DOI: 10.2507/34th.daaam.proceedings.xxx

# IMPLEMENTATION OF TRAJECTORY PLANNING FOR THE MINIATURISED INDUSTRIAL ROBOTS OF THE MOROBOT PLATFORM

Claudia Holzgethan, Ali Aburaia, Kemajl Stuja, Mohamed Aburaia

**This Publication has to be referred as:** Holzgethan, C[laudia]; Aburaia, A[li]; Stuja, K[emajl] & Aburaia, M[ohamed] (2023). IMPLEMENTATION OF TRAJECTORY PLANNING FOR THE MINIATURISED INDUSTRIAL ROBOTS OF THE MOROBOT PLATFORM, Proceedings of the 34th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/34th.daaam.proceedings.xxx

**Abstract**

The International Federation of Robotics (IFR) has reported a 200% increase of installed industrial robots over the last decade, with further growth projected. To prevent a gap between supply and demand for a qualified workforce, Robotics in Education supports students with understanding robotics and sparking interest in technology. For a rapidly growing industry with new technologies and solutions emerging almost daily, the skills of and practice in critical comparison of methods and creative problem-solving are essential. The morobot platform of the University of Applied Sciences Technikum-Wien manufactures 3D-printed miniaturized industrial robots that enable cost-effective and easy access to knowledge in the robotics domain. However, the current implementation of the robot control leads to jerky movement and provides no possibility of creating a geometric path. In this paper, the platform is enhanced with trajectory planning to achieve smooth robot motions and implement continuous point-to-point and linear movements. Suitable velocity profiles are computed to achieve a desired motion. Given the high minimum velocity of the morobot motors, third- and fifth-order polynomials are compared regarding the smoothness of the resulting motion and the absolute positioning accuracy. Both methods achieved the desired smoothness. However, third-order polynomials provide a higher positioning accuracy, and have therefore been implemented for continuous point-to-point movement. For the linear movement, intermediate points are calculated to approximate a straight line between the start and goal robot position. Due to the high minimum velocity of the motors and the close proximity of the intermediate points, trajectory planning cannot resolve the jerky motion. Motors with lower minimum velocity should be identified for a future project phase.

**Keywords:** trajectory planning; third-order polynomials; fifth-order polynomials; geometric path; industrial robotics

## 1. Introduction

Be it industrial manufacturing or assistive technologies [1] robots are expected to play an increasingly important role in society, leading to a radical transformation in skill requirements. Robotics in Education focuses on strengthening the learning skills of the future workforce and motivating students for topics in Science, Technology, Engineering and Mathematics (STEM) [2], [3].

At the University of Applied Sciences Technikum Wien, the morobot platform was created. It focuses on manufacturing 3D-printed miniaturised industrial robots that enable inexpensive and easy access to knowledge in the robotic domain. The platform helps students to learn and understand aspects of industrial robotics, including the composition of different kinematic chains and their strengths and weaknesses, the computation of forward and inverse kinematics and the control of robots via inverse kinematics. However, the implementation of the platform comes with limitations regarding the control of the robot's geometric path and the smoothness of the motion. The current motor control leads to abrupt and jerky movements that induce mechanical stress on the robots' kinematic chains. Furthermore, it is only possible to define goal positions for the robot's tool centre point (TCP) that are approached in a discontinuous point-to-point movement. No continuous geometric path of the TCP can be composed.

To enhance the platform, this paper focuses on employing and comparing third- and fifth-order polynomials as trajectory planning methods for the miniaturised industrial robots. The goal is to minimise jerk and consequently create smooth and continuous movements, whilst maintaining a high positioning accuracy. Furthermore, continuous point-to-point and linear movements are implemented to enable the construction of geometric paths.

The paper is organised as followed: Section 2 gives an introduction to trajectory planning and explores different methods of creating and optimising trajectories. In section 3 the utilized hardware is presented, the limitations of the current implementation are discussed and the goals for this project are defined. Section 4 focuses on the motor limitations and the theoretical as well as practical realisation of trajectory planning and geometric paths for the miniaturised robots of the morobot platform. The paper closes with the discussion of the results in section 5 and the conclusion and suggestions for future projects in section 6.

## 2. State of the Art

In industrial robotics, a distinction is made between path planning and trajectory planning. Path planning mainly focuses on computing a geometric path along which the TCP should move [4]. Trajectory planning on the other hand deals with computing motion profiles, i.e. time sequences of positions, velocities, accelerations and higher derivatives of either the TCP or the robot's joint values [5]. The aim is to create desired motions, like smooth movements, by employing suitable displacement functions.

There are various approaches to generate trajectories for industrial robots. Bai et al. [6] use a five-segment interpolation function based on s-curves to create smooth displacement and velocity profiles for an ABB IRB 1200 industrial robot. S-curves are also employed in [7] and [8]. In the former, the s-curve trajectory is compared to a cubic spline trajectory with respect to execution time and jerk. The results show that the s-curve produces a faster and smoother trajectory than the cubic spline [7]. In [8], the authors use a sigmoid function to generate a desired jerk profile that enables an infinite order of continuity of the motion. This has the effect of minimising vibrations and increasing positioning accuracy. Zheng et al. [9] on the other hand utilise fifth-order polynomials and interpolation to compute a desired trajectory for a 6 degrees of freedom robot manipulator. Parikh and Dave [10] compare high order polynomials to parabolic and cubic functions. The results show that polynomials of order seven and nine generate smoother trajectories than the lower order polynomials but also lead to higher peaks of velocity and acceleration values. In [11] and [12] cubic splines and fifth-order B-splines are used to generate desired motion profiles.

Many papers also focus on optimising computed trajectories. Common optimisation criteria according to literature are minimum execution time [13], minimum energy consumption [14], and minimum jerk [7]. Often multiple of these criteria are combined to generate the optimal trajectory [12], [15].

### *2.1. Minimum execution time*

Increasing productivity is a major aspect when it comes to working economically. One way of achieving this is by minimising the execution time of the robot's task [16], [15]. Li and Wang [13] use a cubic polynomial curve to connect the robot's path points and generate a smooth trajectory. Then a genetic algorithm is applied to optimise the execution time of the robot's joints. The results show that the robot reaches the desired poses and follows the desired trajectory in the shortest time.

### *2.2. Minimum energy consumption*

The criterion of minimum energy consumption does not only focus on saving energy, which is beneficial as it reduces costs, but also on minimising the mechanical stress on the robot's actuators [16], [17]. Chen et al. [14] minimised the energy consumption for a bottle grasping task by applying a gaussian quadrature optimisation method.

*2.3. Minimum Jerk*

Jerk is defined as the time derivative of the acceleration, i.e. the third time derivative of the position. It indicates how quickly the acceleration of an object changes with respect to time. In general, jerk is undesirable in industrial robotics as it leads to an unnatural motion and increases the wear of the actuators and the robot's structure. The minimisation of jerk does not only lead to a smoother execution of the robot's trajectory and reduction of vibration and wear but also reduces trajectory tracking errors [15]–[17].

Devi et al. [7] developed an algorithm to generate a smooth trajectory with minimum jerk. First the inverse kinematics of a 6-DOF PUMA-560 robot is solved with the help of an artificial neural network. The generated joint values are than fed into an s-curve equation to produce a smooth trajectory with minimum jerk and time.

