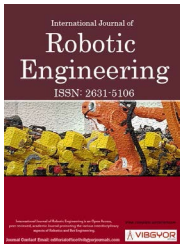
DOI: 10.35840/2631-5106/4137

VIBGYOR ePress

International Journal of Robotic Engineering

In-Depth Analysis of Kinematic, Dynamic, and Control Aspects of a 4-Axis SCARA Robot Manipulator



ISSN: 2631-5106

G. Bala Muniyandi[*]

Department of Mechanical Engineering, University College of Engineering, Tiruchirappalli, Tamil Nadu, India

Abstract

This paper investigates the capabilities of the 4-axis SCARA ABB IRB 930 robot, a pivotal machine in industrial automation renowned for its high payload capacity. Emphasizing cycle time and payload capacity, the exceptional motion control and productivity features of the IRB 930 are highlighted. A comprehensive mathematical model for kinematics and dynamics is presented, implemented in MATLAB for accuracy. The versatility of the IRB 930 across various applications, from assembly to screw driving, makes it an ideal subject for kinematic analysis. Detailed discussions cover mathematical modelling, the methodology for deriving the Jacobian, and dynamic analysis, ultimately leading to a specific control model for this robot. This research enhances understanding of the IRB 930 while also contributing universally applicable methodologies for diverse robotic platforms. Findings ranging from forward kinematics to dynamic control propel advancements in the field of robotics and control methodologies.

Keywords

SCARA Robot, Kinematic analysis, Dynamic modelling, Jacobian analysis, Motion control, Robot manipulator

Introduction

The evolution of industrial automation has ushered in a new era of efficiency and precision, prominently led by programmable manipulators known as industrial robots. These sophisticated machines navigate predefined sequences of motions, exhibiting the remarkable ability to execute tasks with unwavering precision over prolonged periods [1]. The contemporary landscape of industrial production places an escalating demand on the flexibility of manufacturing processes, necessitating a closer examination of the design and adaptability of robotic systems both in the present and in the future. One pivotal aspect that profoundly influences the effectiveness of these robots is their cycle time and payload capacity, key parameters intricately linked to overall productivity [2]. A robot's payload capacity, defined as the amount of mass its wrist can support, extends beyond the mere weight of workpieces handled by the robot. It encompasses the weight of any end-of-arm tooling (EOAT) and bracketing integrated with the robot wrist. Simultaneously, the robot cycle time, representing the duration it takes for a robot to complete one full cycle of its programmed task, includes both the value added time when the robot is actively moving or performing the operation

***Corresponding author:** *G. Bala Muniyandi, Department of Mechanical Engineering, University College of Engineering, Tiruchirappalli, Tamil Nadu, India*

Accepted: April 26, 2024; **Published:** April 28, 2024

Citation: Muniyandi GB (2024) In-Depth Analysis of Kinematic, Dynamic, and Control Aspects of a 4-Axis SCARA Robot Manipulator. Int J Robot Eng 7:037

and any non-value-added wait time. Together, these factors contribute significantly to the overall effectiveness and productivity of industrial robotic systems [3].

In the realm of industrial robotic solutions, the IRB 930 SCARA stands out as a high-payload robot with a capacity of 12-kg or 22-kg. This exceptional SCARA robot enhances throughput by up to 10%, demonstrating class-leading speed, accuracy, internal cabling, and extraordinary downward force [4]. The IRB 930's superior cycle time and payload capacity make it a noteworthy choice for optimizing industrial processes, showcasing its process in delivering enhanced efficiency and productivity.

The IRB 930 SCARA, a versatile powerhouse designed for fast point-to-point applications, demonstrates exceptional strength in handling substantial payloads. Its adaptability extends to delicate tasks through the integration of sophisticated tools and grippers. With an impressive cycle time of 0.38 seconds and a remarkable repeatability deviation position of only 0.01 mm, the IRB 930 excels in motion control, empowering heightened hourly production rates while maintaining high-quality manufacturing standards. Notably, its maximum downward force of 250N sets it apart, providing more than double the average screw driving capacity of other robots. The IRB 930's spatial efficiency, saving up to 20% of the typically required space above the upper arm, enhances overall flexibility in industrial setups. Figure 1 presents a visual representation of the IRB 930 model.

Figure 1: Visual representation of the 4-axis SCARA IRB 930 manipulator.

In the present paper, a thorough mathematical model for kinematics and dynamics of 4 axes SCARA IRB 930 robot is presented. This model encompasses the full mathematical development for both forward and inverse kinematic equations, as well as dynamic equations of motion. Additionally, the utilization of the MATLAB environment to implement and verify the mathematical models presented. Specifically, the Symbolic Math Toolbox within MATLAB is employed for deriving the complex mathematical formulations required for kinematic and dynamic analysis. By leveraging the capabilities of MATLAB, the paper ensures rigorous analysis and accurate verification of the developed models, enhancing their reliability and usefulness for researchers and engineers. The integration of MATLAB not only facilitates the implementation process but also enables comprehensive exploration and understanding of the behaviour and capabilities of the ABB IRB 930 robot across various operational scenarios.

The paper is thoughtfully organized for a comprehensive exploration of the 4-axis SCARA ABB IRB 930 robot. In Section 2, the discussion begins by introducing the robot and diving deep into the mathematical modelling for its kinematics. Moving on to Section 3, the methodology for deriving the Jacobian of the robotic manipulator is outlined. Additionally, the paper discusses singularity analysis and introduces a velocity propagation model. In Section 4, the focus shifts to a dynamic analysis of the robot, drawing insights from the established mathematical model. Building upon this groundwork, Section 5 elaborates on the control model crafted specifically for the 4-axis SCARA ABB IRB 930 robot. Concluding the study in Section 6, a synthesis of key findings and insights garnered throughout the investigation is offered. This contribution to the research field is significant as it provides a comprehensive mathematical framework for analyzing the kinematics and dynamics of the ABB IRB 930 robot, a pivotal component in modern industrial automation. This structured approach ensures a systematic and thorough examination of the robot's kinematics, dynamics, and control model, providing valuable insights into its behavior and capabilities across various operational scenarios.

Kinematic Analysis

Forward kinematics

Forward kinematics in robotic manipulators

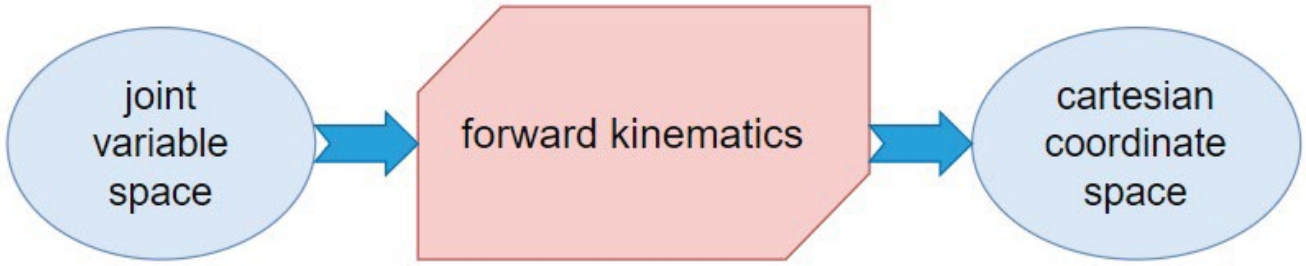


Figure 2: Transformation for forward kinematics.

Table 1: DH table for ABB 930.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	Θ_1
2	a_2	180	0	Θ_2
3	0	0	d_3	0
4	0	0	d_4	Θ_4

involve the conversion of kinematic information from the joint variable space to the Cartesian coordinate space. This process enables the determination of the end-effector's position and orientation based on a given set of joint variables [5]. Figure 2 visually illustrates the transformation process integral to forward kinematics.

