A new scheme based on recursive fuzzy logic is presented in this paper for solving the point-to-point inverse kinematics problem of serial robots. To improve the convergence problem in the whole workspace, the membership functions of the fuzzy logic are searched for, tuned, and optimised using a simple genetic algorithm. A dominant joint, which brings the end-effect closer to the desired target, has to be selected before the implementation of the fuzzy logic in order to reduce the number of fuzzy logic iterations. The inverse kinematics solution of robots is usually obtained by direct inversion of the kinematics equations, but this technique often leads to a singular Jacobian matrix during the calculations. The work presented in this paper provides a direct approach to the calculation of the kinematics inverse problem which bypasses the kinematic singularities. Computer simulations of the proposed scheme confirm the findings of the theoretical developments.

Keywords: Fuzzy logic; Genetic algorithm; Inverse kinematics

1. Introduction

The fundamental problems of robot kinematics are the computation of the forward and inverse mappings between joint space and Cartesian task space. The former, mapping from joint space to Cartesian task space, is called forward kinematics and the latter, i.e. mapping from Cartesian task space to joint space, is called inverse kinematics. In robot control applications, the inverse kinematics is more important since, by default, robot tool trajectories are usually specified in terms of Cartesian task space coordinates. However, some actuators may receive commands in the joint space coordinates only. In this situation, the trajectories must be transformed to the joint space before the robot is controlled. The inverse kinematics of a serial robot is more complex than the forward kinematics since the inverse kinematics equation are more complex and nonlinear. There are two approaches to the solution of the inverse kinematics problem:

1. Closed-form solutions including algebraic and geometric methods; see for example [1–4].
2. Numerical solutions [5,6].

In general, closed-form solution to the inverse kinematics problem may not exist except when the robot is designed with a specific kinematic structure [7]. Moreover, most numerical solutions rely on the calculation of the inverse Jacobian matrix \( J^{-1} \) to solve the inverse kinematics problem. However, if the manipulator is near a singularity region, \( J^{-1} \) does not exist, thus, the inverse kinematics solution will not converge. It is worth noting that, for redundant robots, there may not be a one-to-one mapping between Cartesian task space and joint space. Therefore, the kinematics redundancy problem is usually resolved by adding enough external constraints in the solution procedure. In fact, in most robot applications, it is not necessary to control the robot in every single position during operation.

Fuzzy logic is a powerful tool for the solution of complex nonlinear systems. For many years, fuzzy logic has been applied successfully to a wide variety of problems; see for example [8, 9]. A fuzzy logic robot controller for a serial manipulator was developed by Nedungadi and Wenzel [10]. However, in some cases, the designed controller seemed to involve a blind spot in the fuzzy rules. To clarify this statement, consider the two degrees of freedom (DOF) robot shown in Fig. 1. The tool trajectories for joints 1 and 2 are represented in the figure by the dashed arcs. It is desired to move the robot from position 0 to positions 1 and 2. Using the scheme developed by Nedungadi and Wenzel [10], the joint angles \( \theta_1 \) and \( \theta_2 \) must both increase to reach either position 1 or 2 since the fuzzy logic input is the same. However, physically, joint angle 2 must be increased to reach either position 1 or 2, whereas joint angle 1 must be decreased to reach position 1 and increased to reach position 2. Recently, a few workers have proposed fuzzy rule-based approaches for robot motion
This paper presents a recursive fuzzy logic approach to solving the inverse kinematics of serial robots. This technique uses a simple genetic algorithm (SGA) to tune optimal fuzzy membership functions automatically. Moreover, SGA greatly reduces the effort to tune fuzzy behaviours. In the work presented in this paper, a dominant joint (or the joint which brings the end-effector closer to the desired target) is chosen to reduce the number of iterations. Then, the joint angle variation is evaluated using fuzzy logic. In this way, it is not necessary to add external constraints for redundant robots, so the singularity problem in the calculation procedure is avoided.

The paper is organised as follows. In Section 2, fuzzy logic for the inverse kinematics problem and the genetic algorithm are presented. Simulation results from 2DOF and 4DOF robots are presented and discussed in Section 3. Some concluding remarks are given in Section 4.

2. Synthesis of Fuzzy Logic and Genetic Algorithm

2.1 Fuzzy Logic for Inverse Kinematics Problem

Inverse kinematics mapping from Cartesian task space to joint space for a 1DOF robot has been developed in [14] with a satisfactory tolerance error. The procedure and algorithm developed in [14] are adopted in this work. In order to reduce the computation time in the training procedure, the fuzzy associative memory (FAM) is replaced with a fuzzy logic, genetic algorithm.

The following is an overview of the learning algorithm. For a single degree of freedom robot, the robot forward kinematics Eq. can be written as,

\[ P = [P_x P_y]^T \]  

\[ d(P_k) = A_i dF(\theta_k), \quad (i = 1, \ k = x,y) \]

where \( d(P_k) \) and \( A_i \) are the input variables and \( dF(\theta_k) \) is the output variable used to start the FAM recursive scheme. For example, for a single 1DOF robot with a link length \( L_1 \) and a joint angle \( \theta_1 \), the mapping expressions can be written as follows:

\[ x = L_1 \cos(\theta_1) \]  

\[ y = L_1 \sin(\theta_1) \]

Linearising Eqs (3) and (4), we obtain:

\[ dx = -(L_1 \sin \theta_1) d\theta_1 = A_{1x} d\theta_1, \]  

\[ dy = (L_1 \cos \theta_1) d\theta_1 = A_{1y} d\theta_1. \]

The flowchart for the learning algorithm is shown in Fig. 2. The variable \( dP_i \) represents the difference between the start and desired locations, \( E \) is the error term, and \( E_t \) is the tolerance error. The error \( E \) is given by

\[ E = [(E_x(X))^2 + (E_y(Y))^2 + (E_z(Z))^2]^{0.5} \]

where \( E_x(X) \), \( E_y(Y) \), and \( E_z(Z) \) are the errors between the current position of the end-effector and the desired location in the \( X \)-, \( Y \)-, and \( Z \)-directions, respectively. In this work, the algorithm designed for the 1DOF robot will be extended to multi-degree-of-freedom robots.

Consider a 1DOF robot that is used to reach a desired target not lying in the work space. The distance from the end effector to the origin (global or local, whichever is appropriate) is called the virtual link. This link idea is used here to extend the previous algorithm to multi-degree-of-freedom robots. To clarify the extension procedure, consider the 2DOF robot shown in Fig. 3. The robot can be considered as two 1DOF robots.