*2.4. Multi-criteria*

Often multiple criteria are decisive to obtain an optimal trajectory. Therefore, many papers focus on multi-criteria optimisation. In [12], the contradictory criteria minimum time, minimum acceleration and minimum jerk are combined to produce an optimal trajectory for a glass-handing robot. The authors use a minimised objective function to find an optimal balance between the three criteria and fifth-order B-splines to produce an optimal trajectory. Rout et al. [11] face a similar challenge. Here, the contradictory criteria time, jerk and torque are combined to generate a smooth trajectory with minimum execution time. The best trade-off between the criteria is found by combining the Non-Dominated Sorting Genetic Algorithm and the Nelder-Mead simplex method. The output of the combined algorithms was used to compute the joint angles for the desired cubic spline trajectory path.

Chiddarwar and Babu [15] developed a method to achieve an optimal trajectory in terms of minimising the execution time, energy, jerk and acceleration, whilst accomplishing maximum manipulability, i.e. maintaining all of the robots degrees of freedom at any time. Here a trigonometric spline was used to represent the desired trajectory. The authors then used sequential quadratic programming and genetic algorithm to optimise the trajectory according to the optimisation criteria.

## 3. Problem Description

In this paper, trajectory planning is applied on the miniature robots of the morobot platform of the UAS Technikum-Wien. For a better understanding of the current limitations of the platform, this section starts with an overview of the available robots and the motors that are used. Based on the knowledge of the employed hardware, the limitations of the current robot control and the goals for this project are discussed.

*3.1. Hardware*

The morobot platform consists of several miniaturised industrial robots with different kinematic chains, including both parallel and serial structures. Fig. 1 shows an overview of the available miniaturised industrial robots.

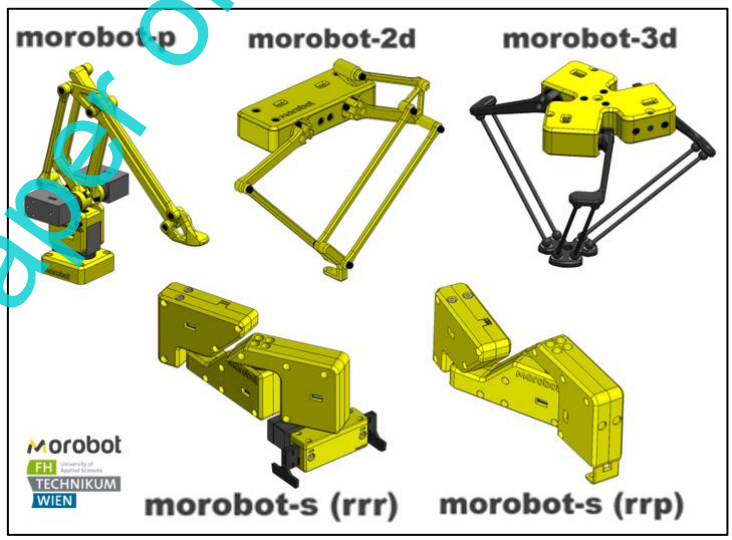


Fig. 1 : Overview of the miniaturised industrial robots of the morobot platform (top row: palletising robot, two-dimensional delta robot, three-dimensional delta robot | bottom row: RRR and RRP SCARA)

The top row in the image shows, from left to right, the parallel structures of a palletising robot, a two-dimensional and a three-dimensional delta robot. The bottom row shows two versions of a SCARA robot, one with three rotational axes (RRR) and one with two rotational and one translational axis (RRP). Each of the robots takes up an approximate size of a DIN A5 sheet of paper. For this project, the robots are controlled with an Arduino Mega 2560.

The motors that are used in the joints of the miniaturised robots are Smart Servos by Makeblock [18]. They dispose of an open-source programming library, which allows to both control the motors and receive information, like the current position, velocity or internal temperature. There are two main groups of commands with which the motors can be controlled:

A) Commands where the motor automatically moves to or by a defined angle and
B) Commands where a fixed motor velocity is set

In case of group A), the Makeblock Smart Servos use an internal velocity regulation to ensure that the target position is reached. On reaching the target position, the motor gets stopped. A continuous motion through via-points is not possible with these commands.

In case of group B), a fixed velocity in form of a PWM (pulse width modulation) value gets sent to the motor. The motor then moves with this velocity until it is set to zero again.

*3.2. Limitations of the implementation*

The problems at this paper's start state are two-fold. First off, the robots can only be controlled by defining target poses for their tool centre point. Inverse kinematics is used to calculate the corresponding joint values and motor angles for the target pose. However, the geometric path, i.e. the path that the TCP follows to reach the target pose, cannot be defined by the user.

Secondly, the calculated goal angles are sent directly to the motors via commands of group A). As described in Section 3.1 above, with these commands, the motors always stop once a target position is reached. Therefore, only a discontinuous point-to-point movement is possible and no continuous geometric path of the robot's end-effector can be composed. Furthermore, the motor's internal velocity regulation leads to abrupt and jerky movements that induce mechanical stress on the robot's kinematic chain. This jerky movement is exemplified in Fig. 2. The image shows the motion of the TCP of the RRR SCARA when moving from a start position to a goal position. The jerk in the motion can be observed by the jumps in the red line and is magnified in the two positions a) and b).

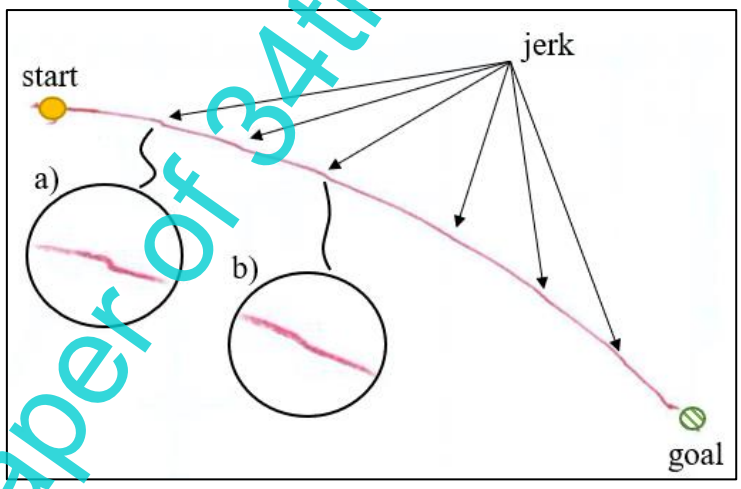


Fig. 2: Exemplary jerky movement of the TCP of the RRR SCARA with the current implementation of the robot control. The red line shows the motion of the TCP. The jerk is indicated by arrows and magnified at the two positions a) and b).

The problems of the current implementation can be summarised as follows:

1. The motor control leads to jerky and discontinuous movements
2. No geometric path can be defined by the user

The aim of this paper is to apply and compare two trajectory planning methods on the RRR SCARA of the morobot platform, with the goal of creating smooth and continuous motions with high position accuracy. Furthermore, the trajectory planning methods shall be used to implement a continuous point-to-point and linear movement of the robot's TCP.

## 4. Methods

The trajectory planning methods that are compared in this paper are third- and fifth-order polynomials. For a better understanding of the reasons that these methods were chosen, it is first necessary to examine the limitations of the employed motors.

### *4.1. Motor limitation – minimum velocity*

As described in section 3.1, in addition to the commands for setting a target pose, the utilised motors also compose of commands with which a velocity can be set via a PWM value. This is important, as it allows to compute time sequences of velocities via trajectory planning and sending these velocities to the motors.