The approach to finding the solution for forward kinematics follows a systematic progression, moving link by link using Denavit-Hartenberg notation and frames. In the DH convention, the link parameters, namely α_i and a_i, remain fixed and are associated with the geometric characteristics of the manipulator. The angle α_i represents the rotation between z_{i-1} and z_i about x_{i-1}, while a_i denotes the distance along x_{i-1} between z_{i-1} and z_i. In contrast, the joint parameters, θ_i and d_i, introduce variability into the kinematic model. Among these joint parameters, one remains constant, providing stability to the system, while the other is variable and adjusts to accommodate different configurations. Specifically, θ_i represents the angle between x_{i-1} and x_i about z_{i-1}, while d_i denotes the distance along z_{i-1} between x_{i-1} and x_i. Figure 3 presents the kinematic diagram for the SCARA manipulator. The DH parameters for the SCARA manipulator are obtained from the kinematic diagram and represented in Table 1.

The subsequent stage in forward kinematics, following the derivation of the Denavit-Hartenberg (DH) table for the robot manipulator, involves the determination of the transformation matrix. MATLAB allows for the implementation of Denavit-Hartenberg (DH) parameters into symbolic variables, representing a_i, α_i, d_i, and θ_i. These parameters are fundamental in defining the geometric and kinematic characteristics of the manipulator's links and joints. With the DH parameters defined symbolically, MATLAB assists in constructing the individual transformation matrices T_i using the DH convention. The transformation matrix from the $(i-1)$-th joint to the i-th joint, represented as T_i, involves trigonometric functions of the joint variables and DH parameters. Additionally, MATLAB facilitates the systematic multiplication of these individual transformation matrices to compute the total transformation matrix T_{total}. By sequentially multiplying the transformation matrices corresponding to each joint, MATLAB provides the complete kinematic relationship from the robot's base to its end-effector.

$$T_4^0 = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_4^3 \tag{1}$$

The transformation matrix from the $(i-1)$-th joint to the i-th joint is represented as

$$T_i^{i-1} = \begin{bmatrix} c_i & -s_i\cos(\alpha_i) & s_i\sin(\alpha_i) & a_i c_i \\ s_i & c_i\cos(\alpha_i) & -c_i\sin(\alpha_i) & a_i s_i \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Where:

$$c_i = \cos(\theta_i)$$

$$s_i = \sin(\theta_i)$$

The individual matrices from the base to the end effector are:

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

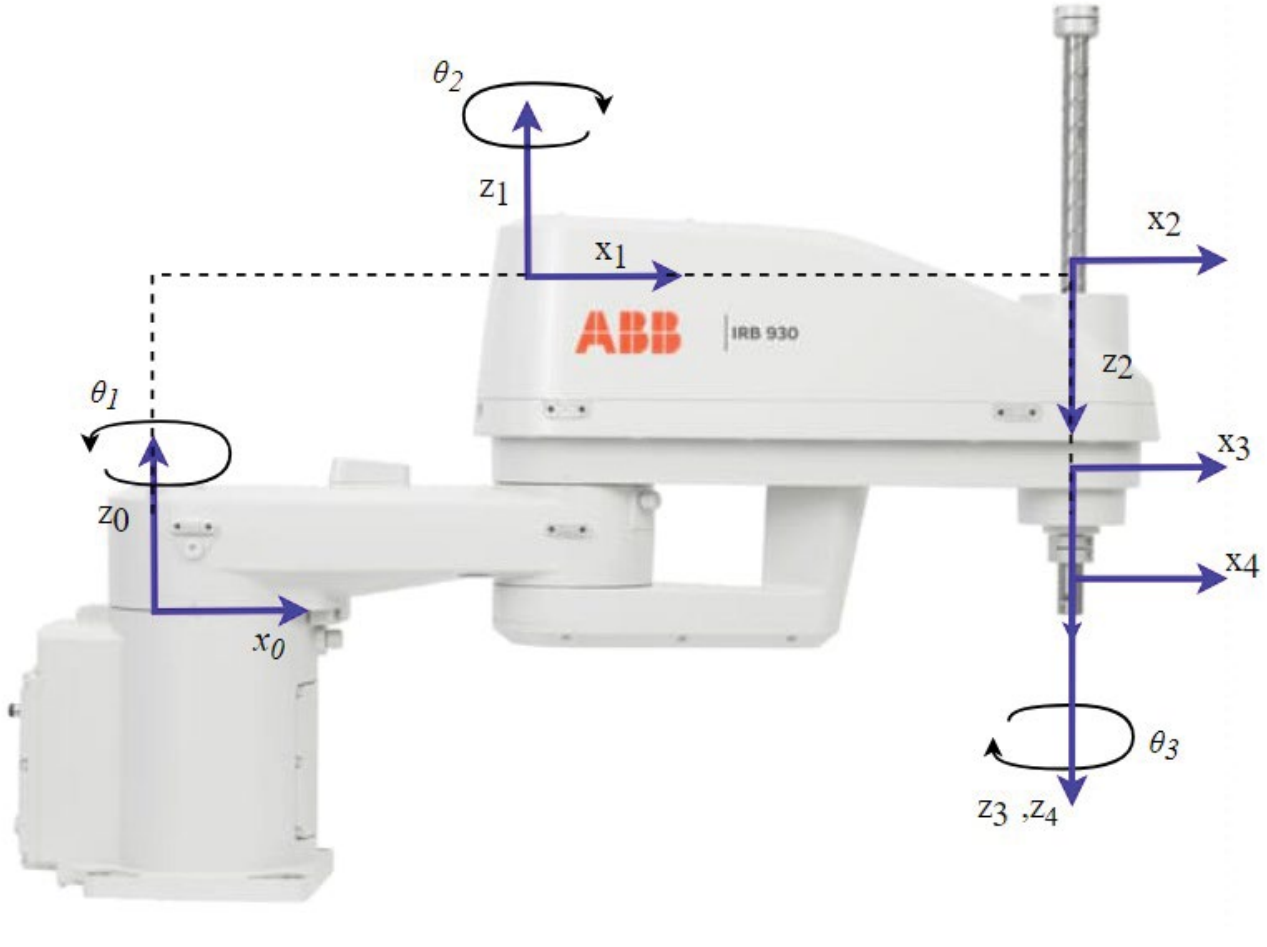


Figure 3: Kinematic diagram of ABB 930.

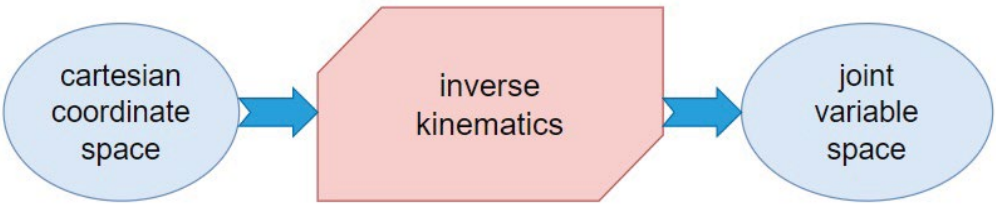


Figure 4: Transformation for inverse kinematics.

$$T_2^1 = \begin{bmatrix} c_2 & s_2 & 0 & a_2c_2 \\ s_2 & -c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$T_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$T_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

The total transformation matrix is

$$T_4^0 = \begin{bmatrix} n_x & s_x & a_x & o_x \\ n_y & s_y & a_y & o_y \\ n_z & s_z & a_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

