ABUAD Journal of Engineering Research and Development (AJERD)
ISSN (online): 2645-2685; ISSN (print): 2756-6811

Volume 5, Issue 1, 110-123

Joint Space Robot Arm Trajectory Planning Using Septic Function

Abubakar Ademola RAJI[1*], Olumuyiwa Sunday ASAOLU[2], Theddeus Tochukwu AKANO[3]

[1,2,3]*Department of Systems Engineering, University of Lagos*
[1]aaraji@unilag.edu.ng, [2]oasaolu@unilag.edu.ng, [3]takano@unilag.edu.ng

Corresponding Author: aaraji@unilag.edu.ng, +2348066785911

Date Submitted: 07/03/2022
Date Accepted: 05/06/2022
Date Published: 30/06/2022

Abstract: *The research examines the use of 7th order polynomials to model the trajectory of an industrial manipulator. Using a joint space scheme, a pick and place trajectory planning is conducted in this paper. Using only the Septic Polynomial, this research proposes a unique method for obtaining zero starting and endpoint velocities and accelerations with decreased jerk. This is to suppress errors in trajectory tracking and unwanted resonance from being induced on the mechanical structure. Whenever the acceleration of a robot arm changes, the corresponding jerk function (the time derivative of acceleration) will exhibit numerous spikes. Therefore, any approach that permits discontinuous acceleration functions to be employed in robot joint-space trajectory development is inappropriate and should be avoided. As a result of their discontinuity in acceleration and hence endless jerk spikes, the linear velocity with polynomial blends and the third-order polynomial becomes problematic when adopted for industrial operations. The strategy employed in this research involves the addition of two "dummy" points in between the start and end points; whilst simultaneously solving the boundary and initial conditions at those points. Utilising the MATLAB robotics toolbox, the trajectory was evaluated using a PUMA 560 serial manipulator. Based on this model-setup, the septic trajectory is established to be quicker than the Linear Segment with Polynomial Blend (LSPB) variation. Also, infinite values and discontinuities of acceleration at the end points were addressed appropriately by adopting the septic function.*

Keywords: *Discontinuity, septic polynomial, trajectory generation, jerk, PUMA 560.*

1. INTRODUCTION

In robotics, a crucial need is to have algorithms that convert high-level human task descriptions into low-level commands which robots can correctly interpret. The planning of robotic trajectories is crucial in the safety and precision of industrial manipulators. The software development of industrial manipulators has two phases involved; trajectory planning and control. The former describes the temporal history of the manipulator's parameters such as the position, velocity, acceleration, and jerk, while the latter converts these measured values into input control signals that are sent into actuators during task execution to mitigate specific deviations.

As a vital facet of a robot control system, considerable attention has been devoted to trajectory planning by researchers. One of the most critical aspects in robotics is trajectory planning. As a result, selecting path with optimal execution time, energy consumption, jerk, gripping forces for a robot is a more crucial task than finding an obstacle-free path [1]. The fundamental task involves gripping, movement and discharge of an object relative to a certain location, thereby providing a restriction to the robot configuration throughout the task execution.

Due to their greater flexibility and simplicity of location in the context of facility design, industrial robots are viable substitutes for specialised machine tools to a considerable extent. Industrial robots are regarded as a pillar of competitive production, which attempts to combine high productivity, quality, and flexibility at the lowest possible cost [2]. In terms of cost, an estimated savings of 20-50 percent compared to typical Computer Numerical Control (CNC) machines has also been widely documented [3].

Inherent in robot manipulators is their great versatility and flexibility ensuring their suitability to different tasks. The proliferation of robotic manipulators has also been attributed to its effective response to dynamic changes in consumer behaviours and adaptation to global competitiveness [2]. Researchers investigate, examine, and suggest numerous optimal techniques and robot settings that offer enhanced geometric accuracy and product quality to assure the general performance of the robot manipulator across the whole production cycle. The development of trajectory planning algorithms for industrial manipulator is a pivotal area in robotics engineering.

Trajectory planning generally involves the calculation of certain required motion sequences for the manipulator's actuation system. Planning smooth trajectories in a timely, energy efficient and smooth manner is thus crucial in most robotic applications in industrial environments. Natural vibrations caused by the actuators on the robot structure should always be kept to a minimum. This also includes the selection of appropriate motion sequence, since deviations from the

ISSN (online): 2645-2685; ISSN (print): 2756-6811

planned trajectory may cause vibrations in the manipulator as a result of the generation of discontinuous applied force and the non-rigid effects of the inherent in the manipulator structure. As a result, it may be advantageous to incorporate trajectories with variable smoothness[4].

2. LITERATURE REVIEW

As regards industrial manipulators, quite a number of trajectory generation techniques to achieve some objective functions have been proposed in literature [5]. Each of these techniques possesses unique characteristics that makes them suited to some particular class of tasks. The issue of trajectory planning has gained importance because, once the displacement and time frame are specified, the selection of a manipulator's motion strategy from start to finish has significant implications for actuator sizing, torque, resonance on the structure, and the specified movement's trajectory following capabilities. Consequently, it is necessary to properly investigate the multitude of possible point-to-point trajectories for a particular system. As such, for specified boundary conditions like the start and end positions, velocities and accelerations, the trajectory profile has a significant influence on the maximum values at each specified via-point [5]. Figure 1 succinctly describes different categories and classification of trajectory types [6].

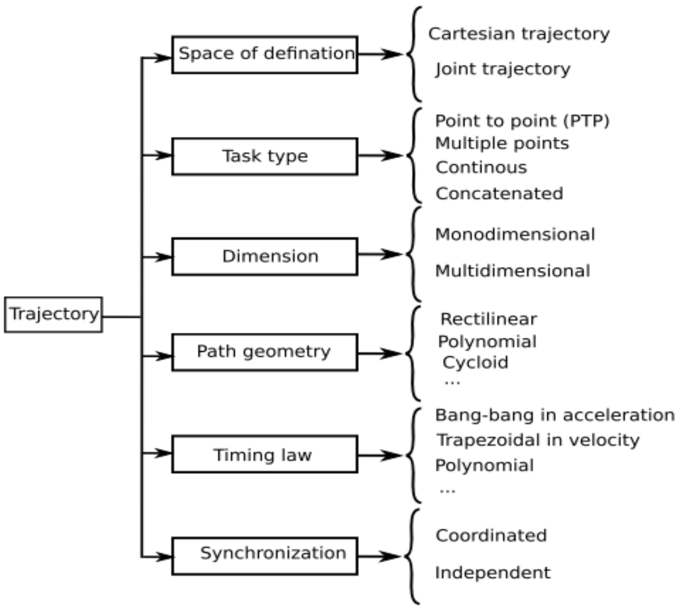


Figure 1: Trajectory Types [6]

2.1 Types of Trajectory Functions

For most robotics research, the most common basic trajectories functions utilised are the polynomial and trigonometric based functions. The mathematical formulations are briefly described in the next sections with relevant works adopting such techniques briefly mentioned.

2.2 Polynomial Trajectories

Fundamentally, robot movement is specified by designating a start and end time instant t_0 *and* t_f, together with constraints on position, velocity and acceleration at t_0 *and* t_f. The task then becomes one of mathematics, which is to find a function:

$$q = q(t), t \in [t_0, t_f] \tag{1}$$

that satisfies all of the requirements that have been specified. Using a polynomial function, this resolution may be gotten by:

$$q(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n \tag{2}$$

in which the $n + 1$ coefficients a_i are calculated in order to satisfy the end-point constraints. The polynomial's degree n is determined by the set of requirements that must be met as well as the requisite "smoothness" of the resultant motion (in this case, a smooth motion). In practice, odd-degree polynomial is employed because of the even order resulting from the set of boundary conditions.

Generally, for linear functions of polynomial of degree one, at $t = t_0, t_f$, The velocity is really not continuous at these points, so the acceleration is infinite. Acceleration is zero in the middle of the trajectory, but it rises sharply at the ends. So, in industrial practice, trajectories of this type are not used.