Experiments, that were conducted on the three Smart Servos of the RRR SCARA showed that the minimum PWM value at which the motors rotate reliably is 8. To determine the actual velocity at the minimum PWM value, each of the motors was rotated by 50 degrees and the travel time was measured. The experiment was conducted 20 times per motor and the average travel time was calculated. The results are shown in Table 1.

| | **Motor 1** | **Motor 2** | **Motor 3** |
|---|---|---|---|
| **Average time [milliseconds]** | 1 923 | 1 826 | 1 812 |
| **Standard deviation [milliseconds]** | 8.82 | 10.15 | 16.10 |
| **Velocity [°/sec]** | 26.00 | 27.37 | 27.59 |

Table 1: Average time that is needed to rotate a motor by 50 degrees with the minimum PWM value of 8, taken from 20 measurements each

The minimum average time could be observed at motor 3 with a value of 1 812 [msec] and a standard deviation of 16.10 [msec]. After dividing the travelled distance of 50 degrees by this average travel time, a minimum velocity of 27.59 [°/sec] or 4.6 [rpm] can be obtained.

To put the minimum velocity in perspective, an example can be considered. Fig. 3 shows the SCARA robot in the two maximum displacement positions of the first joint.

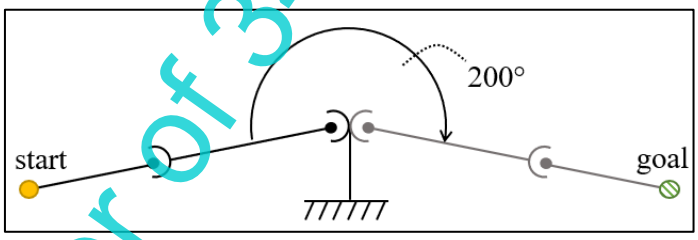


Fig. 3: Poses of the SCARA robot, when the first joint is moved to its minimum (start) and maximum (goal) angle

In total the first joint can be moved by 200 [°]. Dividing this value through the minimum velocity of the motors leads to a maximum travel time of 7.25 [sec]. This means that the first joint of the robot is not allowed to take more than 7.25 seconds to traverse its entire working area. Otherwise, the corresponding PWM value, that is sent to the motor, would be lower than the minimum value of 8 and the robot would not move reliably.

In addition to the maximum distance of a joint, the minimum distance can also be considered. In case of the Makeblock Smart Servos the minimum distance that can be travelled at a time equals 1 [°]. Divided by the minimum velocity, a maximum travel time of 0.036 [sec] or 36 [msec] can be obtained.

*4.2. Trajectory planning*

As mentioned before, trajectory planning focuses on computing desired motion profiles. It can have different goals like minimising the jerk, the energy consumption or the execution time. In this paper, the goal is to create smooth and continuous movements for the miniaturised robots of the morobot platform and therefore reducing their jerk. In robotics and automation in general, a multitude of different trajectory planning methods exits to achieve such a motion [19]. Ranging from polynomial and trigonometric to spline functions, there are different levels of complexity that can lead to varying results of the computed trajectory. In case of the miniaturised robots, two main aspects must be considered when choosing suitable trajectory planning methods. First off, to achieve smooth motions, the first- and second-order derivative of the trajectory function must be continuous [5]. Secondly, due to the high minimum velocity of the motors and the need to calculate the trajectory function at runtime, the computational effort of the employed methods must be as low as possible.

Under these aspects third- and fifth-order polynomials were chosen to be applied and compared. Although third-order polynomials are not continuous in their second derivative and can therefore not guarantee a jerk-free motion, they prove to be promising since their computational effort is low compared to other methods. Fifth-order polynomials do dispose of continuous first and second order derivatives, however, more equations must be computed at runtime.

Trajectory planning can either be conducted for a robot's TCP in the robot coordinate space or for the individual robot joints in the joint space [5]. In this paper, the trajectories in the joint space are examined. In the following sections, the joint trajectory functions will be denoted as $q(t)$, $v(t)$ and $a(t)$ for the position, velocity and acceleration. It shall be noted that these expressions describe the trajectory of one individual joint. In practice, these functions must be computed for all $j$ robot joints.

*4.2.1 Third-order polynomials*

Polynomials of degree three, also commonly known as cubic functions, and their first derivative can be expressed by the following equations (1) and (2) [5]:

$$q(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \tag{1}$$

$$v(t) = c_1 + 2c_2 t + 3c_3 t^2 \tag{2}$$

,where $q(t)$ and $v(t)$ are the joint value and joint velocity depending on time, $t$ is the time that has elapsed since the start ($t = t - t_0$, with $t_0 = 0$) and $c_0$ to $c_3$ are unknown coefficients. Since there are four unknown coefficients, four constraints must be defined. These can for example be the start and final joint value and the velocity at the start and end of the motion:

| At $t_0 = 0$: | At $t_f = t_1$: |
|---|---|
| $q(0) = q_0$ | $q(t_1) = q_1$ |
| $v(0) = v_0$ | $v(t_1) = v_1$ |

By applying the conditions in the equations (1) and (2), the coefficients can be calculated as follows (3)-(6):

$$c_0 = \theta_0 \tag{3}$$

$$c_1 = v_0 \tag{4}$$

$$c_2 = \frac{3(q_1 - q_0) - (2v_0 + v_1)t_1}{t_1^2} \tag{5}$$

$$c_3 = \frac{-2(q_1 - q_0) + (v_0 + v_1)t_1}{t_1^3} \tag{6}$$

The robot's joints will be controlled by setting desired velocities depending on time. Therefore, the equations (4), (5), (6) and (2) must be solved during runtime to determine the corresponding velocity profiles of the motors.

*4.2.2 Fifth-order polynomials*

Similar to third-order polynomials, fifth-order polynomials can be described by the following equations (7)-(9) [5]:

$$q(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \quad (7)$$

$$v(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 \quad (8)$$

$$a(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 \quad (9)$$

Here, $q(t)$, $v(t)$ and $a(t)$ describe the position, velocity and acceleration of the joint as functions of time. The variable $t$, again, stands for the time that has elapsed since the start of the motion with $t = t - t_0$ and $t_0 = 0$ and $c_0$ to $c_5$ are unknown coefficients.

In contrary to third-order polynomials, six constraints must be defined to solve the unknown coefficients. In addition to the initial and final position and velocity of the joint, also the start and final accelerations can be specified:

| At $t_0 = 0$: | At $t_f = t_1$: |
|---|---|
| $q(0) = q_0$ | $q(t_1) = q_1$ |
| $v(0) = v_0$ | $v(t_1) = v_1$ |
| $a(0) = a_0$ | $a(t_1) = a_1$ |

With these conditions, the equations (7)-(9) can be converted as follows, to determine $c_0$ to $c_5$ (10)-(15):

$$c_0 = \theta_0 \quad (10)$$

$$c_1 = v_0 \quad (11)$$

$$c_2 = \frac{1}{2} a_0 \quad (12)$$

$$c_3 = \frac{20(q_1 - q_0) - (12v_0 + 8v_1)t_1 - (3a_0 - a_1)t_1^2}{2t_1^3} \quad (13)$$

$$c_4 = \frac{30(q_0 - q_1) + (16v_0 + 14v_1)t_1 + (3a_0 - 2a_1)t_1^2}{2t_1^4} \quad (14)$$

$$c_5 = \frac{12(q_1 - q_0) - 6(v_0 + v_1)t_1 - (a_0 - a_1)t_1^2}{2t_1^5} \quad (15)$$

As can be seen in the equations, the changes in position, velocity and accelerations are multiplied by higher values than in third-order polynomials. Therefore, even if the same constraints are applied, the coefficients $c_n$ will have bigger values in fifth-order polynomials than in third-order polynomials. This leads to higher peaks in the velocity and acceleration. In case of fifth-order polynomials, six equations ((8),(11)-(15)) must be solved at runtime to calculate corresponding velocities depending on time.