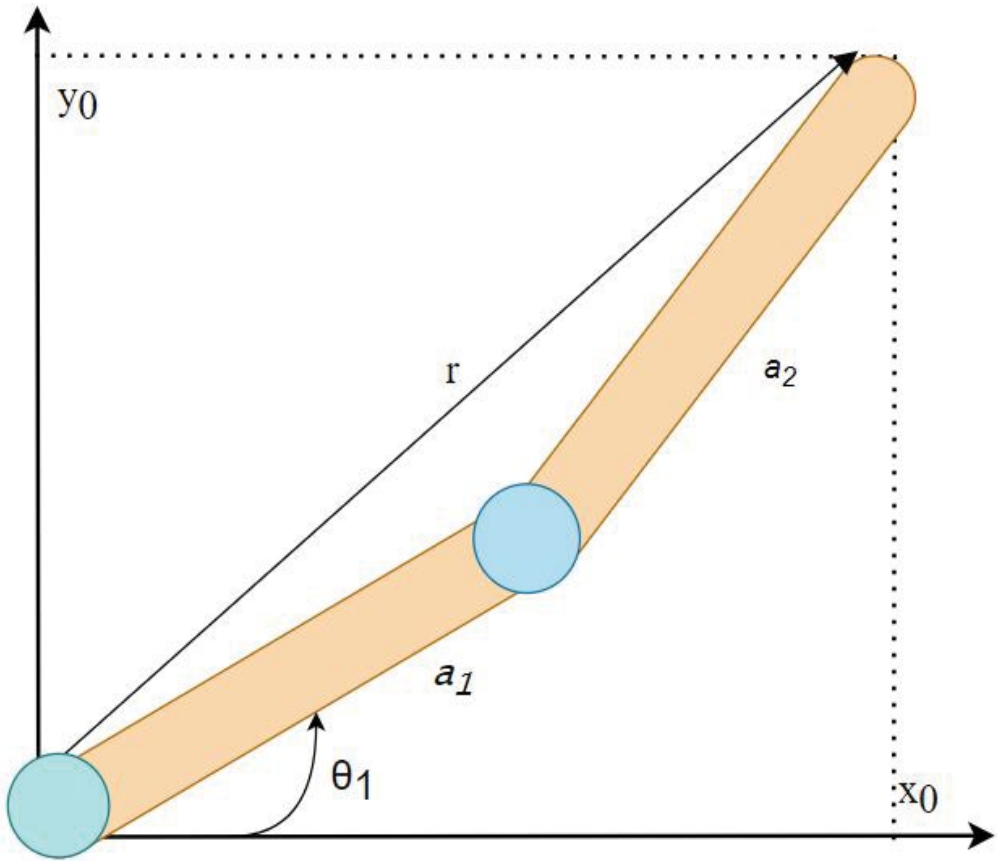


Figure 5: Projection of manipulator.

Compound expansions for the total transformation matrix:

$$n_x = c_{12}c_4 + s_{12}s_4 \quad \text{(Normal vector)} \quad (8)$$

$$s_x = s_{12}c_4 - c_{12}s_4 \quad \text{(Sliding vector)} \quad (9)$$

$$a_x = 0 \quad \text{(Approach vector)} \quad (10)$$

$$o_x = a_1c_1 + a_2c_{12} \quad \text{(Position vector)} \quad (11)$$

$$n_y = s_{12}c_4 - c_{12}s_4 \quad \text{(Normal vector)} \quad (12)$$

$$s_y = -s_{12}s_4 - c_{12}c_4 \quad \text{(Sliding vector)} \quad (13)$$

$$a_y = 0 \quad \text{(Approach vector)} \quad (14)$$

$$o_y = a_1s_1 + a_2s_{12} \quad \text{(Position vector)} \quad (15)$$

$$n_z = 0 \quad \text{(Normal vector)} \quad (16)$$

$$s_z = 0 \quad \text{(Sliding vector)} \quad (17)$$

$$a_z = -1 \quad \text{(Approach vector)} \quad (18)$$

$$o_z = -d_3 - d_4 \quad \text{(Position vector)} \quad (19)$$

The total transformation matrix provides the basis for obtaining the solution to forward kinematics. This solution is derived by comparing the total transformation matrix with the reference matrix, which represents the orientation and position information of the end-effector.

Inverse kinematics

Inverse kinematics is the reverse process of forward kinematics, involving the kinematic transformation from Cartesian coordinate space to joint variable space [6]. In this method, the joint variables are determined based on the desired configuration of the end effector.

The mathematical representation involves the search for these joint variables, with their determination entailing the solution of a set of nonlinear coupled algebraic equations. In practice, computer-controlled robots are predominantly actuated in joint variable space, whereas the objects to be manipulated are typically expressed in a Cartesian coordinate frame. To determine the inverse kinematic solution, the total transformation matrix T_4^0 is considered as

$$T_4^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & o_x \\ r_{21} & r_{22} & r_{23} & o_y \\ r_{31} & r_{32} & r_{33} & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The joint variables θ_1,θ_2, and θ^4 are determined as follows:

$$\theta_1 + \theta_2 - \theta_4 = \alpha = a\tan 2(r_{11}, r_{12})$$

Then, projecting the manipulator on the x_0–y_0 plane, the other values can be found. Figure 5 illustrates the projection process.

$$\theta_2 = a\tan 2\left(c_2, \pm\sqrt{1-c_2}\right)$$

Where $c_2 = \dfrac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2}$

$$\theta_1 = a\tan 2(o_x, o_y) - a\tan 2(a_1 + a_2c_2, a_2s_2)$$

To determine ϑ_4,

$$\theta_4 = \theta_1 + \theta_2 - \alpha = \theta_1 + \theta_2 - a\tan 2(r_{11}, r_{12})$$

And $d_3 = o_z + d_4$.

Thus, the inverse kinematic solution has been obtained from the geometrical method.

Differential kinematics

Differential kinematics serves as the crucial link connecting joint movements to the resulting motion of a robotic manipulator's end-effector. Also referred to as velocity kinematics, this concept delves into the intricate relationship between joint velocities and end-effector velocities. At its core lies the Jacobian matrix, a mathematical tool that encapsulates this relationship, providing a means to analyze and control the motion of robotic systems.

The Jacobian matrix plays a pivotal role in both forward and inverse differential kinematics. In forward differential kinematics, it enables predicting the impact of small changes in joint positions on the end-effector's position and orientation. Conversely, in inverse differential kinematics, it guides the adjustments in joint positions needed to achieve a desired end-effector motion [7]. Both forward and inverse differential kinematics are instrumental in the broader field of robotics, offering distinct perspectives and practical applications. They find relevance in tasks such as trajectory planning and real-time control, contributing to the efficient and precise movement of robotic systems. Figure 6 illustrates the relationship between differential kinematics.

Forward differential kinematics: Forward Differential Kinematics is a fundamental concept in robotics that aims to elucidate how minute adjustments in joint positions ($d\theta$) translate into

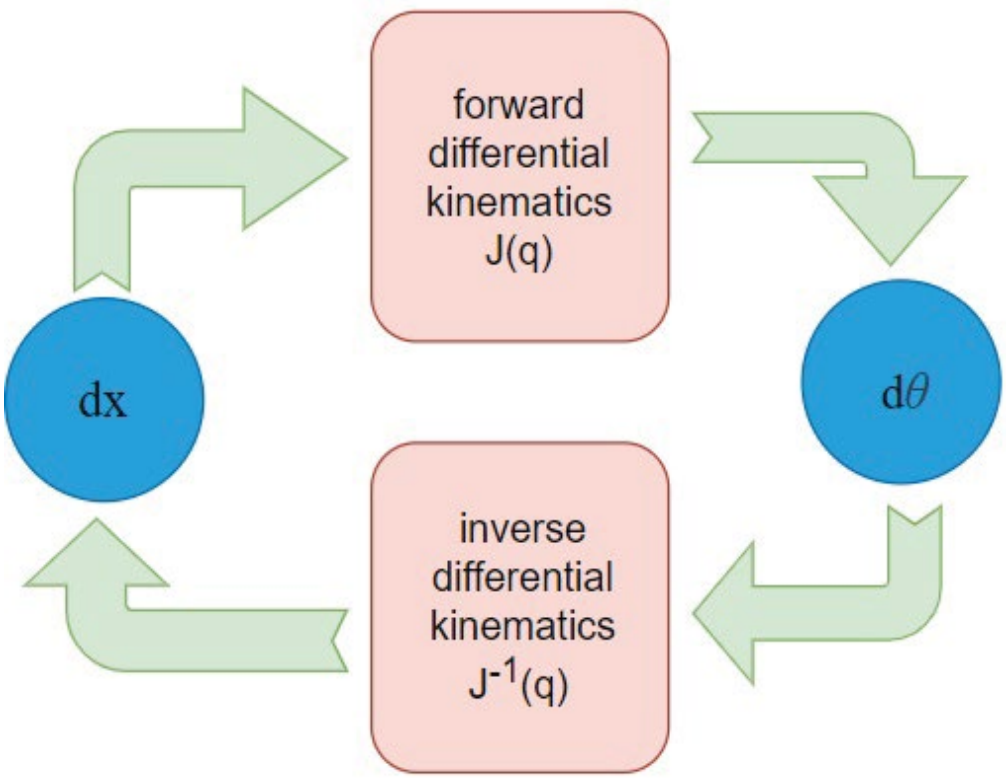


Figure 6: Relationship between differential kinematics.

corresponding changes in the end-effector's position and orientation (*dX*). The objective is to mathematically formulate and simulate the manipulator's behavior by analyzing its velocity ratios. *dX* signifies the differential change in the end-effector's position and orientation, providing insights into how the manipulator responds to alterations in joint configurations. The Jacobian matrix (*J*) acts as a bridge, representing the sensitivity of the end-effector motion to variations in joint velocities.

