AIMS Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 9(6): 13944–13979.
DOI:10.3934/math.2024678
Received: 13 March 2024
Revised: 06 April 2024
Accepted: 09 April 2024
Published: 16 April 2024

Research article

Assessing the accuracy and efficiency of kinematic analysis tools for six-DOF industrial manipulators: The KUKA robot case study

Mohamed S. Elhadidy[1,2,*], **Waleed S. Abdalla**[1,2], **Alaa A. Abdelrahman**[1], **S. Elnaggar**[1] **and Mostafa Elhosseini**[3,4]

[1] Department of Mechanical Design and Production Engineering, Faculty of Engineering, Zagazig University, Zagazig 44511, Egypt
[2] Department of Mechatronics Engineering, Faculty of Engineering, Horus University, New Damietta 34517, Egypt
[3] College of Computer Science and Engineering, Taibah University, Yanbu 46421, Saudi Arabia
[4] Computers and Control Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt

* **Correspondence:** Email: melhadidy@horus.edu.eg, Tel: +201012900010.

Abstract: Accuracy is an important factor to consider when evaluating the performance of a manipulator. The accuracy of a manipulator is determined by its ability to accurately move and position objects in a precise manner. This research paper aims to evaluate the performance of different methods for the kinematic analysis of manipulators. The study employs four distinct techniques, namely mathematical modeling using the closed form solutions method, roboanalyzer, Peter Corke toolbox, and particle swarm optimization, to perform kinematic analysis for manipulators. The KUKA industrial manipulator is used as an illustrative case study in this research due to its widespread use in various industrial applications in addition to its high precision and stability. Its wide usage in the industry makes the results of this research highly relevant and allows for a thorough evaluation of the performance of the different methods being studied. Furthermore, understanding the kinematic analysis of the manipulator can also help in improving the performance and increasing the efficiency of the robot in different tasks. This paper conducts a comparison of the accuracy of the four methods, and the results indicate that particle swarm optimization is the most accurate method. The RoboAnalyzer approach achieved the fastest execution time.

Keywords: kinematic analysis; KUKA industrial manipulator; Peter Corke toolbox; roboanalyzer; particle swarm optimization
Mathematics Subject Classification: 68T40, 70B15, 93C85

1. Introduction

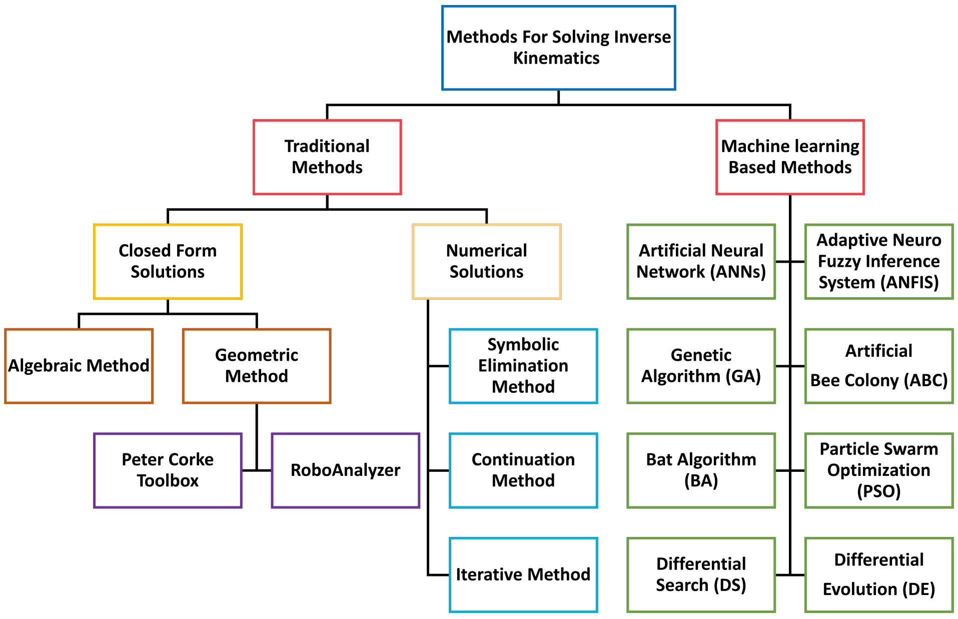
Robotics is an inspiring and ever-growing field of engineering with an array of applications across a broad range of industries. Robotics involves the design, development, simulation, management, use in administrative offices, exploration of space, and application of robots to automate tasks that would otherwise require a human work force. The robots of today help humans through their everyday routines and do many mundane tasks [1]. An industrial manipulator is a versatile robot that operates autonomously. Industrial manipulators are becoming increasingly popular in a broad range of industrial applications such as production lines, painting, welding, assembly, etc., due to their benefits of large workspace, compact construction, and excellent adaptability [2]. For these tasks to be done correctly, the robots must be accurate [3].

The KUKA KR 22 R1610-2 industrial manipulator is a specific model of robot produced by KUKA. It was chosen as an illustrative case study due to the unique characteristics that make it ideal for kinematic analysis. It has six degrees of freedom (6-DOF) with a spherical wrist, which allows for a wide range of motion and a high level of flexibility. This makes it an ideal subject for studying the kinematics of a robot with a high degree of complexity [4]. Compared to other robots, the KUKA KR 22 R1610-2 is a compact robot with a high payload capacity and reach, making it suitable for a wide range of applications, especially in small to medium-sized manufacturing [5].

The intricate issue of robot motion is known as robot kinematics [6]. Finding the conversion from cartesian space to joint space and back is connected to the kinematics issue [7]. Any robot manipulator's kinematics issue has two sorts of solutions: forward kinematics and inverse kinematics [8]. In robotics, forward kinematics is used to determine the position and orientation of the end effector, such as a gripper or tool, based on the angles of rotation of the joints of the robot's manipulator [9]. The process of forward kinematics typically involves solving a set of equations called the forward kinematics equations, which describe the relationship between the angles of rotation of the joints and the position and orientation of the end effector or bones in the system. To move the end effector into the desired position and orientation in a reference frame, a set of joint variables must be identified for the given end effector position and orientation. It is known as inverse kinematics [10].

Robotics research is based on manipulator kinematics, which shows the relationship between the robot's final position and its kinematic parameters [11]. Robot path planning and motion control research start with manipulator kinematics analysis [12]. Manipulators are rigid chains with revolute or prismatic joints. Manipulator joint locations are relative to adjacent joints. Notably, 4×4 homogeneous transformation matrices with robot orientation and position data define joint relationships [13]. Robot degrees of freedom depend on transformation matrices. These transformation matrices generate n-DOF manipulator orientation and position [6]. Forward kinematics calculates the robot manipulator's end effector location and orientation from joint angles [14]. The Denavit-Hartenberg (D-H) Convention determined the homogeneous matrix from four consecutive movements that integrate the end effector's translational and rotational motions concerning the manipulator base [15]. The manipulator's inverse kinematics are harder to solve [16]. Inverse kinematics has nonlinear equations, singularities, and many solutions [17]. The joint variable motions must be computed to discover all possible forms for going from the end effector position to the base position [18].

There are many types of kinematics solution methods that depend on the kinematics type. One solution method approach for forward kinematics serial chains is to simply concatenate transformations between frames set in neighboring links of the chain [19]. In inverse kinematics, several different methods have been proposed to solve this problem, including analytical methods, numerical methods, and machine learning-based methods [20]. Analytical methods, such as closed form solutions, are based on mathematical equations that can be used to calculate the joint angles of a robotic system. These methods are computationally efficient and provide accurate solutions, but they are often limited to specific types of robotic systems, such as serial robots, and may not be able to handle complex or variable systems [21]. Algebraic methods and geometric methods like the roboanalyzer and Peter Corke robotics toolbox are also analytical methods [22]. Various methods for solving inverse kinematics can be summarized and illustrated in Figure 1.



Methods For Solving Inverse Kinematics
Traditional Methods
Machine learning Based Methods
Closed Form Solutions
Numerical Solutions
Algebraic Method
Geometric Method
Peter Corke Toolbox
RoboAnalyzer
Symbolic Elimination Method
Continuation Method
Iterative Method
Artificial Neural Network (ANNs)
Adaptive Neuro Fuzzy Inference System (ANFIS)
Genetic Algorithm (GA)
Artificial Bee Colony (ABC)
Bat Algorithm (BA)
Particle Swarm Optimization (PSO)
Differential Search (DS)
Differential Evolution (DE)

Figure 1. Methods for solving inverse kinematics.

Roboanalyzer is a 3D model-based solution with multiple modules that may be used to efficiently visualize different robotics topics. It can display D-H parameter animation and conduct forward and inverse kinematics, as well as dynamic analyses of serial robots [23]. Peter Corke's robotics toolbox is an open-source toolbox for robotic systems. This toolbox contains a collection of functions and tools for robotic systems that allow users to easily implement and test robot control and vision algorithms [24]. Numerical methods, such as iterative methods, use numerical techniques to find a solution to the inverse kinematics problem. These methods are more flexible than analytical methods and can handle a wide range of robotic systems, but they can be computationally expensive and may not always converge to a solution. Symbolic elimination methods and continuation methods are also numerical methods [16]. Machine learning based methods use techniques from machine learning, such as artificial neural networks (ANNs), to learn from data and make predictions. These methods are effective for solving the inverse kinematics problem, particularly for complex and variable systems [25]. However, these methods require large amounts of data for training, and their performance can be influenced by the choice of hyper parameters. adaptive neuro fuzzy inference system (ANFIS), genetic algorithm (GA), bat algorithm (BA), differential search (DS), artificial bee colony (ABC), particle swarm optimization (PSO), and differential evolution (DE) are also machine learning based methods [26].

Particle swarm optimization is a population-based metaheuristic optimization algorithm developed by Kennedy and Eberhart in 1995. Inspired by the social behavior of bird flocking or fish schooling, PSO simulates the collaborative movement of particles in a search space to find optimal solutions. PSO maintains a population of candidate solutions, called particles, which fly through the search space and update their positions based on the influence of their own best-known position and the best-known position in the swarm [27]. PSO has been proven to be a promising alternative for solving IK problems. In PSO-based IK, the particles represent the joint angles, and their positions are updated according to the fitness function, which measures the distance between the current end-effector position and the desired one. The particles also communicate with each other to share information and improve their positions, mimicking the social behavior of birds or fish. One advantage of using PSO for IK is its ability to handle non-linear constraints. Unlike traditional methods, PSO does not require the computation of Jacobians or the inverse of the kinematic equations, making it more efficient and suitable for real-time applications. Furthermore, PSO is a population-based algorithm, meaning that it can explore a larger area of the search space and is less likely to get trapped in local optima [28].

In recent years, there has been a significant amount of research conducted in the field of inverse kinematics, with numerous studies addressing various aspects of the problem. One tool that has been integral to the advancement of the robotics field is Peter Corke's robotics toolbox for MATLAB. The toolbox provides users with the ability to quickly and easily design, program, and stimulate robotic systems. The robotics toolbox has been used to solve the inverse kinematics of a PUMA 560 robot and has been found to provide a simple and efficient solution [24]. According to Peter Corke, MATLAB has several toolboxes that can be used to improve the learning experience in robotics and vision.

Several studies have been conducted to examine the potential of using the robotics toolbox as a visualization and control tool for manipulator kinematics. In a study performed by Sadanand et al. [29], it was found that a virtual robot module within the toolbox could be an effective visualization tool for introducing robotics to students. Another study found that the robotics toolbox was able to determine the joint angles accurately and efficiently for a given end effector position of a 6-DOF manipulator [30]. In a study by Yu Long et al. [31], it was found that the robotics toolbox could control 7-DOF manipulators with high precision, allowing the robot to move along a desired path. Additionally, research by Long et al. [32] found that MATLAB and the robotics toolbox could be used for the simulation of robotic arms, and the developed simulation model was found to be accurate enough for use in industry applications.

Zhang et al. [33] studied the critical aspect of enhancing the positioning accuracy reliability of industrial robots by employing probability and evidence theories. By incorporating probability theory, which deals with the quantification of uncertainty, and evidence theory, which focuses on handling uncertainties and imprecise information, the researchers aim to advance the reliability of positioning accuracy in industrial robot systems. Probability theory-based methods are commonly used to evaluate the positioning accuracy reliability of industrial robots. The suggested aleatory-epistemic hybrid model describes parameters of industrial robots that are not always clear, which helps build the kinematic equation. The study establishes a probability-evidence hybrid reliability analysis model that provides a reliability interval for industrial robots under various thresholds. The effectiveness of the proposed method is demonstrated through the application of a six-degrees of freedom industrial robot. Zhang and Han [34] propose an efficient reliability analysis method to predict the kinematic reliability of robotic manipulators by accounting for the random dimensions and joint angles of robotic mechanisms. The method defines kinematic reliability as the probability that the actual position of an end-effector falls within a specified tolerance sphere centered at the target position. The motion error is indicated by a compound function of independent standard normal variables constructed by three co-dependent coordinates of the end-effector. The saddle point approximation is applied to compute kinematic reliability, resulting in satisfactory accuracy and efficiency compared to random simulation methods. Zhang et al. [35] performed an in-depth study of the reliability of kinematic trajectory accuracy for industrial robots while considering intercorrelations among multi-point positioning errors. The research highlights the significance of understanding and evaluating the reliability of kinematic trajectories in industrial robotic systems, which play a pivotal role in ensuring precise and efficient operations. By examining the intercorrelations among multi-point positioning errors, the study offers valuable insights into the complexities of industrial robot performance. This analysis is crucial for enhancing the overall accuracy and reliability of industrial robots, ultimately contributing to improved productivity and quality in manufacturing processes. Zhang et al.'s results show how multi-point positioning errors and the reliability of kinematic trajectories are closely connected.