*4.3. Geometric path*

In robotics it is often desired that the end-effector follows a specific geometric path to fulfil its tasks and avoid obstacles. Such a geometric path can be achieved by, for example, implementing and combining basic movements, like moving arbitrarily between points (point-to-point movement), following a line (linear movement) or following a circular arc (circular movement). In this paper, the previously described trajectory planning methods are utilised to implement a continuous point-to-point and linear movement.

*4.3.1 Point to point movement*

In a point-to-point movement a start and goal pose are specified in the robot coordinate system. The path that the TCP follows when moving from the start to the goal pose is unknown. Therefore, this type of movement is effective if no exact path is needed, like in pick-and-place or palletizing tasks. By combining multiple point-to-point movements, it is possible to avoid obstacles and approximate a desired geometric path.

In this paper, the movement between a start and goal point is accomplished by calculating velocity profiles for each of the robot's joints with the described trajectory planning methods. The calculated velocity profiles are approximated by sending the corresponding velocities depending on the time, to the motors at different time stamps during the movement.

Furthermore, in order to achieve a smooth motion, the robot's joints are synchronised so that they start and finish their movement at the same time. Fig. 4 shows an overview of the main process steps that are used to achieve the point-to-point movement for the miniaturised robots.

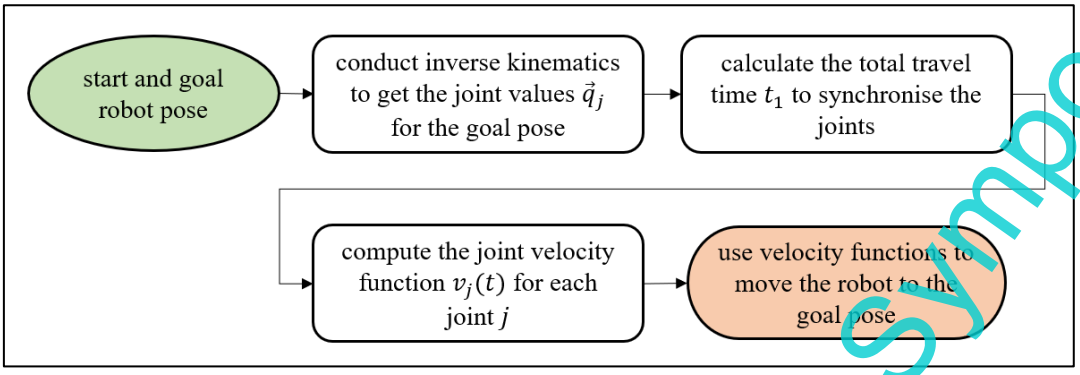


Fig. 4: Process steps for the point-to-point movement

The process begins with a start and goal TCP pose, whereby the start pose corresponds to the current pose of the robot. Inverse kinematics is used to calculate the joint values for the goal pose. Next, the total travel time $t_1$ is calculated. To achieve synchronisation, this travel time is the same for all joints. The velocity function $v_j(t)$ is computed for each joint, taking in consideration the start and goal pose, as well as the total travel time $t_1$. Depending on the employed trajectory planning method, also the start and end velocities and accelerations are required. Finally, the computed velocity functions are employed to move the robot from the start to the goal pose. For a better understanding, the synchronisation and motor control are discussed in more detail below.

**A. Synchronisation**

The synchronisation of the joints is achieved by computing a total travel time that is the same for all the robot's joints. This ensures that, if the motors are started at the same time, they also reach their target position at the same time. Since the employed Smart Servos have a high minimum velocity, the calculation of the total travel time is based on the fastest motor, i.e. the motor that travels the smallest distance to reach its goal position. If this motor were to move with the minimum velocity, it would need $n$ seconds to reach its goal position. To achieve synchronisation, the other motors must reach their target position at the same time. Therefore, the total travel time is calculated as follows (16):

$$t_{1,max} = \frac{distance_{min}}{velocity_{min}} \tag{16}$$

,where $t_{1,max}$ describes the total travel time of the joints, $distance_{min}$ describes the minimum distance that has to be travelled and $velocity_{min}$ describes the minimum velocity of the Smart Servos.

As indicated in the equation the corresponding travel time $t_{1,max}$ is the maximum total time that can be used in the velocity functions. Otherwise, velocities smaller than the minimum velocity might be computed and it cannot be guaranteed that all motors move reliably.

**B. Motor control**

Based on the determined total travel time, the start and goal joint values, velocities and accelerations, the velocity functions are calculated as described in section 4.2, with either third or fifth-order polynomials. Fig. 5 shows an example of such a velocity profile of a joint with a travel distance of 10 [°], a total travel time of $t_1 = 0.3$ [sec] and a start and goal velocity of $v_0 = v_1 = 25$ [°/sec]. A third-order polynomial was used to compute this exemplary motion profile.

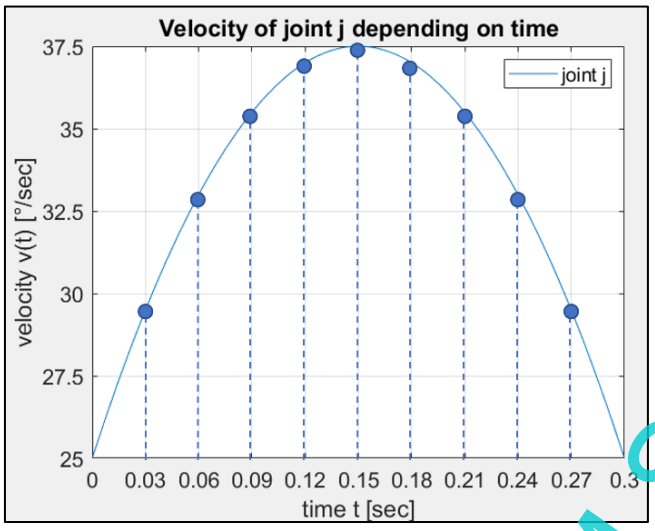


Fig. 5: Exemplary velocity profile with a total travel time of $t_1 = 0.3$ [$sec$]. The dots indicate the velocities that are sent to the motors at different time stamps

Note, that for the start and goal velocity a value unequal to zero was chosen. Instead, these parameters were set to a value slightly below the minimum velocity. As a result, even though the motors start from and end in standstill, no values smaller than the minimum velocity are computed.

In order to approximate the velocity profile, the velocity is calculated at several time stamps. This is indicated by the dots in the graph.

Since in trajectory planning, functions of time are computed, the resulting velocities have the unit [°/sec]. To set the motor velocities, these velocity values must be converted to PWM values. In order to find the correlation between the motor velocity in [°/sec] and the PWM value, the velocity was measured for different PWM values and linear regression was applied. This is illustrated in Fig. 6 for positive velocities.

