Analyzing different trajectory generation methods for serial manipulators

RBE 501 Robot Dynamics Project - Spring 2020

Madhan Suresh Babu
Robotics Engineering
Worcester Polytechnic Institute
msureshbabu@wpi.edu

Muqeet Jibran
Robotics Engineering
Worcester Polytechnic Institute
mmuqeetjibran@wpi.edu

Sabhari Natarajan
Robotics Engineering
Worcester Polytechnic Institute
snatarajan@wpi.edu

Vrushabh Desai
Robotics Engineering
Worcester Polytechnic Institute
vmdesai@wpi.edu

***Abstract*—Any serial manipulator has to perform an action of moving from a its current position to a desired position. This action can be defined using trajectories. There are multiple ways of generating these trajectories which will be discussed and analysed in this project. Based on the action requirement, this can be in the task space or the joint space. Two 3 DoF serial manipulators, cartesian and articulated were used for this study.**

***Index Terms*—Cartesian Manipulator, Articulated Manipulator, Trajectory Generation, Task Space, Joint Space**

I. Introduction

Every serial manipulator, has its limitations in terms its hardware capabilities i.e. Joint Space and the environment i.e. Task space in which it has to function. Any task for a manipulator can be categorized into three stages namely Path planning, Trajectory planning and Robot control as shown in Fig. 1.

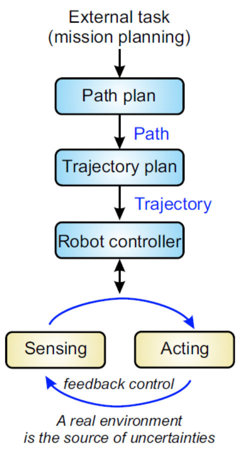


Fig. 1: Motion Planning Overview

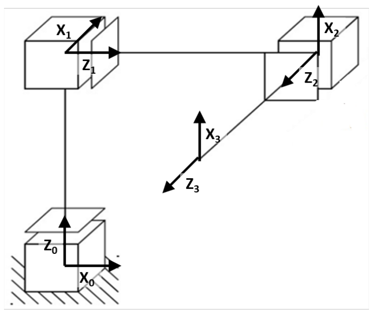
Path planning is the stage, where a pure geometric description of motion is specified or generated. Path is defined as a set of ordered loci of points in the the joint/task space, which the manipulator should follow. Trajectory planning is where we convert the path as a function of time. Trajectory is a path plus velocities and accelerations in its each point. This is the section where we will be focusing in this study. The final step with the trajectory generated, is to send command to manipulator joints based on the trajectory. This is taken care by the robot controller.

The two serial manipulators considered for this study are the two extremes. First is a 3 DoF cartesian manipulator where all joints are prismatic and second, a 3 DoF articulated manipulator where all joints are revolute. Two commonly used trajectory interpolation methods, Polynomial and Linear Segments with Parabolic Blends (LSPB) for the two manipulators will be analysed in the joint and task space in terms of the position, velocity and acceleration profiles.

II. Mathematical Model

A. 3 DOF Cartesian Manipulator

DH Frames and Parameters: The frames are assigned based on the Denavit-Hertenberg (DH) convention. With this conventional we can come up with a set of DH-parameters which can be used to arrive at the homogeneous transformation matrices. The frame assignments and DH-parameters are shown in Fig. 2.



Link	θ (°)	d (m)	a (m)	α (°)
1	90	$l_1 + q_1^*$	0	90
2	90	$l_2 + q_2^*$	0	-90
3	0	$l_3 + q_3^*$	0	0

Fig. 2: Cartesian Manipulator - DH Frames and parameters

Forward Kinematics: Forward kinematics is the use of kinematic equations of the manipulator to determine the position of the end-effector from the joint parameters (q). The use of DH convention makes this process easier for us. We can get the end-effector pose directly from the Homogeneous Transformation matrix and its as follows.

$$
\begin{aligned}
x &= +l_2 + q_2 \\
y &= -l_3 - q_3 \\
z &= +l_1 + q_1
\end{aligned}
$$

Inverse Kinematics: As the name suggests, Inverse Kinematics (IK) is the inverse function of Forward Kinematics. We follow a geometric approach instead of numerical one to solve the Inverse kinematics problem. The Inverse Kinematics for the scara manipulator is as shown below.

$$
\begin{aligned}
q_1 &= z - l_1 \\
q_2 &= x - l_2 \\
q_3 &= -l_3 - y
\end{aligned}
$$

Jacobian: Forward and Inverse kinematics describe the relationships between the static configurations of a mechanism and its end-effector. Jacobian is used to relate the velocities $(\dot{x}, \dot{q})$ and forces (F, τ) between task and joint space. We use the Transformation Matrices to calculate the Jacobian. Here we will be using the Geometric Jacobian (J). J can be decomposed into two components linear (J_v) and angular (J_ω).

$$
J = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix}
$$

Jacobian relates, $\dot{x}$ and $\dot{q}$ as $\dot{x} = J\ \dot{q}$. This is called as Inverse Velocity Kinematics (IVK). The end-effector forces F and joint torques τ as $\tau = J^T F$.

In this study, because its a 3 DoF manipulator, we are only concerned with the linear component J_v. For a prismatic joint, $J_{vi} = z^0_{i-1}$, where z^0_{i-1} is z vector (third column) in Homogeneous transformation matrix T^0_{i-1}. Following is the J_v calculated for the cartesian manipulator.

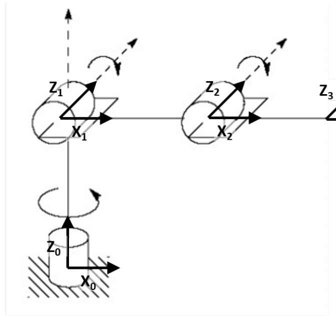
$$
J_v = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix}
$$

B. 3 DOF Articulated Manipulator

DH Frames and Parameters: The same DH convention used for the cartesian manipulator has been used and the DH frames and parameters for Articulated manipulator are shown in Fig. 3.