Bahuguna et al. [36] solved the forward and inverse kinematics of a 6-DOF robot arm and showed that a roboanalyzer helped students understand robot motion and develop and analyze robotic devices. Thus, roboanalyzer are essential for robot kinematics education. Mehta et al. [23] created a roboanalyzer teaching pendant. An operator interfaces input commands, and a robot's graphical representation display the motion. The teaching pendant also generated robot program code for later usage. The technology accurately generated motion on multiple virtual robots. Gupta et al. [37] discussed how roboanalyzers may assist robot technicians to analyze problems quickly and efficiently. Roboanalyzer displays robot data in an easy-to-use graphical user interface, according to the creators. Roboanalyzer also lets technicians zoom in to review data and easily access faulty robot parts for diagnosis and repair. Othayoth et al. [38] developed a roboanalyzer to simplify robot kinematics analysis and simulation. Other roboanalyzer features include the ability to produce code for controlling a real-world robotic system and visualize its motion in 3D. SS Chauhan and Khare [18] conducted a study using the roboanalyzer software. Their investigation demonstrated that the robot could duplicate its desired movements and complete tasks efficiently. The analysis increased the robot's kinematic performance. The study shows that roboanalyzers help industrial robots do kinematic analysis. Chang [39] proposed a closed form solution to the challenging problem of controlling manipulators with kinematic redundancy. This solution focused on the use of a linear combination of the positioning and velocity task space components. By combining the two components, a more robust control system could be implemented, and a more stable solution could be found. Chang [40] provided a closed form approach for robot manipulator inverse kinematics with redundancy. The Generalized Inverse Jacobian (GINJ) approach underpinned this solution, which claimed to be more efficient than previous numerical methods. Chang's technique solved the inverse kinematics issue of robotic manipulators with redundancy simply and efficiently, advancing robotics. Chen and Gao [41] discussed the development of a closed form inverse kinematics solver for reconfigurable robots, with a specific focus on the closed form solution for the inverse kinematics problem of a reconfigurable robot. The algorithm proposed by the authors was tested on a reconfigurable robot, and the results showed that the algorithm provided satisfactory solutions for a variety of configurations.

Gao et al. [42] provided an extensive analysis of the uncertainties involved in the kinematics of cable-driven parallel robots. The research focuses on developing an error transfer model to quantify and assess the impact of uncertainties on the robot's kinematic performance. By incorporating uncertainties into the kinematic model, the study aims to provide a more comprehensive understanding of the robot's behavior under varying conditions. The research establishes an inverse kinematic model for a CDPR used for picking up medicines, considering the radii of fixed pulleys. The influence of the radii of fixed pulleys on errors in cable lengths is explored. An error transfer model and an evidence theory-based uncertainty analysis method (ETUAM) are presented to analyze the uncertain sources of cable lengths in CDPRs. The ETUAM demonstrates accuracy and efficiency in kinematic uncertainty analysis compared to the vertex method and Monte Carlo method. Zhang et al. [43] evaluate the effectiveness of statistical moment-based methods for assessing the reliability of industrial robots' positional accuracy. They explore the significance of statistical moments as key descriptors for characterizing the distribution of positional errors in robotic systems. Three different statistical moment-based methods are applied in the study: the sparse grid numerical integration (SGNI) method, the point estimation method (PEM), and the univariate dimension reduction method (UDRM). The kinematics model of industrial robots is established using the Denavit-Hartenberg method. The SGNI method is found to have the best computational accuracy, while the PEM method exhibits the highest computational efficiency among the three methods. The positional accuracy reliability of industrial robots is quantitatively evaluated using these methods and compared with the results from the monte carlo simulation (MCS) method. Zhao et al. [44] delved into the complexities of assessing the reliability of robotic manipulators in dynamic environments. The study addresses the issue of time-dependent system kinematic reliability analysis for robotic manipulators, which is an area that has receive limited investigation. The proposed method in the research is based on the first-passage method and calculates the outcrossing rate to determine the time-dependent system kinematic reliability. Using Lie group theory, the authors come up with a closed-form answer for the covariance of the joint distribution of the pose error and its derivative. The outcrossing rate is calculated by decomposing the outcrossing event of the pose error and determining the first-order moment of a truncated multivariate Gaussian. The paper deduces an analytical formula for the outcrossing rate based on the independent assumption that outcrossing events occur independently. The effectiveness of the proposed method is demonstrated using a six-degrees-of-freedom (6-DOF) robotic manipulator, and it is compared with Monte Carlo simulation and the finite-difference-based outcrossing rate method.

Danaci et al. [27] proposed a particle swarm optimization approach for solving the inverse kinematics problem in robotic manipulators. The PSO approach provides a comprehensive solution for both the position and orientation of the robot end effector. The results show that QPSO is more efficient than PSO, FA, and ABC for solving IK. Abdor-Sierra et al. [45] delved into a comparative analysis of metaheuristic algorithms to address the challenge of solving the inverse kinematics problem in robot manipulators. The study evaluates various metaheuristic algorithms, including genetic algorithms (GAs), particle swarm optimization (PSO), and artificial bee colony (ABC) optimization, among others, in tackling the inverse kinematics problem. Comparative analysis provides insights into the strengths and limitations of each algorithm in terms of convergence speed, solution quality, and computational complexity when applied to robot manipulators. Liu et al. [46] proposed a novel approach to solving the inverse kinematics problem for manipulators using a combination of Inverse Particle Swarm Optimization (IPSO) and Back-Propagation Neural Network (BPNN) algorithms. The proposed method aims to improve the accuracy and efficiency of inverse kinematics solutions for manipulators by integrating IPSO, which is known for its global optimization abilities, with BPNN, which is a powerful machine learning technique. The proposed method solves the inverse kinematics of the UR3 manipulator, and the output error of the joint angle obtained is less than 0.1 degrees.

Although the listed literature shows promising methods and significant results in addressing manipulator kinematics strategies there is still room for further research and improvements in the field of manipulator kinematics to fill the following research gaps:

- Despite the potential of using the robotics toolbox as a visualization and control tool for manipulator kinematics, there is a lack of studies that have examined KUKA KR 22 R1610-2 in industry applications.
- There is a gap in research on the use of roboanalyzers in industry applications, specifically in terms of their ability to assist with robot diagnosis and repair.
- There is a lack of research comparing different inverse kinematics methods and their potential advantages and disadvantages for specific manipulator types or applications.
- There is a need for further research on closed form inverse kinematics algorithms for continuum manipulators, as current methods may not be accurate when the manipulator is subject to external forces.
- The research on the closed form solution for the inverse kinematics problem of a reconfigurable robot is limited.

Overall, the research indicates that while there have been advances in the field, there is still room for further research and improvements in the field of manipulator kinematics. However, each of these methods has its own set of advantages and limitations, and the selection of the appropriate method is dependent on the specific characteristics of the robotic system and the requirements of the application. The present work aims to fill the presented research gaps and solve the manipulator kinematics problem using mathematical simulation with MATLAB, roboanalyzer software, the Peter Corke robotics toolbox, and the particle swarm optimization. The results of this study are then compared with the results available in the literature to assess their accuracy and efficiency. The goal is to obtain good agreement between the results of this study and those found in previous research. Additionally, this paper aims to evaluate the potential of using the robotics toolbox as a visualization and control tool for manipulator kinematics, the PSO as an optimization tool for solving the inverse kinematics problem, and the roboanalyzer software for robot kinematics education and analysis. This work provides the following contributions:

- Advanced Kinematic Analysis: Provides an in-depth examination of the KUKA KR 22 R1610-2, emphasizing operational strengths and practical limitations in industrial settings.
- Unique Simulation Framework: Introduces a novel, validated simulation framework for evaluating manipulator performance, offering precise simulations and predictions.
- Validated Research Methodologies: Empirically validates our analytical methods against existing literature, ensuring the accuracy, and reliability of our findings.
- Broad Applicability: Delivers actionable insights for the KUKA KR 22 R1610-2's use in industrial automation, robotics research, and education, fostering future advancements.
- Kinematics Analysis Using Particle Swarm Optimization (PSO): Applies the PSO algorithm to accurately solve the inverse kinematics problem for the KUKA KR 22 R1610-2, targeting precise

end-effector positioning.

The structure of this paper is composed of four distinct sections. Section 2 provides a comprehensive examination of the kinematics of the manipulator, which is divided into two subsections: forward kinematics and inverse kinematics. In Section 3, the simulation results of forward and inverse kinematics are presented using mathematical simulation, the roboanalyzer software, the Peter Corke robotics toolbox, and particle swarm optimization. Section 4 offers an illustrative case study. In-depth discussion of the obtained results is presented in Section 5. Lastly, Section 6 presents the conclusions drawn from the research presented in the paper.

2. Materials and methods

2.1. Kinematics analysis of manipulator using mathematical formulation

The study of mechanism motion without considering the driving forces is known as kinematic analysis. The primary goal of the kinematic analysis is to determine the characteristics of the end effector of a manipulator concerning the manipulator base, like its location, displacement, velocity, and acceleration. Kinematic analysis is based on the geometrical connection between the robotic linkages and their joints, which can be described with several scenarios of motion, such as Denavit-Hartenberg parameters, to investigate the robot's motion [47]. Kinematics plays a significant role in controlling serial manipulators in a variety of applications [48]. The solutions of any manipulator kinematics are divided into two categories: forward kinematics and inverse kinematics. The forward kinematics of the manipulator point to the calculation of its end effector orientation and position from the values of joint angles, while the inverse kinematics point to the calculation of the values of joint angles from the end effector orientation and position [49]. The forward and inverse kinematics of the manipulator must be solved for kinematic modeling [50]. Kinematics analysis can be summarized as shown in Figure 2.

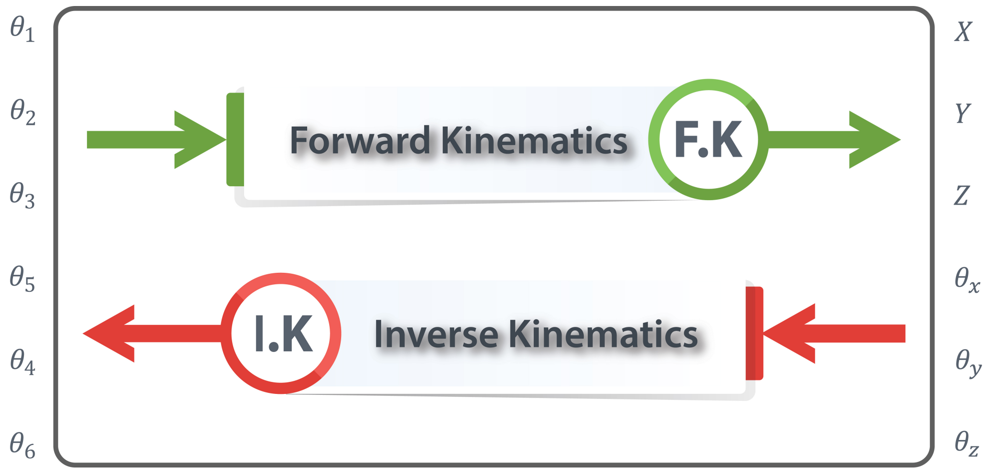


Figure 2. Kinematics analysis summarizing [51].

2.1.1. Forward kinematics

Forward kinematics is known as the method of estimating the pose of the end effector (location and orientation) with specified joint angles [50]. To highlight the connection between the coordinate systems and to define the location and orientation of connecting links, each connecting link was developed in its own coordinate system [14]. The coordinate transformations are defined using the Denavit-Hartenberg convention (D-H parameters) approach, as shown in Figure 3 [52]. Joint angle, joint offset, link length, and twist angle are four factors that determine the transformation between coordinates associated with the joints between two links, which were described as shown in Tables 1 and 2 [53]. Forward kinematics is solved with the Denavit-Hartenberg convention method [30].

