JOURNAL OF MECHANICAL ENGINEERING AND SCIENCES (JMES)
ISSN: 2289-4659 e-ISSN: 2231-8380
VOL. 15, ISSUE 2, 7948 – 7963
DOI: https://doi.org/10.15282/jmes.15.2.2021.02.0627

ORIGINAL ARTICLE

Novel use of the Monte-Carlo methods to visualize singularity configurations in serial manipulators

M. I. M. Abo Elnasr [1,*], Hussein M. Bahaa[1] and Ossama Mokhiamar[2]

[1] Department of Electro-Mechanical Engineering, Faculty of Engineering, Alexandria University, El-Chatby, Alexandria 21544, Egypt
[2] Mechanical Engineering Department, Faculty of Engineering, Alexandria University, El-Chatby, Alexandria 21544, Egypt

ABSTRACT – This paper analyses the problem of the kinematic singularity of 6 DOF serial robots by extending the use of Monte-Carlo numerical methods to visualize singularity configurations. To achieve this goal, first, forward kinematics and D-H parameters have been derived for the manipulator. Second, the derived equations are used to generate and visualize a workspace that gives a good intuition of the motion shape of the manipulator. Third, the Jacobian matrix is computed using graphical methods, aiming to locate positions that cause singularity. Finally, the data obtained are processed in order to visualize the singularity and to design a trajectory free of singularity. MATLAB robotics toolbox, Symbolic toolbox, and curve fitting toolbox are the MATLAB toolboxes used in the calculations. The results of the surface and contour plots of the determinate of the Jacobian matrix behavior lead to design a manipulator's trajectory free of singularity and show the parameters that affect the manipulator's singularity and its behavior in the workspace.

ARTICLE HISTORY
Received: 11th May 2020
Revised: 09th Dec 2020
Accepted: 01st Feb 2021

KEYWORDS
Singularity analysis; kinematic modeling; Monte-Carlo methods; 6 DOF serial robotic arm; forward kinematics

INTRODUCTION

Kinematic singularity is one of the critical topics in the field of robotics but nonetheless no manipulator architecture does not suffer from the singularity problem. Kinematics singularities represent configurations in which the manipulator number of instantaneous degrees of freedom are reduced [1]. This reduction will limit the capabilities of the manipulator and won't be possible to execute arbitrary motions of the end-effector. The presence of singularities in the manipulator's workspace or joint space can profoundly affect the overall performance of the manipulator. Upon approaching a singularity configuration, small velocities in the Cartesian space may generate infinite velocities in the joint space. Resulting in illogical torques or forces on the links of the manipulator, loss of stiffness and rigidity, and malfunction of control algorithms. Therefore, the kinematic singularity is a vital problem that should be heavily considered in the designing and controlling process of robotic manipulators.

Traditionally, the singularity analysis is addressed by checking the manipulability index of the system in question and was introduced by Yoshikawa [2]. The manipulability concept is to identify the linear dependencies in the Jacobian that may cause a singular configuration. The configuration is to be singular if the Jacobian matrix is out of rank. This method was initially used in control algorithms to avoid singular positions. Since then it has been used in a wide variety of applications, such as real-time end-pose planning in walking tasks [3], and planning of human-robot interaction workspaces [4]. The manipulability index has also been extended to include the joint twist limits, self-collision, and the ability to adapt to obstacles in the working area [5].

The singularity analysis could also be approached using the mathematical point-of-view [4]. The configuration is to be singular if the Jacobian matrix ($6 \times n$) is out of rank (i.e. the configurations at which the Jacobian matrix is rank-deficient). These configurations are named Kinematic Singularities [6]. Subsequently, the general solution of the determinant of the Jacobian should be obtained to identify the locus of the singular configurations. Those singular configurations have then been used to generate a singularity free path [1].

Another way to reduce the number of singular configurations is by introducing a shoulder joint. This was firstly introduced by Chiaverini [7]. He proposed to introduce a shoulder joint to the Puma configuration 5 degree-of-freedom (5 DOF) serial robot configuration) to increase the number of degrees of freedom (6 DOF). This helped in decreasing the number of singular configurations but it couldn't eliminate the singularity problem.

Another approach to singularity is the use of dynamic equations as presented in [8, 9] where the forward dynamics approach was used for task space control. The Singularity conditions were detected based on the estimation of the energy stored. Also in [10], the definitions of force polytope and force ellipsoid were studied with the static constraints for planner manipulators.

The Monte Carlo method is a numerical method for solving mathematical problems by means of random sampling. The Monte-Carlo method does not require inverse Jacobian calculation and it is relatively simple to apply. It has been used by many researchers to generate workspace and the corresponding size, the accessibility of workspaces, and the effect of changes in geometric parameters on the workspace [11, 12, 13].

*CORRESPONDING AUTHOR | M. I. M. Abo Elnasr | ✉ aboelnasr1997@gmail.com

The Monte-Carlo methods were only used to visualize the workspace not to figure out the singularity configurations. Therefore, this paper aims to use the Monte-Carlo methods to figure out the locations of the singular configurations in the whole workspace of the 6 DOF serial manipulator. This can be achieved through (a) construct the transformation matrices that describe the kinematic motion of the serial manipulator based on the D-H convention [14]; (b) simulate the workspace of the manipulator using the Monte-Carlo methods [11] depending on the Kinematic equations that were computed, (c) calculate the Jacobian matrix geometrically [6, 15] (hence the Jacobian matrix is available), (d) compute the determinate of the Jacobian matrix, (e) identify the locations of the singularity to simulate the behavior of the Jacobian matrix with the change in the variable affecting it and (f) use the Monte-Carlo methods again to simulate the shape of the singular workspace. The results show which of the joints values affect the singularity of the manipulator and how to generate the singularity-free workspace. Besides, the results give a map to the motion of the end-effector in the Cartesian space and the corresponding value of the Jacobian matrix determinate. This map gives a good intuition of how close the end-effector is to the region of singularity. Consequently, avoiding unreasonable velocities in the joint variable space.

STRUCTURE OF THE AER-1 SERIAL ROBOT

The robotic arm architecture used in this work is a six-degree serial manipulator called the AER-1 robotic arm as shown in Figure 1(a). The configuration of the robot consists of the waist, shoulder, elbow, and wrist. Each of them has one degree of freedom, except for the wrist, which has a spherical configuration (Roll-Pitch-Yaw) that helps in the manipulation and handling of objects. The AER-1 manipulator is fully driven by high precision stepper motor equipped with incremental magnetic encoders to form a reliable closed-loop system.

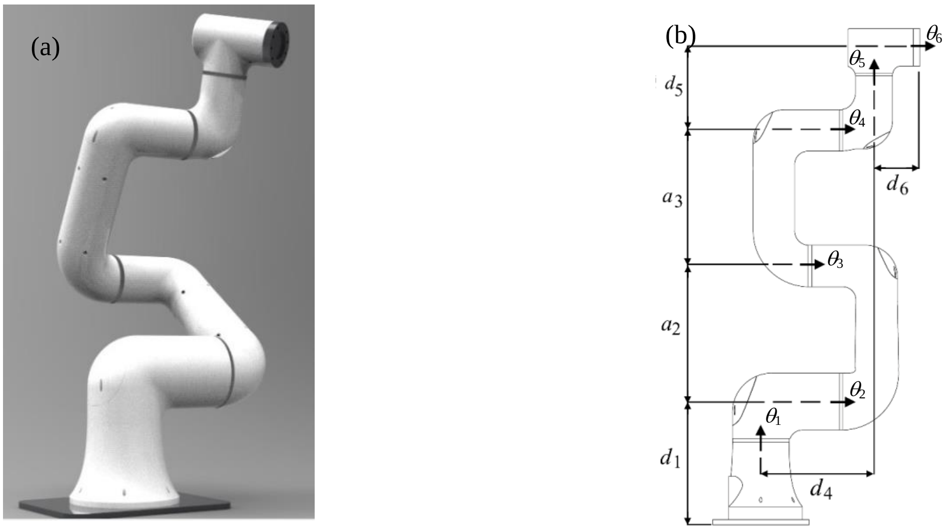

Figure 1. Robotic arm AER-1 architecture: (a) 3D model of the robotic arm and (b) distances between joint's axis

The architecture of the robot consists of six revolute joints as shown in Figure 1(b). In the calculations, the robot structure will be divided into two main parts (the waist, shoulder, and elbow) representing the first three links and the wrist consisting of the last three links as shown in Figure 2(a) and (b) for ease of the analysis [16, 17]. The wrist configuration is Roll-Pitch-Yaw as shown in Figure 2(b), which provides the shortest path to the desired target than the Roll-Pitch-Roll configuration as stated in [17, 18]. As a result, a shorter path generates a chance of having fewer points of singularity in the desired path, which reduces the probability of having a singular point in the generated trajectory.