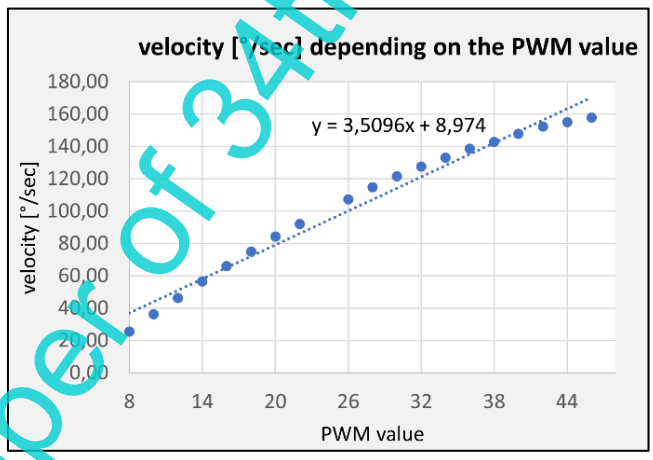


Fig. 6: Motor velocity in [°/sec] in dependence of the PWM value

As can be seen in the graph, a correlation of $y = 3.5096x + 8.974$ can be obtained. In this context $y$ describes the velocity in [°/sec] and $x$ the PWM value. Therefore, the PWM value can be computed as follows (17):

$$x = \frac{y - 8.974}{3.5096} \tag{17}$$

For a complete robot movement, this correlation must also be computed for negative velocities.

To summarize, the motor control process is as follows:

1) Calculate the coefficients $c_n$, based on the total travel time, start and goal position, velocity and acceleration
2) Divide the total travel time $t_1$ in $k$ parts
3) For each time stamp $k$:
   a) Use the coefficients $c_n$ to calculate the velocity in [°/sec]
   b) Convert the velocity into a PWM value
   c) Send the PWM value to the motor

This process must be conducted for each of the $j$ motors.

### *4.3.2 Continuous point-to-point movement*

In the previous section, the fundamentals and operating principle of the point-to-point movement between a start and goal pose were described. However, in some cases, it is desired, that the TCP moves through multiple points in a continuous motion to approximate a geometric path. This can be achieved by defining start and goal velocities for intermediate points that are unequal to zero.

Heuristic rules are applied as defined in equation (18) and (19), to determine optimal intermediate velocity [19].

$$v_k = \begin{cases} 0 & sign(d_k) \neq sign(d_{k+1}) \\ \frac{1}{2}(d_k + d_{k+1}) & sign(d_k) = sign(d_{k+1}) \end{cases} \tag{18}$$

$$d_k = \frac{q_k - q_{k-1}}{t_k - t_{k-1}} \tag{19}$$

In the equations, $v_k$ describes the calculated intermediate velocity, $q_k$ the position and $t_k$ the current time at point $k$.

With these rules, the intermediate velocity is set to zero if the motor changes directions and a value unequal to zero if the direction is maintained. An example of a corresponding position and velocity profile through intermediate points is depicted in Fig. 7

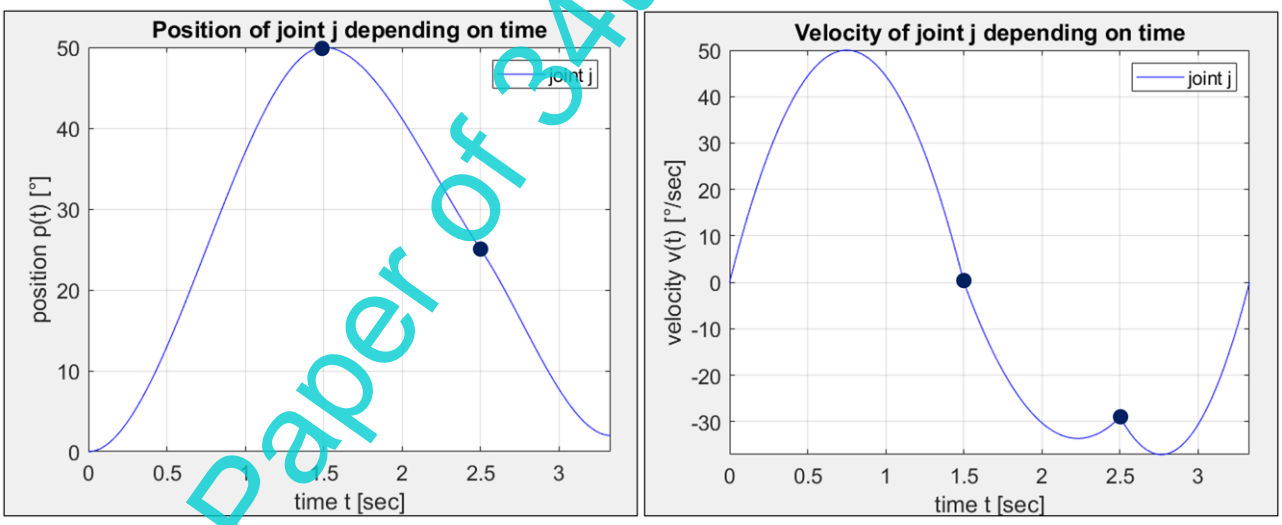


Fig. 7: Exemplary position (left) and velocity (right) profiles for a continuous point-to-point movement. The intermediate points are indicated by the dots at t=1.5 [sec] and t=2.5 [sec].

In the graphs, the intermediate points are located at the time stamps t=1.5 [sec] and t=2.5 [sec], as indicated by the dots. For a better visibility the velocity at the start, end and reversal point were set to zero. In practice, these parameters would, again, be set to a value slightly below the minimum velocity. In this example, the position and velocity profiles were calculated with third-order polynomials.

*4.3.3 Linear movement*

In a linear movement the robot's TCP follows a line to move from a start pose to a goal pose. This type of movement is suitable when an exact path must be maintained to fulfil a task, as it might be the case in assembly, cluing or painting processes.

The implementation of a linear movement is comparable to a continuous point-to-point movement. However, in a linear movement, the intermediate points are not defined by the user, but must be calculated by the control system. The main process steps for the implemented linear movement are depicted in Fig. 8.

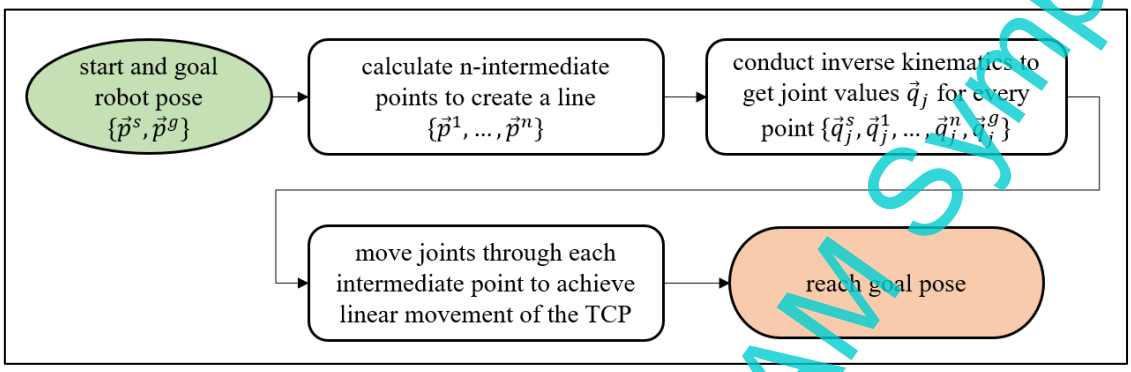


Fig. 8: Process steps for the linear movement

The process starts with a user-defined goal pose for the robot's TCP. The start pose corresponds to the current robot pose. In the first step, $n$ intermediate points are computed to approximate a line. These points are then converted into the joint space by applying inverse kinematics. In order to make the TCP follow the approximated line, the robot must be controlled in a way that all joints pass through their intermediate joint positions simultaneously.

