Journal of Current Science and Technology, September-December 2024 Vol. 14 No. 3, Article 68

ISSN 2630-0656 (Online)

Cite this article: Nguyen, X. V., & Nguyen, N. L. (2024). Using genetic algorithms and bézier curves for automatic path optimization of a 6-DOF robot. *Journal of Current Science and Technology*, *14*(3), Article 68. https://doi.org/10.59796/jcst.V14N3.2024.68

Journal of Current Science and Technology

Journal homepage: https://jcst.rsu.ac.th

Using Genetic Algorithms and Bézier Curves for Automatic Path Optimization of a 6-DOF Robot

Xuan-Vinh Nguyen[1,2,*], and Ngoc-Lam Nguyen[3]

[1]Faculty of Electronics and Telecommunications, University of Science, Ho Chi Minh City, Vietnam
[2]Vietnam National University, Ho Chi Minh City, Vietnam
[3]Vietnam Research Institute of Electronics, Informatics and Automation

*Corresponding author; E-mail: nxvinh@hcmus.edu.vn

Received 25 April 2024; Revised 17 June 2024; Accepted 25 June 2024
Published online 1 September 2024

Abstract

This paper describes a novel method for automatically planning point-to-point motion paths for a robot with six degrees of freedom (6-DOF). A linear motion path between two points on such robots may be impractical due to joint angle constraints or exceeding the manipulator's operational range. The proposed method employs a genetic algorithm to generate suitable motion paths based on the second-, third-, and fourth-orders of Bézier curves. The control points of Bézier curves are determined using a genetic algorithm, which can adjust the fitness function as the end-effector moves closer to the obstacle. As a result, the algorithm can adjust its motion path planning in response to obstacles. The motion paths are generated with the goal of optimizing the robot's inverse kinematic configuration. The results show that using a genetic algorithm and Bézier curves can produce motion paths with smooth transitions, minimal changes in joint angles, and no sudden jerks within the robot's operational area in both obstacle-free and obstacle avoidance scenarios. This solution may be useful for intelligent robots with automated path-planning capabilities in unknown environments.

Keywords**:** *6-DOF robot; kinematics; path planning; genetic algorithm; Bézier curves*

Nomenclature

Symbol	Description
ψ, θ, ϕ	Roll, Pitch, Yaw angle
θ_i	Joint angle boundary
α_i	Connecting rod torque
a_i	Connecting rod length
d_i	Joint offset
A_{i-1}^{i}	Homogeneous transformation matrix
$R_{(\phi,\theta,\psi)}$	Rotation matrix
P_i	Control point of Bézier curve
SAC_i	Sum of available inverse kinematic configurations
δ	safe distance.
λ	reduction factor

1. Introduction

The six-degrees-of-freedom (6-DOF) robot is the most powerful tool in modern industrial production and life. With the advantage of high flexibility, robots with six degrees of freedom can perform flexible operations like human hands and avoid obstacles more easily than robots with fewer degrees of freedom. In current practical applications, 6-DOF robots mainly perform predetermined operations according to known and repeated trajectories in production lines. When initially training the robot, experts will analyze the motion process and based on personal experience, choose a fixed, suitable

inverse kinematics configuration. When operating, the 6-DOF robot will run automatically according to the selected default process to perform repetitive operations in a fixed workspace.

As applications in unpredictable environments become more common, the demand for intelligent robots is growing. To ensure stable operation, these robots must be able to calculate and create various motion paths on their own. This capability enables robots to establish and adapt their work pathways in response to external impacts in the workplace. As a result, the traditional approach of selecting a single fixed trajectory is no longer sufficient. Instead, modern robots need to be able to automatically generate and select operating trajectories that accurately reflect real-world conditions.

Planning the motion path of a 6-DOF manipulator is an important part of the robot control process. Motion path planning allows the creation of a realistic route from a starting point to a goal point. A path is usually made up of a series of connected waypoints. In path planning, we must consider the robot's constraints and limitations, such as rotation angle, workspace, and available inverse configurations. The problem of planning the motion trajectory of a 6-DOF manipulator presents numerous calculation and control challenges.

According to Gasparetto et al., (2015), path planning and trajectory algorithms are important in robotics and automation because they ensure smooth trajectories required for high-speed operation while preventing excessive accelerations and vibrations in robot actuators and structures.

Several articles have been published on the application of evolutionary techniques to robot motion planning. Juříček et al., (2023) provide an overview of evolutionary computation (EC) techniques commonly used in engineering, specifically industrial robotics. Several of these articles focus on a genetic algorithm that generates a moving path for two- or three- link robots. Kazem et al., (2008) used a genetic algorithm (GA) to optimize point-to-point trajectory planning for a three-link robot arm, reducing travel time and space while avoiding collisions.

Some research on the 6-DOF robot focuses on automated and optimized path planning and trajectory methods. Perumaal, & Jawahar (2012) proposed an S-Curve trajectory planner for minimum-time and collision-free pick-and-place operations in the presence of obstacles. Masajedi et al., (2013) used the Bee Algorithm (which simulates the food foraging behavior of honeybee colonies: neighborhood search combined with global search) to assess reliability for both linear and curved trajectories for a 6-DOF robot running ADAMS software.

Path planning for a 6-DOF robot has received little attention in genetic algorithm research. Hou et al., (2023) used the non-dominated sorting genetic algorithm II to improve efficiency and reduce runtime in trajectory planning for 6-DOF manipulators while accounting for kinematic constraints.

Zhang et al., (2018) propose an improved adaptive genetic algorithm for optimizing interpolation point time intervals, resulting in time-optimal trajectory planning by adjusting the crossover and mutation operators in the general adaptive genetic algorithm.

Baressi Šegota et al., (2020) propose utilizing evolutionary computation algorithms to optimize robotic manipulator paths and reduce joint torques. The results show that the genetic algorithm performs the best in terms of torque minimization, with differential evolution providing comparatively good results and simulated annealing producing the weakest results while providing smoother torque curves.

Meng et al., (2021) propose a method to improve the efficiency of a robotic chainsaw when cutting complex timber joints. They employ particle swarm optimization (PSO) to determine the shortest cutting path and an adaptive genetic algorithm (AGA) to optimize the timing of interpolation points, resulting in a time-optimal trajectory. Their findings show that PSO reduces path length while AGA smoothens trajectories and minimizes time intervals, demonstrating the efficacy of their approach for 6-DOF robots in cutting tasks.

Mousa et al., (2023) investigate path planning algorithms for a 6-DOF robotic arm, comparing polynomial methods to the Whale Optimization Algorithm (WOA: an optimization algorithm that mimics humpback whales' natural hunting mechanism) and the Genetic Algorithm (GA). In comparison to GA, WOA is better in terms of implementation speed and accuracy. However, GA has a smoother beginning and end to the moving path.

Bézier curves, a popular method in computer graphics and design, are a versatile way to represent and manipulate curves. According to Hughes et al., (2014), Bézier curves provide smooth and flexible curves that are commonly used in image representation, design, and other graphics applications. Yoshida et al., (2010) present an interactive method for generating a 3D class A Bézier curve segment by specifying two endpoints and their tangents.

Recently, researchers have investigated the use of Bézier curves in planning robot motion trajectories, as reported in recent publications. Elhoseny et al., (2017) describe an efficient dynamic path planning method for mobile robots that uses Bézier curves and the Modified Genetic Algorithm (MGA). This method dynamically determines the robot's path by optimizing the distance between the start and target points with appropriate control point selection for the Bézier curve.

Ma et al., (2020) present a novel smooth path planning technique based on Bézier curves that addresses issues such as redundant nodes and peak inflection points in traditional algorithms for a mobile robot. They use genetic operations to obtain control points and then apply an optimization criterion to choose a shorter path while maintaining movement safety.

AL-Qassara, & Abdulnabib (2018) present an optimal path planning strategy for 5-DOF robots that employs the Bézier curve technique in robot joint space. They use the particle swarm optimization (PSO) method to determine the best path with the shortest distance and least time while avoiding obstacles.

The survey of related studies shows that researchers predominantly employ pure genetic algorithms to generate motion trajectories for manipulators. These studies are often confined to robots with low degrees of freedom (Kazem et al., 2008) or concentrate on specific optimization criteria such as movement time (Zhang et al., 2018; Meng et al., 2021), path distance, joint torque (Baressi Šegota et al., 2020; Meng et al., 2021), speed, and movement accuracy (Mousa et al., 2023).