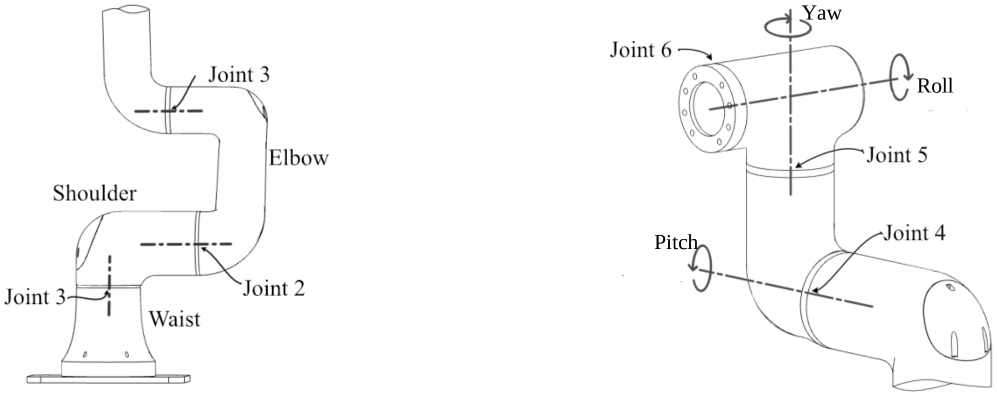

Figure 2. Robot configuration: (a) shoulder configuration and (b) wrist configuration (Roll-Pitch-Yaw

KINEMATIC MODELING

D-H Parameters of the AER-1

The kinematic model of the manipulator is the mathematical relationship between the orientation and position of the end effector and the parameters of the joint. There is more than one method for performing the kinematic problem, the most common methods are based on the Denavit–Hartenberg (DH) parameters and the successive screw displacements. Both methods have a systematic nature and are suitable for modeling serial manipulators.

In this paper, the identification of the joints and the links are done through Denavit–Hartenberg method. The DH parameters are assigned to determine the position and orientation of the robotic arm end-effector [16] and also used to represent how each link reference frame is attached to the robotic arm as illustrated in Figure 3 [6].

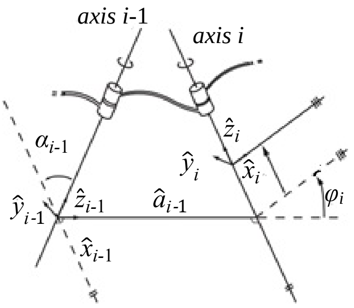

Figure 3. Illustration of D-H convention [19]

Table 1. D-H parameters table

Link	a	α	d	θ
1	0	$\frac{\pi}{2}$	d_1	θ_1
2	a_2	0	0	$\theta_2+\left(\frac{\pi}{2}\right)$
3	a_3	0	d_3	θ_3
4	0	$-\frac{\pi}{2}$	d_4	$\theta_4+\left(\frac{\pi}{2}\right)$
5	0	$\frac{\pi}{2}$	d_5	$\theta_5+\left(\frac{\pi}{2}\right)$
6	0	0	d_6	θ_6

Table 1 shows the DH parameters of the six joints of the AER-1 robotic arm shown in Figure 1(b), where:

α_i : the twist angle between $\hat{z}_{\mathrm{i}}$ & $\hat{z}_{\mathrm{i+1}}$measured about $\hat{z}_{\mathrm{i}}$

a_i: it is the offset length of the common normal between each two successive links

θ_i : it is the value of the angle between $\hat{x}_i$ & $\hat{x}_{i\text{-}1}$ along $\hat{z}_i$

d_i : offset distance between $\hat{x}_i$ & $\hat{x}_{i\text{-}1}$ along $\hat{z}_i$

The main usage of the D-H table is to construct the kinematic equations in order to simulate the entire workspace of the manipulator.

Forward Kinematic of the AER-1

Forward kinematics refers to the process of obtaining the position of the end effector, given the known joint angles. Using parameters from the Denavit-Hartenberg (DH) table, the transformation matrix of each link is then computed where it has a rotation sub-matrix R(3×3), and a translation sub-matrix P(3×1), as in Eq. (1), where i is the link number.

$$ {}^{i\text{-}1}T_i=\begin{bmatrix} cos\theta_i & \text{-}sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\alpha_i \\ sin\theta_i & cos\theta_i cos\alpha_i & \text{-}cos\theta_i sin\alpha_i & a_i sin\alpha_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} \tag{1} $$

By substituting in the general transformation matrix presented in Eq. (1), with the data presented in Table 1, and using the MATLAB Robotics toolbox [20] the transformation matrices of the six links of the AER-1 can be written as in Eqs. (2) to (7).

$$ {}^{0}T_1=\begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & \text{-}c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$ {}^{1}T_2=\begin{bmatrix} c_2 & \text{-}s_2 & 0 & a_2{*}c_2 \\ s_2 & c_1 & 0 & a_2{*}s_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^{2}T_3=\begin{bmatrix} c_3 & \text{-}s_3 & 0 & a_3{*}c_3 \\ s_3 & c_3 & 0 & a_3{*}s_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

$$ {}^{3}T_4=\begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & \text{-}c_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

$$ {}^{4}T_5=\begin{bmatrix} c_5 & 0 & \text{-}s_5 & 0 \\ s_5 & 0 & c_5 & 0 \\ 0 & \text{-}1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

$$ {}^{5}\mathrm{T}_6=\begin{bmatrix} c_6 & 0 & s_6 & 0 \\ s_6 & 0 & \text{-}c_6 & 0 \\ 0 & 1 & 0 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7} $$

where, $\sin(\theta_i)=\mathrm{s_i}$, $\cos(\theta_i)=\mathrm{c_i}$.

Then the transformation matrix from the base to the end effector (on the left-hand side) is computed by multiplying all of the previous matrices presented above in order (on the right-hand side) as follows:

$$ {}^{0}T_{\mathrm{n}}={}^{0}T_1\ldots\ldots\ldots{}^{n\text{-}1}T_n \tag{8} $$

Upon segmenting the manipulator into two main segments as shown in Figure 2 the transformation matrix for each segment can be written as Eqs. (9) and (10).

$$ {}^{0}T_3=T_{shoulder}=\begin{bmatrix} c_{23}c_1 & \text{-}s_{23}c_1 & \text{-}s_1 & c_1(a_3c_{23}+a_2c_2) \\ c_{23}s_1 & \text{-}s_{23}s_1 & c_1 & s_1(a_3c_{23}+a_2c_2) \\ s_{23} & c_{23} & 0 & d_1+a_3s_{23}+a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9} $$

$$ {}^{4}T_6=T_{wrist}=\begin{bmatrix} c_4c_5c_6\text{-}s_4s_6 & \text{-}c_6s_4\text{-}c_4c_5s_6 & \text{-}c_4s_5 & d_5s_4\text{-}d_6c_4s_5 \\ s_4s_6+c_5c_5s_4 & c_4c_6\text{-}c_5s_4\mathrm{s}_6 & \text{-}s_4s_5 & \text{-}d_5c_4\text{-}d_6s_4s_5 \\ s_5c_6 & \text{-}s_5s_6 & c_5 & d_4+d_6c_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10} $$

where, $sin(\theta_i+\theta_j)=s_{ij}$, $cos(\theta_i+\theta_j)=c_{ij}$, where θ_i and θ_j are two different joints values for two different links i and j.

The resulting overall transformation matrix (rotation matrix ${}^{1}R_6$ and translation matrix ${}^{1}P_6$ from the multiplication of the previous matrices presented in Eqs. (9) and (10) will be Eqs. (11) and (12):

$$ {}^{1}R_6=\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \\ 0 & 0 & 0 \end{bmatrix}=\begin{bmatrix} c_{234}c_1c_5c_6\text{-}c_6s_1s_5\text{-}s_{234}c_1s_6 & s_1s_5s_6\text{-}s_{234}c_1c_6\text{-}c_{234}c_1c_5s_6 & c_5s_1+c_{234}s_5c_1 \\ c_1c_6s_5\text{-}s_{234}s_1s_6\text{-}c_{234}s_1c_5c_6 & \text{-}(s_{234}s_1c_6+c_1s_6s_5+c_{234}c_5s_1s_6) & c_{234}s_1s_5\text{-}c_1c_5 \\ c_{234}s_6\text{-}s_{234}c_5c_6 & c_{234}c_6\text{-}s_{234}c_5s_6 & s_{234}s_5 \\ 0 & 0 & 0 \end{bmatrix} \tag{11} $$

$$ {}^{1}P_6=\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}=\begin{bmatrix} a_2c_2c_1+a_3c_1c_{23}+s_1(d_2+d_3+d_4)+d_6(s_1c_5+c_{234}c_1s_5)\text{-}d_5s_{234}c_1 \\ a_2c_2s_1\text{-}s_1(d_2+d_3+d_4)\text{-}d_5c_{234}s_1+a_3s_1c_{23}+d_6(c_{234}s_1s_5\text{-}c_1c_5) \\ d_1+a_3s_{23}+a_2\mathrm{s}_2+d_5\mathrm{c}_{234}+d_6c_{234}s_5 \\ 1 \end{bmatrix} \tag{12} $$

WORKSPACE OF THE AER-1

Following the kinematics of the AER-1 robotic arm presented in the previous section, this section is intended to identify the entire workspace in which the robotic manipulator can operate. The workspace of the robot manipulator is defined as the set of points that can be reached by the end-effector of the robot [11]. The entire workspace will be obtained on the basis of numerical methods that have been developed in recent years. For this purpose, the Monte-Carlo numerical methods are used by applying random values for the joint values between the upper and lower bounds of the joint using the algorithm implemented in [12, 16]. One advantage of the Monte-Carlo method is that it does not require the inverse kinematics solution of the manipulator. The assumption that the manipulator singularity configurations are located upon approaching the boundaries of its workspace was validated in [21]. This emphasizes the importance of finding the workspace of the manipulator.