The practical application of forward differential kinematics is manifold. It finds extensive usage in predicting the motion of the end-effector, particularly when confronted with incremental adjustments in joint positions. This predictive capability is crucial for tasks such as trajectory planning, where understanding the relationship between joint variations and end-effector motion is imperative for devising optimal paths. The general form of forward differential kinematics is given below:

$$dX = J.d\theta \tag{20}$$

Where:

dX is the differential change in the end-effector position and orientation.

J is the Jacobian matrix.

*d*θ is the vector of differential changes in joint positions.

Inverse differential kinematics: Inverse Differential Kinematics establish a clear connection between incremental changes in the end-effector's position and orientation (*dX*) and the corresponding adjustments needed in joint positions (*d*θ). This approach plays a crucial role in the velocity-level control of manipulators. The primary application of inverse differential kinematics lies in the precise control of a robot's joints to achieve specific end-effector motions. This method proves especially valuable in closed-loop control systems, where real-time feedback from the actual end-effector motion is utilized to regulate the robot's behavior.

The mathematical expression for inverse differential kinematics is given below:

$$d\theta = J^{-1} \cdot dX \tag{21}$$

Where:

dϑ is the vector of differential changes in joint positions.

J^{-1} is the inverse of the Jacobian matrix.

dX is the differential change in the end-effector position and orientation.

Jacobian

In robotics, the Jacobian matrix (*J*) plays an important role in the analysis of serial link robot manipulators [8]. The joints are categorized as either revolute or prismatic, with the joint type represented by *ρi* (1 for revolute, 0 for prismatic). The orientation vector $z^0_{(i-1)}$ is determined by

$$z^0_{(i-1)} = R^0_{(i-1)} \cdot k \tag{22}$$

Where $z^0_0 = k = (0,0,1)^T$.

The Jacobian matrix is structured into two halves: The upper half (J_v) Corresponding to angular velocities and the lower half (J_ω) corresponding to angular velocities. The linear velocity Jacobian (J_v) is further divided into columns (J_{v_i}) for each joint. For revolute joints, J_{v_i} is expressed as $\left[z_{i-1} \times (o_n - o_{i-1})\right]$, while for prismatic joints, J_{v_i} simplifies to $\left[z_{i-1}\right]$.

Simultaneously, the angular velocity Jacobian (J_ω) is structured similarly, with columns (J_{ω_i}) for each joint. For revolute joints, J_{ω_i} is given by $\left[z_{i-1}\right]$, and for prismatic joints, J_{ω_i} is [0]. The complete Jacobian matrix (*J*) is then formed by concatenating J_v and J_ω resulting in a matrix of the form

$$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix} = \begin{bmatrix} J_1 & J_2 & \dots & J_n \end{bmatrix} \tag{23}$$

For revolute joints, J_i is expressed as

$$\begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{24}$$

And for prismatic joints, J_i takes the form

$$\begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{25}$$

This comprehensive structure of the Jacobian Matrix facilitates the mapping of joint velocities to end-effector velocities, crucial of the kinematic analysis of robotic manipulators.

In the case of a SCARA manipulator where joints 1, 2, and 3 are revolute, and joint 3 is prismatic, the robot has a parallel link between joints 3 and 4. i.e., $o_4 - o_3$ is parallel to Z_3, therefore,

$$\left(Z_3 \times (o_4 - o_3)\right) = 0,$$

implying that the rotational joint 3 does not contribute to the end-effector motion in the direction of $o_4 - o_3$. Therefore, the modified Jacobian matrix is represented as

$$J = \begin{bmatrix} z_0 \times (o_4 - o_0) & z_1 \times (o_4 - o_1) & z_2 & 0 \\ z_0 & z_1 & 0 & z_3 \end{bmatrix} \quad (26)$$

$$o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix},$$

$$o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix},$$

$$o_4 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ d_3 - d_4 \end{bmatrix}$$

$$z_0 = z_1 = k_1,\ z_2 = z_3 = -k$$

Finally, by performing the indicated calculations, the Jacobian of the SCARA manipulator is

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \quad (27)$$

Singularities

Singular configurations refer to specific robot poses where the end effector experiences a reduction in its degrees of freedom, rendering it inferior to the typical operational dimensions. Singularities occur under certain conditions. In the case of prismatic joints, singular configurations may arise when two axes become parallel, causing a loss of independence in the kinematic equations and resulting in a reduction of the end effector's degrees of freedom (DOF). Similarly, for revolute joints, singularities can occur when two axes become identical, leading to a linear dependence in the kinematic equations and restricting the freedom of movement for the end effector [9]. Singular configurations must be carefully avoided, as they can lead to theoretical challenges, most notably, the velocity required to move the end effector becomes theoretically infinite.

Two primary types of singularities may occur in the workspace of a robot manipulator. Workspace Boundary Singularities manifest at the boundaries of the robot's workspace, and detecting and understanding these singularities are crucial for optimizing the robot's operational range. Additionally, Workspace Interior Singularities can occur within the interior of the robot's workspace, and identifying and managing these singularities are essential for ensuring smooth and predictable robot movements during operation. One effective method for identifying and analyzing singularities is through the use of the Jacobian matrix.

$$J = \begin{bmatrix} J_p \mid J_0 \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \quad (28)$$

From Figure 7, it is evident geometrically that the only singularity of the SCARA arm occurs when the elbow is fully extended or fully retracted. This observation aligns with the fact that the portion of the Jacobian governing SCARA arm singularities is given as follow:

$$J_{11} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} & 0 \\ -a_1 s_{12} & a_1 c_{12} & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (29)$$

The rank of J_{11} will be less than three precisely whenever $\alpha_1 \alpha_4 - \alpha_2 \alpha_3 = 0$.

$S_2 = 0$, which implies $\theta_2 = 0, \pi$.

In summary, the singularities of the SCARA arm occur when the elbow is fully extended or fully retracted, corresponding to the conditions $S_2 = 0$ and $\alpha_1 \alpha_4 - \alpha_2 \alpha_3 = 0$.

Velocity Propagation

The systematic analysis of velocity propagation through the various joints of a robotic arm is required for understanding the dynamic behavior of the manipulator, as it is crucial for optimizing the design and control of robotic systems. The process involves forward propagation, where joint velocities and accelerations are computed, either starting from the base link and progressing towards the end-effector or vice versa [10].