Articles combining genetic algorithms with Bézier curves primarily apply to mobile robots and aim to create flexible paths while avoiding collisions by using some intermediate points for the generated movement paths (Elhoseny et al., 2017; Ma et al., 2020). Only one study has been conducted on the PSO swarm algorithm to optimize the distance and control time for a 5-DOF robot. No specific studies have used genetic algorithms and Bézier curves to generate motion paths for a 6-DOF robot.

Using a single genetic algorithm to plan smooth, jerk-free paths point to point faces numerous challenges, especially because the algorithm's chromosomes must manage a large number of intermediate points. This may necessitate a large population size, resulting in longer computation times and results. Furthermore, combining such algorithms with controllers that can dynamically adjust trajectories in response to obstacles complicates and turns the process unrealistic.

This article presents a method for creating smooth, jerk-free paths for a 6-DOF robot moving from point to point using Bézier curves. The genetic algorithm will be designed to find control points (1–3) on Bézier curves. The fitness function of the genetic algorithm integration for the optimal path selection of the inverse kinematic configurations is described by Nguyen, & Nguyen (2024). This option is proposed to limit the size of the genetic algorithm and simplify the calculation process, with the goal of being feasible when applied to intelligent controllers for robots in unknown working environments.

Our prior study utilized computer simulations to evaluate the robot's workspace, factoring in joint angles as constraints (Nguyen, & Nguyen, 2024). Results show that considering joint angles significantly shrinks the robot's effective workspace. Additionally, we introduce a method enabling the robot to autonomously select the optimal configuration during path planning, especially in uncertain scenarios. This capability facilitates seamless configuration adjustments in line with minimizing changes. The findings from this study will serve as the foundation for the algorithms outlined and discussed in the subsequent sections of this article.

The sections that follow present a solution for motion path planning for a 6-DOF robot using genetic algorithms and Bézier curves. Section 2 presents the results of our inverse kinematics analysis. For forward kinematics, we used the Denavit-Hartenberg (D-H) method (Denavit, & Hartenberg, 1955), and for inverse kinematics, we used a two-stage geometric approach to account for both end-effector position and orientation. A genetic algorithm combined with Bézier curves creates and adjusts point-to-point motion paths in the robot's workspace, considering the possibility of avoiding obstacles during motion path planning, as described in Section 3. Section 4 presents the simulation results of the end-effector's motion paths using Bézier curves with obstacle avoidance. Our results demonstrate the feasibility of manually constructing and adjusting trajectory paths to avoid obstacles in the manipulator’s workspace. Section 5 presents the paper's conclusion.

2. Kinematic model

Kinematic problems, such as forward and inverse kinematics, must be solved to plan the motion path of a robot. The Denavit-Hartenberg's principle of determining the coordinate system and parameters is frequently applied to forward kinematics problems. Finding the joint variables to bring the end-effector to

the desired position for a 6-axis robot is a complex inverse kinematics problem (Hartenberg, & Denavit, 1964). As a result, geometric methods are widely used. However, the interdependent results in the multi-solution problem remain a significant barrier to determining a suitable configuration for a 6-DOF robot (Yang et al., 2016).

2.1 Forward kinematics

According to Spong et al., (2004), the typical 6-DOF manipulator platform has the six homogeneous transformation matrices A_{i-1}^{i} given in Equation (1):

$$A_{i-1}^{i} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where

$c\alpha_i \equiv cos(\alpha_i)$; $s\alpha_i \equiv sin(\alpha_i)$;(i=1,2, ..., 6)

$c\theta_i \equiv cos(\theta_i)$;$s\theta_i \equiv sin(\theta_i)$;(i=1,2, ..., 6)

Lee, & Ziegler (1984) introduced the concept of forward kinematics, with Equation (2) detailing the transformation matrix. This matrix represents the spatial relationship between the end-effector and the robot's base coordinate system by determining its position and orientation.

$$T_0^6 = A_0^1 A_1^2 A_2^3 A_3^4 A_4^5 A_5^6$$

$$T_0^6 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_0^6 & P_0^6 \\ 0 & 1 \end{bmatrix} \quad (2)$$

The vectors *n*, *s*, and *a* define the end-effector's orientation, while the vector *p* represents its position. R_0^6 is the rotational matrix that describes the orientation of the end-effector's coordinate system in relation to the base coordinate system, while P_0^6 is the position vector of the end-effector's coordinate system within the base coordinate system. Together, these vectors describe the end-effector's spatial configuration in the base coordinate system.

2.2 Inverse kinematics

For a 6-DOF robot, solving the inverse kinematics problem involves finding the joint angles that correspond to a specified end-effector position and orientation. According to Piotrowski, & Barylski, (2014), this problem can be approached in two stages: first, determining the inverse kinematics of position to calculate the first three joint angles $(\theta_1, \theta_2, \theta_3)$, and then determining the inverse kinematics of orientation to calculate the final three joint angles $(\theta_4, \theta_5, \theta_6)$. This division simplifies the overall kinematic analysis and is outlined in Nguyen, & Nguyen (2024).

According to Spong et al., (2004), the rotation matrix, based on the rotation angles Roll, Pitch, and Yaw (RPY), is illustrated as follows:

$$R_{(\phi,\theta,\psi)} = R_z(\phi).R_y(\theta).R_x(\psi) = \begin{bmatrix} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & c\phi s\theta c\psi + s\phi s\psi \\ s\phi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix} \quad (3)$$

Equation (4) provides the orientation matrix R_0^6, which is obtained for wrist orientation and has the same structure as the RPY transformation matrix $R_{(\phi,\theta,\psi)}$.

$$R_0^6 = R_0^1 R_1^2 R_2^3 R_3^4 R_4^5 R_5^6 = R_0^3 . R_3^6 \quad (4)$$

The matrixes $R_{(\phi,\theta,\psi)}$ and R_0^6 can be compared as follows:

$$R_3^6 = (R_0^3)^{-1} R_0^6 = (R_0^3)^{-1} R_{(\phi,\theta,\psi)} \quad (5)$$

Equation (5) derives the matrix $(R_0^3)^{-1}$ from angles $\theta_1, \theta_2, \theta_3$, and matrix $R_{(\phi,\theta,\psi)}$is defined by RPY angles. Therefore, it becomes possible to find the remaining corresponding angles $\theta_4, \theta_5, \theta_6$ from matrix R_3^6.

The inverse kinematics problem of a 6-DOF robot usually has four sets of solutions corresponding to four different configurations at the same position and direction angle of the terminal coordinates. Dividing the analysis process into two sequential stages simplifies determining solution sets compared to traditional methods. The following section discusses automatic path planning with a genetic algorithm and Bézier curves, including obstacle avoidance considerations.

3. Automated path optimization using genetic algorithms combined with Bézier curves

3.1 Bézier curve

A Bézier curve is a continuous, smooth curve composed of a starting point, a number of control points, and an end point. Figure 1 demonstrates how the Bezier curve's high-order derivative continuity enables it to change smoothly from beginning to end. As a result, the Bezier curve is useful for planning the robot's travel path because it is both continuous and smooth (Abbas et al., 2011).

In the *Oxyz* coordinate system (Farin, 2006), the Bezier curve of degree '*n*' is defined as:

$$P(t) = \sum_{i=0}^{n} B_i^n(t) P_i \quad 0 \le t \le 1 \qquad (6)$$

where:

$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$, $i=0,1,\ldots,n$ is Bernstein polynomials with degree 'n'.

$P_i = (x_i, y_i, z_i)$, $i=0,1,\ldots,n$ are the control points of Bézier curve.

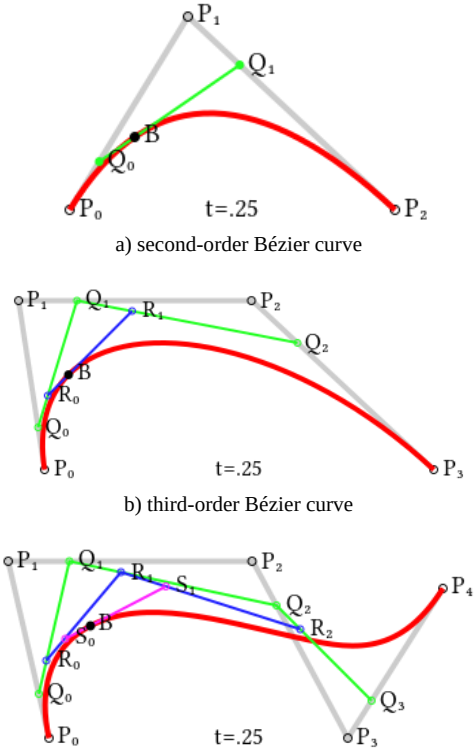


Figure 1 Construction of Bézier curves (Reprinted from Bézier curve, 2014)

Typically, traditional path planning algorithms use Bézier curves, which have fixed starting and ending points while other points change the curve's trajectory. In this paper, the Bézier curve property was used to plan the path for a 6-DOF manipulator's end-effector, with the Bézier curve's starting and ending control points (P_0 and P_n) representing the starting and target positions of the moving path, respectively. The remaining control points P_j ($j=1,2,\ldots,n-1$) on the Bézier curve are used to alter the motion path, allowing the robot arm to travel from its starting point to its final destination in the workspace. In the following section, the proposed genetic algorithm creates control points for the Bézier curve, yielding a smooth motion path with obstacles avoidance in the robot's environment.