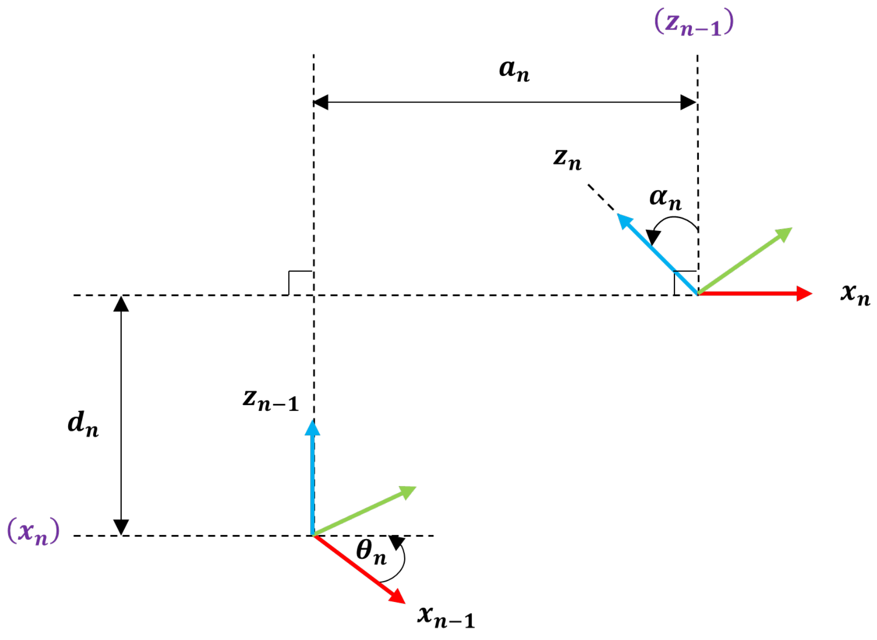


Figure 3. Denavit-Hartenberg convention.

Table 1. D-H parameters description.

Parameter	The description
a_n	The distance between z_{n-1} and z_n (along x_n)
α_n	The twist angle between z_{n-1} and z_n (measured about x_n)
d_n	The offset distance between x_{n-1} and x_n (along z_{n-1})
θ_n	Angle between x_{n-1} andx_n (measured about z_{n-1})

Table 2. D-H link and joint parameters.

Joint parameters	Link parameters
θ_n: joint angle	α_n: link twist
d_n: link offset	a_n: link length

Each homogeneous transformation is the result of four fundamental transformations, as shown in

Eqs (2.1) and (2.2) [54].

$$A_n = \text{Rot}_{z,\theta_n}\ \text{Trans}_{z,d_n}\ \text{Trans}_{x,a_n}\ \text{Rot}_{x,\alpha_n}. \tag{2.1}$$

For a link n the transformation matrix is as follows:

$$A_n = \begin{bmatrix} C\theta_n & -S\theta_n C\alpha_n & S\theta_n S\alpha_n & a_n C\theta_n \\ S\theta_n & C\theta_n C\alpha_n & -C\theta_n S\alpha_n & a_n S\theta_n \\ 0 & S\alpha_n & C\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

As demonstrated in Figure 4 (a), the KUKA KR 22 R1610-2 industrial manipulator has six degrees of freedom. This manipulator is located at Horus University in Egypt's Faculty of Engineering. Figure 4 (b) depicts the manipulator dimensions, workspace, and joint-to-joint distances [4]. In Figure 4 (c), the first three axes of the robot, A1, A2, and A3, are referred to as "major axes" because they are responsible for determining the end effector position, whereas the other three axes, A4, A5, and A6, are referred to as "minor axes" because they are used for describing the end effector to the specified position, and the minor axes contribute to end effector orientation. Figure 4 (d) shows D-H parameter frames for each axis of rotation [30]. First, choose the z-axis along the joint rotation axis, then choose the x-axis based on the joint rotation direction. Industry and research utilize this. For forward and inverse kinematic analysis, this model is used [18].

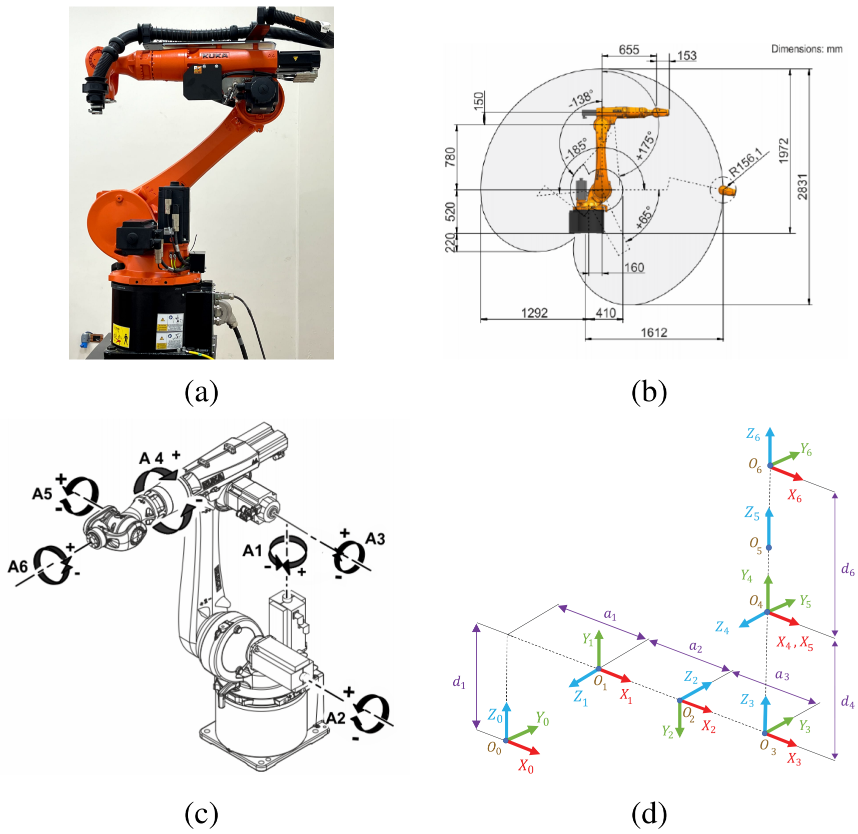


Figure 4. KUKA KR 22 R1610-2 manipulator configuration [4]: (a) Description of KUKA KR 22 R1610-2 manipulator; (b) Description of workspace and link lengths; (c) Description of major and minor axes; (d) Description of coordinate frames.

The D-H parameters of the manipulator are shown in Table 3. If it used the D-H parameter as a guide, it could figure out the matrix transformation of each frame coordinate starting, at link n and going all the way up to link n+1 [2]. Because none of the joints can be entirely rotated by 360 degrees, all the joints have constraints on the lowest and greatest angle of rotation of which they are capable. These rotational limitations of movement for various joints are provided in Table 3. These limitations are going to be used in an inverse kinematics analysis later [18].

Table 3. D-H link and joint parameters.

I_n	θ_n	d_n	a_n	α	θ_n Limitation
1	θ_1	d_1	a_1	90	$-185° \leq \theta_1 \leq 185°$
2	θ_2	0	a_2	180	$-185° \leq \theta_2 \leq 65°$
3	θ_3	0	a_3	90	$-138° \leq \theta_3 \leq 175°$
4	θ_4	d_4	0	90	$-350° \leq \theta_4 \leq 350°$
5	θ_5	0	0	-90	$-130° \leq \theta_5 \leq 130°$
6	θ_6	d_6	0	0	$-350° \leq \theta_6 \leq 350°$

To solve forward kinematics and obtain the end effector's pose, first calculate each coordinate of the transformation separately using Eq (2.2) and then plug the values from Table 3 into Eq (2.2) to obtain transformation matrices of the six joints [12]. It is initiated by analyzing the kinematics of the first joint, situated at the base of the robot, and subsequently proceeding to the second joint, then the third joint, and so on, until the final analysis of the end effector is achieved [55]. The transformation matrices of the six joints are calculated as shown in Appendix A from Eqs (5.1)–(5.6).

$$T_6^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_n^0 & P_n^0 \\ 0 & 1 \end{bmatrix} = A_1^0 \times A_2^1 \times A_3^2 \times A_4^3 \times A_5^4 \times A_6^5. \tag{2.3}$$

T_6^0 is a homogeneous transformation matrix of six degrees of freedom manipulator, and n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z, p_x, p_y, andp_z are functions of θ_1, θ_2, θ_3, θ_4, θ_5, and θ_6 where n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, and a_z express the orientation of the manipulator end effector and p_x, p_y, p_z represent the position of the end effector [56]. The forward kinematics equations (total transformations) of the KUKA KR 22 R1610-2 manipulator are calculated using Eq (2.3) as shown in Eq (2.4)–(2.15). C_n stands for $Cos\theta_n$, and S_n stands for $Sin\theta_n$, and so on.

$$n_x = -S_6\left(C_4S_1 + S_4\left(C_1S_2S_3 + C_1C_2C_3\right)\right) - C_6\left(C_5\left(S_1S_4 - C_4\left(C_1S_2S_3 + C_1C_2C_3\right)\right) - S_5\left(C_1C_2S_3 - C_1C_3S_2\right)\right). \tag{2.4}$$

$$n_y = S_6\left(C_1C_4 - S_4\left(S_1S_2S_3 + C_2C_3S_1\right)\right) + C_6\left(C_5\left(C_1S_4 + C_4\left(S_1S_2S_3 + C_2C_3S_1\right)\right) +\right.$$

$$S_5\left(C_2S_1S_3 - C_3S_1S_2\right)). \tag{2.5}$$

$$n_z = C_6\left(\left(C_2 - C_3\right)S_5 + \left(S_2 - S_3\right)C_4C_5\right) - \left(S_2 - S_3\right)S_4S_6. \tag{2.6}$$

$$o_x = S_6\left(C_5\left(S_1S_4 - C_4\left(C_1S_2S_3 + C_1C_2C_3\right)\right) - S_5\left(C_1C_2S_3 - C_1C_3S_2\right)\right) - C_6\left(C_4S_1 + \right. \\ \left. S_4\left(C_1S_2S_3 + C_1C_2C_3\right)\right). \tag{2.7}$$

$$o_y = C_6\left(C_1C_4 - S_4\left(S_1S_2S_3 + C_2C_3S_1\right)\right) - S_6\left(C_5\left(C_1S_4 + C_4\left(S_1S_2S_3 + C_2C_3S_1\right)\right) + \right. \\ \left. S_5\left(C_2S_1S_3 - C_3S_1S_2\right)\right). \tag{2.8}$$

$$o_z = -S_6\left(\left(C_2 - C_3\right)S_5 + \left(S_2 - S_3\right)C_4C_5\right) - \left(S_2 - S_3\right)C_6S_4. \tag{2.9}$$

$$a_x = S_5\left(S_1S_4 - C_4\left(C_1S_2S_3 + C_1C_2C_3\right)\right) + C_5\left(C_1C_2S_3 - C_1C_3S_2\right). \tag{2.10}$$

$$a_y = C_5\left(C_2S_1S_3 - C_3S_1S_2\right) - S_5\left(C_1S_4 + C_4\left(S_1S_2S_3 + C_2C_3S_1\right)\right). \tag{2.11}$$

$$a_z = \left(C_2 - C_3\right)C_5 - \left(S_2 - S_3\right)C_4S_5. \tag{2.12}$$

$$p_x = a_1C_1 + a_2C_1C_2 - d_4(S_2 - S_3)C_1 + a_3C_1C_2C_3 + a_3C_1S_2S_3 + d_6S_1S_4S_5 - d_6(S_2 - S_3)C_1C_5 \\ - d_6C_1C_2C_3C_4S_5 - d_6C_1C_4S_2S_3S_5. \tag{2.13}$$

$$p_y = a_1S_1 + a_2C_2S_1 - d_4\left(S_2 - S_3\right)S_1 + a_3C_2C_3S_1 - d_6C_1S_4S_5 + a_3S_1S_2S_3 - d_6\left(S_2 - S_3\right)C_5S_1 \\ - d_6C_2C_3C_4S_1S_5 - d_6C_4S_1S_2S_3S_5. \tag{2.14}$$

$$p_z = d_1 + a_2S_2 + d_4(C_2 - C_3) + a_3(S_2 - S_3) - (d_6(S_2 - S_3)(S_4 + S_5))/2 + d_6(C_2 - C_3)C_5 \\ + (d_6(S_2 - S_3)(S_4 - S_5))/2. \tag{2.15}$$

2.1.2. Inverse kinematics

Inverse kinematics is more complicated than forward kinematics [12]. Inverse kinematics plays a crucial role in trajectory planning and motion control; on the other side, forward kinematics estimates the pose of the end effector [14]. Many solutions, such as geometric and algebraic analysis, are employed to obtain the inverse kinematics when the manipulator's system structure is considered [50]. In inverse kinematics, for any location and orientation, the angles of joint θ_1, θ_2, θ_3, θ_4, θ_5, and θ_6 can be calculated [57]. To calculate the joint angles of a six degree of freedom manipulator, multiply the transformation matrix in Eq (2.3) by A_n^{-1} on both sides of the equation sequentially for n = 1, 2, 3, 4, 5, 6 as shown in Eqs (2.16)–(2.20), then solve the resulting equations derived by equating terms on both sides of matrices as shown in Eqs (2.21)–(2.42). The inverse of transformation matrices is calculated as shown in Appendix A, from Eq (5.7) to Eq (5.11).