**A. Computation of intermediate points**

In order to compute appropriate intermediate points, first, a line is placed between the start and goal position. Since, depending on the robot structure, the line can be three-dimensional, it is described with vectors as expressed in equation (20).

$$\vec{x}(s) = \vec{a} + s \cdot \vec{m} \tag{20}$$
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} p_x^s \\ p_y^s \\ p_z^s \end{pmatrix} + s \cdot \begin{pmatrix} p_x^g - p_x^s \\ p_y^g - p_y^s \\ p_z^g - p_z^s \end{pmatrix}$$

In the equation, $\vec{x}(s)$ stands for the x, y and z coordinate of any point on the line, $\vec{a}$ is described by the start point $\vec{p}^s$ and $\vec{m}$ by the difference between the start point $\vec{p}^s$ and end point $\vec{p}^g$. With this definition, the parameter $s$ has an interval of [0,1]. Therefore, an intermediate point $s_i$ is calculated by dividing through the desired number of points $n$ and multiplying the result with $i$ (21).

$$s_i = \frac{1}{n} \cdot i \tag{21}$$

An example of such a three-dimensional line is visualized in Fig. 9. The start point has the coordinates $\vec{p}^s = (4, 5, 2)^T$ and the end point $\vec{p}^g = (10, 20, 5)^T$. Eight intermediate points have been calculated with the equations (20) and (21) and are indicated in the graph by blue dots. The different coloured dots at the start and end of the line show the start and end point.

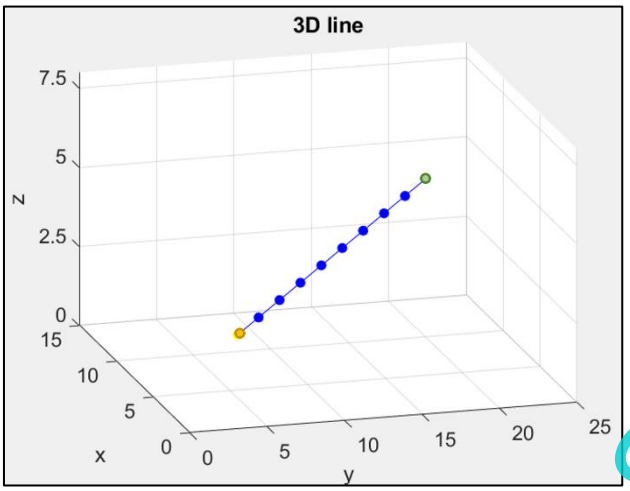


Fig. 9: Example of a three-dimensional line with intermediate points indicated by blue dots. (start point: $\vec{p}^{s} = (4{,}5{,}2)^{T}$, end point $\vec{p}^{g} = (10, 20, 5)^{T}$, number of intermediate point $n = 8$)

The optimal number of points $n$ depends on the length of line and the desired resolution and is calculated as follows (22):

$$n = \frac{length\_of_{line}[mm]}{resolution\ [mm]} \tag{22}$$

**B. Motor Control**

After calculating and transforming the intermediate points in the joint space, appropriate motor commands need to be defined to create the correct TCP movement. To ensure that the TCP moves through the intermediate points and therefore follows the calculated line, all joints need to pass through their intermediate joint positions at the same time. This behaviour can be achieved by synchronising the motors in between the intermediate points, similar to the point-to-point movement. However, since for the approximation of a line, the intermediate points are required to be close together, the distances that need to be traversed by the motors in between points are small, i.e. often an individual joint must only move by 1 [°]. As described in the motor limitations in section 4.1, this means that in these cases the total travel time $t_1$ would assume a value of only 0.036 [sec]. To achieve synchronisation all motors would have to finish their movement in these 0.036 [sec], even if they have to travel higher distances like e.g. 5 [°]. This means that very high velocities are required for the motors, e.g. 139 [°/sec]. However, the motors are physically limited in their acceleration and deceleration and therefore, cannot guarantee to stop their movement in such a short time. This leads to high overshoot and makes following a line with motor synchronisation impossible. Fig. 10 shows an example of the resulting TCP movement with this approach. The line between the start and end point has a length of 123.76 [mm]. By defining a resolution of 5 [mm], in total 24 intermediate points have been computed. The movement between the intermediate points was achieved via a continuous point-to-point movement as described in section 4.3.2. The motion was computed with third-order polynomials. In the image, the actual TCP movement is depicted in blue, the desired linear movement is indicated by a dotted line. As can be seen, with this approach, the TCP has an extreme overshoot and its movement is far away from representing a straight line.

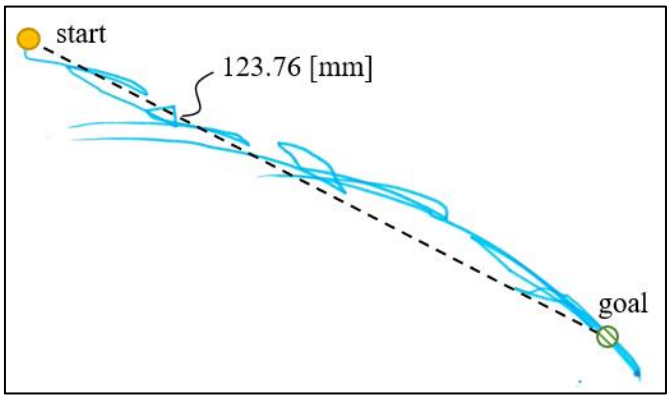


Fig. 10: Linear movement with motor synchronization. The actual TCP movement is illustrated in blue, the desired movement with a dotted line. Total length of line: 123.76 [mm], desired resolution: 5 [mm], number of intermediate points: 24

A different approach is needed to implement the linear movement of the miniaturised robots. For this, to maintain the short travel distances and the resulting short travel times between points, the motors are all set to the minimum velocity. The integrated Smart Servo functions are used to frequently check the current motor position. Once a motor has reached its target position, it is stopped. To achieve continuity, it is determined which motor must travel the greatest distance in between two adjacent points. Once this motor reaches a distance of 1 to 2 [°] from its target position, the joint travel distances to next TCP point are calculated and the motors are, again, set to the minimum velocity.

## 5. Results

Based on the theoretical knowledge of the applied trajectory planning methods and the practical implementation of the geometric paths, this section focuses on analysing and discussing the results. In particular, the smoothness of the TCP movement and the absolute positioning accuracy of the trajectory planning methods and geometric paths are examined.

### *5.1. Comparison of third- and fifth-order polynomials*

To choose the optimal trajectory planning method, first the results of third- and fifth-order polynomials are compared. Fig. 11 shows the TCP movement of the RRR SCARA when applying the respective methods and moving between a start and goal pose.