Joints Limits and Variables Values

The limits of the AER-1 robot are defined in such a way that the links do not collide from their motion in the workspace and also use the maximum space for the end effector to move within it. For the current manipulator design the limits of links 1, 2, 4, 5, and 6 are expressed in Eq. (13). On the other hand, the limits of link 3 are in Eq. (14).

$$-180 \leq q_{1,2,4,5,6} \leq 180 \tag{13}$$

$$-140 \leq q_3 \leq 140 \tag{14}$$

The values of links are chosen to achieve a workspace of a radius of 1m maintaining a reasonable payload on the account of the available driving system. The dimensions of the six links are therefore chosen as presented in Table 2.

Table 2. Dimensions of the six links

Variable	Value (m)
d_1	0.08
a_2	0.41
a_2	0.41
d_4	0.15
d_5	0.15
d_6	0.1

Monte Carlo Algorithm

After discussing the joint limits and variables of the AER-1 in the previous section, this section explains the use of the Monte-Carlo algorithm in this work. The Monte Carlo method is a numerical method for solving mathematical problems by means of random sampling. The Monte Carlo algorithm is used to generate random joint angles and then these angles are used as inputs for forwarding kinematics Eqs. (1) and (8) to calculate the position of the end-effector [11, 12]. Each position represents a point in the workspace of the robotic arm. This is done multiple numbers of iterations (n). A new position for the end effector is generated for each iteration. Those positions are then plotted together to form the workspace of the robotic manipulator. The Monte-Carlo's randomly generated values must be within the bounds of the joints, specified in Eqs. (13) and (14). This is achieved by using Eq. (15), where each joint (i), gets a random joint value(θ_{i_n}), and also the boundaries for the motion are through the MATLAB function Rand ( ), which generates a number between 0 and 1. This number is then multiplied by the difference between the upper and the lower limits of the joints variables this generates a random joint value of each iteration for each joint.

$$\theta_{i_n}=\theta_{i_{min}}+\left(\theta_{i_{max}}-\theta_{i_{min}}\right)*\text{Rand( )} \tag{15}$$

The flowchart illustrated in Figure 4 explains the MATLAB code of the algorithm that was designed to generate the workspace. The algorithm starts by determining the numbers of the iterations to generate the workspace, and then the matrices are constructed to store the data. The next step is to generate a random value for each joint calling function Rand ( ) as presented in Eq. (15). After that, the values of the joints variables for this iteration are used as inputs for Eq. (12) to calculate the position of the end effector. The values of the resulting computations are then saved to be used later to simulate/visualize the workspace. Previous procedures shall be repeated until the number of iterations set out in the first step has been reached. The last step is to show the scattered points representing the Cartesian coordinates of the end-effector resulting from all runs. The Monte Carlo methods define the workspace of the robot each iteration. The increase in the number of iterations can bring the simulated workspace as close as possible to the actual workspace [22]. The appropriate number of iterations is achieved by trial and error until the addition of more iterations is redundant.

A set of 250,000 random numbers for each joint variable was generated from the boundaries of the joints variables intervals described in Eqs. (13) and (14) to generate the workspace as described in the above flowchart. Figure 5 represents the results of the robotic arm workspace simulation. Figure 5(a) describes the motion of the end-effector of the AER-1 6-DOF manipulator in the Cartesian coordinate reference point formed in the workspace. Figures 5(b), (c), and (d) show the three Cartesian coordinates of the projection planes, *x*-*y*, *x*-*z*, *y*-*z* planes, respectively.

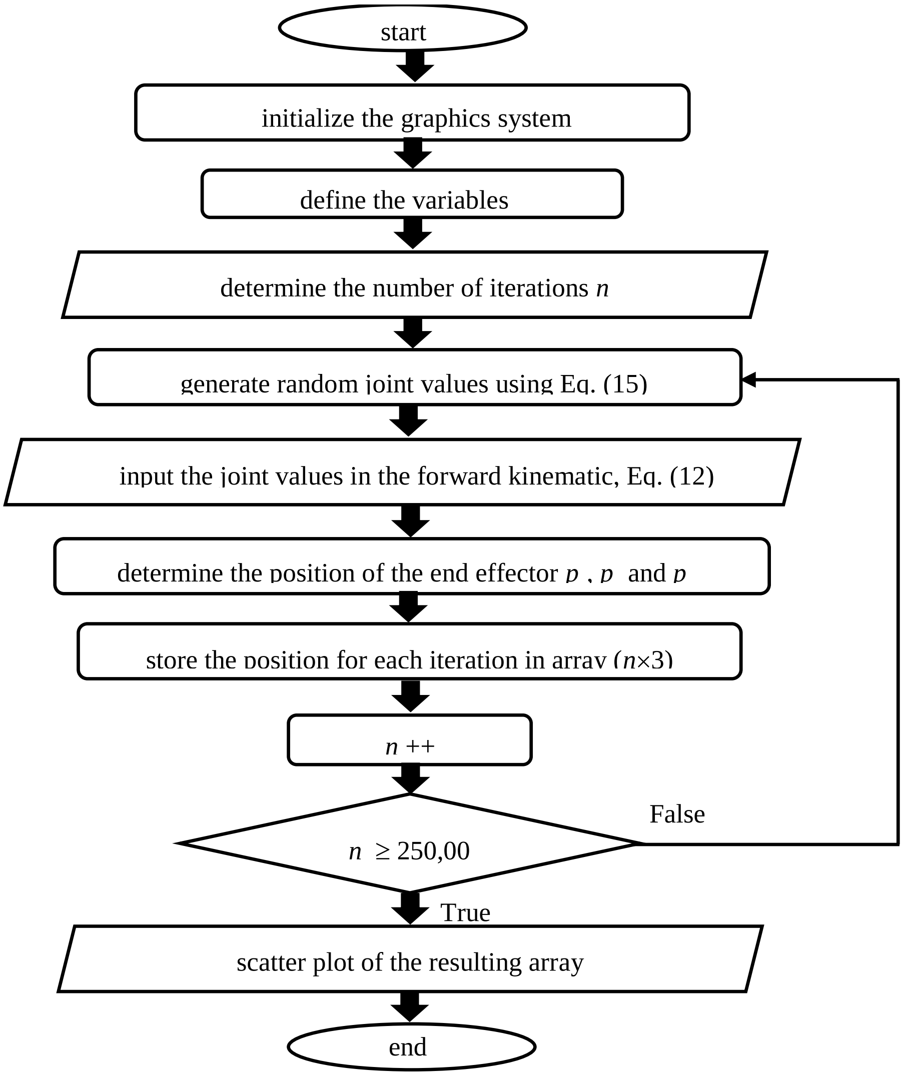
start
initialize the graphics system
define the variables
determine the number of iterations *n*
generate random joint values using Eq. (15)
input the joint values in the forward kinematic, Eq. (12)
determine the position of the end effector p , p and p
store the position for each iteration in array (*n*×3)
n ++
False
n ≥ 250,00
True
scatter plot of the resulting array
end