2.3 Cubic Polynomial Trajectory

When both the position and velocity values at t_0 and t_f (q_0, q_f, and v_0, v_f respectively) are specified, four requirements must be met. As a result, a polynomial of third degree must be utilised.

$$q(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3, \qquad t_0 \le t \le t_f \tag{3}$$

where the four parameters a_0, a_1, a_2, a_3 are derived from the provided conditions

$$\begin{aligned} a_0 &= q_0 \\ a_1 &= v_0 \\ a_2 &= \frac{3h - (2v_0 + v_f)T}{T^2} \\ a_3 &= \frac{-2h+(v_0 + v_f)T}{T^3} \end{aligned} \tag{4}$$

where $T = t_f - t_0$ is the time duration.
$h = q_f - q_0$ is the displacement

Using the approach outlined above, it is quite straightforward to calculate a trajectory with constant velocity through a sequence of n locations. Overall motion is split into $n - 1$ segments, each of which is a separate motion. These segments link the points q_k and q_{k+1} at time t_k, t_{k+1} and have the starting and end velocities, respectively, of v_k, v_{k+1}. Then, equations $a_{0k}, a_{1k}, a_{2k}, a_{3k}$. as seen in equation 4 are used for each of these segments to define the $4(n - 1)$ parameters. Because they provide continuous velocity and acceleration profiles along the desired trajectory, cubic splines are frequently used. Furthermore, the parameters are simple to compute, and huge perturbations of the positional variable and its time derivatives are avoided [7].

Adoption of cubic splines can be seen in the work of Chettibi [8]. Joint trajectories were represented in the research using clipped cubic spline functions with nodes spread evenly throughout the time scale. This approach was justified by the well-known qualities of cubic spline functions, namely their second-order continuity and their lower order, which significantly inhibits oscillations and enables a swift computation of extremum values between two adjacent nodes. The difficulty becomes the transfer time and the location of cubic spline nodes when modelling joint motions.

Another work [9] examined the approximation of trajectories using third, fourth, and fifth-degree polynomial functions. Additionally, comparisons and tests using an industrial robot arm were shown. Increases in the degree of the polynomial have no effect on the approximation, and the cubic approximation takes less time to compute than other high-degree functions. Additionally, they observed the distinction between time-bounded and jerk-bound systems. The work concluded that when the acceleration is low (as in the case of large-radius curves, such as the line and circle), the time-bounded third order polynomial trajectory is preferable, and vice versa. As such for robot manipulator applications, the third-degree polynomial trajectories provide good performance and simplicity.

Bharadwaj et al. [10] employed the cubic B-spline to create a smooth trajectory using the minimax approach to minimise the jerk value. Additionally, Li et al. [11] employed B-spline curves to generate robot trajectories. They stated that B-splines are preferable to Bézier curves because they allow for local change without altering the overall trajectory and because the degree of a polynomial does not depend on a set of control points. Additionally, adopting this method makes possible numerous and robust geometric operation methods, such as spline subdivision and spline merging, that enable flexible manipulation of robot trajectories.

Shi and Zeng [12] also employed a cubic B-spline in joint space to plan a route across the joint angle's data points which was derived by Cartesian space planning.

The trajectory for pick-and-place operations was generated using Cubic spline by Li et al.[13]. Cubic B-spline curves were used in this work to produce zero starting and end velocities and accelerations. Two dummy points were added to the path nodes and the linear system is maintained in a manner similar to solving the velocity or acceleration boundary conditions on their own. Cubic spline curves, according to the authors, are more computationally efficient and have less local support than higher-order B-splines.

In general, a third-degree polynomial-based trajectory across the positions $q_0 \ldots, q_n$ has continuous position and velocity profiles, but discontinuous acceleration. While this trajectory is often "smooth," acceleration discontinuities might have unwanted consequences on the links and joints and on inertial loads in particular applications. This occurs most frequently when time savings are desired and hence high values of acceleration and velocity are specified, or when the actuation system has significant design elasticities.

2.4 Quintic Polynomial Trajectory

While cubic polynomials are widely used to create smooth trajectory profiles, they do not yield a constant jerk profile and so cannot be used for high-level trajectory planning. Higher order polynomials are necessary to obtain a smoother jerk profile. Quintic polynomials are the lowest order polynomials that assure the smoothness of the jerk profile at each via-point sequence, while a quadratic polynomial defines the associated jerk profile. [14]. To generate continuous acceleration trajectories, it is important to specify appropriate beginning and terminal values for the acceleration in addition to the

ISSN (online): 2645-2685; ISSN (print): 2756-6811

position and velocity criteria. A fifth-degree polynomial must be utilised due to the six boundary conditions (position, velocity, and acceleration) [5]:

$$q(t) = q_0 + a_1(t-t_0) + a_2(t-t_0)^2 + a_3(t-t_0)^3 + a_4(t-t_0)^4 + a_5(t-t_0)^5 \qquad (5)$$

with the conditions

$$q(t_0) = q_0, q(t_f) = q_f\ q^{\cdot}(t_0) = v_0, q^{\cdot\cdot}(t_0) = \alpha_0\,,$$
$$q^{\cdot}(t_f) = v_f\ q^{\cdot\cdot}(t_1) = \alpha_f\,.$$

With this situation, by defining $T = t_f - t_0$, the polynomial coefficients are obtained.

$$\begin{aligned} a_0 &= q_0 \\ a_1 &= v_0 \\ a2 &= a_0/2 \\ a_3 &= \frac{1}{2T^3}\,[20h - (8v_1 + 12v_0)T - (3a_0 - a_1)T^2] \\ a_4 &= \frac{1}{2T^4}\,[-30h + (14v_1 + 16v_0)\,T + (3a_0 - 2a_1)T_2] \\ a_5 &= \frac{1}{2T^5}\,[12h - 6(v_1 + v_0)T + (a_1 - a_0)T_2]. \end{aligned} \qquad (6)$$

Song et al. [15] modelled the whole manipulator trajectory planning problem using a quintic polynomial interpolation technique, which included robotic arm kinematics and dynamics. The paper developed a trajectory planning technique for a six degree-of-freedom (DoF) manipulator that considerably improves the manipulator's trajectory tracking performance and motion stability.

Additionally, Zhao et al. [16] simulated the deflagration of an explosive ordnance disposal (EOD) robot job in joint space using the quintic polynomial interpolation approach. The trajectory planning approach of quintic polynomial interpolation guaranteed the manipulator moved smoothly and operated with great precision.

Wang et al.[17] proposed a trajectory planning strategy that hybridises cubic polynomials with quintic B-splines interpolation for path planning and trajectory generation respectively. This resulted in a decrease in execution time and energy utilisation, as well as an increase in stability performance while performing point-to-point motion operations.

Bézier curves have control points that affect the entire curve; however, a uniform B-spline curve avoids these shortcomings while maintaining the Bézier curve's continuous and smooth characteristics. Only the B-spline curve's local trajectory is altered when control points are changed; the remaining segment pathways remain unaffected [18]. Additionally, standard interpolation curves are incapable of adequately representing complicated curves. If precision is to be ensured, the order of the polynomial curve must be increased significantly, resulting in a significant computing burden.

In comparison to lower-order B-spline curves such as the cubic B-spline curve, the quintic B-spline curve can give more adjustable dynamical characteristics, but at the expense of more processing resources and lengthy computations.

To summarise their research, Zhang et al.[19] discovered that the quintic B-spline curve outperforms the cubic B-spline curve in terms of acceleration performance, resulting in a reduction in the system's angular acceleration error and an improvement in system resilience.

2.5 Trigonometric Trajectories

These trajectories present non-null continuous derivatives for any order of derivation in the interval (t_0, t_1). However, these derivatives may be discontinuous in t_0 and t_1. Generally, an $m-th$ order trigonometric spline function $y(t)$ with a total of $2m$ constraints in each of the n closed arcs $[t_{i-1}, t_i](i = 1, \ldots, n)$ *is defined as* [7]:

$$y(t) = y_i(t) \quad t \in [t_{i-1}, t_i] \qquad (2.12)$$

where $y(t)$ *is given by*

$$y_i(t) = a_{i,0} + \sum_{k=1}^{m-1}\left(a_{i,k}\cos kt + b_{i,k}\sin kt\right) + a_{i,m}\sin m(t-\gamma_i) \qquad (7)$$

$$\gamma_i = \sum_{j=0}^{2m-1}\frac{\tau_{ij}}{2m} \qquad (8)$$

Note that τ_{ij} are the values of t where $y_i\ (t)$ has a constraint applied.