For prismatic joints, the angular and linear velocities (ω and V), as well as angular and linear accelerations (α and a), are determined through mathematical expressions that account for the rotational and translational aspects of joint motion. However, for rotary joints, these equations are

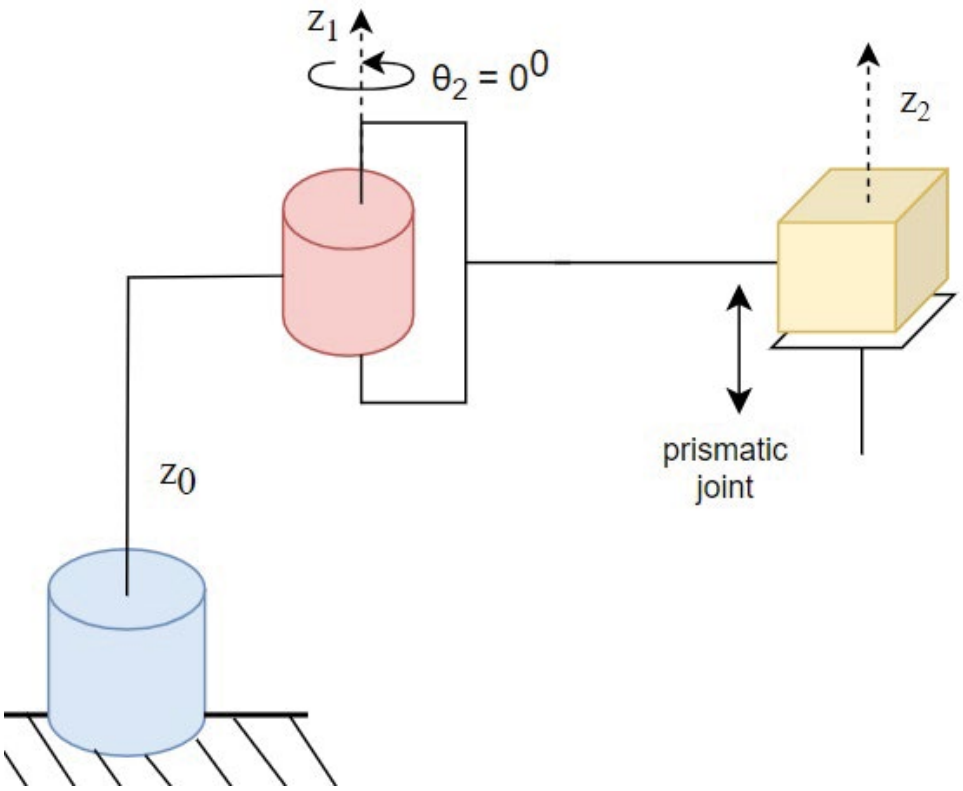


Figure 7: Singularity analysis of SCARA manipulator.

modified to incorporate the joint angle (ϑ).

The angular velocity for a rotary joint through forward propagation is given by

$$\omega_{i+1}^{i+1} = R_i^{i+1}\omega_i^i + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix} \tag{30}$$

For prismatic joints, the angular velocity is given by

$$\omega_{i+1}^{i+1} = R_i^{i+1}\omega_i^i \tag{31}$$

Similarly, the linear velocity for both rotary and prismatic joints through forward propagation is given by

$$V_{i+1}^{i+1} = R_i^{i+1}(V_i^i + \omega_i^i \times P_{i+1}^i) \tag{32}$$

$$V_{i+1}^{i+1} = R_i^{i+1}\left(V_i^i + \omega_i^i \times P_{i+1}^i + \begin{bmatrix} 0 \\ 0 \\ \dot{d}_{1+1} \end{bmatrix}\right) \tag{33}$$

In implementing the mathematical formulations for velocity propagation within a MATLAB environment, the joint parameters such as lengths a_i, joint angles θ_i, and joint velocities $\dot{\theta}_i$ are defined. Using these parameters, the forward propagation equations are systematically applied to compute the velocities for each joint, progressively moving from the base towards the end-effector.

$$V_1^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad V_2^2 = \begin{bmatrix} 0 \\ -a_2\dot{\theta}_1 \\ 0 \end{bmatrix} \tag{34}$$

$$V_3^3 = \begin{bmatrix} 0 \\ -a_2\dot{\theta}_1 \\ \dot{d}_3 \end{bmatrix}, \quad V_4^4 = \begin{bmatrix} -a_2\dot{\theta}_1 \sin(\theta_4) \\ -a_2\dot{\theta}_1 \cos(\theta_4) \\ \dot{d}_3 \end{bmatrix} \tag{35}$$

$$\omega_1^1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}, \quad \omega_2^2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 - \dot{\theta}_1 \end{bmatrix} \tag{36}$$

$$\omega_3^3 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 - \dot{\theta}_1 \end{bmatrix}, \quad \omega_4^4 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 - \dot{\theta}_1 \end{bmatrix} \tag{37}$$

The final linear and angular velocity of the end-effector with respect to the base frame are

$$\omega_4^0 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 - \dot{\theta}_2 \end{bmatrix} \tag{38}$$

$$V_4^0 = \begin{bmatrix} -a_2\dot{\theta}_1 \sin(\theta_1+\theta_2) \\ -a_2\dot{\theta}_1 \cos(\theta_1+\theta_2) \\ -\dot{d}_3 \end{bmatrix} \quad (39)$$

This final velocity of the end effector is critical for parameters that characterize the overall motion of the robotic manipulator. This comprehensive analysis of velocity propagation forms the basis for further advancements in manipulator design, control strategies, and overall performance optimization in diverse applications, ranging from industrial automation to cutting-edge robotics.

Dynamic Modelling

Dynamic modelling involves studying the changes in the state of a robotic system over time. The dynamic equations can be expressed as

$$\tau = M(q)\ddot{q} + n(q,\dot{q})$$

Where τ is the torque, *M(q)* is the inertia matrix, and $n(q,\dot{q})$ represents other terms. Dynamic modelling is divided into two main categories: forward dynamics and inverse dynamics. Forward dynamics involve determining the resultant motion of the manipulator $(\ddot{q})$ for the input vector τ and known states q, $\dot{q}$. The relation for forward dynamics is given by

$$\ddot{q} = M(q)^{-1}(\tau - n(q,\dot{q}))$$

Inverse dynamics deal with finding the required input vector τ to achieve a specific trajectory $(q,\dot{q},\ddot{q})$. The relationship between forward dynamics and inverse dynamics is illustrated in Figure 8.

Forward dynamics simulate or analyze the manipulator's motion, while inverse dynamics address the control aspect of manipulating the robot. Two primary approaches to obtaining these equations are the energy-based approach, such as the Lagrange-Euler method, and the force approach, such as the Newton-Euler method.

Lagrange-Euler method

To determine the Euler-Lagrange equations, the Lagrangian of the system has to be formed, which is the difference between the kinetic energy and the potential energy [11].

$$\mathrm{L} = K.E - P.E$$

Where:

L : Lagrangina

K.E: Kinetic energy

P.E: Potential energy

Using the Lagrangian, the equation of motion can be obtained by the relation:

$$\frac{d}{dt}\left(\frac{\partial \mathrm{L}}{\partial \dot{q}}\right) - \frac{\partial \mathrm{L}}{\partial q} = \tau$$

To find the Lagrange position coordinates $((x_1,y_1),\ldots,(x_n,y_n))$, the total kinetic energy of the robot manipulator can be found by:

$$K.E = \sum_{i=1}^{n} \frac{1}{2} m_i(\dot{x}_i^2 + \dot{y}_i^2)$$

The potential energy can be obtained by:

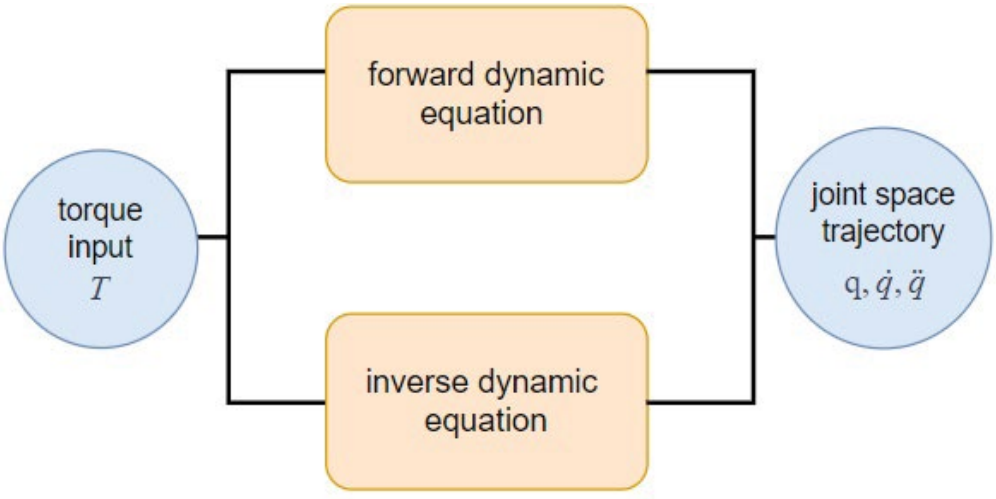


Figure 8: Relationship between dynamic analysis.