$$A_1^{-1} \times T_6 = T_6^1. \tag{2.16}$$

$$A_2^{-1} \times A_1^{-1} \times T_6 = T_6^2. \tag{2.17}$$

$$A_3^{-1} \times A_2^{-1} \times A_1^{-1} \times T_6 = T_6^3. \tag{2.18}$$

$$A_4^{-1} \times A_3^{-1} \times A_2^{-1} \times A_1^{-1} \times T_6 = T_6^4. \tag{2.19}$$

$$A_5^{-1} \times A_4^{-1} \times A_3^{-1} \times A_2^{-1} \times A_1^{-1} \times T_6 = T_6^5. \tag{2.20}$$

The use of trigonometric equations aids in the development of simple solutions. Multiply the Eq (2.3) by A_1^{-1} on R.H.S. and L.H.S.:

$$A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2^1 \times A_3^2 \times A_4^3 \times A_5^4 \times A_6^5. \tag{2.21}$$

$$\begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & a_xS\theta_1 - a_yC\theta_1 & p_xS\theta_1 - p_yC\theta_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & S\theta_4S\theta_5 & d_6S\theta_4S\theta_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.22}$$

Equating (R_{33}) and (R_{34}) elements of both matrices:

$$d_6\left(a_xS\theta_1 - a_yC\theta_1\right) = p_xS\theta_1 - p_yC\theta_1. \tag{2.23}$$

$$\left(p_y - d_6a_y\right)C\theta_1 - \left(p_x - d_6a_x\right)S\theta_1 = 0. \tag{2.24}$$

$$\theta_1 = \tan^{-1}\left[\frac{\left(p_y - d_6a_y\right)}{\left(p_x - d_6a_x\right)}\right], \text{Both } \theta \text{ and } \theta + \pi \text{ are solutions.} \tag{2.25}$$

Multiply the Eq (2.21) by A_6^{-1} on R.H.S. and L.H.S.:

$$A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times A_6^{-1} = A_2^1 \times A_3^2 \times A_4^3 \times A_5^4. \tag{2.26}$$

$$\begin{bmatrix} . & . & . & p_xC\theta_1 - d_6(a_xC\theta_1 + a_yS\theta_1) - a_1 + p_yS\theta_1 \\ . & . & . & p_z - d_1 - a_zd_6 \\ . & . & . & . \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} . & . & . & a_2C\theta_2 + a_3C(\theta_2 - \theta_3) - d_4S(\theta_2 - \theta_3) \\ . & . & . & a_2S\theta_2 + a_3S(\theta_2 - \theta_3) - d_4C(\theta_2 - \theta_3) \\ . & . & . & . \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.27}$$

Equating (R_{14}) and (R_{24}) elements of both matrices:

$$a_2C\theta_2 + a_3C(\theta_2 - \theta_3) - d_4S(\theta_2 - \theta_3) = p_xC\theta_1 - d_6\left(a_xC\theta_1 + a_yS\theta_1\right) - a_1 + p_yS\theta_1. \tag{2.28}$$

$$a_2S\theta_2 + a_3S(\theta_2 - \theta_3) - d_4C(\theta_2 - \theta_3) = p_z - d_1 - a_zd_6. \tag{2.29}$$

$$B = \frac{\left(\sqrt{(p_x - d_6a_x)^2 + \left(p_y - d_6a_y\right)^2} - a_1\right)^2 + (p_z - d_6a_z - d_1)^2 - a_2{}^2 - a_3{}^2 - d_4{}^2}{2a_2\left(\sqrt{a^{22} + d_4{}^2}\right)}. \tag{2.30}$$

$$\theta_3 = \tan^{-1}\left[\frac{d_4}{a_3}\right] + \tan^{-1}\left[\frac{\pm\sqrt{1 - B^2}}{B}\right]. \tag{2.31}$$

$$\theta_2 = \tan^{-1}\left[\frac{p_z - d_6a_z - d_1}{\sqrt{(p_x - d_6a_x)^2 + \left(p_y - d_6a_y\right)^2} - a_1}\right] - \tan^{-1}\left[\frac{a_3S\theta_3}{a_2 + a_3C\theta_3}\right]. \tag{2.32}$$

Multiply the Eq (2.3) by A_1^{-1}, A_2^{-1}, A_3^{-1} on R.H.S. and L.H.S.:

$$A_3^{-1} \times A_2^{-1} \times A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_4^3 \times A_5^4 \times A_6^5. \tag{2.33}$$

$$\begin{bmatrix} . & . & a_zS(\theta_2 - \theta_3) + a_xC(\theta_2 - \theta_3)C\theta_1 + a_yC(\theta_2 - \theta_3)S\theta_1 & . \\ . & . & a_yC\theta_1 - a_xS\theta_1 & . \\ . & . & a_zC(\theta_2 - \theta_3) - a_xS(\theta_2 - \theta_3)C\theta_1 - a_yS(\theta_2 - \theta_3)S\theta_1 & . \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} . & . & -C\theta_4S\theta_5 & . \\ . & . & -S\theta_4S\theta_5 & . \\ . & . & C\theta_5 & . \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.34}$$

Equating (R_{13}), (R_{23}), and (R_{33}) elements of both matrices:

$$\theta_4 = \tan^{-1}\left[\frac{\left(a_y C\theta_1 - a_x S\theta_1\right)}{\left(a_z S\left(\theta_2-\theta_3\right) + a_x C\left(\theta_2-\theta_3\right) C\theta_1 + a_y C\left(\theta_2-\theta_3\right) S\theta_1\right)}\right]. \tag{2.35}$$

$$C\theta_5 = a_z C\left(\theta_2-\theta_3\right) - a_x S\left(\theta_2-\theta_3\right) C\theta_1 - a_y S\left(\theta_2-\theta_3\right) S\theta_1. \tag{2.36}$$

$$\theta_5 = \tan^{-1}\left[\frac{\pm\sqrt{1-\left(a_z C\left(\theta_2-\theta_3\right) - a_x S\left(\theta_2-\theta_3\right) C\theta_1 - a_y S\left(\theta_2-\theta_3\right) S\theta_1\right)^2}}{\left(a_z C\left(\theta_2-\theta_3\right) - a_x S\left(\theta_2-\theta_3\right) C\theta_1 - a_y S\left(\theta_2-\theta_3\right) S\theta_1\right)}\right]. \tag{2.37}$$

Multiply the Eq (2.3) by A_5^{-1}, A_4^{-1}, A_3^{-1}, A_2^{-1}, A_1^{-1} on R.H.S. and L.H.S.:

$$A_5^{-1}\times A_4^{-1}\times A_3^{-1}\times A_2^{-1}\times A_1^{-1}\times\begin{bmatrix} n_x & o_x & a_x & p_x\\ n_y & o_y & a_y & p_y\\ n_z & o_z & a_z & p_z\\ 0 & 0 & 0 & 1\end{bmatrix} = A_6^5. \tag{2.38}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot\\ R_{31} & R_{32} & \cdot & \cdot\\ 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot\\ C\theta_6 S\theta_5 & -S\theta_5 S\theta_6 & \cdot & \cdot\\ 0 & 0 & 0 & 1\end{bmatrix}. \tag{2.39}$$

$$R_{31} = n_z C\left(\theta_2-\theta_3\right) - n_x S\left(\theta_2-\theta_3\right) C\theta_1 - n_y S\left(\theta_2-\theta_3\right) S\theta_1. \tag{2.40}$$

$$R_{32} = o_z C\left(\theta_2-\theta_3\right) - o_x S\left(\theta_2-\theta_3\right) C\theta_1 - o_y\left(C\left(\theta_2-\theta_3\right)\right) S\theta_1. \tag{2.41}$$

Equating (R_{31}) and (R_{32}) elements of both matrices:

$$\theta_6 = \tan^{-1}\left[\frac{-\left(o_z C\left(\theta_2-\theta_3\right) - o_x S\left(\theta_2-\theta_3\right) C\theta_1 - o_y\left(C\left(\theta_2-\theta_3\right)\right) S\theta_1\right)}{n_z C\left(\theta_2-\theta_3\right) - n_x S\left(\theta_2-\theta_3\right) C\theta_1 - n_y S\left(\theta_2-\theta_3\right) S\theta_1}\right]. \tag{2.42}$$

2.2. *Kinematics analysis of manipulator using toolbox and software*

Inverse kinematics is a powerful tool used in robotics, animation, and computer aided design (CAD) to calculate the motion of a robotic arm or other mechanical systems. It is used to calculate the joint angles of a robotic arm or other mechanical systems to achieve a desired end effector position [38]. Inverse kinematics is a complex problem, and there are a variety of toolboxes and software packages available to help solve it, as shown in Figure 5. One of the most popular toolboxes for inverse kinematics is the robotics toolbox for MATLAB (the Peter Corke toolbox) [58]. This toolbox provides a set of functions and classes for solving inverse kinematics problems. It includes functions for calculating joint angles, forward and inverse kinematics, and dynamics. It also includes a graphical user interface (GUI) for visualizing the results of inverse kinematics calculations [24]. Another toolbox for inverse kinematics is the Open Dynamics Engine (ODE). ODE is an open-source library for simulating rigid body dynamics and has been used in robotics, animation, virtual reality, and mechatronics applications [59]. A software platform called roboanalyzer was created to solve forward and inverse kinematics using 3D modeling [60]. Robotica is a computer-aided design tool for robotic manipulators [61]. It includes a set of applications for Mathematica environments that allow for robot analysis [38]. Another program created to assist in the depiction of robot geometry is called Robot Draw [62]. Another software for inverse kinematics is Kinematics Visualizer. Kinematics Visualizer is software that is used to graphically represent the motion of objects in space. It can be used to analyze and understand the motion of mechanisms such as robots or machines [19].

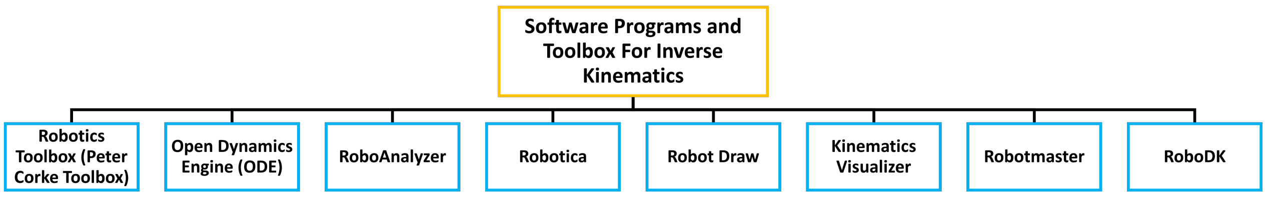

Software Programs and Toolbox For Inverse Kinematics
Robotics Toolbox (Peter Corke Toolbox)
Open Dynamics Engine (ODE)
RoboAnalyzer
Robotica
Robot Draw
Kinematics Visualizer
Robotmaster
RoboDK

Figure 5. Software programs and toolbox for inverse kinematics.

Robotmaster is a software program designed for the offline programming of industrial robots. It is used in the manufacturing and automation industries to create, edit, and optimize robot trajectories, as well as manage and simulate robot work cells [63]. RoboDK is a comprehensive software package that includes a 3D simulation environment, a library of robot models, and a library of robot programs. The 3D simulation environment allows users to visualize and interact with their robot programs in a realistic 3D environment [64]. In this paper, for simulation and solving forward and inverse kinematics, the Peter Corke toolbox was used because the routines are typically written clearly to allow for easy learning, the code is mature and serves as a benchmark for other implementations of the same algorithms, and you can always modify the function to be more effective [24]. Roboanalyzer was used too because it is easy to use and has many features like a serial manipulator with prismatic and revolute joints, D-H parameters as input, the 3D model generated based on D-H parameters, the ability to visualize D-H parameters, forward and inverse kinematics, animation with a trace of the end effector, plot graphs, and so on [60]. Furthermore, the four methods (mathematical model, Peter Corke toolbox, roboanalyzer, and particle swarm optimization) were not used in previous research on the same manipulator.