One of the primary benefits ascribed to trigonometric splines is the avoidance of large overshoots, despite the fact that the function's continuity can be imposed up to a high order. For instance, adopting a fourth-order trigonometric spline ensures the continuity of the jerk function and greatly reduces overshoots in comparison to quartic algebraic splines that meet the same criteria. In general, the position function between two knots of an m-order spline may be found by setting the values of the derivatives up to the m-1 order at the knots. A key theoretical benefit of trigonometric splines over algebraic splines is the ability to fix these values (based on the motion job) without significantly increasing the trajectory's overshoots. The use of trigonometric splines, initially described by Schoenberg [20] has also been advocated by Simon and Isik [21] for trajectory planning of robot manipulators. They emphasised the possibility of generating joint trajectories with constant velocity, acceleration, and jerk, as well as minimum overshoot. Additionally, the computing cost is minimal, and the spline parameters may be selected using an optimisation process to minimise an objective function. In any event, certain things remain unclear and warrant more examination. For instance, the effect of the trigonometric splines' order on

ISSN (online): 2645-2685; ISSN (print): 2756-6811

the total outcome must be analysed. Indeed, Simon and Isik's conclusions analyse only fourth order splines, implying that the jerk function's continuity is imposed. Chiddarwar and Babu [22] also used the trigonometric spline coefficients as optimisation variables in their study, where the KUKA-Kr robot's trajectory was simulated using trigonometric spline coefficients. They contended that interpolation through trigonometric splines has a sound neighbourhood property, i.e., that if the value of a node in the input sequence is modified, only the two splines that share that node as a common point should be recalculated, not the entire trajectory. Also, trigonometric splines maintain the jerk's continuity. However, in practice, the jerk may be discontinuous, even though its limiting typically results in fewer tracking errors and partially prevents resonant excitation. Additionally, the suggested techniques include the assumption that the spline intervals are same, which is a significant assumption in real industrial automated cells [7].

2.6 Combination of Elementary Trajectories

Often, an appropriate motion profile may be derived by combining the functions that characterise the constituent trajectories. As a result, in addition to a continuous function with continuous derivatives up to a specified order, additional features such as minimum values for the maximum acceleration or jerk may be imposed on the trajectory. This requires the creation of trajectories by incorporating the functions in the most effective sequence.

1) *Linear Trajectory with Parabolic Blends*: Combining linear trajectories with parabolic blends produces the typical trapezoidal velocity profiles characterised with a positive displacement, $q_1 > q_o$. The acceleration in the first section is positive and constant, meaning that the velocity is a linear function of time and the position is a parabolic curve. In the second segment, acceleration is zero, velocity is constant, and position is a linear function of time. The last segment has a continuous negative acceleration, a linear decrease in velocity, and a degree two polynomial function for position. For these trajectories, the acceleration phase's duration T_a is often considered to be equal to the deceleration phase's duration T_d [7]. The general case where $t_0 \neq 0$, the trajectory (in position) can be formulated by equation 9 as described in [5]:

$$q(t) = \begin{cases} q_o + \frac{v_v}{2T_a}(t - t_0)^2, & t_0 \leq t \leq t_0 + T_a \\ q_o + v_v\left(t - t_0 - \frac{T_a}{2}\right), & t_0 + T_a \leq t \leq t_1 - T_a \\ q_o - \frac{v_v}{2T_a}(t_1 - t)^2, & t_1 - T_a \leq t \leq t_1 \end{cases} \quad (9)$$

where:

t_0 *and* t_1 are the initial and final times

$T_a =$ the acceleration phase duration.

$T_d =$ is the deceleration phase.

v_v=desired velocity at the end of the acceleration phase.

q_0 *and* q_1 are initial and final joint displacements.

While parabolic blends are capable of controlling limitless acceleration and deceleration at the trajectory's start and end points, they are unable to regulate the instantaneous velocity and acceleration in the trajectory's midpoint [23]. Alshahrani et al. [24] employed a Stanford-type spherical manipulator for trajectory planning in their work. Cycloidal trajectories, Cubic segment, simple harmonic, bang-bang parabolic blend and 3-4-5 polynomial were all employed as functions. Total energy consumption is determined for each trajectory in relation to travel time. It is worth noting that the simple harmonic functions, bang-bang parabolic blends and cubic segment all begin and end with finite acceleration values. These limited acceleration levels will result in jerky motion, which is undesirable in robotics. Cycoidal motion and 3-4-5 polynomial functions, on the other hand, begin and end with null acceleration, resulting in smooth motion. If the path duration is significant, cycloid or 3-4-5 polynomial trajectories should be adopted. The cubic trajectory would be an optimal choice if energy usage is a primary consideration and travel time is limited.

2) *Linear Trajectory with Polynomial Blends:* It is feasible to generate movements with smoother profiles than trapezoidal velocity trajectories by using polynomial functions with a degree greater than two to define the blends between the linear segments. To design an nth-degree linear trajectory with polynomial blends, this approach may also be applied to trajectories via a succession of points, as well as to trajectories that cross via-points. A formal description of this technique is given also by Biagiotti and Melchiorri[5] where the points q_0, q_1 denote initial and final positions, the acceleration/deceleration period $T_a = 2T_s$ and the total displacement time $T = t_1 - t_0$.

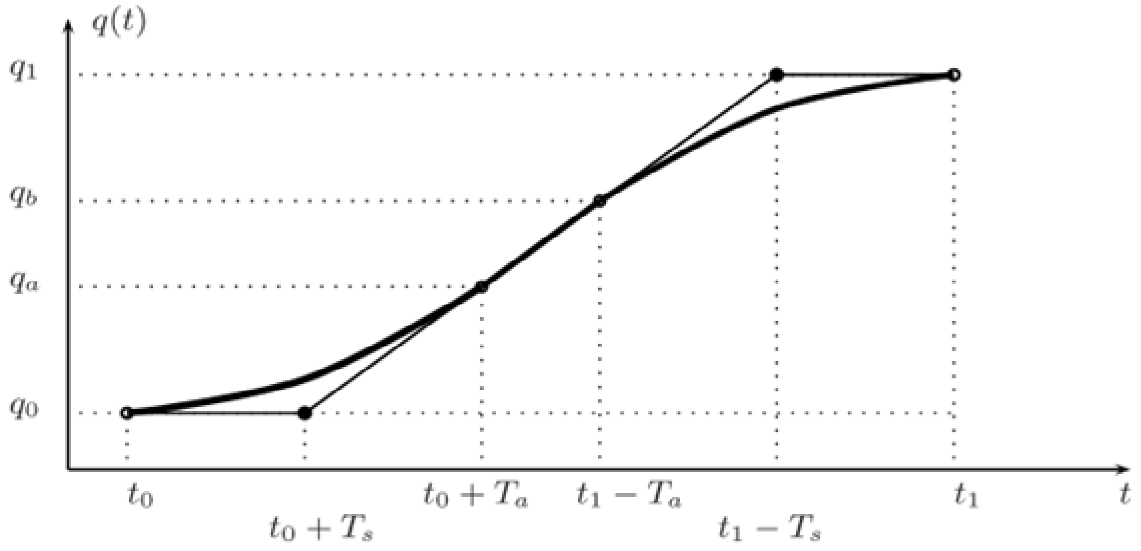


Figure 1: Linear trajectory with polynomial blends [5]

With reference to Figure 2:

- Compute the expression $q_r(t)$ of the line joining the points $(t_0 + T_s, q_0)$and $(t_1 - T_s, q_1)$.
- On this line, compute the values $q_a = q_r(t_0 + T_a)$and $q_b = q_r(t_1 - T_a)$.
- Assign the values

$\dot{q}(t_0) = \dot{q}(t_1) = 0, \quad \ddot{q}(t_0) = \ddot{q}(t_1) = 0 \; and \; \ddot{q}(t0 + Ta) = \ddot{q}(t0 - Ta) = 0$

- Compute the velocity in the segment $(t_0 + T_a) \div (t_1 - T_a) as\; v_c = (q_b - q_a)/(t_1 - t_0 - 2T_a)$.

The trajectory may then be calculated in two acceleration/deceleration phases using equations for polynomials of degree n (for example, n = 5) and in the constant velocity segment using equation 10.

$$\begin{cases} q(t) = v_c t + q_a \\ \dot{q}(t) = v_c \\ \ddot{q}(t) = 0. \end{cases} \tag{10}$$

3) *Double S Velocity Trajectory:* The velocity motion profile of a trapezoidal (or triangular) object exhibits a discontinuous acceleration. As a result, this trajectory may place undue strain and stress on the mechanical system, which may be damaging or result in undesirable vibrational effects. It is feasible to use double S velocity profiles, which are extremely prevalent in industrial usage. As a consequence, a more sophisticated motion profile must be established, such as using a profile that is continuous with linear piecewise acceleration. The resulting velocity is constructed using linear segments connected through parabolic mixes. This trajectory is also referred as the double S or bell trajectory based on the shape of the velocity profile, which consists of seven different sections with continuous jerk[5].