$$P.E = \sum_{i-1}^{n} m_i g_i y_i$$

Where:

m_i: Mass of the component

gi: Gravitational force acting on the component

yi: Vertical position of the component

Then, the Lagrangian is:

$$\mathrm{L} = \sum_{i=1}^{n} \frac{1}{2} m_i (\dot{x}_i^2 + \dot{y}_i^2) - \sum_{i=1}^{n} m_i g_i y_i$$

The input vector τ can be obtained by:

$$\tau = \frac{d}{dt}\left(\frac{\partial \mathrm{L}}{\partial \dot{q}}\right) - \frac{\partial \mathrm{L}}{\partial q}$$

where q represents the angular position. In implementing the mathematical formulation for determining the Euler-Lagrange equations within a MATLAB environment, the process involves translating the equations into code to systematically compute the torque input vector for each joint of the robotic manipulator. Firstly, the Lagrangian of the system is formulated as the difference between the kinetic energy and the potential energy. Using the given joint parameters such as masses m_i, gravitational forces g_i, and joint velocities $\dot{x}_1$ and $\dot{y}_1$, the expressions for kinetic energy and potential energy are defined. Subsequently, the Lagrangian L is constructed as the sum of kinetic and potential energies. This formulation is then utilized to derive the equations of motion using the Euler-Lagrange equation, which relates the time derivative of the partial derivative of the Lagrangian with respect to velocity to the partial derivative of the Lagrangian with respect to position.

$$\begin{aligned}
\tau_1 = {} & 2a_1^2 m_1 \dot{\theta}_1 + 2a_2^2 m_2 \dot{\theta}_1 - 2a_2^2 m_2 \dot{\theta}_2 + 2a_3^2 m_3 \dot{\theta}_1 \\
& - 2a_3^2 m_3 \dot{\theta}_2 - a_1 a_2 m_2 \dot{\theta}_1 + a_1 a_2 m_2 \dot{\theta}_2 \\
& - a_1 a_2 m_3 \dot{\theta}_1 + a_1 a_2 m_3 \dot{\theta}_2 + 2a_2 a_3 m_3 \dot{\theta}_1 \\
& - 2a_2 a_3 m_3 \dot{\theta}_2 + 2a_2 g m_2 \cos(\theta_1 + \theta_2) \\
& + a_2 g m_2 \cos(\theta_1 + \theta_2) + a_3 g m_3 \cos(\theta_1 + \theta_2) \\
& + 2a_1 g m_1 \cos(\theta_1) + a_1 g m_2 \cos(\theta_1) \\
& + a_1 g m_3 \cos(\theta_1) + 2a_1^2 m_1 \dot{\theta}_1 \cos(\theta_1) \\
& + a_1^2 m_2 \dot{\theta}_1 \cos(\theta_1) + a_1^2 m_3 \dot{\theta}_1 \cos(\theta_1) \\
& - 2a_2^2 m_2 \dot{\theta}_1 \cos(\theta_2) + 2a_2^2 m_2 \dot{\theta}_2 \cos(\theta_2) \\
& - a_2^2 m_3 \dot{\theta}_1 \cos(\theta_2) + a_2^2 m_3 \dot{\theta}_2 \cos(\theta_2) \\
& - 2a_1 a_2 m_2 \dot{\theta}_1^2 \sin(\theta_2) - a_1 a_2 m_2 \dot{\theta}_2^2 \sin(\theta_2) \\
& - a_1 a_2 m_3 \dot{\theta}_1^2 \sin(\theta_2) - a_1 a_3 m_3 \dot{\theta}_1^2 \sin(\theta_2) \\
& - a_1 a_3 m_3 \dot{\theta}_2^2 \sin(\theta_2) + 2a_1 a_2 m_2 \dot{\theta}_1 \cos(\theta_1 + \theta_2) \\
& + a_1 a_2 m_3 \dot{\theta}_1 \cos(\theta_1 + \theta_2) + a_1 a_3 m_3 \dot{\theta}_1 \cos(\theta_1 + \theta_2) \\
& + a_1 a_2 m_2 \dot{\theta}_1 \cos(\theta_2) - a_1 a_2 m_2 \dot{\theta}_2 \cos(\theta_2) \\
& + 2a_1 a_3 m_3 \dot{\theta}_1 \cos(\theta_2) - 2a_1 a_3 m_3 \dot{\theta}_2 \cos(\theta_2) \\
& - a_2 a_3 m_3 \dot{\theta}_1 \cos(\theta_2) + a_2 a_3 m_3 \dot{\theta}_2 \cos(\theta_2) \\
& + 2a_1 a_2 m_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2) + 2a_1 a_3 m_3 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2)
\end{aligned} \tag{40}$$

$$\begin{aligned}
\tau_2 = {} & a_1^2 m_1 \dot{\theta}_1 + 2a_2^2 m_2 \dot{\theta}_1 - 2a_2^2 m_2 \dot{\theta}_2 + 2a_3^2 m_3 \dot{\theta}_1 \\
& - 2a_3^2 m_3 \dot{\theta}_2 + 2a_2 a_3 m_3 \dot{\theta}_1 - 2a_2 a_3 m_3 \dot{\theta}_2 \\
& + 2a_2 g m_2 \cos(\theta_1 + \theta_2) + a_2 g m_3 \cos(\theta_1 + \theta_2) \\
& + a_2 g m_3 \cos(\theta_1 + \theta_2) + a_1 g m_1 \cos(\theta_1) \\
& + a_1^2 m_1 \dot{\theta}_1 \cos(\theta_1) - 2a_2^2 m_2 \dot{\theta}_1 \cos(\theta_2) \\
& + 2a_2^2 m_2 \dot{\theta}_2 \cos(\theta_2) - a_2^2 m_3 \dot{\theta}_1 \cos(\theta_2) \\
& + a_2^2 m_3 \dot{\theta}_2 \cos(\theta_2) + 2a_1 a_2 m_2 \dot{\theta}_1 \cos(\theta_1 + \theta_2) \\
& + a_1 a_2 m_3 \dot{\theta}_1 \cos(\theta_1 + \theta_2) + a_1 a_3 m_3 \dot{\theta}_1 \cos(\theta_1 + \theta_2) \\
& - a_2 a_3 m_3 \dot{\theta}_1 \cos(\theta_2) + a_2 a_3 m_3 \dot{\theta}_2 \cos(\theta_2)
\end{aligned} \tag{41}$$

$$\begin{aligned}
\tau_3 = {} & -a_2 m_2 (a_2 \dot{\theta}_1 - a_2 \dot{\theta}_2 + g \cos(\theta_1 + \theta_2) \\
& - a_2 \dot{\theta}_1 \cos(\theta_2) + a_2 \dot{\theta}_2 \cos(\theta_2) + a_1 \dot{\theta}_1 \cos(\theta_1 + \theta_2)) \\
& - a_3 m_3 (2a_3 \dot{\theta}_1 - 2a_3 \dot{\theta}_2 + g \cos(\theta_1 + \theta_2) \\
& - a_2 \dot{\theta}_1 \cos(\theta_2) + a_2 \dot{\theta}_2 \cos(\theta_2) + a_1 \dot{\theta}_1 \cos(\theta_1 + \theta_2))
\end{aligned} \tag{42}$$

$$\tau_4 = -a_3 m_3 (2a_3\dot{\theta}_1 - 2a_3\dot{\theta}_2 + g\cos(\theta_1+\theta_2) - a_2\dot{\theta}_1\cos(\theta_2) + a_2\dot{\theta}_2\cos(\theta_2) + a_1\dot{\theta}_1\cos(\theta_1+\theta) \quad (43)$$

Newton-Euler method

The Newton-Euler method, especially in its recursive formulation, proves to be a robust approach for deriving dynamic equations in robotic applications. Its core advantage lies in its efficiency in calculating joint forces and torques, facilitating a comprehensive analysis of the robot's dynamic behavior. Through forward propagation, joint velocities and accelerations for both prismatic and rotary joints are determined [12].