Figure 4. Working space simulation flowchart based on Monte Carlo method

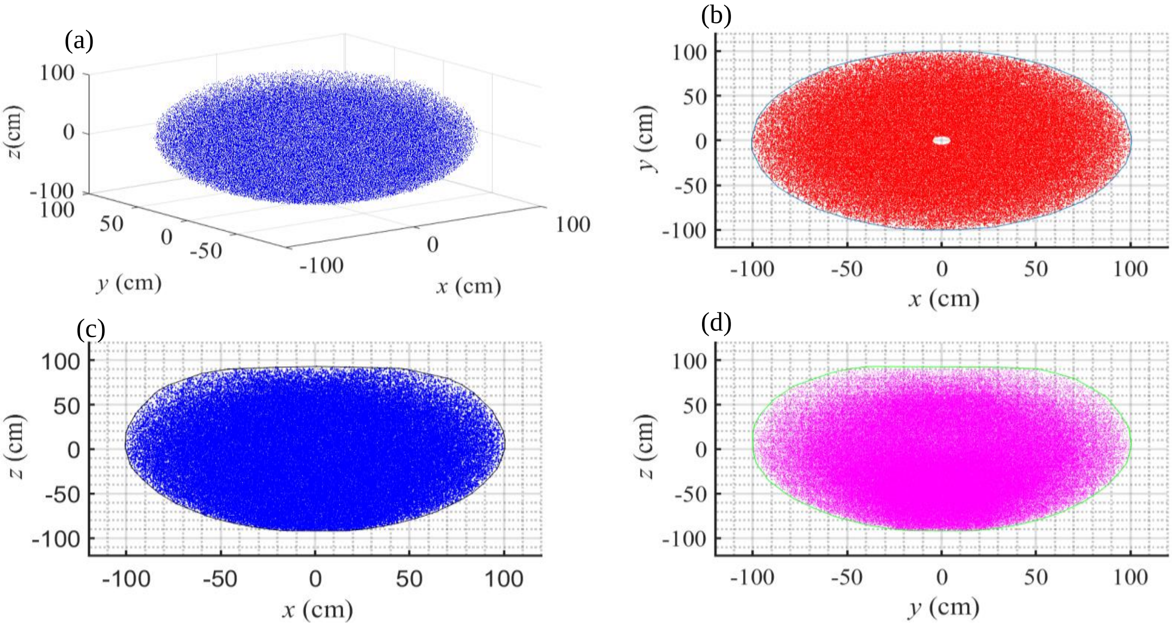

Figure 5. Results of the workspace: (a) the workspace of the manipulator, (b) projection in *x-y* plane, (c) projection in *x-z* plane and (d) projection in *y-z* plane

KINEMATIC SINGULARITY ANALYSIS OF THE MANIPULATOR

In the previous section, the kinematic equations governing the manipulator motion could be obtained. The Jacobian matrix formulas are derived using geometric methods and solved by numerical methods in this section [6, 12]. After that, the determinant of the Jacobian matrix is then computed in order to find the parameters that affect the singularity of the manipulator, and also to further investigate the behavior of these variables in the entire workspace, and to obtain the location of the singular points in both the Cartesian spaces and the joint space variables.

Jacobian Matrix

The time derivative of the kinematics equations when calculated yields the Jacobian matrix of the manipulator. Since the Jacobian matrix is the link between the end-effector velocity in the Cartesian space $v_{(6\times1)}$ and the velocity of the joints space $\dot{q}_{(n\times1)}$ as shown in Eq. (16) [6, 15].

$$v_{(6\times1)}=J(q)_{(6\times n)}\times\dot{q}_{(n\times1)} \tag{16}$$

The left-hand side of the Eq. (16) represents the velocity vector of the end effector in the Cartesian space. This vector has two main components, linear velocities (v_e) and angular velocities (ω_e) as follows:

$$v_{(6\times1)}=\begin{bmatrix} v_e \\ \omega_e \end{bmatrix}=[\dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{\alpha} \quad \dot{\beta} \quad \dot{\gamma}]^{\mathrm{T}} \tag{17}$$

The right-hand side of the Eq. (16) is the cross product of two matrices; $\dot{q}_{(n\times1)}$ which represents the joint space velocities and $J(q)_{(6\times n)}$ which represents the Jacobian matrix of the manipulator. The Jacobian matrix consists of two main parts J_p and J_o, such that the one responsible for linear motion is J_p, while J_o is responsible for the angular motion as in the following equation.

$$J(q)_{(6\times n)}=\begin{bmatrix} J_p \\ J_o \end{bmatrix}_{(6\times n)} \tag{18}$$

where:

$v_{(6\times1)}$ represents the end effector's velocity matrix

v_e is a (3x1) matrix that represents the end effector linear velocity in Cartesian space

ω_e is a (3x1) matrix that represents the end effector angular velocity in Cartesian space

$\dot{x},\dot{y}$ & $\dot{z}$ are the components of the linear velocity in the Cartesian space

$\dot{\alpha},\dot{\beta}$ & $\dot{\gamma}$ are the components of the angular velocity in the Cartesian space

J_p is a (3x1) Jacobian matrix that relates the end effector linear velocity to joints velocities

J_o is a (3x1) Jacobian matrix that relates the end effector angular velocity to joints velocities

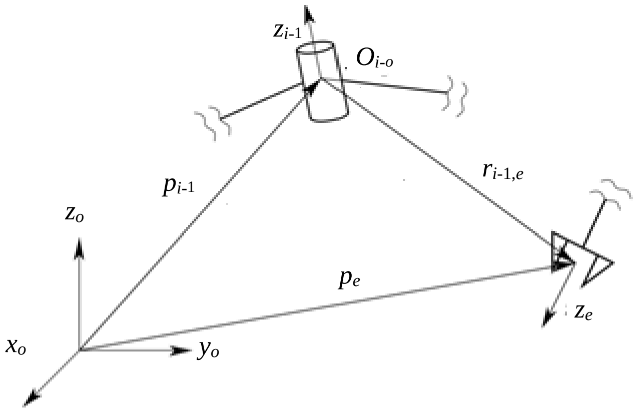

Figure 6. Representation of vectors needed for the computation of the velocity contribution of a revolute joint to the end-effector linear velocity [23]

where:

$z_{i\text{-}1}$ represents the rotation around the Z-axis vector from the transformation matrix $[a_x \quad a_y \quad a_z]^T$, as represented in Eq. (12)

p_e represents the position vector of the end effector

$p_{i\text{-}1}$ represents the position vector of the i^{th}frame relative to the base

${}^{i\text{-}1}r_i$ represents $p_e\text{-}p_{i\text{-}1}$

Figure 6 shows the vectors needed to calculate the Jacobian matrix. It is deduced from the figure that the linear and angular velocities of the end-effector are analogous to that of the vector calculations of the linear and angular velocity of a point [23, 19]. It is therefore concluded that the contribution of each joint to the angular and linear velocity of the end effector can be described using the following Eqs. (19) and (20).

$$ {}^{i\text{-}1}\omega_i=\dot{\theta}_i\times\hat{z}_{i\text{-}1} \tag{19}$$

The computing linear velocity using Eq. (19) is:

$$ {}^{i\text{-}1}v_i={}^{i\text{-}1}\omega_i\times{}^{i\text{-}1}r_i \tag{20}$$

Upon re-substituting in Eq. (16) using what was deduced in Eqs. (19) and (20), Eq. (16) can be rewritten as follows:

$$ J(q)_{(6\times n)}\times\dot{q}_{(n\times 1)}=\begin{bmatrix}J_p\\J_o\end{bmatrix}_{(6\times n)}\times\dot{q}_{(n\times 1)}=\begin{bmatrix}v_e\\\omega_e\end{bmatrix}=\begin{bmatrix}{}^{i\text{-}1}\omega_i\times{}^{i\text{-}1}r_i\\{}^{i\text{-}1}\omega_i\end{bmatrix} \tag{21}$$

Therefore, from the Eqs. (20) and (21), the following equation can be deduced:

$$ \begin{bmatrix}J_p\\J_o\end{bmatrix}_{(6\times n)}\times\dot{q}_{(n\times 1)}=\begin{bmatrix}\dot{\theta}_i\hat{z}_{i\text{-}1}\times{}^{i\text{-}1}r_i\\\dot{\theta}_i\hat{z}_{i\text{-}1}\end{bmatrix}_{(6\times 1)} \tag{22}$$

By eliminating the joint variables from both sides of the above equation, the relationship between $J_{(6\times n)}$ and the parameters of the manipulator can be determined as in the following equation:

$$ \begin{bmatrix}J_p\\J_o\end{bmatrix}_{(6\times n)}=\begin{bmatrix}\hat{z}_{i\text{-}1}\times{}^{i\text{-}1}r_i\\\hat{z}_{i\text{-}1}\end{bmatrix}_{(6\times 1)} \tag{23}$$

The resulting Jacobian matrix of the AER-1 calculated in Eq. (23) will be analyzed. Since the robot configuration can be divided into two main parts; one concerned with the shoulder of the manipulator and the other is concerned with that of the wrist Moreover, each part can be divided into two parts; one is concerned with linear velocity and the other is concerned with that of the angular velocity (J_p and J_o). Finally, the Jacobian matrix is divided into four matrices (3x3) [6, 16] as stated in Eq. (24).

$$J_e = \begin{bmatrix} J_{p_{shoulder}} & J_{p_{wrist}} \\ J_{o_{shoulder}} & J_{o_{wrist}} \end{bmatrix} \tag{24}$$