Because the jerk has a step profile, the stress and vibrational impacts induced by this motion profile on the kinematic structure are decreased in comparison to trapezoidal velocity trajectories with an impulsive jerk profile. It is presumptive that:

$j_{min} = -j_{max}, \; a_{min} = -a_{max}, v_{min} = -v_{max}$ such that $j_{min} \; and \; j_{max}$ denote the minimum and maximum value of the jerk respectively.

With these parameters in mind, it is desirable to construct a trajectory that hits the greatest (lowest) values for jerk, acceleration, and velocity whenever feasible, in order to reduce the overall duration T (minimum time trajectory). To keep things simple, the parameter $t_0 = 0$ is assumed. The following are the boundary conditions:

Initial and final accelerations a_0, a_1 set to zero.

Generic initial and final values of velocity v_0, v_1.

Three phases can be distinguished:

- Acceleration phase, $t \in [0, T_a]$, where the acceleration has a linear profile from the initial value (zero) to the maximum and then back to zero.
- Maximum velocity phase, $t \in [T_a, T_a + T_v]$, with a constant velocity.
- Deceleration phase, $t \in [T_a + T_v, T]$, being $T = T_a + T_v + T_d$, with profiles opposite with respect to the acceleration phase.

The trajectory is determined given the limitations on the maximum values of jerk, acceleration, and velocity, as well as the desired displacement $h = q_1 - q_0$. However, it is required to first determine whether or not a trajectory can be executed. Indeed, there are multiple instances when a trajectory cannot be calculated given the limitations. For instance, if

ISSN (online): 2645-2685; ISSN (print): 2756-6811

the required displacement h is insignificant in comparison to the difference between the beginning and final velocities v_0 *and* v_1, it may not be attainable.

In their research, Devi et al.[25] devised an approach for obtaining the ideal trajectory planning for a robot manipulator with the least amount of jerk. In this work, an industrial robot with six DoF, the PUMA-560, was used to implement the optimisation method. To accomplish the smooth trajectory, this study used a synchronised S-curve. When the same set of joint variables is employed, the execution time and jerk value achieved using the S-curve trajectory are contrasted to the ones generated utilising Cubic spline trajectory. The S-curve trajectory is preferable in terms of runtime and jerk quality.

4) *Piecewise Polynomial Trajectory:* In some circumstances, defining a trajectory as nothing more than a cluster of polynomial pieces may be beneficial. It is critical to set an appropriate number of criteria prior to computing the trajectory under these situations, including boundary conditions, via-point locations, velocity and acceleration continuity.

For instance, when an industrial robot performs pick-and-place activities, it may be advantageous to have movements with extremely smooth start and finish phases. In this scenario, a motion profile derived by connecting three polynomials can be used. $q_l(t), q_t(t), q_s(t)$ denoting lift-off, travel, set (or drop) down segments:

$$q_l(t) \Longrightarrow 4th\ degree\ polynomial$$
$$q_t(t) \Longrightarrow 3rd\ degree\ polynomial$$
$$q_s(t) \Longrightarrow 4th\ degree\ polynomial$$

This trajectory, dubbed 4-3-4, is calculated by assigning five plus four plus five making a total of fourteen parameters, necessitating the definition of 14 conditions. If t_0, t_1 are the start and end instants, t_a, t_b are the "switching" instants between polynomial segments, and q_0, q_a, q_b, q_1 are the relative position values, the requirements for computing the parameters are as follows.

$$6\ crossing\ conditions \quad \begin{cases} q_1(t_0) = q_0, \\ q_l(t_a) = q_t(t_a) = q_a \\ q_t(t_b) = q_s(t_b) = q_b \\ q_s(t_1) = q_1 \end{cases}$$

$$4\ initial\ conditions \quad \dot{q_l}(t_0) = \dot{q}_s(t_1) = 0, \quad \ddot{q}_l(t_0) = \ddot{q}_s(t_1) = 0,$$

$$3 \quad continuity\ \ conditions\ for\ velocity\ and\ acceleration \quad \begin{cases} \dot{q}_l(t_a) = \dot{q}_t(t_a), \\ \ddot{q}_l(t_a) = \ddot{q}_t(t_a) \\ \dot{q}_t(t_b) = \dot{q}_s(t_b) \\ \ddot{q}_t(t_b) = \ddot{q}_s(t_b) \end{cases}$$

2.7 Comparison of Previous Works with Present Research

As a result, the trajectory planning approach described in this research employs a single seventh-order polynomial in generating a multi-joint joint-space trajectory for smooth motion between two via-points. contains eight unknowns that must be resolved by the specification of eight constraints. The 7th order polynomial is used rather than a higher order polynomial because, while theoretically simple, this approach is wasteful computationally and may result in numerical inaccuracies for polynomial orders larger than seven.

3. METHODOLOGY

3.1 Overview of Technique

The generated trajectory is well suited to offline trajectory planning as opposed to an online scheme because many industrial pick and place processes are repetitive which justifies the adoption of the scheme. Offline planning results in a globally optimal trajectory smoothness due to the anticipation of abrupt turns and second-order continuity throughout the trajectory[26]. The trajectory planning algorithm utilises geometrical, Kino dynamic constraints to construct the joint trajectory, which is expressed as a time sequence of position, velocity, and acceleration[22]. When calculating the trajectory, it is important to take into consideration physical limitations of the actuator because they define peak value of velocity and joint acceleration. Due to these constraints, maximum values of velocity and acceleration are established in practice and used to determine time parameters, or the robot's motions between via-points.

In this research, following a kinematic inversion of the specified geometric path described in the task space, the trajectory planning was performed in the joint space of the robot. As such, the joint trajectory was generated by using septic interpolating functions, with the specified velocity and acceleration constraints not violated. As such, the trajectory could be more conveniently modified to suit the requirements while operating in the joint space[22]. This research utilises a single polynomial to generate a trajectory that traverses two way-points for a pick and place task. Core issues such as problem formulation and trajectory modelling using a 7th order polynomial, acceleration and jerk discontinuity, computational simplicity and some limitations of this technique are further explained in the next section.

ISSN (online): 2645-2685; ISSN (print): 2756-6811

3.2 Steps in Pick and Place Trajectory Generation

The trajectory generation issue is defined in this section. These issues take into consideration both the velocity boundary conditions and the acceleration limitation. By adhering to the velocity and acceleration constraints, it is anticipated that created trajectories will exhibit the following qualities.

i. The pick-and-place operation includes the starting position (take point), transfer position (lift or leave point), unload position (drop point) and target position (placement point), with specified boundary conditions and continuous conditions.
ii. MATLAB Robotics Toolbox is used to obtain the four position points of any group of pick-and-place operations at the end of the PUMA560 robot, and obtain the position of each joint through the inverse solution. The version used is the Robotics Toolbox for MATLAB (release 10.1).
iii. In addition, assuming that the joint velocity and acceleration of the starting position and the target position are all set to 0, the motion times of the three segments are 2 seconds, 4 seconds and 3 seconds respectively. The solution to the set of linear equations to solve the unknown coefficients of the trajectory are gotten.
iv. The coefficients are used to generate the trajectory function of the position, velocity and acceleration of each joint of PUMA560.
v. These trajectories are inputted into PUMA560 model to simulate the execution of the pick and place trajectory, while observing and analysing any difference between the execution results of the three-segment quintic trajectory.
vi. Plots of the position, velocity and acceleration of the trajectory are obtained, and it is compared with the three-segment quintic trajectory.

3.3 PUMA 560 Robot Model

The PUMA 560 robot was a game changer in the robotics age, and has been extensively applied in a broad variety of industries. While more powerful robots have found use in business in recent years, PUMA 560 has found a new role in teaching, in part because it is the most mathematically specified robot. Its straightforward structure makes it possible to design new controllers and test novel controlling algorithms for educational and scientific objectives. There are several manufacturers on the market today, but the robots manufactured employ controllers that are not available for study and education. It is critical in educational processes designed for students to be able to assess various parameters (position, error, torque, etc.) from control algorithms used on the controller in real time and compare them to results from other simulations and reference books. As a result, institutions and colleges worldwide have created novel control strategies and controllers for the PUMA 560 robot [27]. The PUMA 560 robot is a member of the anthropomorphic arm family. The basic arrangement resembles a two-link planar arm with an added rotation about the plane's axis. The methodologies given in this thesis have been validated and tested on the PUMA 560 robot. PUMA 560 is a six DoF robot with all joints being revolute.

