.43e3	1.09e4	1.46e4	6.78e3	1.36e4

A swarm of 50 particles was generated to run for only 30 iterations. The fitness function for this swarm is evaluated according to (44). The abandonment limit (Q) was set at 10, so that the swarm is mutated at 10 iterations after stagnation (i.e. Q=10).

The results presented in Table 3 show that the MuPSO best minimized the problem and produced better averages compared to other algorithms. Figure 3 shows the average performance of all metaheuristics. Observe that the mutating PSO algorithm was mutated at the 10th iteration and the best solution was obtained before mutation.

The optimized coefficients of the FFS for the excitation trajectory obtained from the MuPSO are; $a = [-0.3392, 1.1738, -0.20492]^T$, $b = [0.3305, -0.26181, -0.9965]^T$ and $c = 2.0328$. A minimized condition number of 7.3531 was obtained for the observation matrix. Figure 4 shows the resultant trajectory in Cartesian space, while Figure 5 shows the velocity and acceleration of the joints. The solution from the MuPSO and LLS was implemented for dynamic parameter estimation, it was observed that 20 parameters were independently identifiable, 2 were identifiable in linear combinations while 8 were unidentifiable.

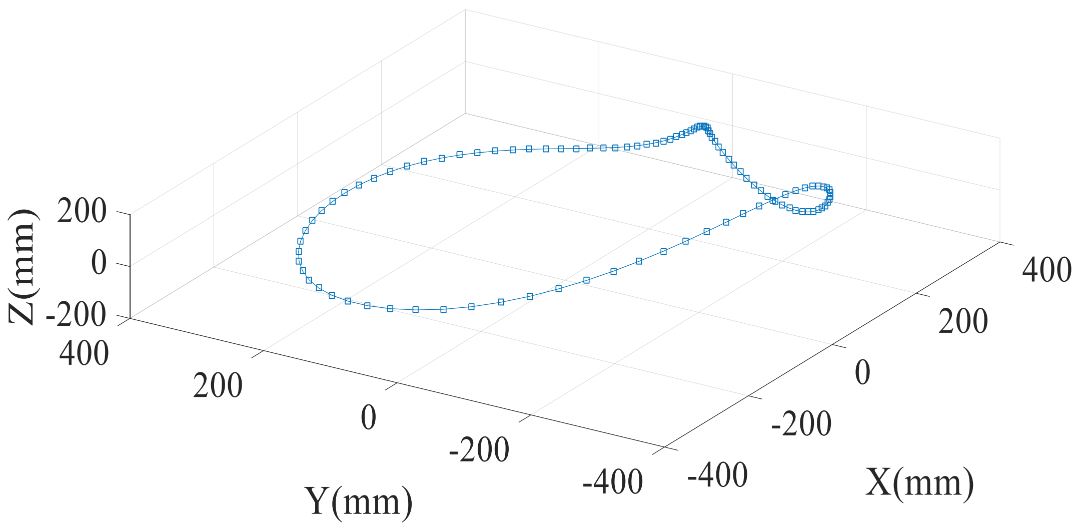

Figure 4: Optimized excitation Trajectory in Cartesian space.

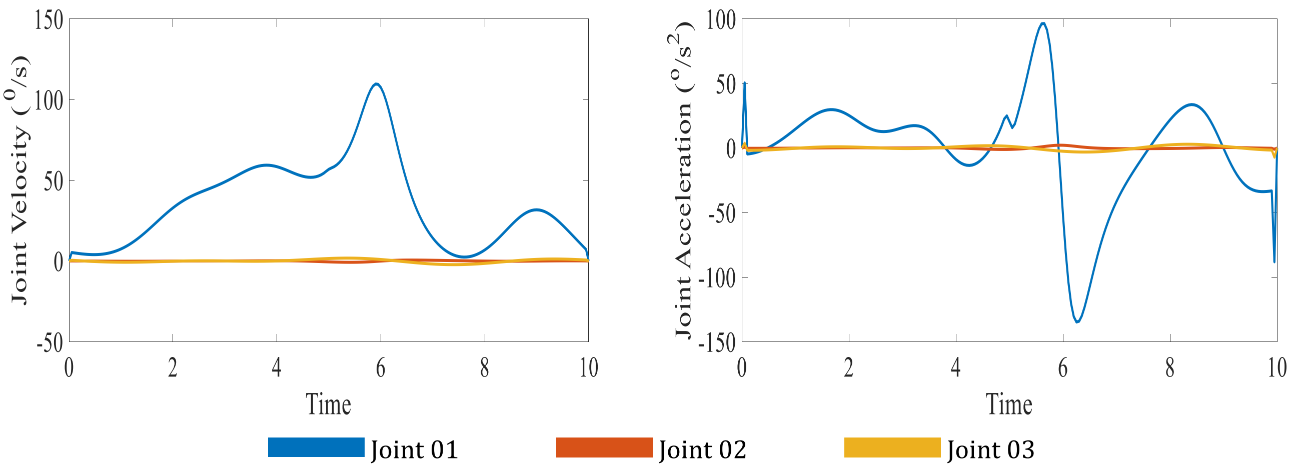

Figure 5: Optimized excitation Trajectory in joint space.