Forward Kinematics: The homogeneous transformation matrix calculated using the DH-parameters was used for Forward Kinematics.

$$
\begin{aligned}
x &= cos(q_1)\ (l_3\ cos(q_2 + q_3) + l_2\ cos(q_2)) \\
y &= sin(q_1)\ (l_3\ cos(q_2 + q_3) + l_2\ cos(q_2)) \\
z &= l_1 - l_3\ sin(q_2 + q_3) - l_2\ sin(q_2)
\end{aligned}
$$



Link	θ (°)	$d\ (m)$	$a\ (m)$	α (°)
1	q_1^*	l_1	0	−90
2	q_2^*	0	l_2	0
3	q_3^*	0	l_3	0

Fig. 3: Articulated Manipulator - DH Frames and parameters

Inverse Kinematics: Similar to the cartesian manipulator, geometric approach was used to solve for Inverse Kinematics.

$$
\begin{aligned}
&q_1 = atan2(y, x) \\
&q_2 = atan2(y^1_3, x^1_3) - atan2(l_3\ sin(q_3), l_2 + l_3\ cos(q_3)) \\
&q_3 = atan2(\pm\sqrt{1 - D^2}, D) \\
&\quad \text{where,} \\
&\quad x^1_3 = x\ cos(q_1) + y\ sin(q_1) \\
&\quad y^1_3 = l_1 - z \\
&\quad D = ({x^1_3}^2 + {y^1_3}^2 - {l_2}^2 - {l_3}^2)/(2\ l_2\ l_3)
\end{aligned}
$$

Jacobian: Again, we are only concerned about the linear component of jacobian. For a revolute joint, $J_{vi} = z^0_{i-1} \times (o^0_3 - o^0_{i-1})$, where z_{i-1} is z vector (third column) and o^0_{i-1} is the position (fourth column) in Homogeneous transformation matrix T^0_{i-1}. Following is the J_v calculated for the articulated manipulator.

$$
J_v = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{pmatrix} \text{ where,}
$$

$$
\begin{aligned}
J_{11} &= -sin(q_1)\ (l_3\ cos(q_2 + q_3) + l_2\ cos(q_2)) \\
J_{12} &= -cos(q_1)\ (l_3\ sin(q_2 + q_3) + l_2\ sin(q_2)) \\
J_{13} &= -l_3\ sin(q_2 + q_3)\ cos(q_1) \\
J_{21} &= cos(q_1)\ (l_3\ cos(q_2 + q_3) + l_2\ cos(q_2)) \\
J_{22} &= -sin(q_1)\ (l_3\ sin(q_2 + q_3) + l_2\ sin(q_2)) \\
J_{23} &= -l_3\ sin(q_2 + q_3)\ sin(q_1) \\
J_{31} &= 0 \\
J_{32} &= -l_3\ cos(q_2 + q_3) - l_2\ cos(q_2) \\
J_{33} &= -l_3\ cos(q_2 + q_3)
\end{aligned}
$$

III. Task Space vs Joint Space

Task space is the operation of the manipulator in cartesian space. It has x, y, z orthonormal axes. Where as, joint space is defined by a vector whose components are the translational and angular displacements of each joint of a robotic link i.e. q_1, q_2, q_3 for the manipulators in this study.

Based on the nature of the task to be performed by the manipulator, we can work either in task space or joint space. Task space is preferred in scenarios where we need navigate around obstacles in the environment. Fig. 4 shows how the path varies in the task and joint space while navigating around obstacles [5].

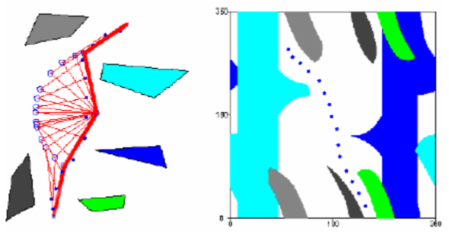


Fig. 4: Task Space(Left) to Joint Space(Right) for a 2 DoF manipulator

Trajectory planning in task space leads to undesirable results at some instances. Near the singularity configurations, the velocity required in joint space for a specified task space velocity increases rapidly, which can exceed the limits of the manipulator joints. This is also computationally expensive as compared to joint space trajectory generation. This will be seen in detail in the upcoming sections.

Trajectory planning in joint space overcomes the issues in task space, but the trajectory followed by the end-effector might not be a desirable one. Both these method will explore and analyzed in the upcoming sections.

IV. Trajectory Generation

To generate the trajectory we need the path and the constraints required as shown in Fig. 5. We will start with point-to-point motion where we need to move the end-effector from a starting point to ending point. The trajectory generated can be in the task space or the joint space. For the manipulators used in the study, in joint space we would be having 3 functions $q_1(t), q_2(t), q_3(t)$ whereas, in the task space it would be $x(t), y(t), z(t)$. As the robot controller takes input in terms of q_1, q_2, q_3, in the task space we need to perform an additional activity of converting x, y, z to q_1, q_2, q_3 using IK at every time-step. This also leads to higher computational requirements as compared to joint space.

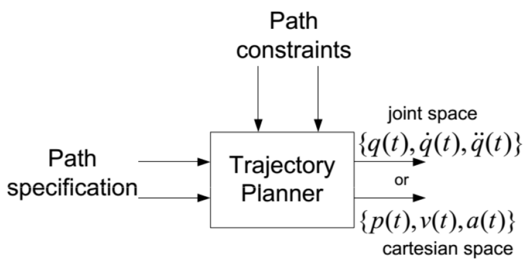


Fig. 5: Path to Trajectory

There are multiple interpolation methods, and we will be analysing two of them. Polynomial interpolation and Linear segments with Parabolic Blends.