For simulation purposes, Professor Peter Corke's Robotics Toolbox for MATLAB/Simulink[28] was employed. The toolbox contains a detailed description of the mathematical model of the robot PUMA 560. The solution approach involves modelling the robot as a consequence of generalised coordinates passing through a complex task space that has been discretised in cartesian coordinate system. This greatly simplifies the trajectory generation process, which trajectory between the robot's starting and end configurations, with the aid of two through locations. This model is constructed using rigid links coupled by kinematic joints. Siciliano et al. [6] outlined the forward and inverse mathematical model of the robot PUMA 560.

3.4 Trajectory Modelling using Septic Polynomial

A trajectory can be specified by providing a succession of intermediate places (via-points) in Cartesian space that the end-effector must travel through. Thus, a matched sequence of angles for each joint to assume at defined locations is constructed. To link these points sequences with a guarantee of velocity, acceleration, and jerk continuity across the trajectory, an appropriate order of splines is employed. As a result, the nonlinear effects generated by direct kinematics pose a difficulty. This complicates anticipating the execution of a planned movement in the joint space in the cartesian space [29].

Analytically, polynomials, exponential, trigonometric functions, and others may be used to characterise the basic trajectory. The septic polynomial is utilised in this thesis. The "smoothness" of the resulting motion and the number of conditions necessary determine the selected polynomial's degree. Polynomial functions have odd degrees because the set of boundary conditions is generally even, hence the degree could be 3rd, 5th, 7th, and so on. A seventh-degree polynomial is used in this case.

In the seventh-degree polynomial planning trajectory. There are just eight conditions that are known a priori which are the joint position, velocity and acceleration of the initial and final positions, and joint position of the two intermediate points. The polynomial of seventh degree contains eight unknown coefficients. It prevents the trajectory from being segmented as is the case with piecewise polynomials. This facilitates the usage of a single polynomial.

Splines are an excellent alternative if the robot must precisely pass the intermediate spots. By selecting an appropriate spline order, the continuity of velocity, acceleration, and jerk inputs across the whole trajectory is ensured.

ISSN (online): 2645-2685; ISSN (print): 2756-6811

All beginning and end values for all joints are known in this example, with the start and end angular velocity and acceleration being both zero. It is worth noting that t_0 and t_n denote the start and end times, respectively. Additionally,, t_1 *and* t_2 represent the vis-point timings. The initial and ultimate joint variables of the robot arm are denoted by q_0 *and* q_n. The angular velocity and acceleration thus represents the joint variable's first and second order derivatives.

Suppose we have a sequence $\boldsymbol{q} = [q_0, q_1, \dots, q_n]$ of via-points that the joint is required to traverse at time instant $\boldsymbol{t} = [t_0, t_1, \dots, t_f]$, respectively. Spline duration is represented as $T_i := t_i - t_{i-1}, i = 1, \dots, n$. Conversely, velocity at initial time instant t_0 is *0* and at time t_n is *n*. Thus, a set of polynomial functions can be written as:

$$q_i(t) = a_i t^7 + b_i t^6 + c_i t^5 + d_i t^4 + e_i\, t^3 + f_i t^2 + g_i t + h_i \qquad (11)$$

where $i = 1, \dots, n,$

$$\dot{q}_i(t) = 7a_i t^6 + 6b_i t^5 + 5c_i t^4 + 4d_i t^3 + 3e_i\, t^2 + 2f_i t + g_i \qquad (12)$$

$$\ddot{q}_i(t) = 42a_i t^5 + 30b_i t^4 + 20c_i t^3 + 12d_i t^2 + 6e_i t + 2f_i \qquad (13)$$

It is possible to specify eight constraints. These are six boundary conditions and two via-point constraints.

Boundary conditions:

$$Initial\ Position = q(t_0) = q_0, \qquad Final\ Position = q(t_n) = q_n$$
$$Initial\ Velocity = \dot{q}(t_0) = v_0, \qquad Final\ Velocity = \dot{q}(t_n) = v_n,$$
$$Initial\ Acceleration = \ddot{q}(t_0) = \alpha_0, \qquad FinalAcceleration = \ddot{q}(t_n) = \alpha_n.$$

Via-Point constraints:

$$Position\ at\ time\ t_1(lift\ position) = q(t_1) = q_1,$$
$$Position\ at\ time\ t_2(unload\ position) = q(t_2) = q_2,$$

A single 7th order polynomial is needed for eight constraints. To solve for the eight undetermined polynomial coefficients, eight linear equations must be established.

Equations 11 to 13 may be solved by substituting the boundary conditions for zero velocity and acceleration at the beginning and end of the trajectory.

By defining $T = t_1 - t_0$ and $h = q_1 - q_0$, the coefficients $a_i, i = 0, \dots, 7$ are:

$h = q_0$

$g = \dot{q}_0$

$f = \dfrac{\ddot{q}_0}{2}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & t_1^5 & t_1^6 & t_1^7 \\ 1 & t_2 & t_2^2 & t_2^3 & t_2^4 & t_2^5 & t_2^6 & t_2^7 \\ 1 & t_n & t_n^2 & t_n^3 & t_n^4 & t_n^5 & t_n^6 & t_n^7 \\ 0 & 1 & 2t_n & 3t_n^2 & 4t_n^3 & 5t_n^4 & 6t_n^5 & 7t_n^6 \\ 0 & 0 & 2 & 6t_n & 12t_n^2 & 20t_n^3 & 30t_n^4 & 42t_n^5 \end{bmatrix} \begin{bmatrix} h \\ g \\ f \\ e \\ d \\ c \\ b \\ a \end{bmatrix} = \begin{bmatrix} q_0 \\ \dot{q}_0 \\ \ddot{q}_0 \\ q_1 \\ q_2 \\ q_n \\ \dot{q}_n \\ \ddot{q}_n \end{bmatrix} \qquad (14)$$

Equation 14 is in the form of a set of linear systems of equation represented by $\boldsymbol{MC} = \boldsymbol{Q}$;

The coefficients are solved by multiplying the inverse of the joint variable with $C = M^{-1}Q$. The remaining coefficients are solved using MATLAB’s linear algebra and symbolic toolbox.

4. RESULTS AND DISCUSSIONS

An AMD A4-9120 RADEON R3, 4 Computer core system, 2.20 GHz processor with 4 Gigabyte of RAM and a 64-bit windows 10 operating system was used. All simulations were done using MATLAB R2018a. The version 10.1 of the MATLAB robotics toolbox was used.

4.1 Joint Variable of the Specified Cartesian Points

Using Table 4.1, one can see how the robot's end-effector should move in order to complete the job at hand. End-effectors must travel from the start node to the destination node through the lift and drop points in order to complete the job. For the specific example used in generating the trajectory, the points specified in the cartesian coordinate $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ are given as:

$$Starting\ position\ (take\ point) = \begin{pmatrix} -0.5 \\ 0.5 \\ -0.5 \end{pmatrix}; \quad transfer\ position\ (lift\ or\ leave\ point) = \begin{pmatrix} -0.5 \\ 0.5 \\ 0.3 \end{pmatrix}$$

$$Unloading\ position\ (drop\ point) = \begin{pmatrix} 0.5 \\ -0.5 \\ -0.3 \end{pmatrix}; \quad Target\ position\ (placement\ point) = \begin{pmatrix} 0.5 \\ -0.5 \\ -0.5 \end{pmatrix}$$

ISSN (online): 2645-2685; ISSN (print): 2756-6811

MATLAB is used to generate a trajectory between two positions specified by joint angles via the usage of two way-points in between the start and end positions. Generated trajectories between individual positions were connected and, in that manner, the entire trajectory was modelled and its validity verified by using MATLAB Robotics Toolbox. Figure 4.1 gives result of the simulation showing the points generated in the task space. The joint angles of the four positions are also highlighted in Table 1.