2.3. Kinematics analysis of manipulator using particle soptimization

The Particle Swarm Optimization (PSO) approach is a metaheuristic algorithm based on the concept of swarm intelligence, which is a powerful technique for solving complex mathematical engineering problems. The PSO solves problems by using multiple agents called particles to investigate and compare the quality of a population of candidate solutions. Each particle iteratively moves from one candidate solution to another based on a mathematical formula. The formula involves the particle's current state, its own local best state (lbest), and the influence from the particle with the best solution (gbest) in the search space. The formula consists of two equations [27]:

$$v_n = c_1 r_1 \left(p_{\text{lbest}} - x_n\right) + w v_n + c_2 r_2 \left(p_{\text{gbest}} - x_n\right). \tag{2.43}$$

$$x_n = v_n + x_n. \tag{2.44}$$

The first equation calculates the velocity of the particle at iteration n (v_n) based on its inertia (wv_n), where current velocity (v_n), inertia weight (w), cognitive parameter (c_1), random numbers (r_1) and (r_2), and the difference between its local best state (p_{lbest}) and its current state (x_n), as well as the difference between the global best state (p_{gbest}) and its current state (x_n). The personal influence term in the PSO algorithm is represented by $c_1 r_1 (p_{\text{lbest}} - x_n)$ and the social influence term in the PSO algorithm is represented by $c_2 r_2 \left(p_{\text{gbest}} - x_n\right)$, where c_2 is the social parameter. The second equation updates the current state of the particle (x_n) by adding its velocity (v_n) to its current state (x_n). The inertia weight (w) and cognitive parameter (c_1) are parameters that control the behavior of the PSO algorithm. The PSO algorithm uses these equations to iteratively update the states of the particles and search for the best solution in the search space.

The fitness function, represented as f_n, is defined to evaluate the local and global best positions among the particles in the swarm. The fitness function is calculated as the distance between the desired end-effector pose, referred to as P_d, and the manipulator end-effector pose associated with the particle state at iteration n, referred to as P_n, as shown in Eq (2.45) [28].

$$f_n = ||P_d - P_n|||. \tag{2.45}$$

The algorithm applies the PSO approach to solve the inverse kinematics problem for the KUKA KR 22 R1610-2 manipulator, with the desired end-effector pose set as the target position P_d.

3. Results

The mathematical model, Peter Corke toolbox, particle swarm optimization, and roboanalyzer software are applied to the studied manipulator (KUKA KR 22 R1610-2). By changing the manipulator end effector positions and orientation as shown in Table 4, which describes the input positions of the manipulator end effector, different sets of angles can be observed as outputs of inverse kinematics analyses as shown in Tables 5–7. By changing the manipulator joint angle inputs as shown in Tables 8–10. Table 4, which describes different sets of angles, the output positions, and the orientation of the manipulator end effector, can be observed as a forward kinematics analysis. KUKA KR 22 R1610-2 manipulator has the following dimensions: d_1=520 mm, d_4=655 mm, d_6=153 mm, a_1=160 mm, a_2=780 mm, a_3=150 mm.

3.1. Inverse kinematics results

To obtain the possible solutions of θ_1, θ_2, θ_3, θ_4, θ_5, θ_6 for different cases, we used four kinematics analysis methods (the Peter Corke toolbox, roboanalyzer, mathematical model, and particle swarm optimization). For the mathematical model, the values listed in Table 4 are substituted for the previous Eqs (2.24), (2.30), (2.31), (2.34), (2.36), and (2.41), which describe the position and orientation of the end effector at a certain position. The same listed values in Table 4 were used for the Peter Corke toolbox and roboanalyzer to obtain the possible solutions of θ_1, θ_2, θ_3, θ_4, θ_5, θ_6. In each case, we have eight sets of solutions for joint rotation for a given position and orientation, but we have constraints on rotation angles as shown in Table 3, so some results will be one solution and other results will be more than one solution. Six random cases and one possible solution for each case have been selected to be studied.

Table 4. End effector positions and orientation of selected cases.

Cases	P_x	P_y	P_z	n			s			a		
				x	y	z	x	y	z	x	y	z
1	1090	0	1328	1	0	0	0	1	0	0	0	1
2	-283	1442	378	1	0	0	0	1	0	0	0	1
3	1260	177	459	1	0	0	0	1	0	0	0	1
4	311	1379	1077	1	0	0	0	1	0	0	0	1
5	546	431	1025	1	0	0	0	1	0	0	0	1
6	655	-213	886	1	0	0	0	1	0	0	0	1

3.1.1. Inverse kinematics results using Peter Corke toolbox

Table 5 describes the output joints' angles for the selected cases using the Peter Corke Toolbox. One solution has been selected after avoiding the rejected solutions according to the manipulator joints' limitations.

Table 5. Set of solutions of each case using Peter Corke toolbox.

Cases	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	0	0	0	0
2	101	8	122	-180	-115	78
3	7	-47	-1	0	45	-8
4	77	-6	27	-179	-33	102
5	38	-23	-52	-1	-30	-39
6	-19	-36	-58	0	-23	18

By using GUI in MATLAB and collaborating with Peter Corke toolbox, a representative manipulator simulation can be obtained for each case, as shown in Figure 6.

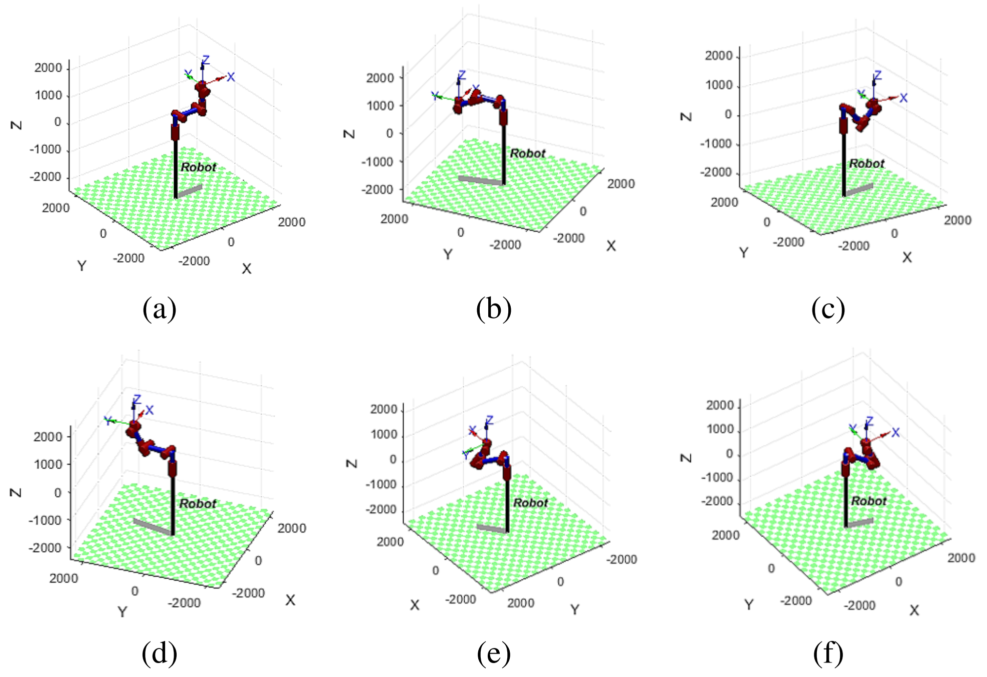


Figure 6. Simulation of solutions of each joint using Peter Corke toolbox (GUI): (a) Description of case one; (b) Description of case two; (c) Description of case three; (d) Description of case four; (e) Description of case five; (f) Description of case six.

3.1.2. Inverse kinematics results using roboanalyzer

Table 6 describes the output joints' angles for the selected cases using roboanalyzer. One solution has been selected after avoiding the rejected solutions according to the manipulator joints' limitations.

Table 6. Set of solutions for each case using roboanalyzer.

Cases	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	0	0	0	0
2	101.103	8.015	122.048	-180	-114.033	78.897
3	7.996	-46.311	-0.603	0	45.709	-7.996
4	77.291	-5.116	27.178	-180	-32.293	102.709
5	38.287	-22.094	-51.161	0	-29.067	-38.287
6	-18.014	-35.251	-57.417	0	-22.166	18.014

By using the roboanalyzer simulator platform, a representative manipulator simulation can be obtained for each case, as shown in Figure 7.

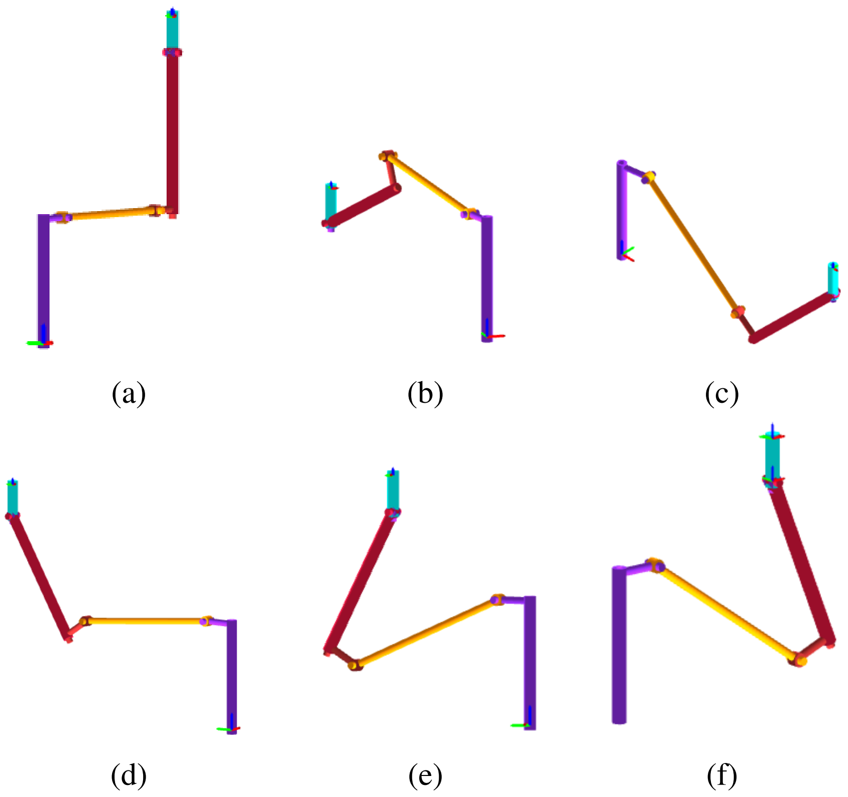


Figure 7. Simulation of solutions of each joint using roboanalyzer: (a) Description of case one; (b) Description of case two; (c) Description of case three; (d) Description of case four; (e) Description of case five; (f) Description of case six.

3.1.3. Inverse kinematics results using the mathematical model

Using the mathematical model of inverse kinematics equations as described in Eqs (2.24), (2.30), (2.31), (2.34), (2.36), and (2.41). Table 7 presents the output joints' angles for the selected cases. To avoid the rejected solutions due to the constraints of the manipulator's joints, one solution has been chosen.

Table 7. Set of solutions for each case using the mathematical model.

Cases	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	0	0	0	0
2	101.41	8.059	121.56	-179.47	-116.022	76.746
3	8.02	-46.41	-0.63	0	45.84	-7.74
4	77.382	-6.06	27.72	-180.02	-31.92	101.097
5	38.783	-22.402	-50.97	0.079	-28.677	-37.878
6	-18.76	-35.52	-57.41	0	-22.79	18.05

3.1.4. Inverse kinematics results using particle swarm optimization

Table 8 presents the angles of the output joints for the selected cases, which were determined using particle swarm optimization. A solution has been chosen after eliminating the rejected options based on the constraints of the manipulator joints.

Table 8. Set of solutions for each case using particle swarm optimization.

Cases	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	0	0	0	0
2	101	8	122	-180	-114	78.9
3	8	-46.3	-0.6	0	45.71	-7.7
4	77.3	-5.12	27.18	-180	-32.3	102.709
5	38.338	-22.074	-51.219	0	-29.146	-38.338
6	-18	-35.25	-57.4	0	-22.2	18

3.2. Forward kinematics results

In forward kinematics, to obtain the position of the end effector of the manipulator (p_x, p_y and p_z), the equations derived from the forward kinematics solution, from Eq (2.4) to Eq (2.15), are used. The outputs in Tables 5–8, which include the output angles of each joint according to inverse kinematics solutions, are used as inputs for forward kinematics for the same cases selected to be studied. Distinct positions are observed by changing the output of the inverse kinematics results of the four kinematics analysis methods (Peter Corke toolbox, roboanalyzer, mathematical model, and particle swarm optimization).

3.2.1. Forward kinematics results using Peter Corke toolbox

Table 9 shows the results of the forward kinematics of the chosen cases using the Peter Corke Toolbox. The inputs for solving forward kinematics were the output result angles of inverse kinematics solutions for the same case using Peter Corke toolbox, as indicated in Table 5.

Table 9. Input angles and output position of each case using Peter Corke toolbox.

Cases	Input						Output		
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	P_x	P_y	P_z
1	0	0	0	0	0	0	1090	0	1328
2	101	8	122	-180	-115	78	-280	1440	378
3	7	-47	-1	0	45	-8	1260	154	449
4	77	-6	27	-179	-33	102	317	1382	1059
5	38	-23	-52	-1	-30	-39	547	426	1013
6	-19	-36	-58	0	-23	18	649	-224	878

3.2.2. Forward kinematics results using roboanalyzer

Table 10 describes the output positions of the selected cases using a roboanalyzer. The output result angles of inverse kinematics solutions for the same case using the roboanalyzer shown in Table 6 were the inputs that have been used to solve forward kinematics.