Polynomial: In this method a n^{th} degree polynomial $f(t)$ is used for interpolation. In this study we are using a 3^{rd} degree polynomial, hence there are four coefficients that need to be determined for generating the trajectory. With a cubic polynomial we only have control over the start and final, position and velocity. If we need to control acceleration as well, then we would have to use quintic polynomial. Few parameters need to be defined, which are as follows:

Start Time	t_s
End Time	t_f
End-Effector Position at t_s	$X_s = (x_s, y_s, z_s)$
End-Effector Position at t_f	$X_f = (x_f, y_f, z_f)$
End-Effector Velocity at t_s	$\dot{X}_s = (\dot{x}_s, \dot{y}_s, \dot{z}_s)$
End-Effector Velocity at t_f	$\dot{X}_f = (\dot{x}_f, \dot{y}_f, \dot{z}_f)$
Joint Position at t_s	$q_s = (q_{1s}, q_{3s}, q_{3s})$
Joint Position at t_f	$q_f = (q_{1f}, q_{3f}, q_{3f})$
Joint Velocity at t_s	$\dot{q}_s = (\dot{q}_{1s}, \dot{q}_{3s}, \dot{q}_{3s})$
Joint Velocity at t_f	$\dot{q}_f = (\dot{q}_{1f}, \dot{q}_{3f}, \dot{q}_{3f})$

Knowing the parameters for End-Effector position and velocity we can use IK and IVK to convert them to the joint position and velocity's and vice versa. The four constraints used to obtain the 4 co-efficients are:

Constraint	Joint Space	Task Space
Position at t_s	$f(t_s) = q_s$	$f(t_s) = X_s$
Position at t_f	$f(t_f) = q_f$	$f(t_f) = X_f$
Velocity at t_s	$\dot{f}(t_s) = \dot{q}_s$	$\dot{f}(t_s) = \dot{X}_s$
Velocity at t_f	$\dot{f}(t_f) = \dot{q}_f$	$\dot{f}(t_f) = \dot{X}_f$

Solving this set of equations we obtain 3 polynomials i.e. $q_1(t), q_2(t), q_3(t)$ in joint space or $x(t), y(t), z(t)$ in the task space. This trajectory generated is fed to the robot controller as input.

1) Linear Segments with Parabolic Blends (LSPB): In this interpolation method the generated trajectory consists of two parabolic curves connected by a line segment. This results in a velocity profile that is trapezoidal in shape. It is generally preferred when a constant velocity is required for a portion of the trajectory.

This method requires as conditions the position, velocity at the start and end time (as shown under the Polynomial method), and the maximum velocity required in the trajectory. Using these values, the blend time (t_b) is computed which is the time for which the parabolic trajectory will be followed. Hence, from time $t = 0$ to $t = t_b$ and from time $t = t_f - t_b$ to time $t = t_f$, the trajectory is fitted with 2 quadratic polynomials. And in between time $t = t_b$ to $t = t_f - t_b$, the velocity is made constant and equal to the velocity value at time $t = t_b$.

If the max desired velocity of the trajectory is too large compared to the distance to be covered, the blend time t_b

would be almost equal to t_f, and would not produce the desired trajectory. Hence, in such cases, the blend time t_b is made equal to half of end time t_f, i.e., $t_b = t_f/2$. And in this case the velocity profile is triangular in shape.

We get three t_b values, one each for q_1, q_2, q_3 or x, y, z. We use the minimum of these values as the t_b for all three. This ensures that we have a straight line path in the generated trajectory for point-to-point motion. The constraints for the first parabolic segment from $t = t_s$ till $t = t_b$ are:

Constraint	Joint Space	Task Space
Position at t_s	$f_1(t_s) = q_s$	$f_1(t_s) = X_s$
Velocity at t_s	$\dot{f}_1(t_s) = \dot{q}_s$	$\dot{f}_1(t_s) = \dot{X}_s$
Velocity at t_b	$\dot{f}_1(t_b) = \dot{q}_{t_b}$	$\dot{f}_1(t_b) = \dot{X}_{t_b}$

The constraints for the second parabolic segment (end) from $t = t_f - t_b$ till $t = t_f$ are:

Constraint	Joint Space	Task Space
Position at t_f	$f_2(t_f) = q_f$	$f_2(t_f) = X_f$
Velocity at t_f	$\dot{f}_2(t_f) = \dot{q}_f$	$\dot{f}_2(t_f) = \dot{X}_f$
Velocity at $t_f - t_b$	$\dot{f}_2(t_f - t_b) = \dot{q}_{t_b}$	$\dot{f}_2(t_f - t_b) = \dot{X}_{t_b}$

Based on these conditions, the coefficients of the polynomials are determined. Then the trajectory of the robot is computed using the appropriate polynomial at any given time instant.

V. Experiment I : Point-to-Point Motion

For both the manipulators we have considered $l_1 = 0.3\,m, l_2 = 0.3\,m, l_3 = 0.3\,m$. Following assumptions were considered for the analysis:

- Path given does not interfere with the robot's body or environment.
- Given path is within the task space. Therefore, solutions exist for the given path and trajectory can be reliably calculated.
- For articulated manipulator, it will be used in elbow down configuration. This ensures that there will be only one unique IK solution.
- The path and the time constraints used will be same for both manipulators to ensure a fair comparison.

Following are the parameters defined for these experiments:

Start Time	$t_s = 0$
End Time	$t_f = 10$
End-Effector Position at t_s	$X_s = (+0.2, 0.01, 0.7)$
End-Effector Position at t_f	$X_f = (-0.2, 0.01, 0.5)$
End-Effector Velocity at t_s	$\dot{X}_s = (0, 0, 0)$
End-Effector Velocity at t_f	$\dot{X}_f = (0, 0, 0)$

A. Polynomial Interpolation

As the name suggests, we need to move from a starting point to ending point. The exact path it needs to take between the two points is not defined. Based on the constraints provided, trajectory interpolator generates the path and trajectory.