For prismatic joints, the angular velocity $\left(\omega_{i+1}^{i+1}\right)$, linear velocity $\left(V_{i+1}^{i+1}\right)$ are determined by using velocity propagation, angular acceleration $\left(\alpha_{i+1}^{i+1}\right)$, and linear acceleration $\left(a_{i+1}^{i+1}\right)$ are computed using the following expressions:

$$\left(\alpha_{i+1}^{i+1}\right) = R_i^{i+1}\alpha_i^i \quad (44)$$

$$\left(a_{i+1}^{i+1}\right) = R_i^{i+1}\left(a_i^i + \alpha_i^i \times P_{i+1}^i + \omega_i^i \times (\omega_i^i \times P_{i+1}^i) + \begin{bmatrix}0\\0\\\ddot{d}_{i+1}\end{bmatrix} + 2\omega_{i+1}^{i+1} \times \begin{bmatrix}0\\0\\\ddot{d}_{i+1}\end{bmatrix}\right)$$

For rotary joints, these expressions are modified to accommodate the joint angle θ_{i+1}:

$$\alpha_{i+1}^{i+1} = R_i^{i+1}\left(\alpha_i^i + \omega_i^i \times \begin{bmatrix}0\\0\\\dot{\theta}_{i+1}\end{bmatrix}\right) + \begin{bmatrix}0\\0\\\ddot{\theta}_{i+1}\end{bmatrix}$$

$$a_{i+1}^{i+1} = R_i^{i+1}\left(a_i^i + \alpha_i^i \times P_{i+1}^i + \omega_i^i \times \left(\omega_i^i \times P_{i+1}^i\right)\right)$$

From these equations, force $\left(F_i^i\right)$ and moment $\left(N_i^i\right)$ for each link can be determined. Force $\left(F_i^i\right)$ is calculated by the product of mass (m_i) and linear acceleration $\left(a_{ci}^i\right)$, while moment $\left(N_i^i\right)$ involves the product of the distance to the center of mass $\left(l_{ci}\right)$ and angular acceleration $\left(\alpha_i^i\right)$, along with the contribution from rotational motion:

$$F_i^i = m_i a_{ci}^i$$

$$a_{ci}^i = a_i^i + \omega_i^i \times \left(a_i^i \times P_{ci}^i\right) + \omega_i^i \times \left(\omega_i^i \times P_{ci}^i\right)$$

$$N_i^i = l_{ci}\alpha_i^i + \omega_i^i \times \left(l_{ci}\omega_i^i\right)$$

The backward propagation step facilitates the computation of joint forces and torques in a recursive manner:

$$f_i^i = R_{i+1}^i f_{i+1}^i + F_i^i \quad (45)$$

$$n_i^i = R_{i+1}^i n_{i+1}^{i+1} + N_i^i + P_{ci}^i \times F_i^i + p_{i+1}^i \times R_{i+1}^i f_{i+1}^{i+1} \quad (46)$$

Ultimately, the joint forces $\left(\tau_i\right)$ and joint torques are obtained by:

$$\tau_i = [0,0,1]n_i^i \quad \text{for rotary joints} \quad (47)$$

$$\tau_i = [0,0,1]f_i^i \quad \text{for prismatic joints} \quad (48)$$

This method allows for a systematic analysis of each link, either starting from the end-effector link and progressing towards the base link or vice versa.

Integrated Motion Control Framework

Motion control is a crucial subfield of automation tasked with managing systems or subsystems responsible for precisely maneuvering components of machine. It involves monitoring the actual trajectory and making real-time adjustments to correct position and velocity errors. Integrated control algorithms, such as PID control, play a fundamental role in governing the performance of robots within this framework. In PID control, the proportional (P), integral (I), and derivative (D) terms are utilized to adjust the control signal based on the error between the desired setpoint and the actual output. The proportional term responds to the current error, the integral term addresses accumulated pas errors, and the derivative term predicts future error trends, collectively ensuring robust closed-loop stability [13]. Within motion control, a complex interplay occurs as motion profiles and target positions are defined, creating trajectories for motors and actuators. Central to this process is the integration of a controller that elevates a robot from a mere mechanical structure to a dynamic system capable of precise actions. However, accurately capturing a robot's dynamic model remains a significant challenge. Motion control broadly encompasses kinematic and dynamic control categories. Kinematic control, relying on kinematic models, further divides into joint space and task space schemes [14]. The Jacobian matrix plays a vital role in both schemes, facilitating the translation between joint and task spaces for achieving precise and controlled motion in robotic systems.

Kinematic control

Joint space scheme: In the joint space scheme, the desired inputs include the target joint positions,

$q_d(t)$, and joint velocities, $\dot{q}_d(t)$, with the latter being set to zero for serpoint control. The available information comprises the actual joint positions, $\dot{q}(t)$, and the Jacobian matrix $J(q)$ along with its inverse, $J^{-1}(q)$. The control inputs $\dot{q}(t)$ are then determined through the relationship

$$\dot{q} = \dot{q}d + \lambda\ddot{q}$$

The control flow for the joint space scheme is illustrated in Figure 9.

Task space scheme: In kinematic control within the task space, the desired parameters are the end-effector positions, $\mu_d(t)$, and the desired end-effector velocity, $\dot{\mu}_d(t)$, with the latter being zero for setpoint control. The available data includes the actual end-effector positions, $\dot{\mu}(t)$, and the Jacobian matrix $J(q)$ with its inverse, $J^{-1}(q)$. The control inputs $\dot{q}(t)$ are obtained through the equation

$$\dot{q} = J(q)^{-1}(\dot{\mu}_d + r\ddot{\mu})$$

Where r is a constant factor. The control flow for the task space scheme is illustrated in Figure 10.

Dynamic control

Joint space scheme: In dynamic control within the joint space, the desired parameters are the target joint positions, $q_d(t)$, and joint velocities, $\dot{q}_d(t)$, with the latter being set to zero for setpoint control. The available information includes the actual joint positions, $q(t)$, actual joint velocities $\dot{q}(t)$, as well as the inertia matrix $M(q)$ and the non-linear term $n(q,\dot{q})$. The control flow for the joint space scheme is illustrated in Figure 11.

The control inputs τ are determined by the equation

$$\tau = k_p(q_d - q) + k_d(\dot{q}_d - \dot{q}) + g(q)$$

Where k_p and k_d are proportional and derivative gain matrices, respectively, and $g(q)$ represents gravitational effects.

Task space scheme: In dynamic control within the task space, the desired parameters are the target joint positions, $\mu_d(t)$, and joint velocities, $\dot{\mu}_d(t)$. For setpoint control, $\dot{\mu}_d(t)$ is set to zero. The available information includes the actual

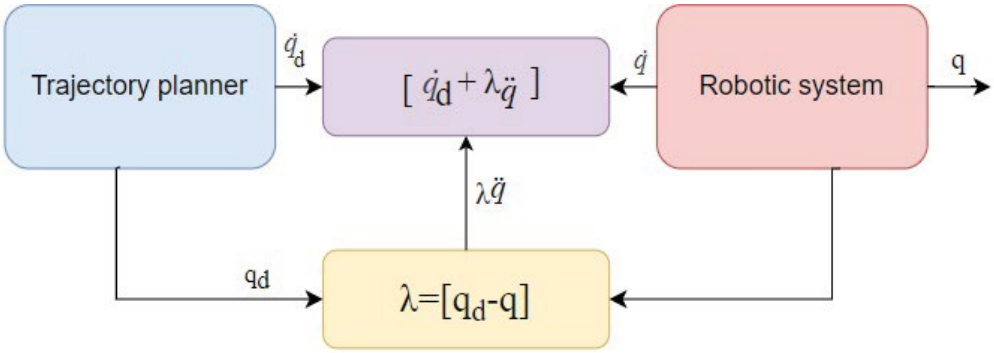


Figure 9: Dynamic analysis on joint space scheme.