3.2 Genetic algorithm.

Genetic algorithms are a technique in the fields of machine learning and artificial intelligence inspired by genetic mechanisms and natural evolution. The basic idea of this algorithm is to use genetic operations such as crossover and mutation to create new generations from the best individuals of the current generation. Genetic algorithms are often applied in optimization problems and searching large solution spaces, including the problems of finding optimal paths, optimizing objective functions, and training neural networks in machine learning. The outstanding feature of this algorithm is its ability to automatically find potential solutions without the need for deep expertise. This genetic algorithm is also suitable for creating a manipulator's movement trajectory, particularly in unknown working environments.

The purpose of this article is to use genetic algorithms to generate an automatic path for a 6-DOF manipulator's end-effector moving from point A (x_A, y_A, z_A) to point B (x_B, y_B, z_B), using the Bézier curve inside the workspace. The starting and ending points of the Bézier curve are assigned to the positions of points A and B on the moving path. A genetic algorithm is used to determine the position of the remaining control points P_j (x_j, y_j, z_j) of the Bézier curve by assigning values x_j, y_j, and z_j along the xyz coordinate axes to the population member's chromosomes. After each evolutionary process, the positions of points P_j (x_j, y_j, z_j) will be determined using the changing value of the optimal function. In this algorithm, the objective optimization function is the total number of available inverse kinematic configurations of the robot during the end-effector's motion along the path from the start point to the endpoint. The number of inverse kinematic configurations during manipulator end-effector movement is determined using a tool developed and presented by the authors in a recent publication (Nguyen, & Nguyen, 2024).

The objective function will be different for each path formed by the Bézier curve with control points P_j. The best P_j position combinations from each evolutionary cycle will be carried over to the next evolutionary cycle. After completing the evolution process with the optimal objective function, the manipulator end-effector's moving path will be built using a Bézier curve with the starting, ending, and control points P_j. Figure 2 depicts the proposed genetic algorithm combined with the Bézier curve, while Table 1 provides information on the fitness function and adjusting for obstacle avoidance.

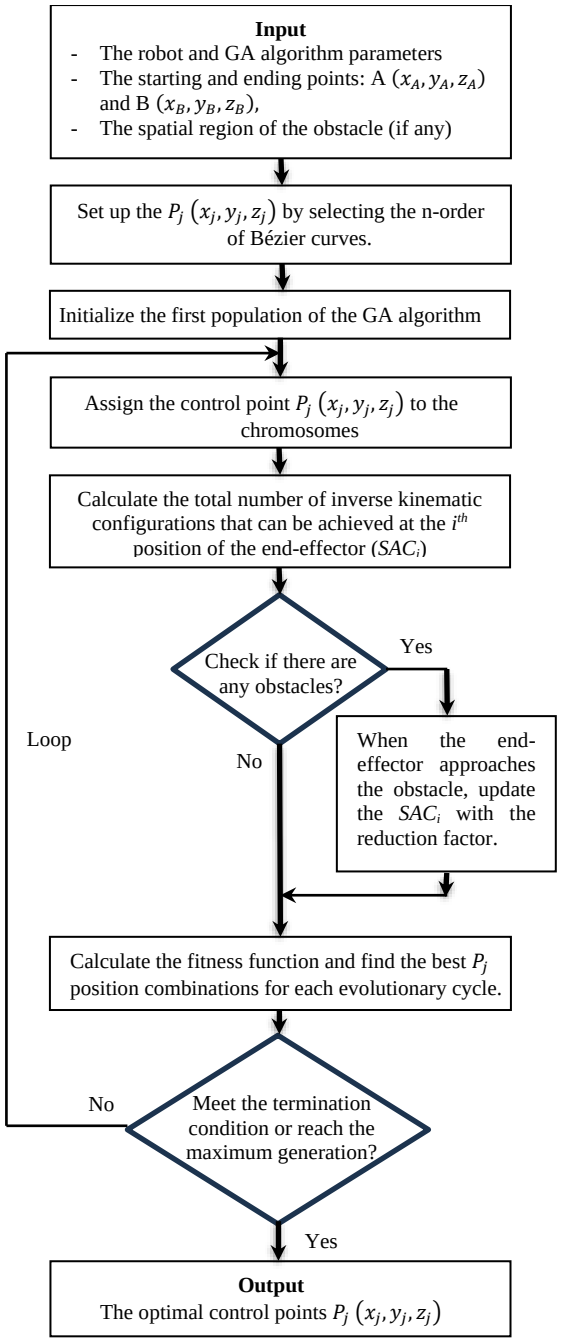


Figure 2 Flow chart of path planning using genetic algorithms combined with the Bézier curve

Table 1 The fitness function and adjusting to obstacle avoidance

Input: The 6-DOF robot's parameters and workspace, path start and end points, the n-order Bézier formula, GA algorithm parameters, and the spatial region of the obstacle.

Procedure: Automated path planning and obstacle avoidance

Path-planning algorithm:

1. Initialize the first population.
2. Main loop:

- Assign $P_j\ (x_j, y_j, z_j)$ into the chromosome.
- Calculate the optimal function and then apply selection, crossover, and mutation to the population's chromosomes.

The optimal function is given below:

$$fitness = \frac{\sum_{i=1}^{n} SAC_i}{n} \quad (7)$$

where:

- n is the number of switched points on the shortest path on the path from point A to point B according to Nguyen and Nguyen (2024).
- SAC_i is the total number of inverse kinematic configurations that can be achieved at the i^{th} position of end-effector, calculated using the following pseudocode:

The pseudocode computes SAC_i

for x_i=1:n
  for y_i = 1:n
    for z_i = 1:n
      if (x_{min} - δ ≤ x_i ≤ x_{min} + δ) and (y_{min} - δ ≤ y_i ≤ y_{min} + δ) and (z_{min} - δ ≤ z_i ≤ z_{min} + δ)
        $SAC_i = \lambda . \sum_{k=1}^{4} C_k$
      else
        $SAC_i = \sum_{k=1}^{4} C_k$
      end
    end
  end
end

where:

- x_j, y_j, z_j are the positions of the robot's end-effector.
- $x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}$ are obstacle region limits in the *Oxyz* coordinate system.
- δ is the safe distance between the manipulator end-effector and the obstacle.
- λ is the reduction factor when the end-effector is close to the obstacle.
- C_k is equal to 1 if the k^{th} inverse configuration satisfies the constraint of all joint angles at the i^{th} position and is equal to 0 otherwise ($k = 1,2,3,4$).

- Repeat the main loop until the termination condition is met or the maximum generation is achieved.

Output: The $P_j\ (x_j, y_j, z_j)$ positions of the Bézier curve, moving paths, and fitness values.

This genetic algorithm also considers the ability to avoid obstacles on the constructed paths. It identifies a box-shaped structure as the obstacle on a previously generated path. The algorithm then adjusts its objective function to consider the path segments located in the obstacle's spatial region. Furthermore, to ensure the manipulator's safety while moving to avoid obstacles, a safe distance is included in the condition function for recalculating the objective function in the algorithm. Equation (7) modifies the SAC_i value when the robot's end-effector position (x_i, y_i, z_i) is within both the obstacle's area (defined by $x_{min}, y_{min}, z_{min}$, $x_{max}, y_{max}, z_{max}$) and a nearby safe zone with a spatial offset (δ). To achieve an accurate and varied examination, the genetic algorithm generates moving trajectories using 2nd, 3rd, and 4th-order Bézier curves. Furthermore, a simulation program is developed to evaluate the manipulator's movement along these paths.

4. Simulation results and discussion

The typical 6-DOF manipulator used in this paper is comparable to the KUKA KR 15/2 industrial robot described by Gracia et al., (2009). To simulate the algorithm, the authors suggest a robot model with Denavit-Hartenberg (D-H) parameters similar to the AKB-IRV1 robot from AKB Machinery (n.d.). This proposed robot has similarities to the object described in our previous study (Nguyen, & Nguyen, 2024), and its D-H parameters are shown in Table 2. Furthermore, Table 3 provides a foundational set of general parameters for the proposed genetic algorithm to use in simulation.