The four matrices are computed using the numerical method presented by Peter Corke in his Robotics Toolbox for MATLAB [20] as in the following equation:

$$J_{p_{shoulder},p_{wrist}} = \begin{bmatrix} \frac{\partial P_x}{\partial \theta_1} & \frac{\partial P_x}{\partial \theta_2} & \frac{\partial P_x}{\partial \theta_3} \\ \frac{\partial P_y}{\partial \theta_1} & \frac{\partial P_y}{\partial \theta_2} & \frac{\partial P_y}{\partial \theta_3} \\ \frac{\partial P_z}{\partial \theta_1} & \frac{\partial P_z}{\partial \theta_2} & \frac{\partial P_z}{\partial \theta_3} \end{bmatrix} \tag{25}$$

By using Eqs. (23) and (25), the Jacobian matrix responsible for the angular velocity of the manipulator as a vector multiplication can be deduced as presented in Eq. (26):

$$J_{p_{shoulder},p_{wrist}} = J_{O_{shoulder},O_{wrist}} \times {}^{i\text{-}1}r_i \tag{26}$$

The singularity is reached if the determinate of the Jacobian matrix equal to zero. The determinate is written as follows:

$$det(J_e) = det\left(J_{p_{shoulder}}\right) \times det\left(J_{o_{wrist}}\right) - det\left(J_{o_{shoulder}}\right) \times det\left(J_{p_{wrist}}\right) \tag{27}$$

Using the MATLAB, the Jacobian matrix of the wrist positions, Eq. (25), can be obtained as follows:

$$J_{p_{wrist}} = \begin{bmatrix} d_5c_4+d_6s_4s_5 & \text{-}d_6c_4c_5 & 0 \\ d_5s_4\text{-}d_6c_4s_5 & \text{-}d_6c_5s_4 & 0 \\ 0 & \text{-}d_6s_5 & 0 \end{bmatrix} \tag{28}$$

Clear inspection of the above equation yields:

$$det\left(J_{p_{wrist}}\right) = 0 \tag{29}$$

Then

$$det(J_e) = det\left(J_{p_{shoulder}}\right) \times det\left(J_{o_{wrist}}\right) \tag{30}$$

Since the singularity of the manipulator is reached when $det(J_e)=0$, it is clear from Eq. (30) that there are two conditions:

1. $det(J_{Pshoulder}) = 0$, this leads to the shoulder's singularity.
2. $det(J_{owrist}) = 0$, this leads to the wrist's singularity.

Using the numerical methods, Eq. (25) and Eq. (26), the Jacobian matrices of the shoulder positions and wrists orientations can be obtained as in Eqs. (31) and (32):

$$J_{p_{shoulder}} = \begin{bmatrix} \text{-}s_2*(a_3c_{23}+a_2c_2) & \text{-}c_1*(a_3s_{23}+a_2s_2) & \text{-}a_3s_{23}c_1 \\ c_1*(a_3c_{23}+a_2c_2) & \text{-}s_1*(a_3s_{23}+a_2s_2) & \text{-}a_3s_{23}s_1 \\ 0 & (a_3c_{23}+a_2c_2) & a_3c_{23} \end{bmatrix} \tag{31}$$

$$J_{o_{wrist}} = \begin{bmatrix} 0 & s_4 & \text{-}c_4s_5 \\ 0 & \text{-}c_4 & \text{-}s_4s_5 \\ 1 & 0 & c_5 \end{bmatrix} \tag{32}$$

Finally, to study the singularity behavior of the end-effector, the determinate value of J_e, $J_{p_{shoulder}}$, and $J_{o_{wrist}}$can be deduced from Eqs. (31) and (32) as follows:

$$det\left(J_{p_{shoulder}}\right) = -a_2 a_3 \left(c_2 s_3 (a_2 + a_3 c_3) - a_3 s_2 s_3^2\right) = -a_2 a_3 s_3 (c_2 a_2 + a_3 c_{23}) \tag{33}$$

$$det\left(J_{o_{wrist}}\right) = -s_5 \tag{34}$$

ANALYSIS OF THE RESULTS

After determining the components of the Jacobian matrix, Eq. (30), and substituting with the results of Eqs. (33) and (34). It was found that the determinate is a function of q_2, q_3, and q_5 where:

$$det(J_e) = f\left(q_2, q_3, q_5\right) = a_2 a_3 s_5 s_3 (c_2 a_2 + a_3 c_{23}) \tag{35}$$

As can be seen from Eq. (35) that the determinate consists of three variable terms. These are s_5, $s_3$and $(c_2 a_2 + a_3 c_{23})$. The determinate is directly proportional to the multiplication of s_5 and s_3. Therefore, if any of these terms are equal to zero, the manipulator will reach a singularity. Consequently, the manipulator will be in a singular position when s_3 equals zero and this will happen when $q_3 = 0, \pi, 2\pi, ...$ etc. On the other hand, the manipulator will be in a singular position when s_5 equals zero which will occur when $q_5 = 0, \pi, 2\pi, ...$etc.

When it comes to the third term, the substitution method is used to find the values of q_2 and q_3 which make the term equals to zero. Substituting the joints limits specified in Eqs. (13) and (14), the absolute value of the determinant is tested to verify singularity.

The determination of the values of the joints variable that singularity of the shoulder is done using MATLAB implementation of the Monte-Carlo algorithm. The use of the Monte-Carlo algorithm was previously limited for the construction of workspace only as it was presented in [12, 13, 22]. In this work, its use is further extended for the singularity analysis in order to get the values of the joint variables that lead to singular configurations. This extension is done by checking the randomly generated joint values to build the workspace of the manipulator.

The extended use of the Monte-Carlo algorithm is explained in the following flowchart illustrated in Figure 7 through the following procedures. The algorithm starts by determining the numbers of the runs that the algorithm would repeat to construct data storage matrices. The next step would be to generate a random value for each joint using function Rand ().

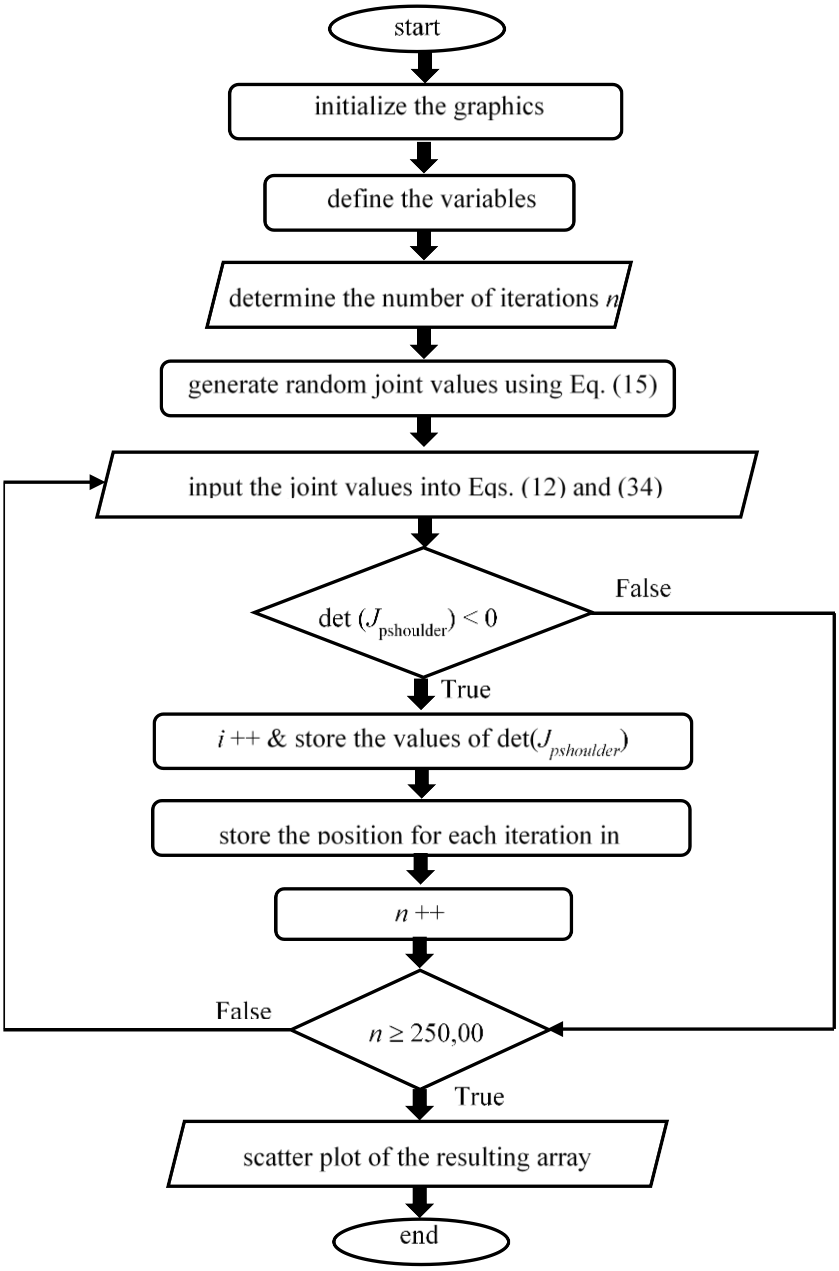
start
initialize the graphics
define the variables
determine the number of iterations n
generate random joint values using Eq. (15)
input the joint values into Eqs. (12) and (34)
det (J pshoulder) < 0
False
True
i ++ & store the values of det(J pshoulder)
store the position for each iteration in
n ++
False
n ≥ 250,00
True
scatter plot of the resulting array
end

Figure 7. Flowchart of the process to determine the shape of the workspace of singular configurations

The values of the joints variables for this iteration are then used as inputs for Eq. (33). Then the result of the previous step is compared to check if $det\left(J_{p_{shoulder}}\right)$<0 or not. If the $det\left(J_{p_{shoulder}}\right)$=0, the variables for this iteration are being saved. Previous procedures shall be repeated until the number of runs specified in the first step has been reached. The last step is to show the scattered points representing the Cartesian coordinates of the end-effector resulting from all runs

The use of this process is done simultaneously with the process of generating the workspace as in Figure 4, to ensure that the randomly generated sample is the same for both simulations. Points that have been identified as singular configurations are stored and used as an input for Eq. (12) to construct the workspace of the singular positions of the end-effector in the Cartesian space as shown in Figure 8.