The 22 identified parameters include *mcy1*, *Iyy1*, *mm2*, *mcx2*, *mcy2*, *mcz2*, *Iyy2*, *Izz2*, *Ixz2*, *Iyz2*, *mm3*, *mcx3*, *mcy3*, *mcz3*, *Ixx3*, *Iyy3*, *Izz3*, *Ixy3*, *Ixz3*, *Iyz3*, *Ixz1*, and *Ixx2*. Table 4 presents the identified dynamic parameters against their corresponding ideal values.

6.0 CONCLUSION

The parameter identification problem of robot manipulators is generally computationally demanding, therefore finding a fast, computationally efficient, and accurate solution is important for which swarm-based algorithms are suitable.

Table 4: Dynamic parameter estimation for an Industrial robot.

Dynamic Variables	Ideal Values	Identified Values	Dynamic Variables	Ideal Values	Identified Values	Dynamic Variables	Ideal Values	Identified Values
my1	0	-1.08E-06	Ixz2	0	-4.86E-07	Izz3	7360	7360
Iyy1	13651	13651	Iyz2	0	0.000491	Ixy3	0	-3.22E-05
m2	4	4	m3	3	3	Ixz3	0	-1.24E-06
mx2	350	350	mx3	60	60	Iyz3	0	-3.24E-05
my2	0	-6.83E-09	my3	0	1.77E-09	Ixz1	0	-0.00079
mz2	0	-5.64E-08	mz3	0	-4.72E-08	Ixx2	103901	103901
Iyy2	79173	79173	Ixx3	9855	9855			
Izz2	30809	30809	Iyy3	9759	9759			

A mutation PSO algorithm with enhanced parameters (MuPSO) was shown to be capable of analysing the dynamic problems of robot manipulators, performing better than other algorithms in minimizing the condition number of the observation matrix achieving between (2 – 500) % gain in the minimum solution against other algorithms, and requiring the second least computational time with a 5.46% increase in computational time against the GWO algorithm and between (3 – 90) % reduction in computational time against the other algorithms.

Observing from the average convergence plots presented in Figure 3 that the best solution of the MuPSO algorithm was obtained early in the run signifies that the algorithm permits the swarms to be populated with less particles thereby reducing the number of computations.

REFERENCES

[1] Lange, D. “Cognitive robotics: Making robots sense, understand, and interact,” *Computer (Long. Beach. Calif)*, 52(12), (2019), 39–44, doi: 10.1109/MC.2019.2942579.

[2] Carvalho, N., Chaim, O., Cazarini, E. and Gerolamo, M. "Manufacturing in the fourth industrial revolution: A positive prospect in sustainable manufacturing." *Procedia Manufacturing*, 21, (2018), 671-678 doi: https://doi.org/10.1016/j.promfg.2018.02.170.

[3] Krinitsyn, N.S., Babaev, A.S., Stolov, E.V., Laptev, N.V., Pivkin, P.M. and Khisamutdinov, R.M. "Precision in Single-Coordinate Positioning of an Industrial Robot Manipulator." *Russian Engineering Research*, 40(1), (2020), 83-85. doi: 10.3103/S1068798X20010116.

[4] Jensen, K., Larsen, M., Nielsen, S.H., Larsen, L.B., Olsen, K.S. and Jørgensen, R.N. "Towards an open software platform for field robots in precision agriculture." *Robotics*, 3(2), (2014), 207-234. doi: 10.3390/robotics3020207.

[5] Ben-Ari, M. and Mondada, F. "Robots and their applications." *Elements of robotics*. Springer, Cham, (2018), 1-20.

[6] Klimchik, A., Ambiehl, A., Garnier, S., Furet, B. and Pashkevich, A. "Efficiency evaluation of robots in machining applications using industrial performance measure." *Robotics and Computer-Integrated Manufacturing*, 48, (2017), 12-29. https://doi.org/10.1016/j.rcim.2016.12.005.

[7] Zargarbashi, S.H.H., Khan, W. and Angeles, J. "The Jacobian condition number as a dexterity index in 6R machining robots." *Robotics and Computer-Integrated Manufacturing*, 28(6), (2012), 694-699. doi: 10.1016/j.rcim.2012.04.004.