Using this interpolation method with constraints as described in section IV we get the following sets of equations.The plots for position, velocity and acceleration in task and joint space are also shown.

Articulated Manipulator - Joint Space: The plots for this are shown in in Fig. 6 and following are the equations used.

- $q_1(t) = -0.0061t^3 + 0.0913t^2 + 0.0500$
- $q_2(t) = -0.0013t^3 + 0.0202t^2 - 0.3772$
- $q_3(t) = +0.0014t^3 - 0.0210t^2 - 1.4589$

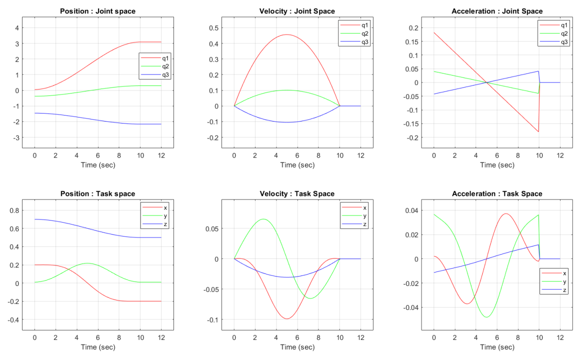


Fig. 6: Articulated Manipulator : Point-to-Point motion in Joint Space with cubic interpolation

Articulated Manipulator - Task Space: The plots for this are shown in in Fig. 7 and following are the equations used.

- $x(t) = +0.0008t^3 - 0.0120t^2 + 0.2000$
- $y(t) = +0.0100$
- $z(t) = +0.0004t^3 - 0.0060t^2 + 0.7000$

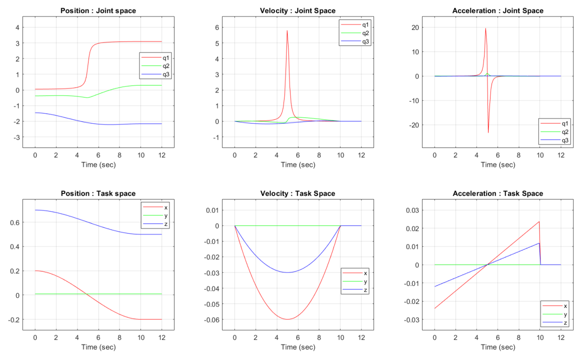


Fig. 7: Articulated Manipulator : Point-to-Point motion in Task Space with cubic interpolation

Cartesian Manipulator - Joint Space: The plots for this are shown in in Fig. 8 and following are the equations used.

- $q_1(t) = -0.0004t^3 - 0.0060t^2 + 0.4000$
- $q_2(t) = +0.0008t^3 - 0.0120t^2 - 0.1000$
- $q_3(t) = -0.3100$

Cartesian Manipulator - Task Space: The plots for this are shown in in Fig. 9 and following are the equations used.

- $x(t) = 0.0008t^3 - 0.0120t^2 + 0.2000$

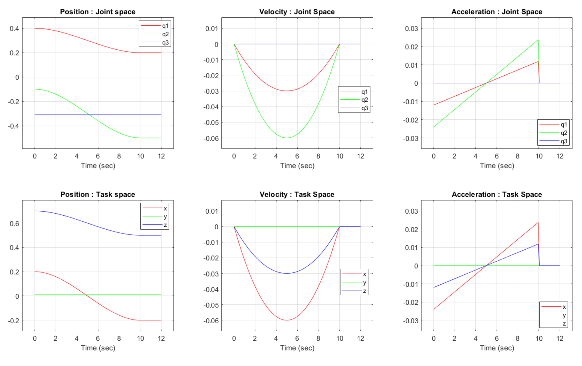


Fig. 8: Cartesian Manipulator : Point-to-Point motion in Joint Space with cubic interpolation

- $y(t) = 0.0100$
- $z(t) = 0.0004t^3 - 0.0060t^2 + 0.7000$

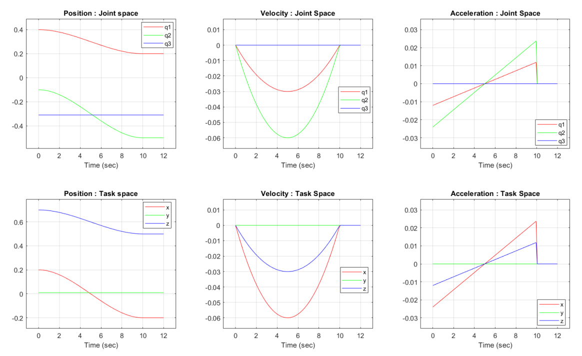


Fig. 9: Cartesian Manipulator : Point-to-Point motion in Task Space with cubic interpolation

B. *LSPB Interpolation*

In polynomial interpolation we were not able control the maximum velocities. LSPB interpolation lets us control that. Apart from the conditions in Polynomial Interpolation we use the following conditions. Max End-Effector Velocity $\dot{X}_m = (0.05, 0.05, 0.05)m/s$ and max Joint Velocity $\dot{q}_m = (0.5, 0.5, 0.5)rad/sec$ for articulated manipulator. For cartesian manipulator, max End-Effector Velocity $\dot{X}_m = (0.05, 0.05, 0.05)m/s$ and max Joint Velocity $\dot{q}_m = (0.05, 0.05, 0.05)m/s$

Using this interpolation method with constraints as described in section IV we get the following sets of equations.The plots for position, velocity and acceleration in task and joint space are also shown.

Articulated Manipulator - Joint Space: The plots for this are shown in in Fig. 10 and following are the equations used.