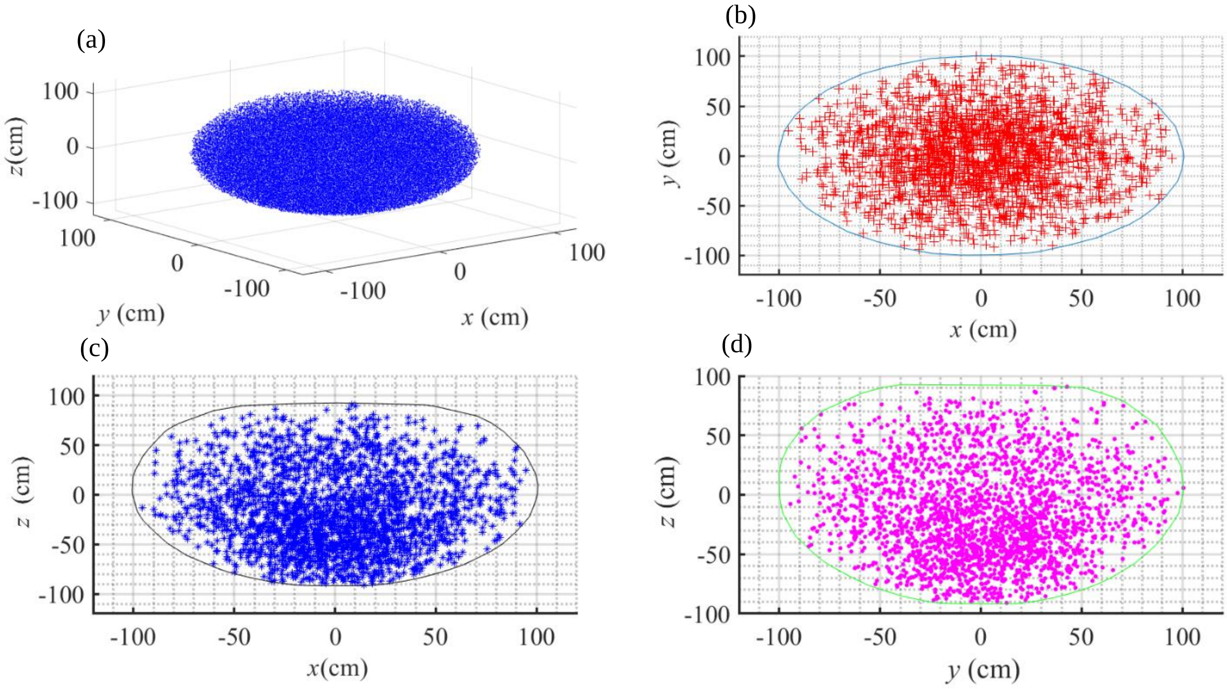

Figure 8. Simulation results of the singularity points of the manipulator in the Cartesian space. (a) 3d projection of singularity points, (b) *x*-*y* projection on the singularity points, (c) *x*-*z* projection on the singularity points, (d) *y*-*z* projection on the singularity points

The results presented in Figure 8 give a good indication of how the singular configurations affect the effective part of the entire workspace and also show the locations where the singularities are crammed.

The expended Monte-Carlo method previously described is used to find the singular points in the Cartesian space. In order to solve the third term presented in Eq. (33) to find the angles by which the determinant is zero to develop a map of joint variables and the determinate value, a MATLAB code is constructed to calculate the determinant value discretely with a step of 1°. In the code, the angles which make the absolute value of the determinant less than 10-5 are considered to make the manipulator in singularity. The process of finding all the possible values for the determinant, Eq. (33), is described in the flow chart illustrated in Figure 9. This process uses 2 nested for loops to evaluate all the possible values.

The data obtained from the previous process is used to visualize the relationship between the joints variables and the det(J_{shoulder}) presented in Figures 10 and 11 in both surface plot and contour plot. In the surface plot, the x and y axes represent the joint values of q_2 and q_3, where the z-axis represents the value of the determinant. On the other hand, in the contour plot the axes, x and y are the joint values of q_2 and q_3 where the value of the determinant is presented in the legend. The contour plot presented in Figure 11 is used to set a threshold value beyond which the value of the Jacobian matrix does not achieve reasonable controlling speed for the manipulator as in Eq. (21). Whereupon designing the trajectory, a spline curve would be generated on that contour, it has to be ensured this the spline curve doesn't approach the threshold value to avoid any damage to the manipulator.

Figure 12 represents a map of the value of the determinant with respect to the location of the end effector in Cartesian coordinates. This data is obtained by combining the data resulting from the process presented in the flowcharts presented in Figures 4 and 7. This map gives a good intuition on how close the end-effector to the singularity configuration during its operation. This map helps to design a trajectory without any singularities, this to be done by assigning an absolute threshold value for the Jacobian determinate, below which the end-effector's trajectory wouldn't be allowed to go.

The results presented in this paper was able to locate the singularities of the manipulator in both the joint space and the operational space of the manipulator. The results are obtained based on a novel use of the Monte-Carlo algorithms. The results are obtained only using the kinematic equations which are not computationally expensive to be solved compared to what was done in [8, 9]. In [8, 9], the equations of motion were used to determine and locate the singularity of the manipulator. The equations are computationally exhausting and require a closed-loop system to perform these operations. It's noteworthy that in this work the singularity was avoided by introducing a bandwidth around the value of the determinant of the Jacobian matrix rather than introducing an extra joint as was suggested in [7]. Adding extra joints leads to a complicated analysis and also makes the mechanical structure more expensive and more complex. Moreover, in this work, the exact locations of the singularities are being allocated by the novel use of Monte-Carlo algorithms. On the contrary, the use of manipulability ellipsoid was introduced in [10] where the velocity and force ellipsoid concepts were utilized to determine the manipulability which in turn determines the locations of the singularity. The manipulability measure has the advantage of being easily computed. However, this value does not give a good intuition on how close the manipulator is to the singularity as stated in [23].