Table 2 The 6-DOF robot's D-H parameters

Joint i	α_i ***(rad)***	a_i **(mm)**	d_i ***(mm)***
1	$\pi/2$	100	370
2	0	300	0
3	$\pi/2$	111.36	0
4	$-\pi/2$	0	300
5	$\pi/2$	0	0
6	0	0	105

Table 3 Genetic algorithm parameters

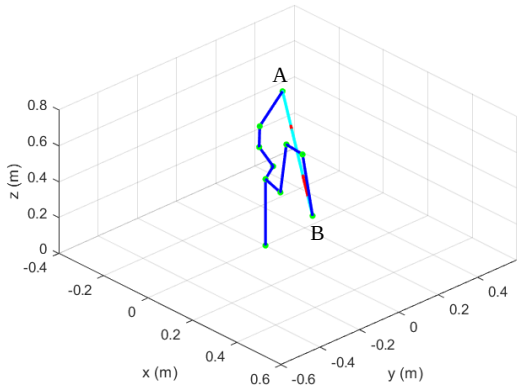
The number of individuals in the population	100
Probability of mutation (%)	5%
Probability of crossover (%)	89%
Max generation (cycles)	200
Selection type	Tournament
Crossover type	Two points
Safe distance (δ) (m)	0.05
Reduction factor (λ)	-1

All simulation results in this section are evaluated using the proposed manipulator D-H parameters in Table 2, with the suggested limit on the manipulator's joint angle determined as $-\pi < \theta_i < \pi$, and the genetic algorithm parameters in Table 3.

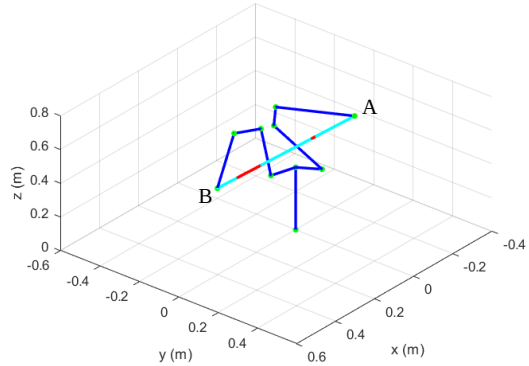
4.1 Problem description

The basic motion planning problem is to find a suitable pathway from point A to point B for the end-effector in the *Oxyz* coordinate system. The first and simplest idea is to create a straight-line path for the end-effector. With this straight-line motion, the end-effector moves linearly along points that divide equally in *Oxyz* coordinates.

Consider a 6-DOF manipulator with parameters D-H according to Table 2. Let the end-effector move in a straight line from point A (-0.1, 0.2, 0.7) to point B (0.3, -0.1, 0.4) with a fixed direction angle $(\phi, \theta, \varphi = 0)$ in Oxyz coordinate. This straight-line moving path is divided into n = 1000 equally spaced discrete points. At each discrete point on the motion path, the feasibility of inverse kinematic configurations is considered with the limit on the manipulator's joint angles.



a) initial viewpoint



b) subsequent angle

Figure 3 The end-effector moves in a straight line from point A (-0.1, 0.2, 0.7) to point B (0.3, -0.1, 0.4)

The survey results depicted in Figure 3, captured from two distinct viewing angles, illustrate the straight-line trajectory of the end-effector from point A to point B. Some specific red-colored points have no available inverse kinematic configurations. Those red-colored points show impossible motion path segments for the end-effector. This may be due to constraints in joint angles or the motion path occurring partially outside the manipulator's operating range. Along the path, the cyan segments represent the discrete points that have at least one available inverse kinematic configuration.

According to Nguyen, & Nguyen (2024), Figure 4 shows the sequence of configurations as the end-effector transitions from point A to point B through 1000 evenly distributed discrete points along the pathway. Configurations 1 and 3 are satisfied by separate segments in the response, but configurations 2 and 4 are not available. The interval points (1-281), (309-680), and (934-1000) align with configuration 1, whereas points (851-1000) correspond to configuration 3. When comparing the four configurations, no inverse configuration is available that matches the end-effectors' straight motion path in the (282–308) and (681–850) interval points.

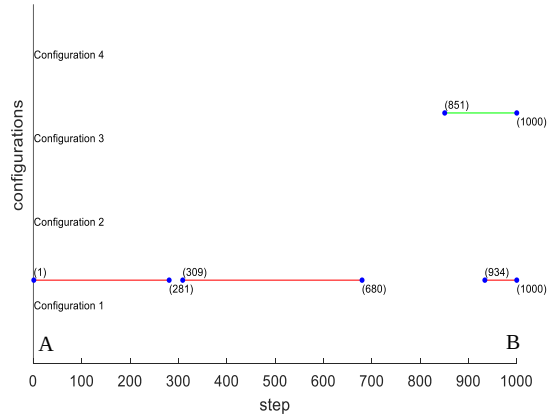


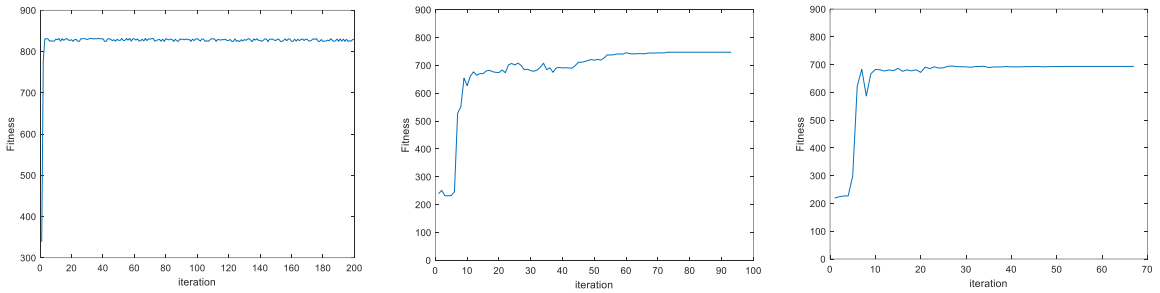
Figure 4 Response of inverse kinematic configurations along the straight line from point A to point B

Straight-line motion paths to move the end-effector from one location to another are not always feasible. In this case, it is necessary to create a new, fully accessible path from A to B. Every discrete point on this path must have at least one available inverse kinematic configuration. This path planning can be based on the operator's expertise or on intelligent methods such as genetic algorithms. The following section will show the simulation results for the point-to-point motion planning algorithm, which applies a genetic algorithm to combine Bézier curves of varying orders while considering obstacle avoidance.

4.2 Automated path optimization

Consider the second-, third-, and fourth- order Bézier curves, as shown in Figure 1, with the moving path's starting and ending points P_0 and P_n, respectively. A genetic algorithm will be used to determine the position of the control points P_j in the Oxyz coordinate system and create motion paths for the end-effector to move from point A to point B. These curves are required to accommodate joint angle limit conditions and inverse kinematic configurations within the robot's workspace.

Figure 5 illustrates the optimal result obtained by applying the genetic algorithm with the parameters in Table 4 to determine the position of the control points P_j in the Oxyz space using the objective function (7). For the quadratic (second-order) Bézier curve, the genetic algorithm gives the best results at the 32nd iteration cycle, with a maximum fitness value of 832, where the control point is P_1 (0.0611, -0.5386, 0.5517). For the third-order Bézier curve, the genetic algorithm gives the best results at the 73rd iteration cycle, with a maximum fitness value of 747.5, where the control points are P_1 (0.3299, 0.3317, 0.4009) and P_2 (0.5517, 0.5517, 0.5517). For the fourth-order Bézier curve, the genetic algorithm has the best results at the 27th iteration cycle, with a maximum fitness value of 695, where the control points P_1 (-0.2425, 0.2177, 0.2307), P_2 (0.2627, 0.5517, 0.5517), and P_3 (0.5517, 0.5517, 0.5517) are located.



a) second-order Bézier curve b) third-order Bézier curve c) fourth-order Bézier curve

Figure 5 Fitness value using Bézier curves

Table 4 Optimal result using a genetic algorithm to create moving paths based on Bézier curves