Table 1: Joint Angles for the four Positions

	1st Joint	2nd Joint	3rd Joint	4th Joint	5th Joint	6th Joint
Joint angle of starting position:	2.57	-0.7873	-1.2022	-3.1416	-1.9895	0.5716
The joint angle of the transfer position:	2.57	-0.1026	-0.5	-3.1416	-0.6026	0.5716
Joint angle of removal position	-0.5716	-0.1026	-0.5	-3.1416	-0.6026	-2.57
The joint angle of the target position:	-0.5716	-0.7873	-1.2022	-3.1416	-1.9895	-2.57

*All angles are in radians

The configuration of the points was such that the PUMA 560 manipulator was in the left arm, elbow down, wrist flip (rotate 180 degrees) configuration. This was specified essentially to get a particular solution for the inverse kinematics when using the toolbox's ikine6s function, which results in an analytical solution that avoids the errors associated with using the numerical solution alternative.

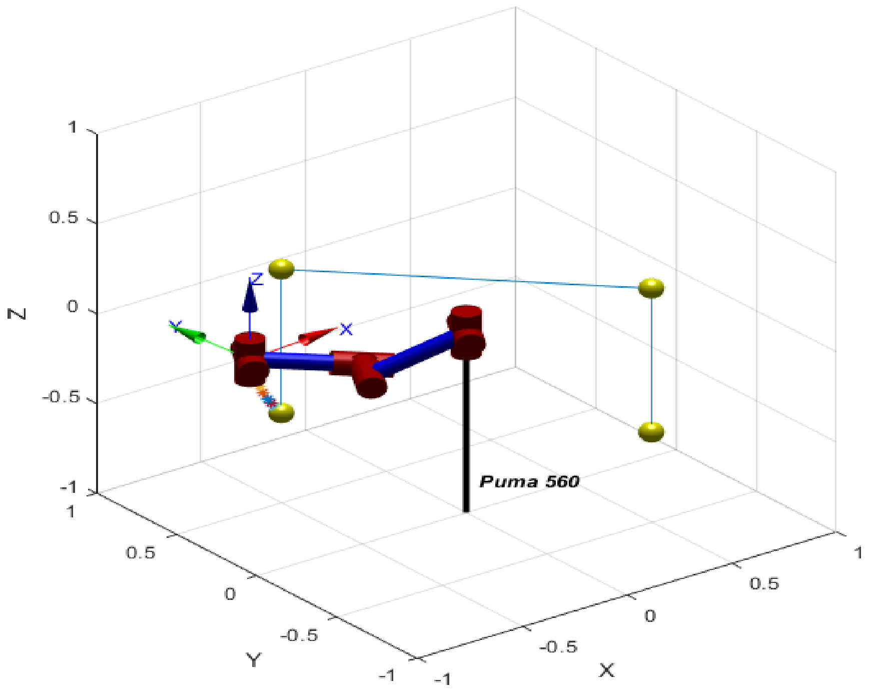


Figure 3: The Robot Workspace showing the start and end points with two via points

4.1.2 Coefficients of the Septic polynomial

For the specific locations used in specifying the path for the trajectory, coefficients of the septic function used for trajectory modelling is shown as seen in Table 2. In this way, the resulting kinematic parameter values are determined by the values of the computed coefficients. The coefficients of the 7th degree polynomials of each of the 6 joints are shown (from left to right corresponding with the 1st to 6th joints).

ISSN (online): 2645-2685; ISSN (print): 2756-6811

Table 2: Coefficients of the Septic Polynomial used in trajectory modelling

Coefficient	1st Joint	2nd Joint	3rd Joint	4th Joint	5th Joint	6th Joint
a	2.5700	-0.7873	-1.2022	-3.1416	-1.9895	0.5716
b	0	0	0	0	0	0
c	0	0	0	0	0	0
d	0.0938	0.2301	0.2359	0.0000	0.4660	0.0938
e	-0.0750	-0.1008	-0.1033	-0.0000	-0.2041	-0.0750
f	0.0169	0.0165	0.0170	-0.0000	0.0335	0.0169
g	-0.0015	-0.0012	-0.0012	0.0000	-0.0024	-0.0015
h	0.0001	0.0000	0.0000	0.0000	0.0001	0.0001

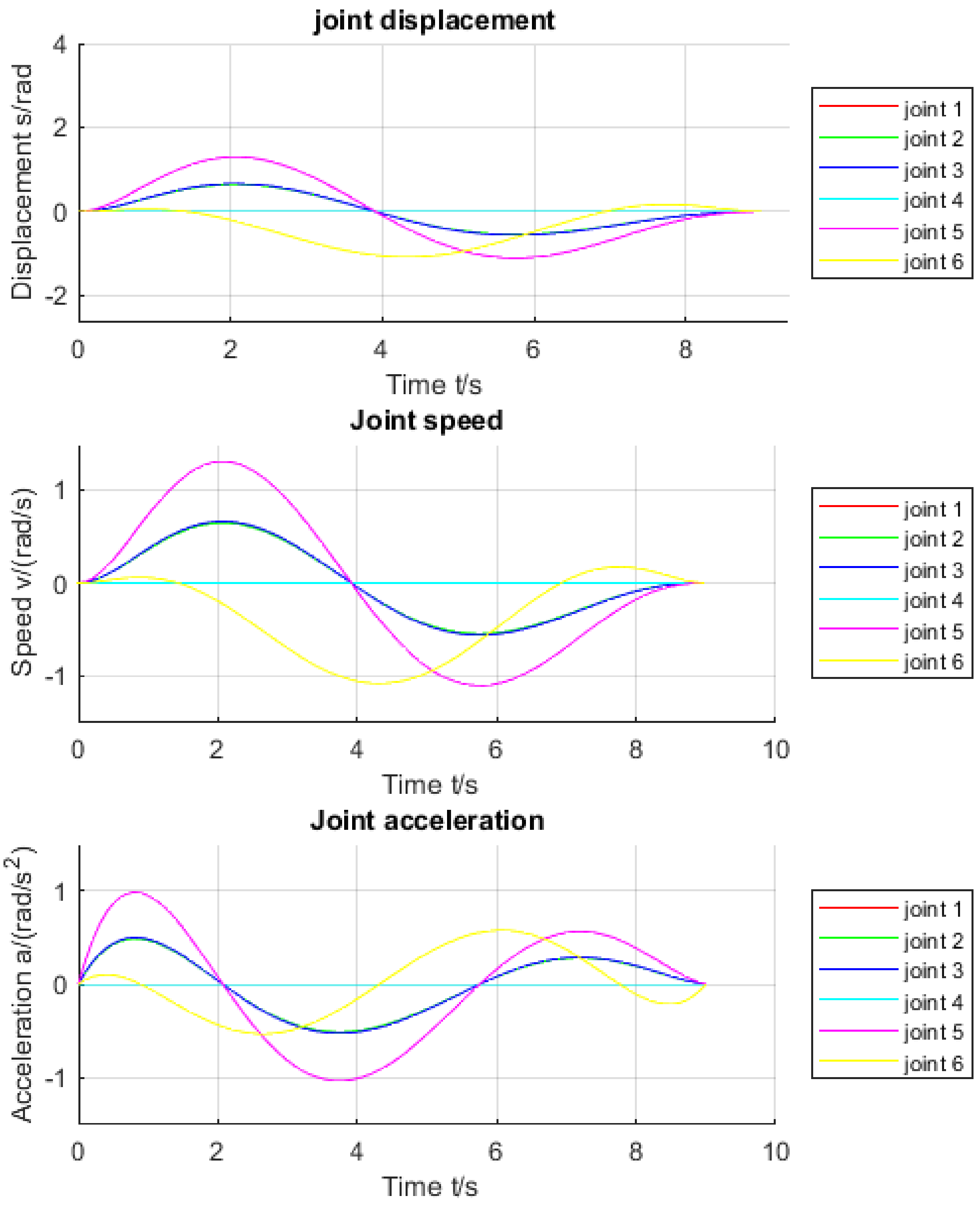


Figure 4: Septic polynomial trajectory Plots

ISSN (online): 2645-2685; ISSN (print): 2756-6811

4.3 Comparison of the Septic Polynomial function and Linear Segment with Polynomial Blend

Comparisons of plots for the septic and linear segment with parabolic blend trajectories are made in this section. Angular displacement, velocity and acceleration are compared and plotted for both trajectories. The MATLAB Robotics Toolbox function *mstra*j builds a multisegmented, multi-axis trajectory from intermediate points. The *mstraj* function is used in generating the LSPB trajectory.