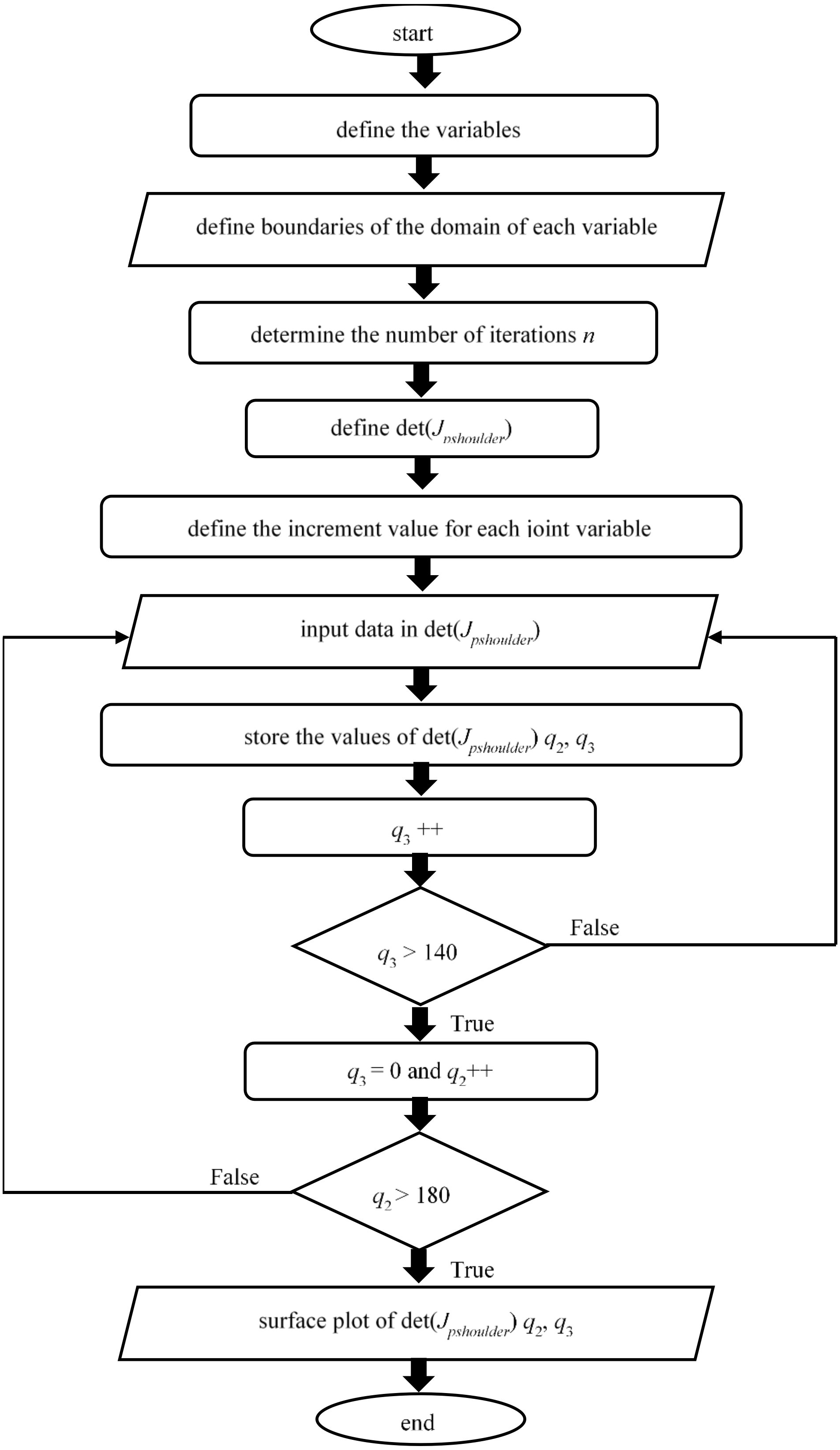
start
define the variables
define boundaries of the domain of each variable
determine the number of iterations n
define det($J_{pshoulder}$)
define the increment value for each joint variable
input data in det($J_{pshoulder}$)
store the values of det($J_{pshoulder}$) q_2, q_3
q_3 ++
$q_3 > 140$
False
True
$q_3 = 0$ and q_2++
$q_2 > 180$
False
True
surface plot of det($J_{pshoulder}$) q_2, q_3
end

Figure 9. Flowchart of the process to determine the possible values of the shoulder determinant

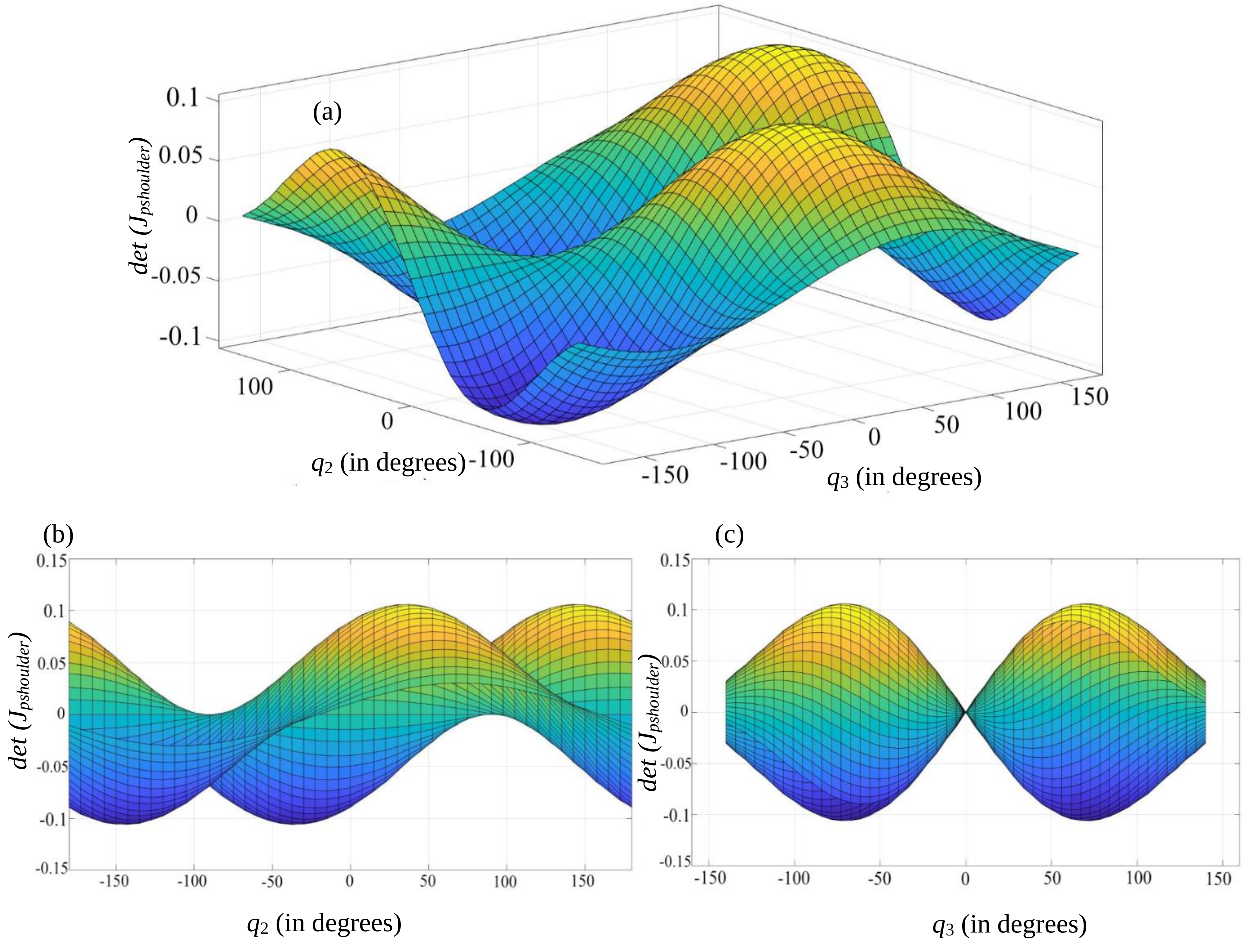

Figure 10. The shoulder determinate behaviour: (a) 3d surface plot of the shoulder determinant versus q_2, q_3; (b) surface plot of the shoulder determinant behaviour versus q_2, q_3 on *x-y* plane and (c) surface plot of the shoulder determinant behaviour versus q_2, q_3 on *y-z* plane