Table 10. Input angles and output position of each case using roboanalyzer.

Cases	Input						Output		
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	P_x	P_y	P_z
1	0	0	0	0	0	0	1090	0	1328
2	101.103	8.015	122.05	-180	-114.033	78.897	-282.98	1442	378
3	7.996	-46.311	-0.603	0	45.709	-7.996	1259.99	176.99	459.01
4	77.291	-5.116	27.178	-180	-32.293	102.71	310.99	1379	1076.99
5	38.287	-22.094	-51.161	0	-29.067	-38.287	545.99	431	1025
6	-18.014	-35.251	-57.417	0	-22.166	18.014	654.99	-212.99	886

3.2.3. Forward kinematics results using the mathematical model

The mathematical model is used to solve forward kinematics, Table 11 shows the output positions of the selected cases. Table 7 shows the angles of the solutions to inverse kinematics for the same case using a mathematical model. These angles were used to solve forward kinematics.

Table 11. Input angles and output position of each case using the mathematical model.

Cases	Input						Output		
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	P_x	P_y	P_z
1	0	0	0	0	0	0	1090	0	1328
2	101.41	8.059	121.56	-179	-116.022	76.746	-288.9	1439.7	376.4
3	8.02	-46.41	-0.63	0	45.84	-7.74	1259.2	177.4	457.4
4	77.382	-6.06	27.72	-180	-31.92	101.1	312.2	1376.9	1075.6
5	38.783	-22.4	-50.97	0.079	-28.677	-37.878	545.6	432.5	1023.7
6	-18.76	-35.52	-57.41	0	-22.79	18.05	655.46	-212.62	883.51

3.2.4. Forward kinematics results using particle swarm optimization

The output positions of the selected cases are shown in Table 12, which displays the results of using particle swarm optimization to solve forward kinematics. Angles of solutions to inverse kinematics utilizing particle swarm optimization are shown in Table 8 for the identical instance. Forward kinematics was solved using these angles.

Table 12. Input angles and output position of each case using particle swarm optimization.

Cases	Input						Output		
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	P_x	P_y	P_z
1	0	0	0	0	0	0	1090	0	1328
2	101	8	122	-180	-114	78.9	-283.01	1441.99	378.02
3	8	-46.3	-0.6	0	45.71	-7.7	1260	177	458.98
4	77.3	-5.12	27.18	-180	-32.3	102.709	311	1379.01	1077
5	38.338	-22.074	-51.219	0	-29.146	-38.338	546.02	431	1025
6	-18	-35.25	-57.4	0	-22.2	18	655	-213	885.998

4. Discussion

The above-mentioned results will be discussed in terms of the purpose of this paper, which is to define the most accurate method for accessing the manipulator's end effector to its target destination and to compare the time factor for executing the method in six different cases.

4.1. Performance metrics

The percentage of accuracy is calculated by estimating the mean absolute percentage error (MAPE) and subtracting it from 100% using Eq (4.2) [65]. The mean absolute percentage error is computed as shown in Eq (4.1) by dividing the absolute error of a point by the observed values for that point. Then, the average of those set percentages is taken, where X_1 is the actual value (input), X_2 is the observed value (output), and n is the number of fitted points. MAPE displays the amount of prediction error in comparison to the actual value [66].

$$MAPE = \frac{\sum \frac{|X_1 - X_2|}{X_1}}{n} \times 100\%. \tag{4.1}$$

$$Percentage\ of\ Accuracy = (100\% -\ MAPE). \tag{4.2}$$

4.2. Hardware and software specifications

A laptop with an intel Core i7 2nd Generation processor, eight gigabytes (GB) of random access memory (RAM), a 64 bit operating system, Windows 10, 128 gigabytes (GB) of solid state drive (SSD) hard disk, and Intel (R) High Definition (HD) Graphics 3000 graphics card specifications are used.

4.3. Forward and inverse results discussion

First, Table 13 shows an accurate comparison between the input positions of inverse kinematics as target destinations and the output result positions of forward kinematics using the Peter Corke toolbox as the final destination of the end effector of the manipulator. The accuracy of the Peter Corke toolbox method in accessing the target can be determined through this comparison, as the average access accuracy for the six cases was 98.350%. As for the execution time of each case, 1.707 seconds were calculated as the average execution time for each case.

Table 13. Comparison between the input positions of inverse kinematics and output result positions of forward kinematics using Peter Corke toolbox.

Cases	Input			Output		
	P_x	P_y	P_z	P_x	P_y	P_z
1	1090	0	1328	1090	0	1328
2	-283	1442	378	-280	1440	378
3	1260	177	459	1260	154	449
4	311	1379	1077	317	1382	1059
5	546	431	1025	547	426	1013
6	655	-213	886	649	-224	878

Second, Table 14 illustrates an exact comparison of inverse kinematics input positions as target destinations and forward kinematics output result positions using a roboanalyzer as the manipulator's end effector's final destination. Because the average access accuracy for the six cases was 99.987%, we can define how effectively the roboanalyzer approach performs at reaching targets using this comparison. As for the execution time of each case, 0.278 seconds were calculated as the average execution time.

Table 14. Comparison between the input positions of inverse kinematics and output result positions of forward kinematics using roboanalyzer.

Cases	Input			Output		
	P_x	P_y	P_z	P_x	P_y	P_z
1	1090	0	1328	1090	0	1328
2	-283	1442	378	-282.98	1442	378
3	1260	177	459	1260	176.99	459.01
4	311	1379	1077	310.99	1379	1077
5	546	431	1025	545.99	431	1025
6	655	-213	886	654.99	-213	886

Third, Table 15 provides a detailed comparison of the input positions of inverse kinematics as target destinations and the output result positions of forward kinematics using a mathematical model as the final destination of the end effector of the manipulator. This comparison shows that the average access accuracy for the six examples was 99.719%, so it can be used to judge how well the mathematical model method works for reaching goals. As for the execution time of each case, 0.281 seconds were calculated as the average execution time for each case.

Table 15. Comparison between the input positions of inverse kinematics and output result positions of forward kinematics using the mathematical model.

Cases	Input			Output		
	P_x	P_y	P_z	P_x	P_y	P_z
1	1090	0	1328	1090	0	1328
2	-283	1442	378	-288.9	1439.7	376.4
3	1260	177	459	1259	177.4	457.4
4	311	1379	1077	312.2	1376.9	1076
5	546	431	1025	545.6	432.5	1024
6	655	-213	886	655.46	-212.62	883.51

Table 16 shows a detailed comparison of the results of forward kinematics with particle swarm optimization as the manipulator's end effector's end point and the input positions of inverse kinematics as target targets. In order to evaluate the efficacy of the particle swarm optimization method in reaching objectives, this comparison reveals that the average access accuracy for the six instances was 99.999%. The average execution time for each case was calculated to be 0.904 seconds.

Table 16. Comparison between the input positions of inverse kinematics and output result positions of forward kinematics using the particle swarm optimization.

Cases	Input			Output		
	P_x	P_y	P_z	P_x	P_y	P_z
1	1090	0	1328	1090	0	1328
2	-283	1442	378	-283.01	1441.99	378.02
3	1260	177	459	1260	177	458.98
4	311	1379	1077	311	1379.01	1077
5	546	431	1025	546.02	431	1025
6	655	-213	886	655	-213	885.998

4.4. End-effector pose error

The position error of the end-effector can be determined using the values P_x, P_y, and P_z in Eq (2.3) by calculating the Euclidean distance between the actual position and the evaluated position, represented by Eq (4.3) [45]. Through Tables 13–16, and utilizing Eq (4.3), the end-effector pose error was computed for each of the four methods. Table 17 presents the result comparison of the end-effector pose errors among the four methods.

$$P_{\text{error}} = \sqrt{(p_{x,E} - p_{x,A})^2 + \left(p_{y,E} - p_{y,A}\right)^2 + (p_{z,E} - p_{z,A})^2}. \tag{4.3}$$

Table 17. End-effector pose error result comparison.

Cases	P_{error}			
	Peter Cork	Roboanalyze	Mathematical Model	PSO
1	0	0	0	0
2	3.606	0.020	6.531	0.024
3	25.080	0.014	1.929	0.020
4	19.209	0.010	2.617	0.010
5	13.038	0.010	1.847	0.020
6	14.866	0.010	2.560	0.002
Average	12.633	0.011	2.581	0.012

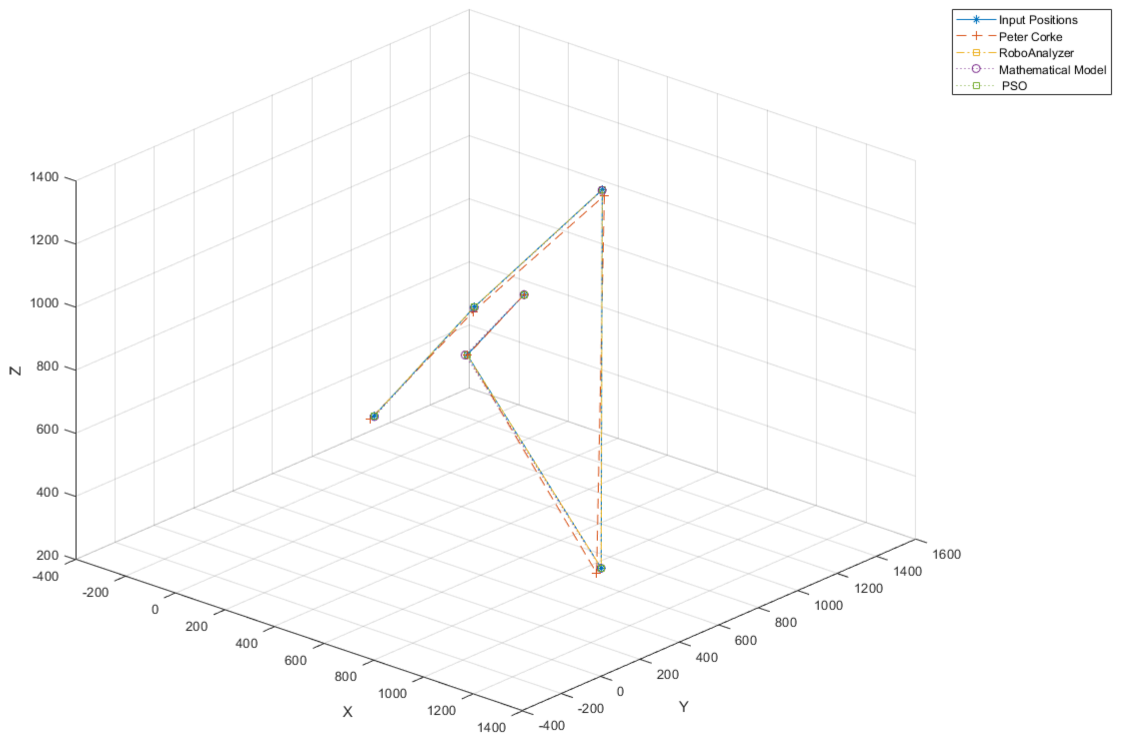
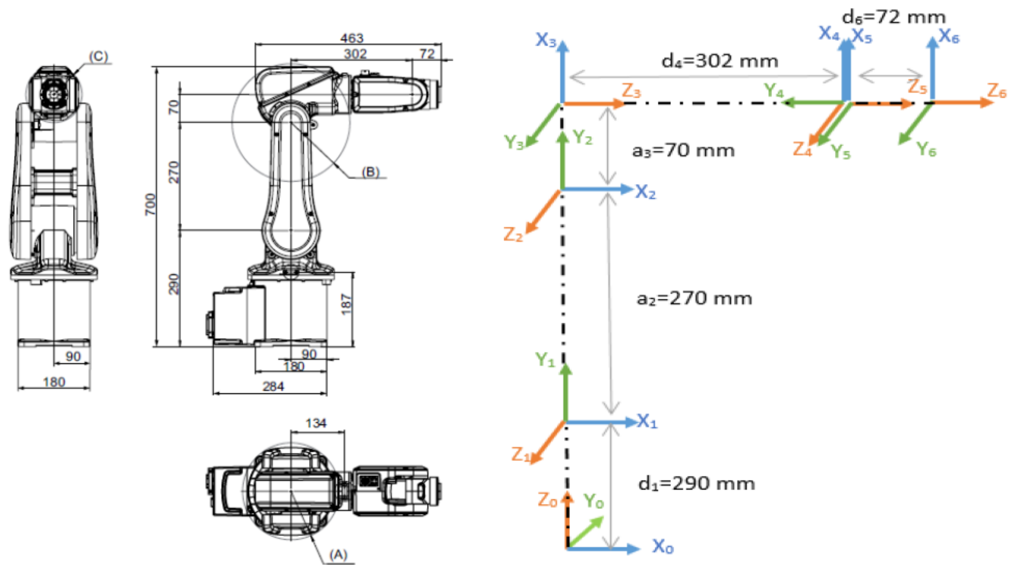


Figure 8. Comparison between the input positions of inverse kinematics and output result positions of forward kinematics of the four method configurations.