- $t_b = 3.9166$
- if $t <= t_b$
  - $q_1(t) = 0.0638t^2 + 0.05$
  - $q_2(t) = 0.0141t^2 - 0.3772$
  - $q_3(t) = -0.0147t^2 - 1.4589$
- if $t > t_b$ *and* $t < t_f - t_b$
  - $q_1(t) = q_1(t_b) + \dot{q}_1(t_b)\ (t - t_b)$
  - $q_2(t) = q_2(t_b) + \dot{q}_2(t_b)\ (t - t_b)$
  - $q_3(t) = q_3(t_b) + \dot{q}_3(t_b)\ (t - t_b)$
- if $t >= t_f - t_b$
  - $q_1(t) = -0.0638t^2 + 1.2766t - 3.2914$
  - $q_2(t) = -0.0141t^2 + 0.2820t - 1.1154$
  - $q_3(t) = 0.0147t^2 - 0.2939t - 0.6896$

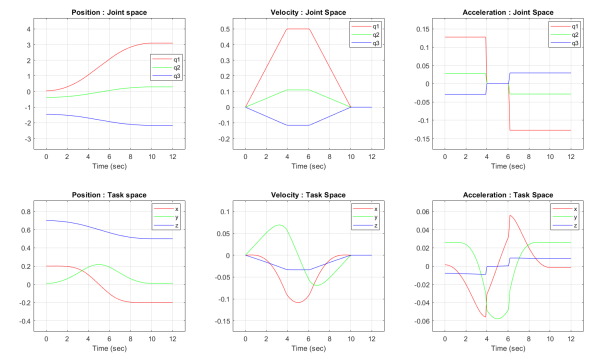


Fig. 10: Articulated Manipulator : Point-to-Point motion in Joint Space with LSPB interpolation

Articulated Manipulator - Task Space: The plots for this are shown in in Fig. 11 and following are the equations used.

- $t_b = 2$
- if $t <= t_b$
  - $x(t) = -0.0125t^2 + 0.2000$
  - $y(t) = 0.0100$
  - $z(t) = -0.0063t^2 + 0.7000$
- if $t > t_b$ *and* $t < t_f - t_b$
  - $x(t) = x(t_b) + \dot{x}(t_b)\ (t - t_b)$
  - $y(t) = y(t_b) + \dot{y}(t_b)\ (t - t_b)$
  - $z(t) = z(t_b) + \dot{z}(t_b)\ (t - t_b)$
- if $t >= t_f - t_b$
  - $x(t) = 0.0125t^2 - 0.2500t + 1.0500$
  - $y(t) = 0.0100$
  - $z(t) = 0.0063t^2 - 0.1250t + 1.1250$

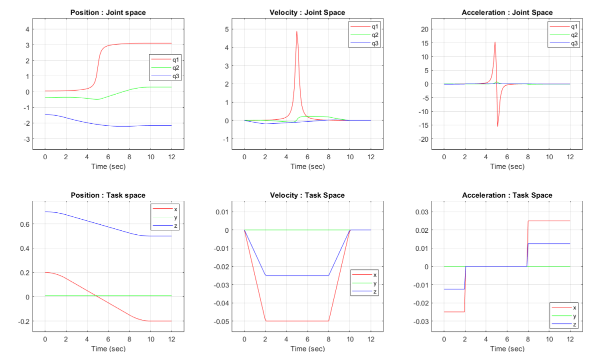


Fig. 11: Articulated Manipulator : Point-to-Point motion in Task Space with LSPB interpolation

Cartesian Manipulator - Joint Space: The plots for this are shown in in Fig. 12 and following are the equations used.

- $t_b = 2$
- if $t <= t_b$
  - $q_1(t) = -0.0063t^2 + 0.4000$
  - $q_2(t) = -0.0125t^2 - 0.1000$
  - $q_3(t) = -0.31$
- if $t > t_b$ *and* $t < t_f - t_b$
  - Same as in previous Joint space experiment
- if $t >= t_f - t_b$
  - $q_1(t) = 0.0063t^2 - 0.1250t + 0.8250$
  - $q_2(t) = 0.0125t^2 - 0.2500t + 0.7500$
  - $q_3(t) = -0.31$

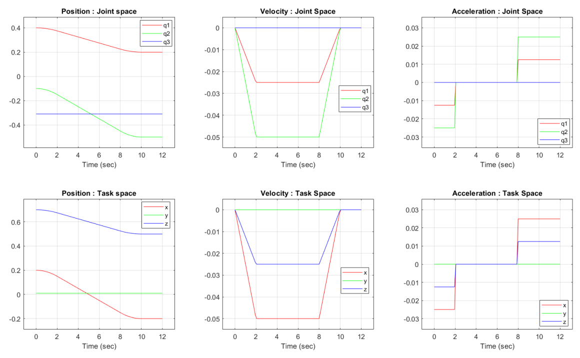


Fig. 12: Cartesian Manipulator : Point-to-Point motion in Joint Space with LSPB interpolation

Cartesian Manipulator - Task Space: The plots for this are shown in in Fig. 13 and following are the equations used.

- $t_b = 2$
- if $t <= t_b$
  - $x(t) = -0.0125t^2 + 0.2000$
  - $y(t) = 0.0100$
  - $z(t) = -0.0063t^2 + 0.7000$
- if $t > t_b$ *and* $t < t_f - t_b$
  - Same as in previous task space experiment
- if $t >= t_f - t_b$
  - $x(t) = 0.0125t^2 - 0.2500t + 1.0500$
  - $y(t) = 0.0100$
  - $z(t) = 0.0063t^2 - 0.1250t + 1.1250$

C. Inferences

Since the link lengths, trajectory time and end positions were kept same in all cases, a number of differences could be inferred-both visually and mathematically.

- The Cartesian manipulator's joint space and task space are similar since each of the 3 joints changes each of the 3 Cartesian dimensions. The joint space and task space trajectories' offset is determined by the link lengths which can be seen in the trajectories plotted-both for polynomial and LSPB. Articulated manipulators do not have this advantage of simplicity as each joint motion can result in change of all 3 Cartesian positions.