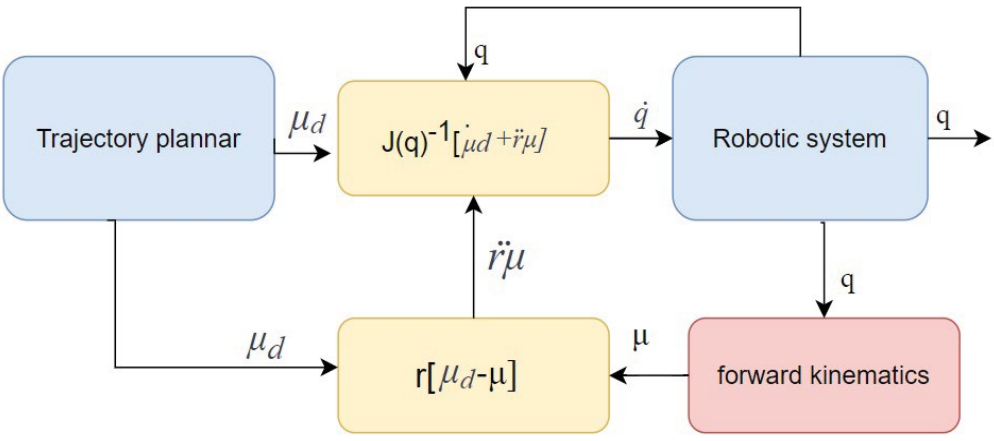


Figure 10: Dynamic analysis on task space scheme.

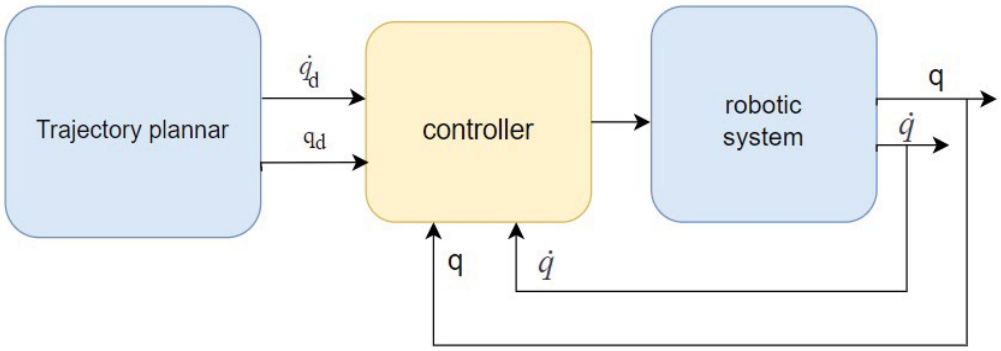


Figure 11: Dynamic analysis on joint space scheme.

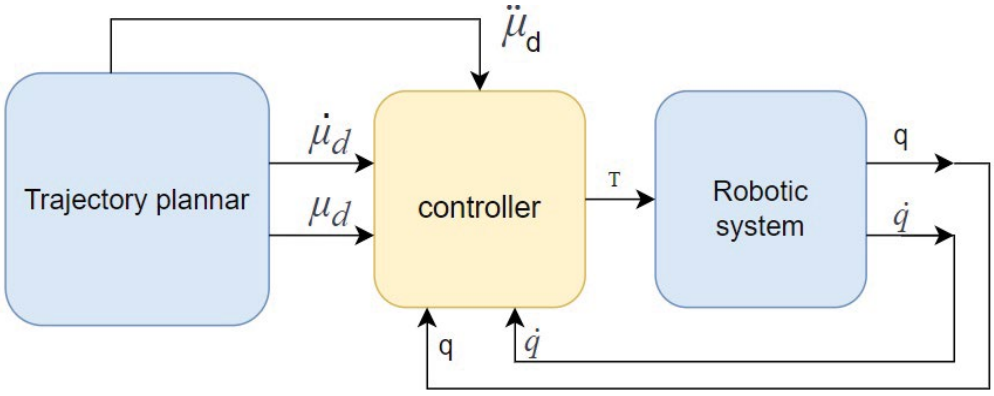


Figure 12: Dynamic analysis on task space scheme.

joint positions, $\mu(t)$, actual joint velocities $\dot{\mu}(t)$, inertia matrix $M(\mu)$, non-linear term $n(\mu,\dot{\mu})$, and the Jacobian matrix $J(q)$ along with its inverse $J^{-1}(q)$. The control flow for the task space scheme is illustrated in Figure 12.

The control inputs τ are determined by the equation

$$\tau = M(q)J(q)^{-1}\left[\ddot{\mu}_d - \dot{J}(q_d)\dot{q}_d + k_p(\mu_d - \mu) + k_d(\dot{\mu}_d - \dot{\mu})\right] + V(q,\dot{q}) + g(q) \quad (49)$$

Where $V(q,\dot{q})$ represents centrifugal and Coriolis effects, and $g(q)$ captures gravitational effects. This formulation highlights the intricate relationship between joint and task spaces in achieving dynamic control for robotic systems.

Conclusion

In conclusion, the comprehensive investigation conducted on the 4-axis SCARE IRB 930 robot has yielded profound insights into both its kinematic and dynamic behaviors. The thorough exploration encompassing forward kinematics, inverse kinematics, and differential kinematics (Section 2) has not only elucidated the intricacies of the robot's motion but also enriched our comprehension of its positioning capabilities. Moreover, the in-depth Jacobian analysis presented in Section 3, incorporating singularity analysis and velocity propagation, has unearthed critical revelations with broader implications transcending the confines of the specific robot model examined. In section 4, the development of the dynamic model employing both Lagrangian and Newton Euler methods has provided a deeper understanding of the robot's dynamic response and behavior. Significantly, the methodologies delineated in Section 5 for kinematic and dynamic control have been intentionally crafted to constitute an adaptable and integrated motion control framework applicable across diverse robotic systems. By refraining from solving for a singular robot model, this research not only contributes to the collective knowledge regarding

the 4-axis SCARA ABB IRB 930 robot also offers a versatile and universally applicable framework with far-reaching implications for the advancement of robotics and control methodologies. These findings represent a significant stride towards the evolution and enhancement of the research field, offering invaluable insights poised to propel future innovations in robotics and control theory.

References

1. Pagilla PR, Biao Yu (2004) An experimental study of planar impact of a robot manipulator. IEEE-ASME Transactions on Mechatronics (IEEE) 9: 123-128.
2. Stuhlenmiller F, Weyand S, Jungblut J, Schebek L, Clever D, et al. (2021) Impact of cycle time and payload of an industrial robot on resource efficiency. Robotics 10: 33.
3. Zacharia P Th, Aspragathos NA (2004) Optimization of industrial manipulators cycle time based on genetic algorithms. IEEE.
4. IRB 930 Datasheet.
5. Kucuk S, Bingul Z (2006) Robot kinematics: Forward and inverse kinematics, industrial robotics: Theory, Modelling and Control, Sam Cubero, ISBN: 3-86611-285-8, InTech.
6. Kucuk S, Bingul S (2004) The inverse kinematics solutions of industrial robot manipulators. Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM '04. Istanbul, Turkey: 274-279.
7. Webster RJ III, Jones BA (2010) Design and kinematic modeling of constant curvature continuum robots: A review. The International Journal of Robotics Research 29: 1661-1683.
8. Merlet JP (2007) Jacobian, manipulability, condition number and accuracy of parallel robots. In: Thrun S, Brooks R, Durrant-Whyte H, Robotics Research. Springer Tracts in Advanced Robotics, vol 28. Springer, Berlin, Heidelberg.
9. Donelan P (2010) Kinematic singularities of robot manipulators. Advances in Robot Manipulators. Ernest Hall, InTech. ISBN: 978-953-307-070-4.
10. Fahimi F, Ashrafiuon H, Nataraj C (2002) An improved inverse kinematic and velocity solution for spatial hyperredundant robots. IEEE Transactions on Robotics and Automation 18: 103-107.
11. Kardoš J (2010) The simplified dynamic model of a robot manipulator. In Proceedings of the 18th International Conference: 1-6.
12. De Luca A, Ferrajoli L (2009) A modified newton-euler method for dynamic computations in robot fault detection and control. IEEE International Conference on Robotics and Automation, Kobe, Japan, 3359-3364.
13. Astrom KJ, Hagglund Tore (1995) PID Controllers: Theory, Design, and Tuning. The International Society of Measurement and Control (2nd edn).
14. Chung WK, Fu L-C, Kröger T (2016) Motion control. Springer handbook of robotics, 163-194.

DOI: 10.35840/2631-5106/4137