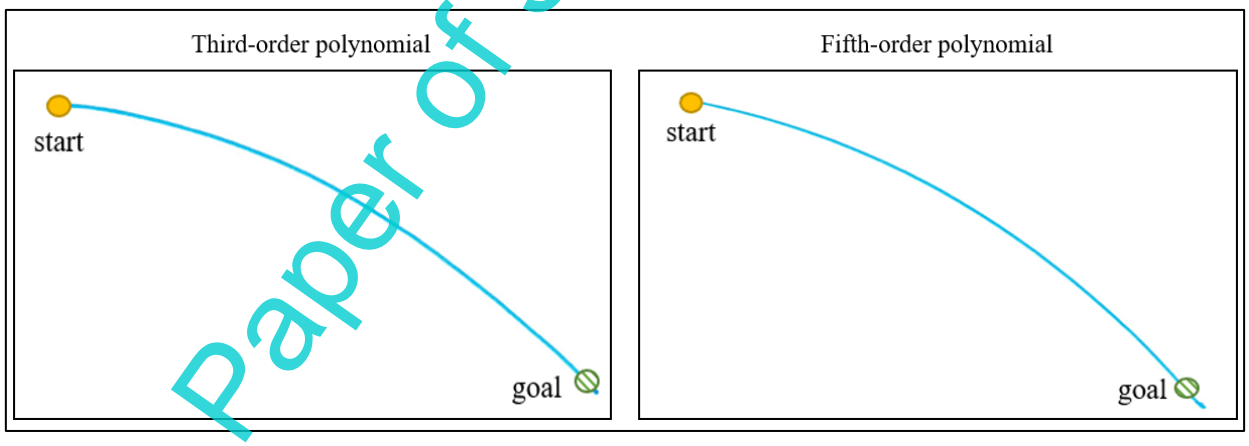


Fig. 11: TCP movement of the RRR SCARA with third-order polynomials (left) and fifth-order polynomials (right)

For both examples the robot started from the home position and was sent to the same goal position. As can be seen in the images, in both cases, the movement of the TCP is a lot smoother than in the start state (see Fig. 2). No jumps, in the form of jerk, are visible in the recorded movements. Therefore, both third- and fifth-order polynomials satisfy the criterion of smoothness.

In terms of the accuracy, varying results could be observed. The robot was started from and sent to different positions and the absolute positioning error in x and y as well as the angular error of the motors were calculated. Due to the low repetitive accuracy of the Smart Servos, the actual TCP positions were computed from the joint positions via forward kinematics. Table 2 summarises the minimum, maximum and average positioning errors as well as the median of five individual point-to-point movements.

| Method | Min [mm] | Max [mm] | Average [mm] | Median [mm] |
|---|---|---|---|---|
| **3rd-order polynomial** | 0.55 | 29.4 | 10.26 | 5.05 |
| **5th-order polynomial** | 0.95 | 31.81 | 11.91 | 11.17 |

Table 2: Minimum, maximum and average absolute positioning error and median of five individual point-to-point movements for both third- and fifth-order polynomials

As can be seen, with third-order polynomials, smaller errors can be achieved in all four columns. The slightly higher positioning errors of fifth-order polynomials can be explained by the fact that with this method overall higher velocity values are computed. Fig. 12 shows the velocity profiles of all three joints of the RRR SCARA for a point-to-point movement as computed by third- and fifth-order polynomials. In the graphs, the synchronisation of the motors is visible, as their motion starts and ends at the same time. In the depicted example, the second joint of the robot travels the shortest distance and consequently the velocity profile stays close to the minimum velocity. In both third- and fifth-order polynomials this joint reaches its exact target position of 18 [°]. The first joint has to move by the furthest distance of 53 [°]. In third-order polynomials, the velocity of this joint peaks at a value of about 100 [°/sec]. In fifth-order polynomials a maximum of about 120 [°/sec] is calculated. Both velocity values are so high, that the motor overshoots and misses its target position. However, since the third-order polynomial in generally produce smaller velocity values, it achieves an overall higher positioning accuracy.

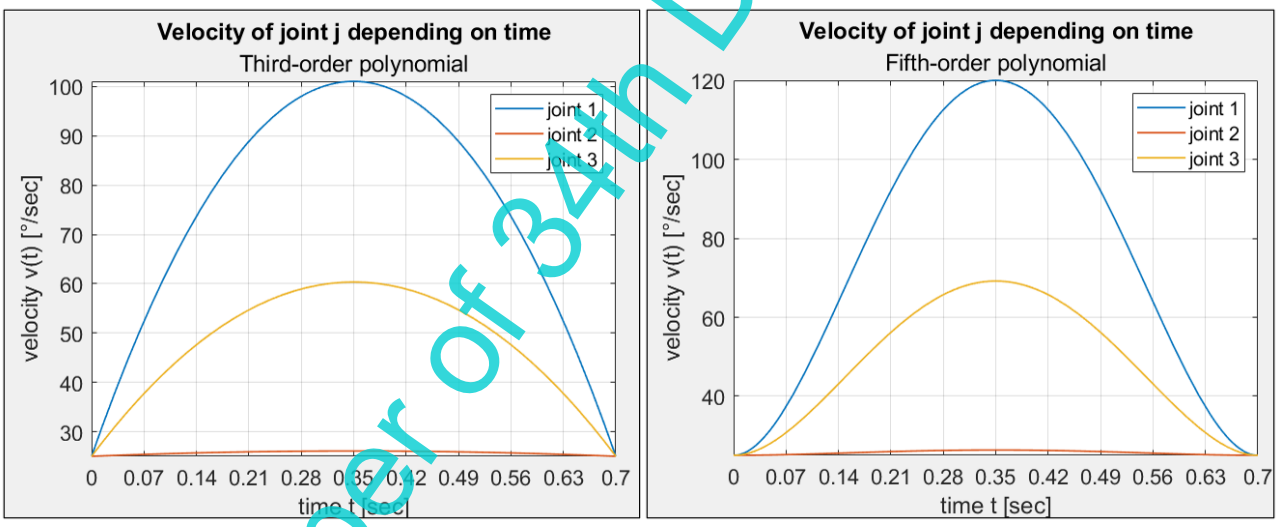


Fig. 12: Velocity profiles of all three RRR SCARA joints for a point-to-point movement, calculated with third-order polynomials (left) and fifth-order polynomials (right)

The variation of the absolute positioning errors of the individual trajectory planning methods can be traced back to the synchronisation of the motors. The greater the differences in distances that the individual motors have to travel, the higher the velocity peaks. High velocity peaks lead to high overshoot and consequently low positioning accuracy.

In overall, third-order polynomials prove to be more suitable for the employed Smart Servos. The method achieves a better positioning accuracy than fifth-order polynomials whilst showing the same path smoothness. The fact, that the second derivative of third-order polynomials is not continuous, does not have visible effects on the motion of the robot.

*5.2. Continuous point-to-point movement*

Third-order polynomials were chosen to implement a continuous point-to-point movement for the RRR SCARA. The smoothness of the geometric path is the same as in the discontinuous point-to-point movement. However, differences in the accuracy can be observed.

Experiments revealed that the angular positioning accuracy of the motors is higher at intermediate points where a change of direction occurs and the velocity is set to zero. These reversal points show an average error of 0 to 2 [°], whereas the average angular error at points with an intermediate velocity unequal to zero lies between 3 and 11 [°]. For the experiments the TCP of the RRR SCARA was moved five times through three intermediate points and the angular positioning error of the individual joints where determined. Overall, an absolute positioning error of the TCP of 12 [mm] in x and 4 [mm] in y could be obtained.

*5.3. Linear movement*

In case of the linear movement, due to the limitations of the minimum velocity, no smooth geometric path of the robot's TCP can be obtained. Fig. 13 shows the resulting TCP movement of the RRR SCARA with the implemented approach of the linear movement where all motors move with the minimum velocity. For the movement, the same constraints for the start and goal pose as well as the number of intermediate points as in Fig. 10 in section 4.3.3 were chosen.