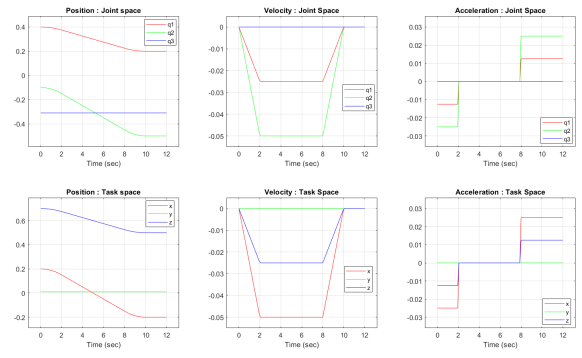


Fig. 13: Cartesian Manipulator : Point-to-Point motion in Task Space with LSPB interpolation

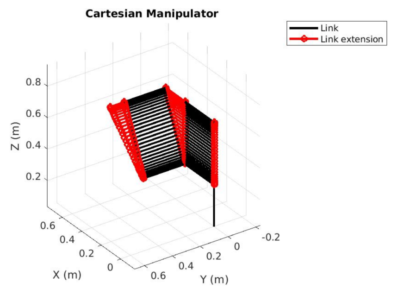


Fig. 14: Cartesian Manipulator: Trajectory Generation Joint Space and Task Space are identical (Note : For visual representation different start and end points have been used)

- A similarity in the executed trajectory is apparent from the stick models. The intensity of lines in a region is inversely proportional to the speed in the region.
- In the articulated manipulator, We can also notice the high joint velocity due to passing of end-effector near a singularity configuration in the task space as seen in Fig. 15. The intensity of lines decreases drastically during that motion.

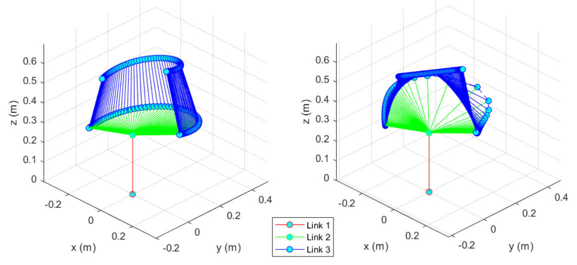


Fig. 15: Articulated Manipulator : Joint Space(Left) vs Task Space(Right) Point-to-Point polynomial trajectory generation

- In terms of the position profiles, there is no significant difference as this is just a point-to-point motion following a straight in their corresponding spaces

VI. Experiment II : Point-to-Point with Waypoints Motion

This is an extension of the previous experiment where its required to go from a start to end point by passing through one or more points. These additional points are called the waypoints. In the experiment, there are two waypoints. Following are the parameters defined for these experiments:

Start Time	$t_s = 0$
End Time	$t_f = 30$
End-Effector Position at t_s	$X_s = (+0.2, 0.01, 0.7)$
End-Effector Position at t_f	$X_f = (-0.2, 0.01, 0.5)$
End-Effector Velocity at t_s	$\dot{X}_s = (0, 0, 0)$
End-Effector Velocity at t_f	$\dot{X}_f = (0, 0, 0)$
Waypoint 1 at $t = 10$	$X_{w1} = (+0.4, 0.20, 0.5)$
Waypoint 2 at $t = 20$	$X_{w2} = (-0.2, 0.20, 0.5)$

A. *Polynomial Interpolation*

The approach followed is similar to the one used for point to point motion. The degree of polynomial increases due to additional constraints that needs to be satisfied. Previously there were four constraints i.e. Start, End Positions and Velocities. Now we have two more constraints which are the waypoint positions. The velocity at the waypoint is not specified as a constraint. Hence we use a 5^{th} degree polynomial for this purpose. In general, a polynomial of $n + 3$ degree is needed, where n is the number of waypoints.

The plots for the experiments with Articulated Manipulator in Joint Space, Task Space an Cartesian Manipulator in Joint/Task Space are shown in Fig. 16, Fig. 17 and Fig. 18 respectively. As in cartesian manipulators, both Joint and Task space are the same only one of them is shown. Also, for the sake of brevity, the equations are not shown.

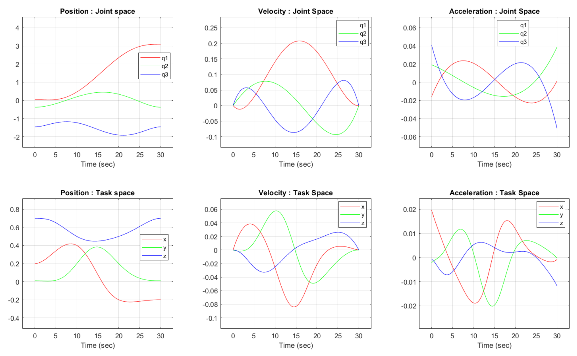


Fig. 16: Articulated Manipulator : Point-to-Point with waypoints motion in Joint Space with Polynomial interpolation

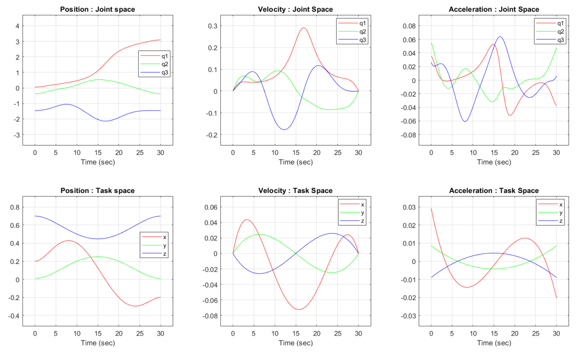


Fig. 17: Articulated Manipulator : Point-to-Point with waypoints motion in Task Space with Polynomial interpolation