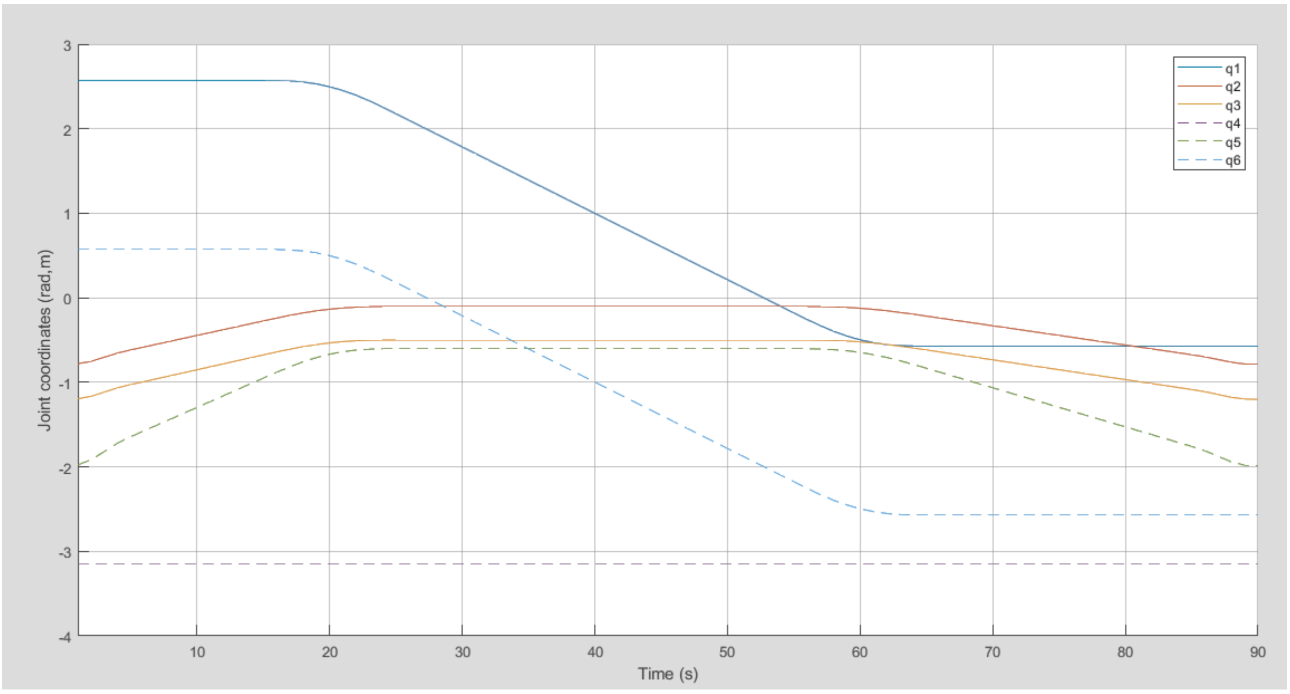
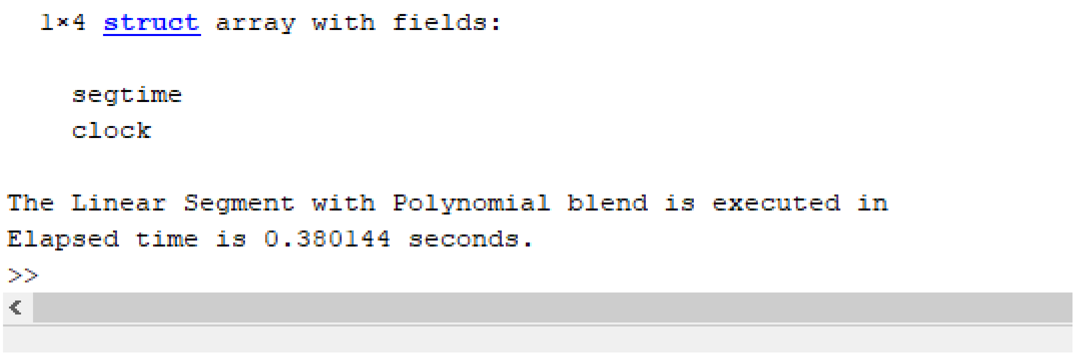


Figure 5: Position Plot of Linear segment with polynomial blend (LSPB)

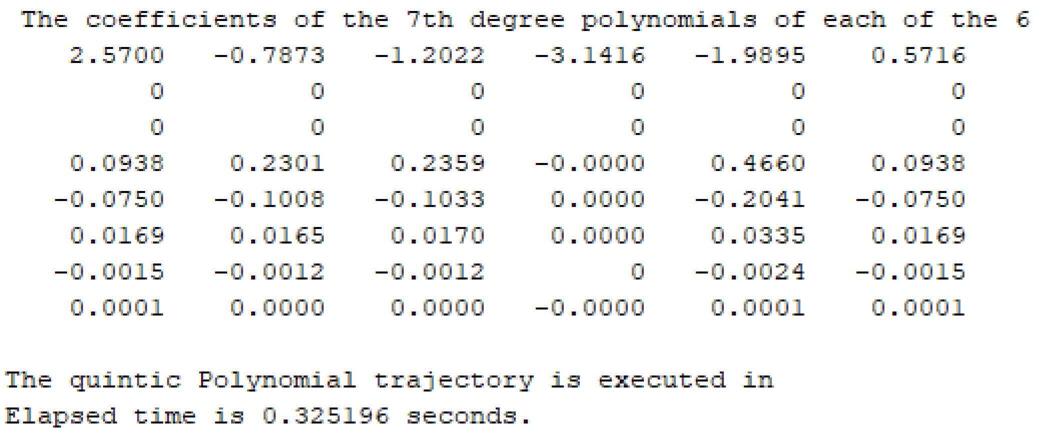
Considering the angular displacement plot in figure 4 which gives the septic polynomial plot and that of figure 5 which provides the linear segment with polynomial blend (in this case a quintic polynomial blend is adopted which is the default multi-segment trajectory generator for the Robotics toolbox), the LSPB trajectory is characterised by over shoot at the beginning of the trajectory. This can be seen when joint 1 is considered as the value of the displacement is about 2.6 radians, while that of the septic polynomial results in angular displacement of around 1.6 radians. Because the LSPB is an approximation approach, the issue of two via-points is over-constrained and the ability to reach each intermediate configuration is compromised in order to achieve continuous velocity.
Figure 4 shows the computed values of velocity and acceleration based on the plots. The septic polynomial trajectory generated zero velocity and acceleration at the endpoints, while the linear segment with polynomial blend trajectory produced zero velocity at the beginning but failed to produce zero velocity at the endpoint. This provides a continuous jerk profile which is desirable for minimisation of jerk and an excellent trajectory tracking ability. A downside of the septic polynomial is that the time the velocity is at maximum is small compared to that of the LSPB.
Considering the execution times of both trajectories, the tic toc function was used to evaluate roughly how fast the trajectory parameters were computed. Figures 6 and 7 show the result of the time taken. According to the result, the septic-based trajectory was faster with an execution time of 0.325196 seconds compared to 0.380144 seconds of the LSPB.


```
  1×4 struct array with fields:

    segtime
    clock

The Linear Segment with Polynomial blend is executed in
Elapsed time is 0.380144 seconds.
>>
```
Figure 6: Execution time of the Linear Segment with Polynomial Blend Trajectory

ISSN (online): 2645-2685; ISSN (print): 2756-6811



```
 The coefficients of the 7th degree polynomials of each of the 6
    2.5700   -0.7873   -1.2022   -3.1416   -1.9895    0.5716
         0         0         0         0         0         0
         0         0         0         0         0         0
    0.0938    0.2301    0.2359   -0.0000    0.4660    0.0938
   -0.0750   -0.1008   -0.1033    0.0000   -0.2041   -0.0750
    0.0169    0.0165    0.0170    0.0000    0.0335    0.0169
   -0.0015   -0.0012   -0.0012         0   -0.0024   -0.0015
    0.0001    0.0000    0.0000   -0.0000    0.0001    0.0001

The quintic Polynomial trajectory is executed in
Elapsed time is 0.325196 seconds.
```

Figure 7: Execution time of the Septic polynomial Trajectory

5. CONCLUSION AND FUTURE WORK

Existing and novel approaches for creating joint-space trajectories may all be applied independently to many joints at the same time. Using a single polynomial for movement through a two-via points, this work implements a unique joint-space trajectory creation technique. In order to maintain a smooth motion profile characterised by a continuous function, the robot does not have to halt at the intermediate points. As a replacement for the Linear segment with polynomial blend (LSPB), which is often used nowadays, a single seventh-order polynomial was proposed as there is no need for three segment function in the new technique. A septic polynomial is used in this study to produce null start and end-point velocities and accelerations. By placing two "control" via-points to the actual trajectory endpoints, a linear network comparable to resolving the velocity and acceleration boundary conditions independently is generated. Whenever the acceleration of a function changes, the corresponding jerk function (the time derivative of acceleration) will show endless spikes. Trajectory generation schemes emphasise that limitless jerk spikes are undesirable. Also, the generation of trajectories using 7th order polynomial was faster when compared to that of the LSPB as shown in the results section. Generally, the manipulator joint-space trajectory generation method employed in this paper is simple; as the septic polynomial employed is more apt for all joint movements with two via-points in point-to-point tasks. In addition to keeping the jerk to a minimum, the seventh-degree polynomial accomplishes the dual via-point constrained motion with just a single function.