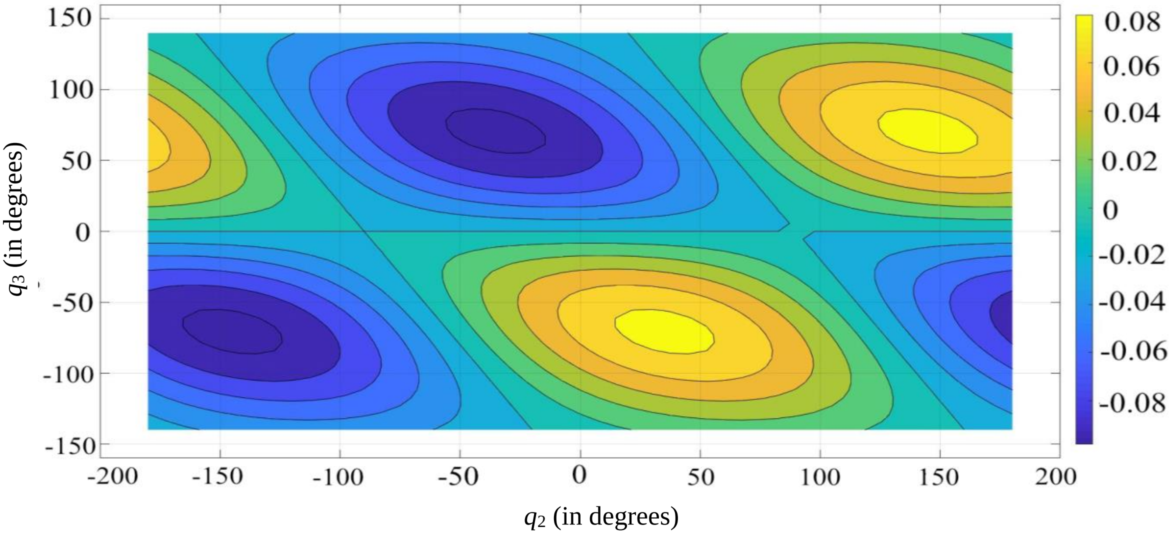

Figure 11. Contour plot of the shoulder singularity versus q_2,q_3

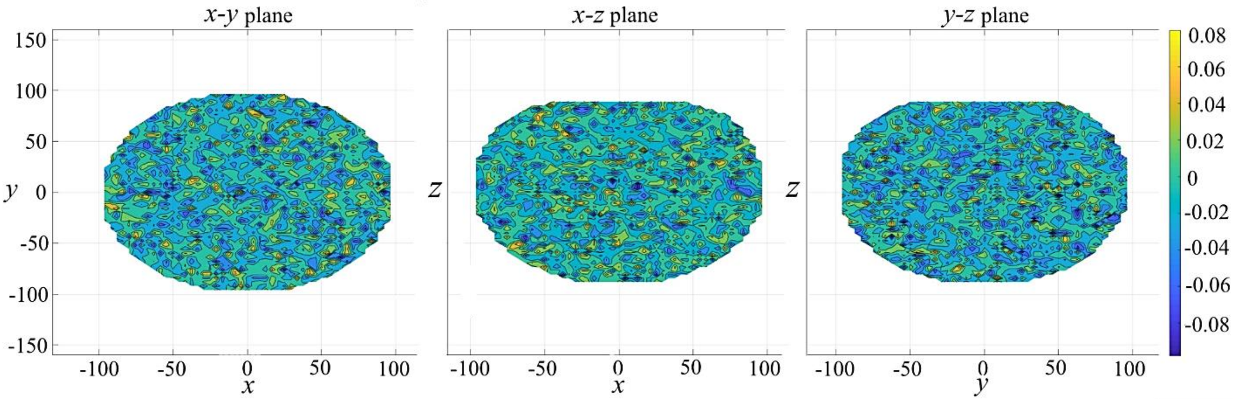

Figure 12. Contour plot linking the position of the end-effector and the value of the Jacobian determinant.

CONCLUSION

In this paper, complete modelling of the 6-DOF robotic manipulator was carried out. First, the robot's kinematic was characterized and the mathematical model was derived and has been implemented using MATLAB Robotics Toolbox. Second, the entire workspace of the manipulator was presented using the numerical method of Monte-Carlo to provide a clear image of how does the manipulator behaves and to determine the boundaries of the manipulator's workspace. Third, the Jacobian matrix and its determinant are calculated using MATLAB symbolic Toolbox to give a general solution of the determinant for the Jacobian matrix to test its rank to locate the singularities in the workspace. Finally, by extending the use of the Monte-Carlo method the singularities are simulated to visualize the singularity of the manipulator in the workspace. After performing the previous procedures, the following points are extracted:

1. The singularity occurs if:
   a. s_5=0, (leads to wrist singularity)
   b. s_3=0, (leads to shoulder singularity)
   c. a_2c_2=-a_3c_{23}, (leads to shoulder singularity)
2. Extended use of the Monte-Carlo methods leads to ease of calculation and visualization of the singularities in Cartesian space.
3. A contour plot is an easy way to figure out the values of the joints variables in order to design a trajectory free of a singularity.
4. The singularities are accumulated around the boundaries of the workspace.
5. The singularities are figured out by Monte Carlo Algorithm and showed in Figures 8, 11, and 12, They will be used to design a path without singularity
6. The manipulator's spherical configuration was easier to treat analytically.
7. To design a manipulator trajectory free of singularity, a spline curve is fitted to the contour plot to ensure that the determinate value does not approach zero.
8. The use of the numerical methods has been shown to be beneficial in this study, and therefore all configurations of the joints that cause the robot to be singular are obtained using them.
9. The results presented in this work are simply achieved in comparison to other methodologies.

REFERENCES

[1] P. S. Donelan, "Kinematic singularities of robot manipulators," in *Advances in Robot Manipulators*, 2010, doi: 10.5772/9548.

[2] Y. Nakamura, Advanced robotics: redundancy and optimization, Addison-Wesley Longman, ISBN: 978-0201151985, 1991.

[3] Y. Yang, V. Ivan, Z. Li, M. Fallon and S. Vijayakumar, "iDRM: Humanoid motion planning with realtime end-pose selection in complex environments," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, November 2016, doi: 10.1109/HUMANOIDS.2016.7803288.

[4] N. Vahrenkamp, H. Arnst, M. Wachter, D. Schiebener, P. Sotiropoulos, M. Kowalik and T. Asfour, "Workspace analysis for planning human-robot interaction tasks," in *IEEE-RAS* 16*th International Conference on Humanoid Robots (Humanoids)*, November 2016, doi: 10.1109/HUMANOIDS.2016.7803437.

[5] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini and R. Dillmann, "Manipulability analysis," in *IEEE-RAS 12th International Conference on Humanoid Robots*, November 2012, doi: 10.1109/HUMANOIDS.2012.6651576.

[6] R. Y. Abdolmalaki, “Geometric Jacobians derivation and kinematic singularity analysis for smokie robot manipulator & the barrett WAM,” *5th International Conference on Robotics and Mechatronics (ICROM),* 2017.

[7] S. Chiaverini , B. Siciliano and O. Egeland, “Kinematic analysis and singularity avoidance for a seven-joint manipulator,” in *American Control* Conference, June 1990, doi: 10.23919/ACC.1990.4791140.

[8] R. Xu, J. Luo and M. Wang, “Kinematic and dynamic manipulability analysis for free-floating space robots with closed chain constraints,” *Robotics and Autonomous Systems*, vol 130 (103548), 2020, https://doi.org/10.1016/j.robot.2020.103548.

[9] V. Duindam and S. Stramigioli, “Singularity-free dynamic equations of open-chain mechanisms with general holonomic and nonholonomic joints,” *IEEE Transactions on Robotics,* vol. 24, no. 3, pp. 517 - 526, 2008, doi: 10.1109/TRO.2008.924250.

[10] D. Lee, W. Lee, J. Park and W. K. Chung, “Task space control of articulated robot near kinematic singularity: forward dynamics approach,” *IEEE Robotics and Automation Letters,* vol. 5, no. 2, pp. 752 - 759, 2020, doi: 10.1109/LRA.2020.2965071.

[11] Y. Cao, K. Lu, X. Li and Y. Zang, “Accurate numerical methods for computing 2D and 3D robot workspace,” *International Journal of Advanced Robotic Systems* , vol. 8, no. 6, 2011, https://doi.org/10.5772/45686.

[12] Y. Cao, S. Qi, K. Lu, Y. Zang and G. Yang, “Shape and size computation of planar robot workspace,” in *Computer Science and Information Engineering, World Congress on*, Los Angeles, California USA, 2009, doi: 10.1109/CSIE.2009.266.

[13] J. Rastegar and D. Perel, “Generation of manipulator workspace boundary geometry using the monte carlo method and interactive computer graphics,” *Journal of Mechanical Design,* vol. 112, no. 3, pp. 452-454, 1990, https://doi.org/10.1115/1.2912630.

[14] P. M. Kebria, S. Al-wais, H. Abdi and S. Nahavandi, “Kinematic and dynamic modelling of ur5 manipulator,” in *2016 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*), Budapest, 2016, doi: 10.1109/SMC.2016.7844896.

[15] H. M. Alwan and H. Z. Rashid, “Investigation and simulation the kinematic singularity of three links robot manipulator with spherical wrist,” *Journal of University of Babylon for Engineering Sciences,* vol. 26, no. 8, 2018.

[16] T. Balkan, M. K. Özgören and M. A. Sahir Arikan, “Structure based classification and kinematic analysis of six-joint industrial robotic manipulators,” in *Industrial Robotics: Theory, Modelling and Control*, 2006, doi: 10.5772/5009.

[17] R. Y. Abdolmalaki, “Development of direct kinematics and workspace representation for smokie robot manipulator & the barret WAM,” in *5th International Conference on Robotics and Mechatronics (ICROM),* Tehran, Iran, 2017.

[18] P. Milenkovic, “Continuous path control for optimal wrist singularity avoidance in a serial robot,” *Mechanism and Machine Theory,* vol. 140, pp. 809-824, 2019, https://doi.org/10.1016/j.mechmachtheory.2019.05.004.

[19] K. M. Lynch and F. C. Park, Modern robotics:mechanics, planning, and control, Cambridge University Press, 1st ed., ISBN: 978-1-107-15630-2, 2017.

[20] P. I. Corke, “A Robotics toolbox for MATLAB,” *IEEE Robotics and Automation Magazine,* vol. 3, no. 1, pp. 24-32, 2007, doi: 10.1109/100.486658.

[21] F. Litvin, “Application of theorem of implicit,” *Mechanism and Machine Theory,* vol. 15, no. 2, pp. 115-125, 1980.

[22] J. Zhu and F. Tian, “Kinematics analysis and workspace calculation of a 3-DOF manipulator,” *IOP Conference Series Earth and Environmental Science*, Vol 170, no. 4, 2018, doi: 10.1088/1755-1315/170/4/042166.

[23] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, Robotics, modelling, planning and control, London: Springer-Verlag London Limited, ISBN 978-1-84628-642-1, 2009.