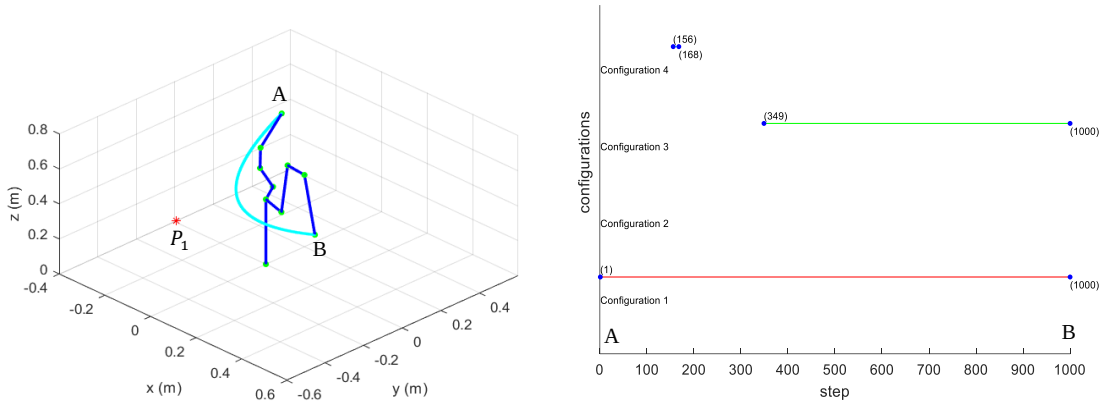
Bézier curve order	**Second order**	**Third order**	**Fourth order**
Max fitness value	832	747.5	695
Best iteration cycle	32^{nd}	73^{rd}	27^{th}
Control point $P_1(x_1, y_1, z_1)$	(0.0611, -0.5386, 0.5517)	(0.3299, 0.3317, 0.4009)	(-0.2425, 0.2177, 0.2307)
Control point $P_2(x_2, y_2, z_2)$	-	(0.5517, 0.5517, 0.5517)	(0.2627, 0.5517, 0.5517)
Control point $P_3(x_3, y_3, z_3)$	-	-	(0.5517, 0.5517, 0.5517)
Total available inverse configuration	1,663	1,494	1389

Figures 6a, 7a, and 8a show the end-effector's motion path relative to the order of the Bézier curves, displaying its smooth and responsive movement. These figures show the motion curve generated by the genetic algorithm in the *Oxyz* coordinates, which corresponds to the constraints of joint angles and selected inverse kinematic configurations. Figures 6b, 7b, and 8b show various inverse kinematic configurations based on these motion paths. These diagrams show how the inverse configuration aligns with the end-effector's movement from point A to point B using Bézier curves.

Figures 6b, 7b, and 8b show that the first inverse kinematic configuration corresponds to the entire motion path of the end-effector. Meanwhile, the remaining inverse kinematic configurations are only available within specific ranges. In each of the three cases, the first configuration will be chosen primarily for the motion controller. The number of the third inverse kinematic configurations for the second-order Bézier curve increases significantly (394–1000) when compared to linear motion (Figure 4). Additionally, a small segment (156–168) of the fourth inverse kinematic configuration appears in Figure 6b. In comparison to the linear motion in Figure 4, the third inverse kinematic configuration reduces the number of suitable configurations for the third-order Bézier curve from 149 to 55. Meanwhile, configuration 2 in the range (360–799) appears to be new, responding to motion paths as an additional controller option. This is similar to the 4th order Bézier curve; the number of the third inverse kinematic configurations is reduced to 40, and configuration 2 gains an additional segment (496–845).

After calculating the total number of available inverse configurations capable of responding to the end-effector's motion paths, it was discovered that the straight-line motion path shown in Figure 4 has 847 configurations. Using the genetic algorithm, the second-order Bézier curve has 1,663 possible inverse configurations, while the third-order Bézier curve has 1,494. There are 1,389 inverse configurations of the fourth-order Bézier curve. It is obvious that the new motion paths are Bézier curves, resulting in a greater total number of available inverse configurations than straight-line motion.

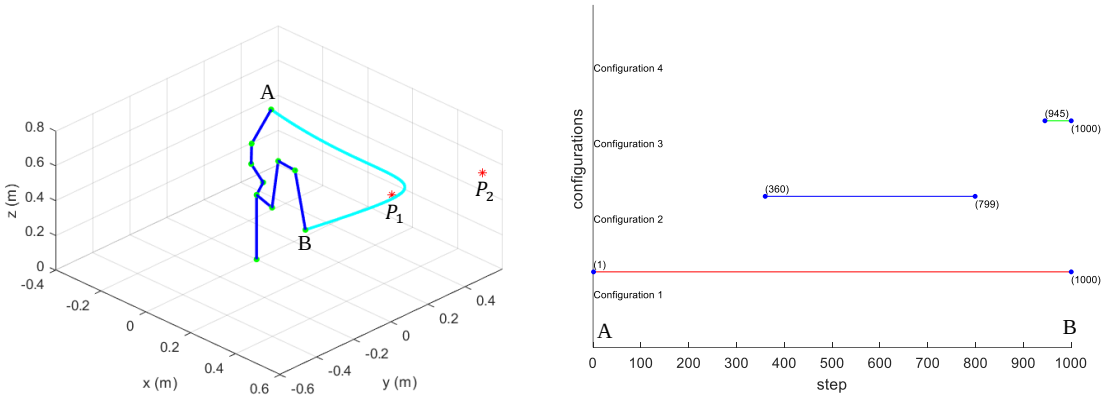
This indicates that, in addition to providing the movement of the end-effector between points A and B and in the opposite direction (as described in configuration 1), new inverse kinematic configurations arise (configurations 2 and 4). This makes it easier to select inverse kinematic configurations during the manipulator's control programming process.



a) The end-effector's motion path

b) Response of inverse kinematic configurations

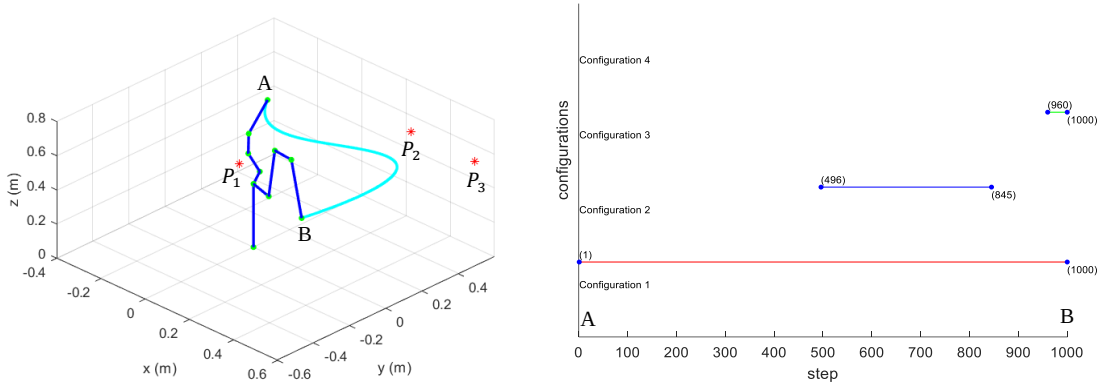
Figure 6 End-effector motion path (a) and inverse kinematic configurations (b) with second-order Bézier curve



a) The end-effector moving path

b) Response of inverse kinematic configurations

Figure 7 End-effector motion path (a) and inverse kinematic configurations (b) with third-order Bézier curve



a) The end-effector moving path

b) Response of inverse kinematic configurations

Figure 8 End-effector motion path (a) and inverse kinematic configurations (b) with fourth-order Bézier curve

As a result, the genetic algorithm efficiently generates free impact motion paths from A to B using second- to fourth-order Bézier curves. These paths adhere to joint angle constraints and operate in inverse kinematic configurations. Higher-order Bézier curves have more control points than second-order Bézier

curves, resulting in more inflection points in the workspace. This increases control over the end-effector's movement from point A to point B, and vice versa. This approach is useful for creating motion paths for manipulators with six degrees of freedom.