When the accuracy of access for each of the four studied methods is known and compared, it is clear that PSO is the most accurate in terms of the average accuracy of reaching each case by 99.999% with execution time 0.904 second, compared to roboanalyzer with 99.987% with an execution time of 0.278 seconds, the mathematical model with 99.719% with an execution time of 0.281 seconds, and the Peter Corke toolbox with 98.35% with an execution time of 1.707 seconds. Following the computation of pose errors, it was observed that the RoboAnalyzer method exhibited the lowest average pose error, measuring at 0.011 mm. Subsequently, the PSO method ranked second, with a marginal difference, registering at 0.12 mm. The mathematical modeling method followed in third place, with a pose error value of 2.581 mm. Lastly, the Peter Cork method demonstrated the largest pose error, measuring 12.633 mm. Figure 8 describes the six points connected as a path for each of the four methods used compared with the input positions of inverse kinematics. The selection of the model depends on the specific requirements of the application. Choosing the PSO and RoboAnalyzer approaches is recommended when considering accuracy. On the other hand, the Peter Corke method is preferred when the most important factor is simplicity in the structure of the code. When it comes to applications where the speed of execution is crucial, the best choice is the RoboAnalyzer and mathematical model.

4.5. *Universality and applicability*

An additional case study was proposed to ensure the applicability of all the methods studied in this paper to any type of mechanism. These methods were applied to a 6-DOF manipulator called the IRB 120 ABB. Figure 9 (a) shows the work space and dimensions of the manipulator. Figure 9 (b) also shows the cartisian frames for each joint of the manipulator. Table 18 present the D-H parameters for IRB 120 ABB Robot.



(a) (b)

Figure 9. IRB 120 ABB robot configuration: (a) Work space; (b) Cartisian frames.

Table 18. IRB 120 ABB robot D-H parameters.

I_n	θ_n	d_n	a_n	α	θ_n Limitation
1	θ_1	0.290	0	90	$-165° \leq \theta_1 \leq 165°$
2	θ_2	0	0.270	0	$-110° \leq \theta_2 \leq 110°$
3	θ_3	0	0.070	90	$-110° \leq \theta_3 \leq 70°$
4	θ_4	0.302	0	-90	$-160° \leq \theta_4 \leq 160°$
5	θ_5	0	0	90	$-120° \leq \theta_5 \leq 120°$
6	θ_6	0.072	0	0	$-400° \leq \theta_6 \leq 400°$

The homogeneous matrix parameters, which describe the position and orientation of the manipulator end effector as a forward kinematic, are represented in Table 19. And it will be taken as input for the four studied methods. Each of the four studied methods was applied to two cases representing two different positions within the accessible workspace in order to ensure the universality of the studied methods. The results for the four methods were compared with the inputs in Table 20. The main absolute square error was used to evaluate the accuracy of the results using these methods, just as we had previously done.

Table 19. Positions and orientations of the end effectors for the two stated cases.

Cases	P_x	P_y	P_z	n			s			a		
				x	y	z	x	y	z	x	y	z
1	130	27	510	1	0	0	0	-1	0	0	0	-1
2	121	-135	314	1	0	0	0	-1	0	0	0	-1

Table 20. Comparison between the input positions of inverse kinematics and the output positions of the four methods.

Methods	Input			Output		
	P_x	P_y	P_z	P_x	P_y	P_z
Mathematical model	130	27	510	127.73	25.42	508.34
	121	-135	314	122.26	-132.13	313.57
Peter Corke	130	27	510	127	26	506
	121	-135	314	117	-131	312
RoboAnalyzer	130	27	510	131.038	26.91	509.67
	121	-135	314	120.976	-134.96	313.52
PSO	130	27	510	129.98	27.01	510
	121	-135	314	121	-134.99	313.97

By evaluating the accuracy of the results using the absolute square error, it was found that the most accurate method was PSO with an accuracy of 99.988%, then, RoboAnalyzer came in second place with an accuracy of 99.767%, the mathematical model came in third place with an accuracy of 98.129%, and finally Peter Corke ranked last with an accuracy of 97.716%. The quality of the PSO method confirms the previous conclusions from the previous studied case, demonstrating the universality of both the methods and the results.

5. Conclusions

In this paper, a comprehensive kinematics analysis of the KUKA KR 22 R1610-2 industrial six degree of freedom manipulator is conducted utilizing four distinct methods: the Peter Corke toolbox and GUI implemented in MATLAB, the roboanalyzer software platform, mathematical model, and particle swarm optimization implemented in MATLAB. The accuracy of the inverse kinematics solutions, represented as target destinations, was evaluated by comparing them to the output positions obtained through forward kinematics, represented as final destinations, in six different cases. The accuracy was determined using the mean absolute percentage error. The PSO method emerged as the most accurate approach for achieving the target, followed by the RoboAnalyzer method, then the Peter Corke method, and finally the mathematical model method. The RoboAnalyzer method was the most effective in terms of execution time, followed by the mathematical model method, PSO, and the Peter Corke method.

The future work of this research is to study the analysis of continuum manipulators under external forces, which falls into the realm of dynamics. Closed-form solutions to inverse kinematics problems for reconfigurable robots are also among the future work. The advanced machine learning techniques can be applied to the inverse kinematics problem solution. Finally, we plan to explore enhancements to the iterative cycle methodology, with a specific focus on addressing factors that influence solution accuracy, including the end-effector pose error.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia, for funding this research work through project number 445-9-953.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. M. A. A. Mousa, A. T. Elgohr, H. A. Khater, Trajectory optimization for a 6 DOF robotic arm based on reachability time, *Annals of Emerging Technologies in Computing*, **8** (2024), 22–35. https://doi.org/10.33166/AETiC.2024.01.003
2. A. Krisbudiman, T. H. Nugroho, A. Musthofa, Analysis industrial robot arm with Matlab and RoboAnalyzer, *International Journal of Advanced Engineering, Management and Science*, **7** (2021), 75–80. https://doi.org/10.22161/ijaems.73.10
3. J. W. Lee, G. T. Park, J. S. Shin, J. W. Woo,, Industrial robot calibration method using denavit-Hatenberg parameters, *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, Jeju, Korea (South), 2017, 1834–1837. https://doi.org/10.23919/ICCAS.2017.8204265
4. Z. Y. He, J. C. Li, Six-degree-of-freedom robot trajectory planning based on MATLAB, *International Conference on Automation, Robotics and Computer Engineering (ICARCE)*, Wuhan, China, 2022, 1–3. https://doi.org/10.1109/ICARCE55724.2022.10046483

5. KR 22 R1610-2-KUKA AG. Available from: https://www.infinitysolutions.co.jp/wprenew/wp-content/uploads/2021/02/kr_cybertech_en.pdf.
6. D. Constantin, M. Lupoae, C. Baciu, B. D. Ilie, Forward kinematic analysis of an industrial robot, *International Conference on Mechanical Engineering (ME 2015)*, Vienna, Austria 2015, 90–95.
7. W. Chen, X. Li, H. L. Ge, L. Wang,Y. H. Zhang, Trajectory planning for spray painting robot based on point cloud slicing technique, *Electronics*, **9** (2020), 908. https://doi.org/10.3390/electronics9060908
8. T. P. Singh, P. Suresh, S. Chandan, Forward and inverse kinematic analysis of robotic manipulators, *International Research Journal of Engineering and Technology*, **4** (2017), 1459–1469.
9. J. Villalobos, I. Y. Sanchez, F. Martell, Singularity analysis and complete methods to compute the inverse kinematics for a 6-DOF UR/TM-type robot, *Robotics*, **11** (2022), 137. https://doi.org/10.3390/robotics11060137
10. D. Sivasamy, M. D. Anand, K. A. Sheela, Robot forward and inverse kinematics research using MATLAB, *International Journal of Recent Technology and Engineering* , **8** (2019), 29–35 https://doi.org/10.35940/ijrte.b1006.0782s319
11. A. Patwardhan, A. Prakash, R. G. Chittawadigi, Kinematic analysis and development of simulation software for nex dexter robotic manipulator, *Procedia Computer Science*, **133** (2018), 660–667. https://doi.org/10.1016/j.procs.2018.07.101
12. M. Kaur, S. Sondhi, V. K. Yanumula, Kinematics analysis and jacobian calculation for six degrees of freedom robotic arm, *2020 IEEE 17th India Council International Conference (INDICON)*, New Delhi, India, 2020, 1–6. https://doi.org/10.1109/INDICON49873.2020.9342093
13. S. KuCuk, Z. Bingul, The inverse kinematics solutions of industrial robot manipulators, *Proceedings of the IEEE International Conference on Mechatronics*, Istanbul, Turkey, 2004, 274–279. https://doi.org/10.1109/ICMECH.2004.1364451
14. M. G. Krishnan, S. Ashok, Kinematic analysis and validation of an industrial robot manipulator, *2019 IEEE Region 10 Conference (TENCON)*, Kochi, India, 2019, 1393–1399. https://doi.org/10.1109/TENCON.2019.8929712
15. D. P. Nayak, K. C Rath, Robot kinematics analysis with simulation of manipulator trajectory utilising the DH parameter, *YMER*, **21** (2022), 273–285. https://doi.org/10.37896/ymer21.08%2F24
16. A. El-Sherbiny, M. A. Elhosseini, A. Y Haikal, A comparative study of soft computing methods to solve inverse kinematics problem, *Ain Shams Eng. J.*, **9** (2018), 2535–2548. https://doi.org/10.1016/j.asej.2017.08.001
17. I. Chavdarov, B. Naydenov, Algorithm for determining the types of inverse kinematics solutions for sequential planar robots and their representation in the configuration space, *Algorithms*, **15** (2022), 469. https://doi.org/10.3390/a15120469
18. S. S. Chauhan, A. K. Khare, Kinematic analysis of the ABB IRB 1520 industrial robot using RoboAnalyzer software, *Evergreen*, **7** (2022), 510–518. https://doi.org/10.5109/4150470

19. M. W. Spong, S. Hutchinson, M. Vidyasagar, *Robot modeling and control*, 2 Eds., Hoboken: Wiley, 2020. https://doi.org/10.1109/MCS.2006.252815

20. B. Siciliano, O. Khatib, Robotics and the handbook, In: *Springer handbook of robotics*, Cham: Springer, 2016, 1–6. https://doi.org/10.1007/978-3-319-32552-1

21. Z. Bingul, H. M. Ertunc, C. Oysu, Comparison of inverse kinematics solutions using neural network for 6R robot manipulator with offset, *2005 ICSC Congress on Computational Intelligence Methods and Applications*, Istanbul, Turkey, 2005, 5. https://doi.org/10.1109/CIMA.2005.1662342

22. P. Corke, *Robotics and control: Fundamental algorithms in MATLAB*, Cham: Springer, 2022. https://doi.org/10.1007/978-3-030-79179-7

23. I. Mehta, K. Bimbraw, R. G. Chittawadigi, S. K. Saha, A teach pendant to control virtual robots in Roboanalyzer, *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*, Amritapuri, India, 2016, 1–6. https://doi.org/10.1109/RAHA.2016.7931881

24. P. I. Corke, A robotics toolbox for MATLAB, *IEEE Robot. Autom. Mag.*, **3** (1996), 24–32. https://doi.org/10.1109/100.486658

25. A. El-Sherbiny, M. A. Elhosseini, A. Y. Haikal, A new ABC variant for solving inverse kinematics problem in 5 DOF robot arm, *Appl. Soft Comput.*, **73** (2018), 24–38. https://doi.org/10.1016/j.asoc.2018.08.028

26. M. A. A. Mousa, A. T. Elgohr, H. A.Khater, Path planning for a 6 DoF robotic arm based on whale optimization algorithm and genetic algorithm, *J. Eng. Res.*, **7** (2023), 160–168. https://doi.org/10.21608/erjeng.2023.237586.1256

27. H. Danaci, L. A. Nguyen, T. L. Harman, M. Pagan, Inverse kinematics for serial robot manipulators by particle swarm optimization and POSIX threads implementation, *Appl. Sci.*, **13** (2023), 4515. https://doi.org/10.3390/app13074515

28. S. Djeffal, C. Mahfoudi, Inverse kinematic model of multi-section continuum robots using particle swarm optimization and comparison to four meta-heuristic approaches, *SIMULATION*, **99** (2023), 817–830. https://doi.org/10.1177/00375497231164645

29. R. Sadanand, R. G. Chittawadigi, R. P. Joshi, S. K Saha, Virtual robots module: An effective visualization tool for robotics toolbox, *Proceedings of the 2015 Conference on Advances In Robotics*, Goa, India, 2015, 1–6. https://doi.org/10.1145/2783449.2783475

30. A. N. Barakat, K. A. Gouda, K. A Bozed, Kinematics analysis and simulation of a robotic arm using MATLAB., *2016 4th International Conference on Control Engineering & Information Technology (CEIT)*, Hammamet, Tunisia, 2016, 1–5. https://doi.org/10.1109/CEIT.2016.7929032

31. Y. L. Bao, K. M. Hamza, K. D. Kallu, S. J. Abbasi, M. C. Lee, A study on 7-DOF manipulator control by using MATLAB robotics toolbox, *2019 16th International Conference on Ubiquitous Robots*, Jeju, Korea, 2019, 24–27.