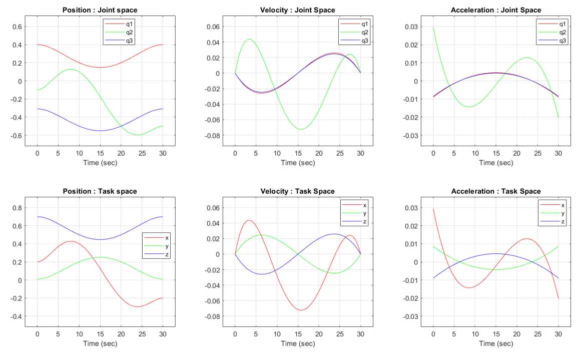


Fig. 18: Cartesian Manipulator : Point-to-Point with waypoints motion in Joint/Task Space with Polynomial interpolation

B. *LSPB Interpolation*

For LSPB interpolation we need to know two out of three parameters 'time between waypoints' , 'velocity desired' and 'acceleration desired'. In the point-to-point motion we used time and velocity, but here we use time and acceleration. In case of task space $acceleration_{max} = 0.025\ m/s^2$, and in joint space it has been set to $acceleration_{max} = 0.05\ rad/s^2$. All other parameters and constraints as same as the one used in the polynomial interpolation part.

The calculation process is similar to that as described in the point-to-point motion. The motion between each waypoint consists of three same three phases quadratic, linear and quadratic. We consider a linear motion between the waypoints and then based on the acceleration limit. smooth/blend the velocity change at a waypoint using quadratic functions. The exact mathematical calculations can be found in [2].

The plots for the experiments with Articulated Manipulator in Joint Space, Task Space an Cartesian Manipulator in Joint/Task Space are shown in Fig. 19, Fig. 20 and Fig. 21 respectively.

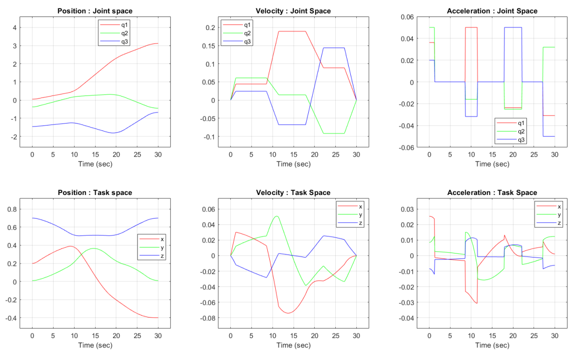


Fig. 19: Articulated Manipulator : Point-to-Point with waypoints motion in Joint Space with LSPB interpolation

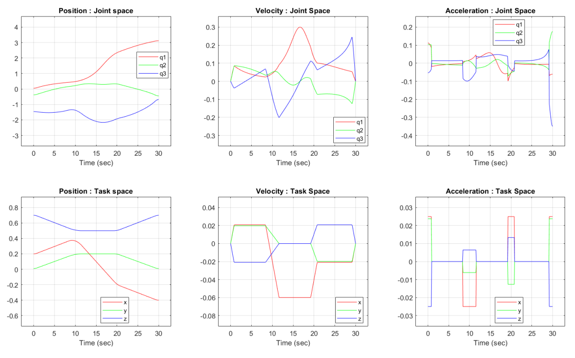


Fig. 20: Articulated Manipulator : Point-to-Point with waypoints motion in Task Space with LSPB interpolation

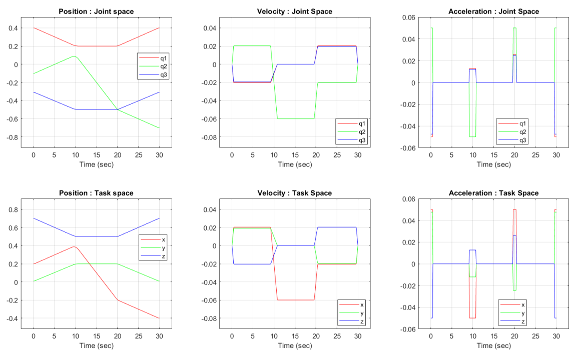


Fig. 21: Cartesian Manipulator : Point-to-Point with waypoints motion in Joint/Task Space with LSPB interpolation

C. *Inferences*

The addition of waypoints gave us more information on the behaviour of the two trajectory generation methods. Similar to the previous set of experiments the waypoints and timings were same throughout to ensure a fair comparison.

- Velocity profile in case of LSPB has sharp changes as compared to polynomial which is smooth.
- The nature of LSPB interpolation causes the trajectory to miss the waypoints due to the parabolic blends. This can be corrected by using fake waypoints, which would make the trajectory to pass though the actual waypoint.
- We can notice from Fig. 22 and Fig. 23 that the LSPB gives a straight line path in task space between waypoints, which is not the case with polynomial.
- With more waypoints, the degree of polynomial would increase the oscillations which is not desirable which might lead to undesirable velocities and accelerations. To reduce the oscillations we can use multiple lower order polynomials which are connected to each other also known as splines.
- Polynomial approach is mathematically simpler as compared to LSPB.

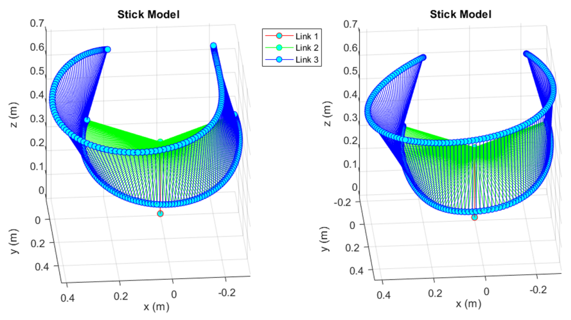


Fig. 22: Articulated Manipulator : Joint Space(Left) vs Task Space(Right) Point-to-Point with waypoints polynomial trajectory generation