To ensure that parameters like the jerk is continuous and bounded, trajectory optimisation becomes expedient. Some of these objective functions might be minimum energy consumption, trajectory time, dynamic load-carrying capacity, minimum jerk, singularities avoidance, manipulability, minimum gripping forces and so on. Essentially, when it comes to robotic manipulator trajectory optimisation, there are a number of criteria to consider. Solving these problems may profit from multi-objective optimisation, particularly when the objectives are conflicting[30]. The implication is that an improvement of one criterion might unnecessarily lead to the degradation of another. As such, the optimal compromise between them is selected and a trade-off made between competing objectives. In future, the generated trajectory would be optimised for better performance using a heuristic approach because of the computational complexity involved. premised on their ability to handle complex problems with little or no understanding of the search space. This ensures the lack of computational tractability involved in multi-objective trajectory optimisation is handled well.

NOMENCLATURE

Some abbreviations used in this research work include:

B-spline:	Basis Spline
CNC:	Computer Numerical Control
DoF:	Degrees of Freedom
EOD:	Explosive Ordinance Disposal
LSPB:	Linear Segment with Polynomial Blends
PTP:	Point-to-Point
PUMA:	Programmable Universal Machine for Assembly

REFERENCES

[1] Fakharian, A. (2018). Energy efficiency in the robot arm using genetic algorithm, in 2018 8th Conference of AI & Robotics and 10th RoboCup Iranopen International Symposium (IRANOPEN), 2018, 14–20.
[2] Hägele, M., Nilsson, K., Pires, N., & Bischoff, R. (2016). Industrial Robotics: Springer Handbook of Robotics, 2nd ed., Springer Handbooks. Cham, Switzerland: Springer.

[3] Berselli, G., Gadaleta, M., Genovesi, A., Pellicciari, M., Peruzzini, M., & Razzoli, R. (2016). Engineering methods and tools enabling reconfigurable and adaptive robotic deburring: Advances on Mechanics, Design Engineering and Manufacturing, Lecture Notes in Mechanical Engineering. Cham, Switzerland: Springer.
[4] Koloc, Z., & Václavík, M. (1993). Cam mechanisms. Amsterdam, Netherlands: Elsevier Science Limited, vol. 41.
[5] Biagiotti, L., & Melchiorri, C. (2008). Trajectory Planning for Automatic Machines and Robots. Springer Science & Business Media, Berlin, Germany.
[6] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). Robotics Modelling, Planning and Control. Springer.
[7] Visioli, A. (2000). Trajectory planning of robot manipulators by using algebraic and trigonometric splines. Robotica, 18 (6), 611-631.
[8] Chettibi, T., Lehtihet, H. E., Haddad, M., & Hanchi, S. (2004). Minimum cost trajectory planning for industrial robots. European Journal of Mechanics - A/Solids, 23(4), 703–715.
[9] Zhao, R., & Ratchev, S. (2017). Polynomial trajectory approximation along specified paths for robot manipulators, in 2017 13th IEEE Conference on Automation Science and Engineering (CASE), 2017, 106–111.
[10] Shreyas Bharadwaj, B. G., Vaishnavi, L., Purrab, D., Devi, M. A., & Prakash, C. P. S. (2021). Application of Mini-Max Method for Trajectory Planning of PUMA-560 using Cubic B Spline Interpolation Technique for Jerk Minimization, in 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), 2021, 1–6.
[11] Schilling, R. J. (2003). Fundamentals of Robotics Analysis and Control, Prentice Hall of India Private Limited, Delhi, India.
[12] Shi, B., & Zeng, H. (2021). Time-Optimal Trajectory Planning for Industrial Robot based on Improved Hybrid-PSO, in 2021 40th Chinese Control Conference (CCC), 2021, 3888–3893.
[13] Li, X., Gao, X., Zhang, W., & Hao, L. (2021). The Cubic B-spline Trajectories with the Boundary Conditions of Null Velocities and Accelerations, in 2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2021, 259–264.
[14] Lu, S., Ding, B., & Li, Y. (2020). Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation, Advances in Mechanical Engineering, 12(3), 1-18.
[15] Song, Q., Li, S., Bai, Q., Yang, J., Zhang, A., Zhang, X., & Zhe, L. (2021). Trajectory Planning of Robot Manipulator Based on RBF Neural Network, Entropy, 23(9), 1-21.
[16] Zhao, J., Han, T., Ma, X., Ma, W., Liu, C., Li, J., & Liu, Y. (2021). Research on Kinematics Analysis and Trajectory Planning of Novel EOD Manipulator, Applied Sciences, 11(20), 1-21.
[17] Wang, Z., Li, Y., Sun, P., Luo, Y., Chen, B., & Zhu, W. (2021). A multi-objective approach for the trajectory planning of a 7-DOF serial-parallel hybrid humanoid arm, Mechanism and Machine Theory, 165, 1-22.
[18] Piegl, L., & Tiller, W. (2012). The NURBS Book, Springer Science & Business Media, Berlin, Germany.
[19] Zhang, L., Wang, Y., Zhao, X., Zhao, P., & He, L. (2021). Time-optimal trajectory planning of serial manipulator based on adaptive cuckoo search algorithm, Journal of Mechanical Science and Technology, 35(7), 3171–3181.
[20] Schoenberg, I. J. (1964). On trigonometric spline interpolation. Journal of Mathematics and Mechanics, 795–825.
[21] Simon, D., & Isik, C. (1993). A trigonometric trajectory generator for robotic arms, International Journal of Control, 57(3), 505–517.
[22] Chiddarwar, S. S., & Babu, N. R. (2012). Optimal trajectory planning for industrial robot along a specified path with payload constraint using trigonometric splines, Int. J. Automation and Control, 6(1), 39–65.
[23] Parikh, P. A., Joshi, K. D., & Trivedi, R. (2021). Vision-Based Trajectory Planning for a Five Degree of Freedom Assistive Feeding Robotic Arm Using Linear Segments with Parabolic Blend and Cycloid Functions, Mechatronics and Machine Vision in Practice 4, 193–206.
[24] Alshahrani, S. A., Diken, H., & Aljawi, A. A. N. (2002). Optimum trajectory function for minimum energy requirements of a spherical robot, in 6th Saudi Engineering Conf., KFUPM.–Dhahran, 2002, 613–625.
[25] Devi, M. A., Prakash, C. P. S., Jadhav, P. D., Hebbar, P. S., Mohsin, M., & Shashank, S. K. (2021). Minimum Jerk Trajectory Planning of PUMA560 with Intelligent Computation using ANN, in 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, 544–550.
[26] Iyad, F. J. A., Rasha, F. A., & Mohamed, M. A. (2014). Statistical evaluation of an evolutionary algorithm for minimum time trajectory planning problem for industrial robots, The International Journal of Advanced Manufacturing Technology Manuscript, 89(1), 389-406.
[27] Jokić, D., Lubura, S., Rajs, V., Bodić, M., & Šiljak, H. (2020). Two Open Solutions for Industrial Robot Control: The Case of PUMA 560. Electronics, 9(6), 972.
[28] Corke, P. I. (2017). Robotics, Vision & Control, Springer, Cham, Switzerland.
[29] Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2012). Trajectory Planning in Robotics, Mathematics in Computer Science, 6(3), 269–279.
[30] Lazinica, A., & Kawai, H. (2010). Robot Manipulators New Achievements, InTechOpen, London, UK.