[8] Zargarbashi, S.H.H., Khan, W. and Angeles, J. "Posture optimization in robot-assisted machining operations." *Mechanism and Machine Theory*, 51, (2012), 74-86. doi: 10.1016/j.mechmachtheory.2011.11.017.

[9] Ding, L., Wu, H., Yao, Y. and Yang, Y. "Dynamic model identification for 6-DOF industrial robots." *Journal of Robotics*, (2015). doi: 10.1155/2015/471478.

[10] Hayat, A.A., Boby, R.A. and Saha, S.K. "A geometric approach for kinematic identification of an industrial robot using a monocular camera." *Robotics and computer-integrated manufacturing*, 57, (2019), 329-346. doi: 10.1016/j.rcim.2018.11.008.

[11] Yan, D., Lu, Y. and Levy, D. "Parameter identification of robot manipulators: A heuristic particle swarm search approach." *PloS one*, 10(6), (2015), e0129157 doi:

10.1371/journal.pone.0129157.

[12] Gautier, M. and Poignet, P. "Extended Kalman filtering and weighted least squares dynamic identification of robot." *Control Engineering Practice,* 9(12), (2001), 1361-1372. doi: 10.1016/S0967-0661(01)00105-8.

[13] Fei, F., Hongjie, H. and Zhongtong, G. "Application of genetic algorithm PSO in parameter identification of SCARA robot." *2017 Chinese Automation Congress (CAC)*. IEEE, 2017. doi: 10.1109/CAC.2017.8242898.

[14] Mizuno, N. and Nguyen, C.H. "Parameters identification of robot manipulator based on particle swarm optimization." *2017 13th IEEE International Conference on Control & Automation (ICCA)*. IEEE, 2017, 307–312, doi: 10.1109/ICCA.2017.8003078.

[15] Umar, A., Shi, Z., Khlil, A. and Farouk, Z.I. "Developing a new robust swarm-based algorithm for robot analysis." *Mathematics* 8.2 (2020): 158. doi: 10.3390/math8020158.

[16] Atkeson, C.G., An, C.H. and Hollerbach, J.M. "Estimation of inertial parameters of manipulator loads and links." *The International Journal of Robotics Research,* 5(3), (1986), 101-119. doi: doi.org/10.1177/027836498600500306.

[17] Khosla, P. K. "Categorization of parameters in the dynamic robot model." *IEEE Transactions on Robotics and Automation,* 5(3), (1989), 261-268.

[18] Armstrong, B. "On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics." *The International journal of robotics research,* 8(6), (1989), 28-48. doi: 10.1177/027836498900800603.

[19] Swevers, J., Ganseman, C., De Schutter, J. and Van Brussel, H. "Experimental robot identification using optimised periodic trajectories." *Mechanical Systems and Signal Processing,* 10(5), (1996), 561-577. doi: 10.1006/mssp.1996.0039.

[20] Wu, W.X., Zhu, S.Q. and Jin, X.L. "Dynamic identification for robot manipulators based on modified fourier series." *Journal of ZheJiang University (Engineering Science,)* 47(2), (2013), 231-237. doi: 10.3785/j.issn.1008-973X.2013.02.006.

[21] Wu, W., Zhu, S., Wang, X. and Liu, H. "Closed-loop dynamic parameter identification of robot manipulators using modified fourier series." *International Journal of Advanced Robotic Systems,* 9(1), (2012), 29. doi: 10.5772/45818.

[22] Kennedy, J. and Eberhart, R. “Particle Swarm Optimization,” in *Proceedings of the 1995 IEEE International Conference on Neural Networks. Part 1 (of 6),* Apr. 1995, 4, 1942–1948.

[23] Mirjalili, S. and Lewis, A. "The whale optimization algorithm." *Advances in engineering software,* 95, (2016), 51-67. doi: 10.1016/j.advengsoft.2016.01.008.

[24] Zorarpacı, E. and Özel, S.A. "A hybrid approach of differential evolution and artificial bee colony for feature selection." *Expert Systems with Applications,* 62, (2016), 91-103. doi: 10.1016/j.advengsoft.2013.12.007.

[25] Saremi, S., Mirjalili, S. and Lewis, A. “Grasshopper optimisation algorithm: theory and application." *Advances in Engineering Software,* 105, (2017), 30-47. doi: 10.1016/j.advengsoft.2017.01.004.

[26] Storn, R. "Differential evolution design of an IIR-filter." In *Proceedings of IEEE international conference on evolutionary computation*, (1996),268–273. doi: 10.1109/icec.1996.542373.