32. D. T. Long, T. V. Binh, R. V. Hoa, L. V. Anh, N. V. Toan, Robotic arm simulation by using matlab and robotics toolbox for industry application, *International Journal of Electronics and Communication Engineering*, **7** (2020), 1–4. https://doi.org/10.14445/23488549%2Fijece-v7i10p101

33. D. Q. Zhang, Z. Y. Peng, G. S. Ning, X. Han, Positioning accuracy reliability of industrial robots through probability and evidence theories, *J. Mech. Des.*, **143** (2021), 011704. https://doi.org/10.1115/1.4047436

34. D. Q. Zhang, X. Han, Kinematic reliability analysis of robotic manipulator, *J. Mech. Des.*, **142** (2020), 044502. https://doi.org/10.1115/1.4044436

35. D. Q. Zhang, S. S. Shen, J. H. Wu, F. Wang, X. Han, Kinematic trajectory accuracy reliability analysis for industrial robots considering intercorrelations among multi-point positioning errors, *Reliab. Eng. Syst. Safe.*, **229** (2023), 108808. https://doi.org/10.1016/j.ress.2022.108808

36. J. Bahuguna, R. G. Chittawadigi, S. K. Saha, Teaching and learning of robot kinematics using RoboAnalyzer software, In: *Proceedings of conference on advances in robotics*, New York: Association for Computing Machinery, 2013, 1–6. https://doi.org/10.1145/2506095.2506142

37. V. Gupta, R. G. Chittawadigi, S. K. Saha, RoboAnalyzer: Robot visualization software for robot technicians, In: *Proceedings of the advances in robotics*, Association for Computing Machinery, 2017, 1–5. https://doi.org/10.1145/3132446.3134890

38. R. S. Othayoth, R. G. Chittawadigi, R. P. Joshi, S. K. Saha, Robot kinematics made easy using RoboAnalyzer software, *Comput. Appl. Eng. Educ.*, **25** (2017), 669–680. https://doi.org/10.1002/cae.21828

39. P. Chang, A closed-form solution for the control of manipulators with kinematic redundancy, *1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, 1986, 9–14. https://doi.org/10.1109/ROBOT.1986.1087725

40. P. Chang, A closed-form solution for inverse kinematics of robot manipulators with redundancy, *IEEE Journal on Robotics and Automation*, **3** (1987), 393–403. https://doi.org/10.1109/jra.1987.1087114

41. I. M. Chen, Y. Gao, Closed-form inverse kinematics solver for reconfigurable robots, *IEEE International Conference on Robotics and Automation*, Seoul, South Korea, 2001, 2395–2400. https://doi.org/10.1109/ROBOT.2001.932980

42. J. Gao, B. Zhou, B. Zi, S. Qian, P. Zhao, Kinematic uncertainty analysis of a Cable-Driven parallel robot based on an error transfer model, *J. Mechanisms Robotics*, **14** (2022), 051008. https://doi.org/10.1115/1.4053219

43. D. Q. Zhang, Z. H. Han, F. Wang, X. Han, Proficiency of statistical moment-based methods for analysis of positional accuracy reliability of industrial robots, *Int. J. Mech. Mater. Des.*, **17** (2021), 403–418. https://doi.org/10.1007/s10999-021-09532-2

44. Q. Q. Zhao, J. K. Guo, D. T. Zhao, D. W. Yu, J. Hong, Time-dependent system kinematic reliability analysis for robotic manipulators, *J. Mech. Des.*, **143** (2021), 041704. https://doi.org/10.1115/1.4049082

45. J. A. Abdor-Sierra, E. A. Merchán-Cruz, R. G. Rodríguez-Cañizo, A comparative analysis of metaheuristic algorithms for solving the inverse kinematics of robot manipulators, *Results in Engineering*, **16** (2022), 100597. https://doi.org/10.1016/j.rineng.2022.100597

46. C. J. Liu, X. Y. Wang, H. S. Jiang, X. Y. Wang, H. Y. Guo, Inverse kinematics solution of manipulator based on IPSO-BPNN, *2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, Chengdu, China, 2022, 175–179. https://doi.org/10.1109/PRAI55851.2022.9904288

47. A. X. Wu, Z. P. Shi, Y. D. Li, M. H. Wu, Y. Guan, J. Zhang, et al., Formal kinematic analysis of a general 6R manipulator using the screw theory, *Math. Probl. Eng.*, **2015** (2015), 549797. https://doi.org/10.1155/2015/549797

48. Q. D. Li, H. H. Ju, P. F. Xiao, Kinematics analysis and optimization of 6R manipulator, *IOP Conf. Ser.: Mater. Sci. Eng.*, **816** (2020), 012016. https://doi.org/10.1088/1757-899X/816/1/012016

49. M. T. Nguyen, C. Yuan, J. H. Huang, Kinematic analysis of a 6-DOF robotic arm, In: *Mechanisms and machine science*, Cham: Springer, 2019, 2965–2974. https://doi.org/10.1007/978-3-030-20131-9_292

50. H. A. R. Akkar, A. N. A-Amir, Kinematics analysis and modeling of 6 degree of freedom robotic arm from DFROBOT on Labview, *Research Journal of Applied Sciences, Engineering and Technology*, **7** (2016), 569–575. https://doi.org/10.19026/rjaset.13.3016

51. A. Talli, A. C. Giriyapur, Kinematic analysis and simulation of industrial robot based on RoboAnalyzer, In: *Recent advances in mechanical infrastructure*, Singapore: Springer, 2021, 473–483. https://doi.org/10.1007/978-981-33-4176-0_40

52. J. Z. Vidaković, M. P. Lazarević, V. M. Kvrgić, Z. Z. Dančuo, G. Z. Ferenc, Advanced quaternion forward kinematics algorithm including overview of different methods for robot kinematics, *FME Trans.*, **42** (2014), 189–199. https://doi.org/10.5937/fmet1403189v

53. T. Aravinthkumar, M. Suresh, B. Vinod, Kinematic analysis of 6 DOF articulated robotic arm, *International Research Journal of Multidisciplinary Technovation*, **3** (2021), 1–5. https://doi.org/10.34256/irjmt2111

54. K. S. Gaeid, A. F. Nashee, I. A. Ahmed, M. H. Dekheel, Robot control and kinematic analysis with 6DoF manipulator using direct kinematic method, *Bulletin of Electrical Engineering and Informatics*, **10** (2021), 70–78. https://doi.org/10.11591/eei.v10i1.2482

55. M. Dahari, J. D. Tan, Forward and inverse kinematics model for robotic welding process using KR-16KS KUKA robot, *2011 Fourth International Conference on Modeling, Simulation and Applied Optimization*, Kuala Lumpur, Malaysia, 2011, 1–6. https://doi.org/10.1109/ICMSAO.2011.5775598

56. J. X. Yu, D. Z. You, J. S. Liu, Analysis of inverse kinematics method for six degrees of freedom manipulator based on MATLAB, *2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE)*, Beijing, China, 2017, 211–215. https://doi.org/10.1109/CCSSE.2017.8087925

57. S. Asif, P. Webb, Kinematics analysis of 6-DoF articulated robot with spherical wrist, *Math. Probl. Eng.*, **2021** (2021), 6647035. https://doi.org/10.1155/2021/6647035

58. P. Corke, MATLAB toolboxes: Robotics and vision for students and teachers, *IEEE Robot. Autom. Mag.*, **14** (2007), 16–17. https://doi.org/10.1109/m-ra.2007.912004

59. E. Drumwright, J. Hsu, N. Koenig, D. Shell, Extending open dynamics engine for robotics simulation, In: *Simulation, modeling, and programming for autonomous robots*, Berlin: Springer, 2010, 38–50. https://doi.org/10.1007/978-3-642-17319-6_7

60. N. A. S. Laksana, R. Ariawan, U. S. Jati, J. Sodikin, Ulikaryani, Analisis kinematik singularty pada manipulator 7 DOF dengan software simulasi RoboAnalyzer, *Infotekmesin*, **13** (2022), 265–271. https://doi.org/10.35970/infotekmesin.v13i2.1538

61. J. F. Nethery, M. W.Spong, Robotica: A mathematica package for robot analysis, *IEEE Robot. Autom. Mag.*, **1** (1994), 13–20. https://doi.org/10.1109/100.296449

62. M. F. Robinette, R. Manseur, Robot-draw, an internet-based visualization tool for robotics education, *IEEE T. Educ.*, **44** (2001), 29–34. https://doi.org/10.1109/13.912707

63. M. Morozov, S. G. Pierce, C. N. MacLeod, C. Mineo, R. Summan, Off-line scan path planning for robotic NDT, *Measurement*, **122** (2018), 284–290. https://doi.org/10.1016/j.measurement.2018.02.020

64. A. Garbev, A. Atanassov, Comparative analysis of RoboDK and robot operating system for solving diagnostics tasks in off-line programming, *2020 International Conference Automatics and Informatics (ICAI)*, Varna, Bulgaria, 2020, 1–5. https://doi.org/10.1109/ICAI50593.2020.9311332

65. M. K. Elshaarawy, A. K. Hamed, Predicting discharge coefficient of triangular side orifice using ANN and GEP models, *Water Science*, **38** (2024), 1–20. https://doi.org/10.1080/23570008.2023.2290301

66. U. Khair, H. Fahmi, S. A. Hakim, R. Rahim, Forecasting error calculation with mean absolute deviation and mean absolute percentage error, *J. Phys.: Conf. Ser.*, **930** (2017), 012002. https://doi.org/10.1088/1742-6596/930/1/012002

Appendix A

The transformation matrices of the six joints.
A_1^0 is a the transformation matrix for $\theta = \theta_1, d = d_1, a = a_1, \alpha_1 = 90^\circ$

$$A_1^0 = A_1 = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & a_1 C\theta_1 \\ S\theta_1 & 0 & -C\theta_1 & a_1 S\theta_1 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.1}$$

A_2^1 is the transformation matrix for $\theta = \theta_2, d = 0, a = a_2, \alpha_2 = 180^\circ$

$$A_2^1 = A_2 = \begin{bmatrix} C\theta_2 & S\theta_2 & 0 & a_2C\theta_2 \\ S\theta_2 & -C\theta_2 & 0 & a_2S\theta_2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.2}$$

A_3^2 is the transformation matrix for $\theta = \theta_3, d = 0, a = a_3, \alpha_3 = 90^\circ$

$$A_3^2 = A_3 = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & a_3C\theta_3 \\ S\theta_3 & 0 & -C\theta_3 & a_3S\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.3}$$

A_4^3 is the transformation matrix for $\theta_=\theta_4, d = d_4, a = 0, \alpha_4 = 90^\circ$

$$A_4^3 = A_4 = \begin{bmatrix} C\theta_4 & 0 & S\theta_4 & 0 \\ S\theta_4 & 0 & -C\theta_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.4}$$

A_5^4 is the transformation matrix for θ_5 ,$d, d = 0$,$a = 0$,$\alpha_5 = -90^\circ$

$$A_5^4 = A_5 = \begin{bmatrix} C\theta_5 & 0 & -S\theta_5 & 0 \\ S\theta_5 & 0 & C\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.5}$$

A_6^5 is the transformation matrix for $\theta_=\theta_6, d = d_6, a = 0, \alpha_6 = 0^\circ$

$$A_6^5 = A_6 = \begin{bmatrix} C\theta_6 & -S\theta_6 & 0 & 0 \\ S\theta_6 & C\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.6}$$

The inverse of transformation matrices of the six joints:

$$A_1^{-1} = \begin{bmatrix} C\theta_1 & S\theta_1 & 0 & -a_1 \\ 0 & 0 & 1 & -d_1 \\ S\theta_1 & -C\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.7}$$

$$A_2^{-1} = \begin{bmatrix} C\theta_2 & S\theta_2 & 0 & -a_2 \\ S\theta_2 & -C\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.8}$$

$$A_3^{-1} = \begin{bmatrix} C\theta_3 & S\theta_3 & 0 & -a_3 \\ 0 & 0 & 1 & 0 \\ S\theta_3 & -C\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.9}$$

$$A_4^{-1} = \begin{bmatrix} C\theta_4 & S\theta_4 & 0 & 0 \\ 0 & 0 & 1 & -d_4 \\ S\theta_4 & -C\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

$$A_5^{-1} = \begin{bmatrix} C\theta_5 & S\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -S\theta_5 & C\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.11}$$