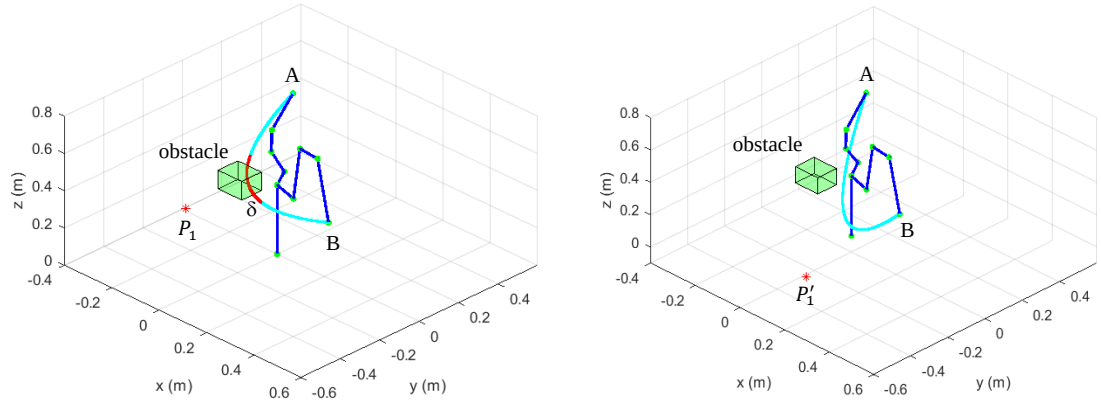
4.3 Obstacle avoidance

To test the algorithm's adaptability in generating the end-effector's motion path in uncertain environments, a rectangular obstacle will be randomly placed along the current motion path. As shown in Table 5, these obstacles define distinct regions within the Bézier curves. Assuming this, the obstacle's location could be determined using a variety of position measurement techniques, such as cameras, space laser scanning, and so on. In such cases, the genetic algorithm adjusts the SAC_i parameter inside the optimization function (Table 1) to improve the current path. Figures 9a, 11a, and 13a show the obstacle and the available travel path, with the restricted segment highlighted in red. After running the genetic algorithm with the parameters from Table 5, new moving paths are reconstructed using the new control points P_j' shown in Figures 9b, 11b, and 13b.

Using the SAC_i parameter adjustment algorithm described in Table 1 with λ = -1 and δ = 0.05 (m), the simulation results determine the new control position P_j' for the Bézier curves shown in Table 5. Figures 10a, 12a, and 14a illustrate the fitness values, while Figures 10b, 12b, and 14b show the response of the inverse kinematic configurations after adjusting the SAC_i parameter for avoiding obstacles.

Table 5 Optimal result using a genetic algorithm based on Bézier curves in case obstacle avoidance

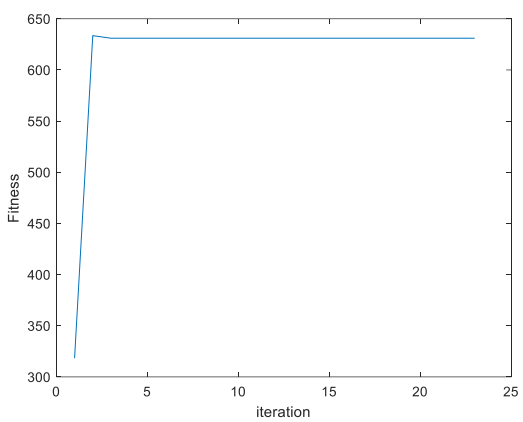
Bézier curve order	**Second order**	**Third order**	**Fourth order**
Obstacle region limits (m) $(x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max})$	(0, 0.1, -0.3, -0.2, 0.5, 0.6)	(0.2, 0.3, 0.3, 0.4, 0.5, 0.6)	(-0.1, 0, 0.3, 0.4, 0.4, 0.5)
Max fitness value	633.5	535.5	663.5
Best iteration cycle	2nd	25th	47th
Control point $P_1' = (x_1', y_1', z_1')$	(0.0435, -0.2958, -0.0544)	(-0.2650, 0.0424, 0.0436)	(-0.2862, 0.0347, 0.0660)
Control point $P_2' = (x_2', y_2', z_2')$		(0.3937, 0.4334, 0.5517)	(0.3929, 0.4779, 0.5517)
Control point $P_3' = (x_3', y_3', z_3')$			(0.5517, 0.5517, 0.5517)
Total available inverse configuration	1266	1070	1338



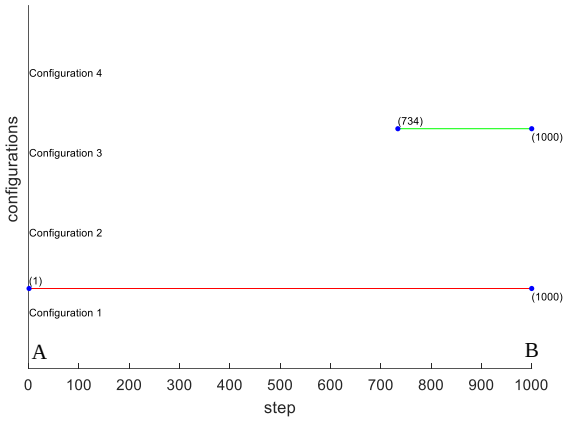
a) Obstacles and impossible paths of movement (red-colored) with a safe distance (δ)

b) Obstacle avoidance path with new control point P_1'

Figure 9 End-effector motion path: before (a) and after (b) obstacle avoidance with second-order Bézier curve

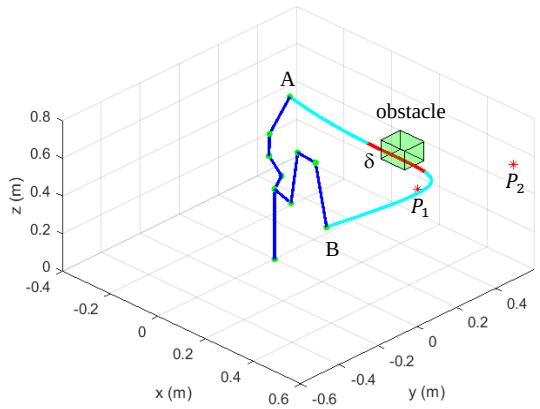


a) Fitness value in case obstacle avoidance

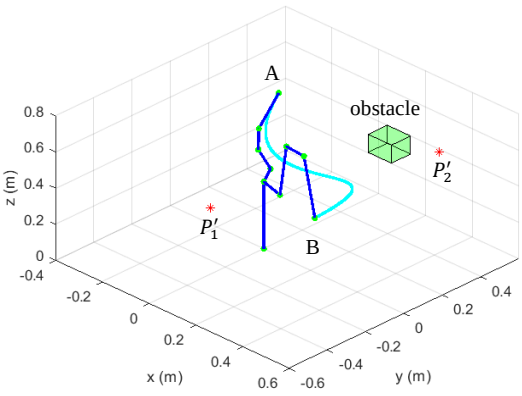


b) Response of inverse kinematic configurations in case obstacle avoidance

Figure 10 Fitness value (a) and available configuration (b) using a second-order Bézier curve for obstacle avoidance

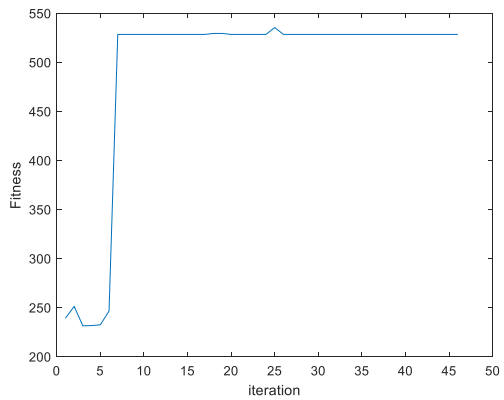


a) Obstacles and impossible paths of movement (red-colored) with a safe distance (δ)

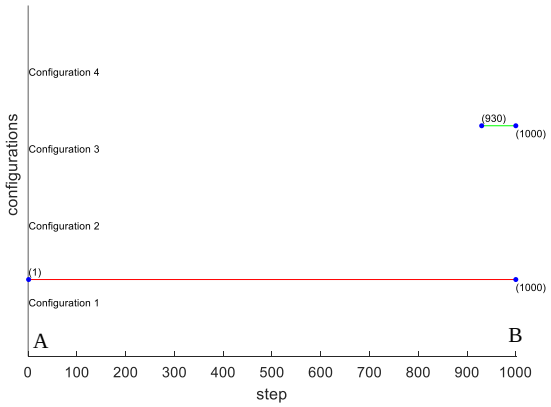


b) Obstacle avoidance path with new control points P_1' and P_2'

Figure 11 End-effector motion path: before (a) and after (b) obstacle avoidance with third-order Bézier curve