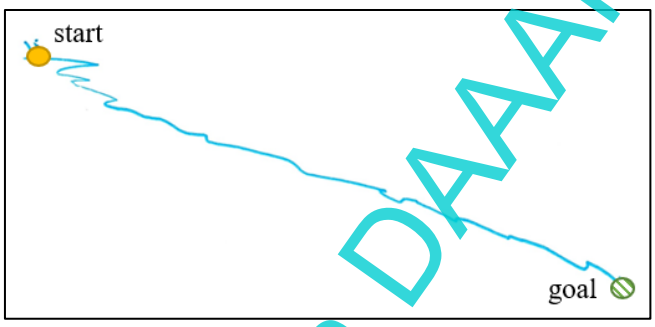


Fig. 13: TCP movement of the RRR SCARA with the implemented linear movement

As can be seen, with the implemented approach, the TCP movement does resemble a better line than in the continuous point-to-point movement in Fig. 10. However, since some of the motors are constantly stopped in between the intermediate points, there are a lot of jumps in the resulting TCP path. In the actual movement, these jumps correspond to jerk.

However, the employed implementation of the linear movement leads to a high positioning accuracy. After five experiments with different start and goal positions a median of the positioning error of 0.85 [mm] was obtained.

## 6. Conclusion and outlook

The morobot platform focuses on producing miniaturised industrial robots for educational purposes. However, the original implementation of these robots came with the limitation of jerky movement and no possibility to create a continuous geometric path. In this paper, trajectory planning was applied to enhance the morobot platform and achieve smooth motions as well as implement a continuous point-to-point and linear movement. Due to the high minimum velocity of the employed motors in the robot's joints, third- and fifth-order polynomials were chosen to be compared. The methods are used to calculate velocities in dependence of time that are sent to the motors to achieve a desired movement of the TCP. To ensure smooth motions, motor synchronisation is employed by defining an overall total travel time for all motors. The results showed that third-order polynomials achieve a higher absolute positioning accuracy whilst producing the same smooth motions as fifth-order polynomials. Therefore, third-order polynomials were chosen to implement the continuous point-to-point movement. Compared to the start state, the movement of the robot is a lot smoother as there is no jerk visible anymore. The positioning error of the developed motor control ranges from 0.5 [mm] up to 30 [mm]. It can be explained by high velocity peaks, that are caused by the already high minimum velocity, and result in overshoot.

Additionally, to the continuous point-to-point movement, also a linear movement was implemented. Because of the high minimum velocity, the use of trajectory planning and motor synchronisation was not possible. Therefore, only a very jerky movement could be achieved.

Both the high positioning error as well as the jerky linear movement might be solved by utilizing motors with a smaller minimum velocity. It would lead to smaller velocity peaks and consequently less overshoot and a higher accuracy. A smaller velocity would also allow to use trajectory planning to create smooth motions for the linear movement.

## 7. References


[1] Cooper, S., Di Fava, A., Vivas, C., Marchionni, L., Ferro, F. (2020). ARI: the Social Assistive Robot and Companion. In: *29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 31 August – 4 September 2020, Naples. Naples, Italy: IEEE, pp. 745-751.

[2] Ahmed, H., La, H., (2019). Education-Robotics Symbiosis: An Evaluation of Challenges and Proposed Recommendations. In: *2019 IEEE Integrated STEM Education Conference (ISEC)*, 16 March 2019, Princeton. Princeton, NJ, USA: IEEE, pp. 222-229.

[3] Curto, B., Moreno, C., (2016). Robotics in Education. *Journal of Intelligent & Robotic Systems*. 81(1), pp.3-4.

[4] Abu-Dakka, F., Rubio, F., Valero, F., Mata, V., (2013). Evolutionary indirect approach to solving trajectory planning problem for industrial robots operating in workspaces with obstacles. *European Journal of Mechanics – A/Solids*. 2013(42), pp. 210-218.

[5] Pratihar, D., (2017). *Fundamentals of Robotics*. Oxford: Alpha Science International Ltd.

[6] Bai, Y., Yang, X., Huang, W., (2020). Research on trajectory optimization of handling robot based on ADE. In: *2020 3rd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, 4-6 December 2020, Shanghai. Shanghai, China: IEEE, pp. 109-113.

[7] Devi, M., Prakash, C., Jadhav, P., Hebbar, P., Mohsin, M., Shashank, S., (2021). Minimum Jerk Trajectory Planning of PUMA560 with Intelligent Computation using ANN. In: *6th International Conference on Inventive Computation Technologies (ICICT)*, 20-22 January 2021, Coimbatore. Coimbatore, India: IEEE, pp. 544-550.

[8] Fang, Y., Hu, J., Liu, W., Shao, Q., Qi, J., Peng, Y., (2019). Smooth and time-optimal S-curve trajectory planning for automated robots and machines. *Mechanism and Machine Theory*. 2019(137), pp. 137-153.

[9] Zheng, X., Zheng, Y., Shuai, Y., Yang, J., Yang, S., Tian, Y., (2019). Kinematics analysis and trajectory planning of 6-DOF robot. In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 15-17 March, Chengdu. Chengdu, China: IEEE, pp. 1749-1754.

[10] Parikh, P., Dave, J., (2020). Trajectory Planning for the Five Degree of Freedom Feeding Robot Using Septic and Nonic Functions. International Journal of Mechanical Engineering and Robotics Research. 9(7), pp. 1043-1050.

[11] Rout, A., Mohanta, G., Gunji, B., Deepak, B., Biswal, B., (2019). Optimal time-jerk-torque trajectory planning of industrial robot under kinematic and dynamic constraints. In: *9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, 13-15 March 2019, Jaipur. Jaipur, India: IEEE, pp. 36-42.

[12] Duan, H., Zhang, R., Yu, F., Gao, G., Chen, Y., (2016). Optimal Trajectory Planning for Glass-Handing Robot Based on Execution Time Acceleration and Jerk. *Journal of Robotics*. Vol. 2016, pp. 1-9.

[13] Li, G., Wang, Y., (2019). Industrial Robot Optimal Time Trajectory Planning Based on Genetic Algorithm. In: *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 4-7 August 2019, Tianjin. Tianjin, China: IEEE, pp. 136-140.

[14] Chen, X., Zhang, W., Shi, Y., Fan, D., Zhang, T., Liu, G., Li, Q., Huang, Q., (2017). An energy optimization based planning approach for moving bottle grasping task using a seven-DoF robotic arm. In: *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, 6-9 August, 2017, Takamatsu. Takamatsu, Japan: IEEE, pp. 833-839.

[15] Chiddarwar, S., Babu, N., (2012). Optimal trajectory planning for industrial robot along a specified path with payload constraint using trigonometric splines. *International Journal of Automation and Control*. 6(1), pp. 39-65.

[16] Gasparetto, A., Boscariol, P., Lanzutti, A., Vidoni, R., (2012). Trajectory Planning in Robotics. *Mathematics in Computer Science*. 6(3), pp. 269-279.

[17] Ratiu, M., Prichici, M., (2017). Industrial robot trajectory optimization- a review. In: *Annual Session of Scientific Papers IMT ORADEA 2017*, 27-29 May (2017), Sanmartin. Sanmartin, Romania: EDP Sciences, pp. 54-59.

[18] Makeblock, (2021). *Smart Servo MS-12A* [Datasheet] Available at: https://www.manualshelf.com/manual/makeblock/95080/data-sheet-english/page-5.html [Accessed on 14 December 2021]

[19] Biagiotti, L., Melchiorri, C., (2008). *Trajectory Planning for Automatic Machines and Robots*. Berlin: Springer Science & Business Media.