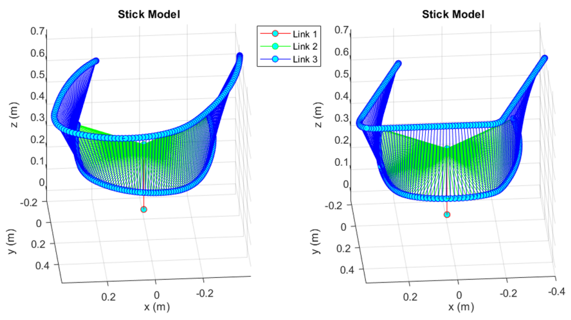


Fig. 23: Articulated Manipulator : Joint Space(Left) vs Task Space(Right) Point-to-Point with waypoints LSPB trajectory generation

VII. Experiment III : Stretch Goal Continuous Path for Articulated Manipulator

In this case, the end effector path is given as a continuous functions instead of points and waypoints. For experiment we are considering a circular path in the x-z place defined by the following equation.

$$x^2 + (z - 0.2)^2 = 0.1^2$$
$$y = 0.5$$

Our goal is to complete one revolution in 10 sec with initial and final velocities as 0. We cannot fit two functions to x and z separately as they are interdependent. Also, the circle equation being a one-to-many function does not help. Hence we convert it to a parametric form as shown below.

$$x = 0.1cos(\theta)$$
$$y = 0.5$$
$$z = 0.2 + 0.1sin(\theta)$$

With this parametric form our constraints become

- $\theta(t = 0) = 0$
- $\theta(t = 10) = 2\pi$
- $velocity(t = 0) = 0$
- $velocity(t = 10) = 0$

Velocity in case of a circular motion is $r\omega$. In this case it would be $r\frac{d\theta}{dt}$. With four constraints we can use a cubic polynomial for $\theta(t) = at^3 + bt^2 + ct + d$ and solve for the four coefficients. Fig. 24 show the various profiles for the articulated manipulator and Fig. 25 the motion of the arm. From the intensity of lines we can notice that it starts from a zero velocity and ends with zero velocity as required.

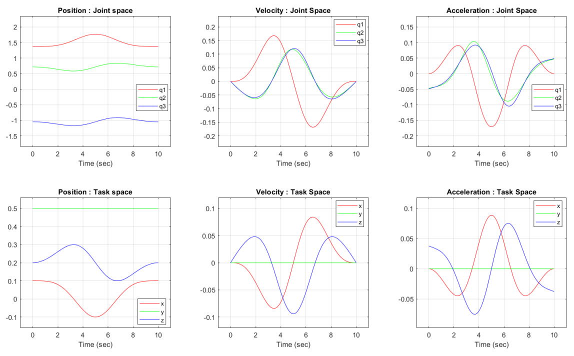


Fig. 24: Articulated Manipulator : Circular Path

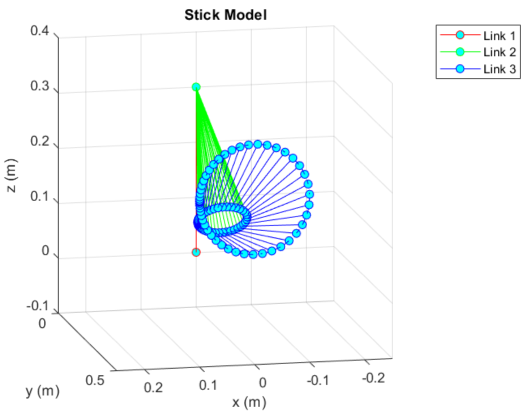


Fig. 25: Articulated Manipulator : Circular Path

VIII. Conclusion

Since the task space approach is concerned with trajectories of end effector, the individual joint trajectories do not follow any characteristic curve but they do so in task space. This is the same in LSPB. So the task space and joint space approaches can also be viewed as a trade-off between smooth end effector trajectory versus smooth joint trajectories. If a task requires smooth maneuvering, task space trajectory would be advantageous, whereas if a task involves heavy loading of joints with less regard for the way the trajectory is executed, joint space approach would be arguably better.

This argument can also be extended to Cartesian versus Articulated manipulators. As explained before and as seen in the executed trajectories, the joint space and task space are identical. This means that Cartesian manipulators can combine the best qualities of both approaches, the smooth end effector trajectory combined with smooth joint trajectories. This is perhaps the reason why they are suited in conditions where extreme precision is required like CNC Machines. Articulated manipulators are more flexible and have a high work-space volume to robot volume ratio making it useful in environments where dexterity is required.

With increasing number of waypoints LSPB performs better as using a higher degree of polynomial leads to highly oscillating trajectories. But LSPB trajectories, do not pass through the waypoint exactly due to the parabolic blends. Also, polynomial trajectories are mathematically easier to calculate. Though both have disadvantages, they can be improved. Using fake waypoints will make the LSPB trajectories pass through the actual waypoints and in case of polynomials, splines can be used to reduce the oscillations. LSPB can be used in scenarios where a constant velocity is required as that is the salient feature of it. In cases, where the trajectory needs to be smooth, polynomials or preferably splines can be used.

References

[1] B. Siciliano et al. Robotics (Modeling, planning and control), Springer, Berlin, 2009, chapter 4: Trajectory planning, pages 161-189.
[2] Saeed, Saad. (2018). Creating Through Points in Linear Function with Parabolic Blends Path by Optimization Method. Al-Khawarizmi Engineering Journal. 14. 10.22153/kej.2018.10.005.
[3] people.ciirc.cvut.cz/~hlavac/TeachPresEn/55AutonomRobotics/090RobotTrajectoryGenerationEn.pdf
[4] www.ent.mrt.ac.lk/~rohan/teaching/EN3562/LectureNotes/Lec%208%20Trajectory%20Planning.pdf
[5] www.cs.cmu.edu/~motionplanning/lecture/Chap3-Config-Space_howie.pdf
[6] ttuadvancedrobotics.wikidot.com/trajectory-planning-for-point-to-point-motion