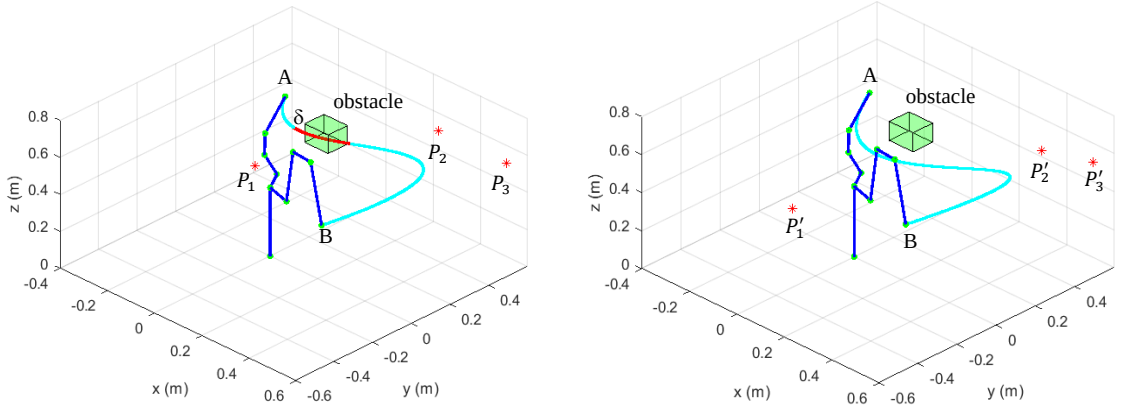


a) Fitness value in case obstacle avoidance



b) Response of inverse kinematic configurations in case obstacle avoidance

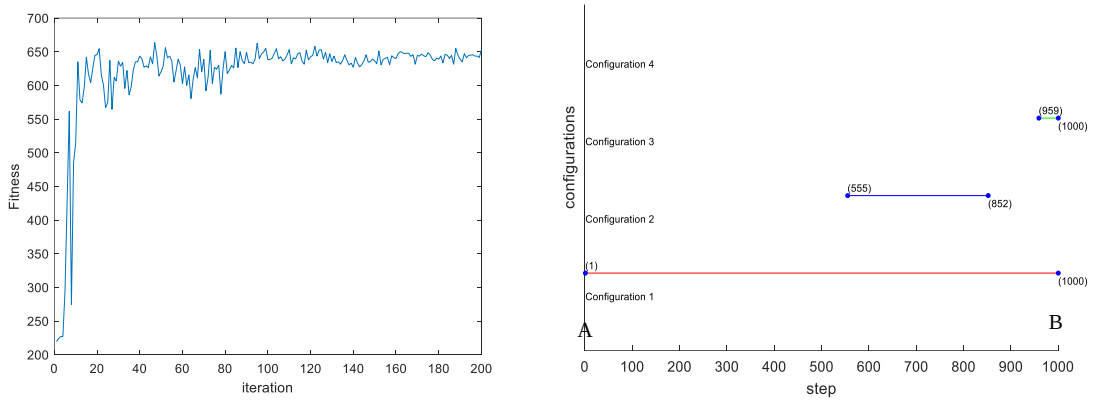
Figure 12 Fitness value (a) and available configuration (b) using a third-order Bézier curve for obstacle avoidance



a) Obstacles and impossible paths of movement (red-colored) with a safe distance (δ)

b) Obstacle avoidance path with new control points P_1', P_2' and P_3'

Figure 13 End-effector motion path: before (a) and after (b) obstacle avoidance with fourth-order Bézier curve



a) Fitness value in case obstacle avoidance

b) Response of inverse kinematic configurations in case obstacle avoidance

Figure 14 Fitness value (a) and available configuration (b) using a fourth-order Bézier curve for obstacle avoidance

Figures 9b, 11b, and 13b show the algorithm's ability to dynamically adjust motion paths. These updated paths successfully maneuver around obstacles while ensuring a safe distance and smooth curvature for end-effector motion. Despite the reduced number of available inverse configurations, the algorithm intelligently selects new control points (P_j') to generate alternative motion paths. Even when there are obstacles, the paths remain within joint angle limits and the robot's operational range. Although the revised paths may have increased length and curvature to avoid obstacles, the manipulator smoothly guides the end-effector from point A to point B without the need for inverse configuration changes, ensuring a certain level of safety when near obstacles.

Figures 10a, 12a, and 14a show the optimal results obtained by running the genetic algorithm with the parameters listed in Table 5. The objective function (7) computes the new control points P_j', which have different positions from the current control points P_j. The genetic algorithm performs best on the quadratic (second-order) Bézier curve at the second iteration cycle, with a maximum fitness value of 633.5, where the control point is P_1' (0.0435, -0.2958, -0.0544). The genetic algorithm performs best on the third-order Bézier curve at the 25th iteration cycle, with a maximum fitness value of 535.5, where the control points are P_1' (-0.2650, 0.0424, 0.0436) and P_2' (0.3937, 0.4334, 0.5517). The genetic algorithm improves on the fourth-order Bézier curve in the 47th iteration cycle, with a maximum fitness value of

663.5. In this case, the new location-specific control points are P_1' (-0.2862, 0.0347, 0.0660), P_2' (-0.3929, 0.4779, 0.5517), and P_3' (0.5517, 0.5517, 0.5517).

When evaluating the available inverse kinematic configurations in Figures 10b, 12b, and 14b, configuration 1 remains the most complete configuration for controlling the end-effector move from point A to point B. In comparison to the results in Figures 6b, 7b, and 8b, configuration 4 is no longer possible with the new motion paths. Configurations 2 and 3 achieve fewer points than the case without obstacles. The number of controller configurations decreases from 1,663 to 1266 with quadratic Bézier curves, 1,494 to 1070, and 1389 to 1338 with third- and fourth-order Bézier curves. This reduces the possibility of switching inverse kinematic configurations during control programming. However, the most important aspect is that the proposed algorithm can effortlessly generate new motion paths based on Bézier curves and self-adjust to successfully avoid obstacles.

4.4 Smooth motion analysis

To evaluate the smoothness of movement along Bézier curve trajectories, we look at changes in the robot's joint angle values. Because the last three joint angles $(\theta_4, \theta_5, \theta_6)$ only affect the direction angle of the end-effector, the first three joint angles $(\theta_1, \theta_2, \theta_3)$ will be considered. We will examine the graphs of angle and angular difference for $\theta_1, \theta_2, \theta_3$ when considering the motion paths based on Bézier curves constructed in both cases with and without obstacles, as demonstrated in Sections 4.2 and 4.3. Figures 15 show the results for angle and angular difference of $\theta_1, \theta_2, \theta_3$.

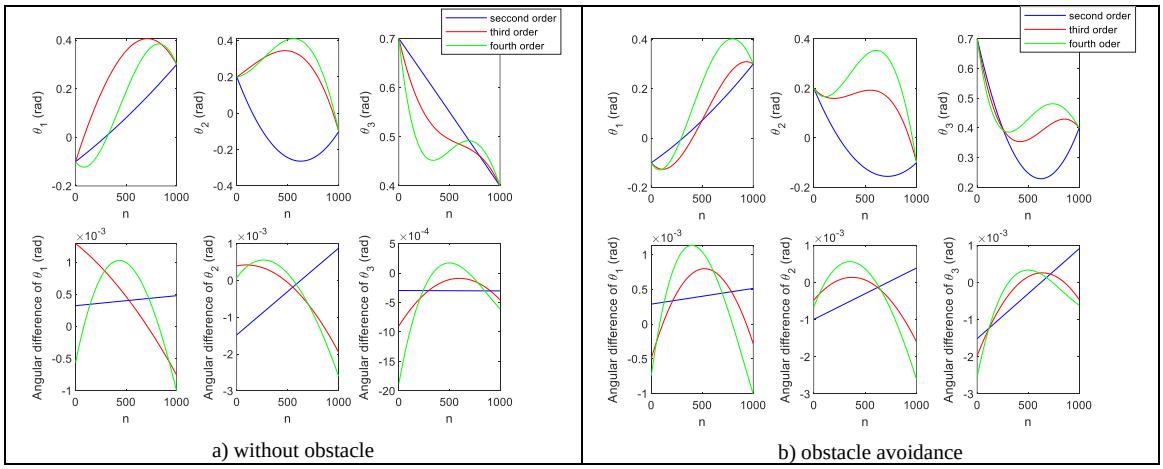


Figure 15 Joint angles and angular differences with the second-, third-, and fourth-order Bézier curves in the case of no obstacle (a) vs. obstacle (b)

Figures 15 demonstrate the angles and angular differences of the first three joints $(\theta_1, \theta_2, \theta_3)$ relative to the generated motion paths in Sections 4.2 and 4.3. The joint angular difference is measured as the end-effector transitions between consecutive positions along the motion paths. These charts are based on second-, third-, and fourth-order Bézier curves in situations with and without obstacles.

When all factors are considered, in the presence of an obstacle, the angles $(\theta_1, \theta_2, \theta_3)$ have greater angular displacement than in the absence of an obstacle. The second-order Bézier curve (blue lines) has the lowest absolute value of amplitude and the most linear path of the angular difference, while the fourth-order Bézier curve (green lines) has the highest values. Figure 15b shows the maximum absolute value of 0.0011 (rad) for the joint angular difference (θ_1 of the fourth-order Bézier curve).

This demonstrates that the joint angular difference is quite small when the manipulator is operating along the moving paths. In this case, the end-effector moves smoothly with no jerks. It allows the manipulator's control to be easier and smoother. As a result, the motion paths generated by genetic algorithms using Bézier curves are smooth, have low angular disparity, and can navigate around obstacles while remaining at a safe distance.

The goal of this paper is to find a new path that follows the inverse kinematic configuration when moving the end-effector from point A to point B,

rather than a straight-line path of motion. These motion paths have smooth movements that maintain joint angle constraints in the robot's workspace. According to the survey findings, the genetic algorithm, when combined with Bézier curves, can automatically generate a smooth and gentle motion path from one point to another in the *Oxyz* coordinates. This algorithm can self-adjust parameters in the optimization function to rebuild the moving path to avoid obstacles. Using higher-order Bézier curves makes the motion paths for the 6-DOF manipulator more flexible by providing more control points. This algorithm is valuable for smart manipulators that can plan and adjust their motion paths in uncertain environments.

5. Conclusion

This article describes how to automatically generate point-to-point motion paths for a 6-DOF robot using genetic algorithms combined with Bézier curves. The order of the Bézier curve determines the number of control points, thereby influencing the flexibility of the motion paths. The suggested genetic optimal function considers the total number of available inverse kinematic configurations along the moving paths. The survey results show a straight-line motion path may occur due to the robot's limitations. The combination of a genetic algorithm and Bézier curves generates motion paths with smooth shifts, minimal changes in joint angles, and no sudden jerks within the robot's operational area. The proposed algorithm can also assist the robot in adjusting its motion paths when encountering obstacles at a safe distance. This algorithm is suitable for 6-DOF robots and is able to manage unexpected situations in an unknown environment. Finally, future research for the 6-DOF robot could investigate using a single genetic algorithm to generate optimal motion paths that avoid obstacles. This could include comparing results from previous studies or using optimization search algorithms to determine the best number of control points for the Bézier curve, as described in this article. Subsequent studies will present the related findings.

6. Acknowledgements

This research is funded by University of Science, VNU-HCM under grant number T2024-57.

7. References

Abbas, M., Jamal, E., & Ali, J. M. (2011). Bezier Curve Interpolation Constrained by a Line. *Applied Mathematical Sciences*, *5*(37), 1817 – 1832.

AKB Machinery. (n.d.), *AKB-IRV1 6-DOF robot*. Retrieved February 2, 2024, from https://akbmachinery.com/san-pham/canh-tay-robot-6-bac-tu-do-akb-irv1/

AL-Qassara, A. A., & Abdulnabib, A. N. (2018). Optimal Path Planning Obstacle Avoidance of Robot Manipulator System using Bézier Curve. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, *40*(01), 6-17.

Baressi Šegota, S., Anđelić, N., Lorencin, I., Saga, M., & Car, Z. (2020). Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. *International Journal of Advanced Robotic Systems*, *17*(2). https://doi.org/10.1177/1729881420908076

Bézier curve. (2014, April 4). *Wikipedia*. Retrieved February 2, 2024, from https://en.wikipedia.org/wiki/B%C3%A9zier_curve

Denavit, J., & Hartenberg, R. (1955). A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics*, *22*(2), 215–221. https://doi.org/10.1115/1.4011045

Elhoseny, M., Tharwat, A., & Hassanien, A. E. (2017). Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm. *Journal of Computational Science*, *25*, 339–350. https://doi.org/10.1016/J.JOCS.2017.08.004

Farin, G. (2006). Class A Bézier curves. *Computer Aided Geometric Design*, *23*(7), 573-581. https://doi.org/10.1016/j.cagd.2006.03.004

Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2015). Path Planning and Trajectory Planning Algorithms: A General Overview. In: Carbone, G., Gomez-Bravo, F. (eds), *Motion and Operation Planning of Robotic Systems* (pp. 3–27). Mechanisms and Machine Science, 29. Cham: Springer, https://doi.org/10.1007/978-3-319-14705-5_1

Gracia, L., Andres, J. & Tornero, J. (2009). Trajectory Tracking with a 6R Serial Industrial Robot with Ordinary and Non-ordinary Singularities. *International Journal of Control, Automation, and Systems*, *7*, 85-96. https://doi.org/10.1007/s12555-009-0111-1

Hartenberg, R., & Denavit, J. (1964). *Kinematic Synthesis of Linkages* (1st ed.). New York, US: McGraw-Hill McGraw Hill.

Hou, J., Du, J., & Chen, Z. (2023). Time-Optimal Trajectory Planning for the Manipulator Based on Improved Non-Dominated Sorting Genetic Algorithm II. *Applied Sciences*, *13*(11), Article 6757. https://doi.org/10.3390/app13116757

Hughes, J. F., Dam, A. V., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., & Akeley, K. (2014). *Computer Graphics - Principles and Practice*, (3rd ed). Boston. US: Addison-Wesley.

Juříček, M., Parák, R., & Kůdela, J. (2023). Evolutionary Computation Techniques for Path Planning Problems in Industrial Robotics: A State-of-the-Art Review. *Computation*, *11*(12), Article 245. https://doi.org/10.3390/computation11120245

Kazem, B. I., Mahdi, A. I., & Oudah, A. T. (2008). Motion Planning for a Robot Arm by Using Genetic Algorithm. *Jordan Journal of Mechanical and Industrial Engineering*, *2*(3), 131–136.

Lee, C. S. G., & Ziegler, M. (1984). A Geometric Approach in Solving the Inverse Kinematics of Puma Robots. *IEEE Transactions on Aerospace and Electronic Systems*, *AES-20*(6), 695–706. https://doi.org/10.1109/TAES.1984.310452

Ma, J., Liu, Y., Zang, S., & Wang, L. (2020). Robot Path Planning Based on Genetic Algorithm Fused with Continuous Bezier Optimization. *Computational Intelligence and Neuroscience*. Article 9813040. https://doi.org/10.1155/2020/9813040

Masajedi, P., Heidari Shirazi, K., & Ghanbarzadeh, A. (2013). Verification of bee algorithm based path planning for a 6DOF manipulator using ADAMS. *Journal of Vibroengineering*, *15*(2), 805–815

Meng, Y., Sun, Y., & Chang, W. S. (2021). Optimal trajectory planning of complicated robotic timber joints based on particle swarm optimization and an adaptive genetic algorithm. *Construction Robotics*, *5*, 131–146. https://doi.org/10.1007/s41693-021-00057-w

Mousa, M. A. A., Elgohr, A., & Khater, H. (2023). Path Planning for a 6 DoF Robotic Arm Based on Whale Optimization Algorithm and Genetic Algorithm. *Journal of Engineering Research*, *7*(5), 160–168. https://doi.org/10.21608/erjeng.2023.237586.1256

Nguyen, X.-V., & Nguyen, N.-L. (2024). Automated Inverse Kinematics Configuration Selection for Path Planning of a 6-DOF Robot. *Journal of Current Science and Technology*, *14*(1), Article 10. https://doi.org/10.59796/jcst.V14N1.2024.10

Perumaal, S. S., & Jawahar, N. (2012). Automated Trajectory Planner of Industrial Robot for Pick-and-Place Task. *International Journal of Advanced Robotic Systems*, *10*(2), Article 53940. https://doi.org/10.5772/53940

Piotrowski, N., & Barylski, A. (2014). Modelling a 6-DOF Manipulator Using Matlab Software. *Archives of Mechanical Technology and Automation*, *34*(3), 45–55.

Spong, M. W., Hutchinson, S., &Vidyasagar, M., (2004). *Robot Dynamic and Control*. (2nd ed.), Academia.edu Publishing.

Khawlaor, W., & Nakphan, K. (2021). Forward Kinematic Analysis of Denso vs-6577 Robot Manipulator. *Journal of Energy and Environment Technology of Graduate School Siam Technology College*, *8*(1), 60–68. https://ph01.tci-thaijo.org/index.php/JEET/article/view/244118

Yang, C., Ma, H., & Fu, M. (2016). *Advanced technologies in modern robotic applications*. Springer Singapore.

Yoshida, N., Fukuda, R., & Saito, T. (2010). Interactive Generation of 3D Class A Bézier Curve Segments. *Computer-Aided Design and Applications*, *7*(2), 163–172. https://doi.org/10.3722/cadaps.2010.163-172

Zhang, J., Meng, Q., Feng, X., & Shen, H. (2018). A 6-DOF robot-time optimal trajectory planning based on an improved genetic algorithm. *Robotics and Biomimetics*, *5*(1). https://doi.org/10.1186/s40638-018-0085-7