Robotics and Mechatronics

Homework Three

Mohammad Montazeri
School of Mechanical Engineering
College of Engineering, University of Tehran
Tehran, Iran; 810699269
mohammadmontazeri@ut.ac.ir

***Abstract*—In this homework we mainly explore the vast field of robotics regarding robot Jacobians and Trajectories. These mathematical constructs play a crucial role in controlling robotic arms for precise movements. Our report covers the essentials of these two matters, including subjects like Jacobians, singularity, static analysis, and trajectory generation using MATLAB.**
***Index Terms*—Jacobian, trajectory, singularity, velocity, acceleration, interpolation, end-effector**

I. Introduction

In robotics, the Jacobian matrix bridges joint velocities (how fast each joint moves) to the velocity of the end effector (the robotic arm's tip). It helps us understand how changes in joint motion affect the overall movement of the arm. Jacobians find applications in controlling robotic arms during tasks like painting or precise movements. For instance, if the servo motors rotate at certain velocities, the Jacobian helps calculate how fast the end effector moves in both linear (x, y, z) and angular (roll, pitch, yaw) directions. Trajectory planning involves finding a time series of successive joint angles or end-effector positions that guide a robot from its starting configuration to a desired goal. It's like plotting the robot's path through space to achieve specific tasks, such as picking up an object and placing it on a shelf. As we delve into this fascinating realm, we'll explore how Jacobians enable precise control and discuss the importance, applications, methods and challenges of trajectory planning where motion becomes ambiguous.

II. Problem 1: Wrist Robot Jacobian

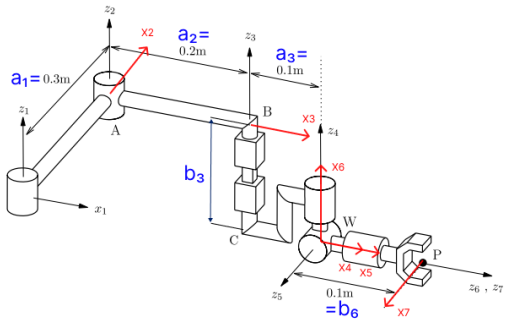


Fig. 1. Schematic view of 6-DOF robot with its D-H vectors

The velocity of end-effector, also known as its ***twist***, can be found from equation below:

$$\vec{t} = \begin{bmatrix} \vec{\omega} \\ \dot{\vec{P}} \end{bmatrix} = \underset{\sim}{J}\dot{\vec{\theta}} \tag{1}$$

Where,

- $\vec{t}$ is the twist of end-effector,
- $\vec{\omega}$ is the angular velocity of end-effector,
- $\dot{\vec{P}}$ is the linear velocity of end-effector,
- $\underset{\sim}{J}$ is the Jacobian matrix,
- $\dot{\vec{\theta}}$ is the joints' velocity.

Since the third joint is a prismatic one in this robot, to find its Jacobian at the current configuration, we use:

$$\underset{\sim}{\mathcal{J}} = \begin{bmatrix} \vec{e_1} & \vec{e_2} & \vec{O} & \vec{e_4} & \vec{e_5} & \vec{e_6} \\ \vec{e_1} \times \vec{r_1} & \vec{e_2} \times \vec{r_2} & \vec{e_3} & \vec{e_4} \times \vec{r_4} & \vec{e_5} \times \vec{r_5} & \vec{e_6} \times \vec{r_6} \end{bmatrix} \tag{2}$$

Where, in this problem, using the fixed coordinate origin of (x_1, y_1, z_1) in Figure 1 we have:

$$\vec{e_1} = \vec{e_2} = \vec{e_3} = \vec{e_4} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \vec{e_5} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \quad \vec{e_6} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\vec{r_1} = \begin{bmatrix} 0.4 \\ 0.3 \\ -b \end{bmatrix} \quad \vec{r_2} = \begin{bmatrix} 0.4 \\ 0 \\ -b \end{bmatrix} \quad \vec{r_3} = \begin{bmatrix} 0.2 \\ 0 \\ -b \end{bmatrix} \quad \vec{r_4} = \vec{r_5} = \vec{r_6} = \begin{bmatrix} 0.1 \\ 0 \\ 0 \end{bmatrix}$$

So, by doing the math, the Jacobian is computed as:

$$\underset{\sim}{\mathcal{J}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ -0.3 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.1 & 0 \end{bmatrix}$$

Now, having the given joint space velocities and the obtained Jacobian, we can find the end-effector velocity by equation 1.

$$\vec{t} = \begin{bmatrix} \vec{\omega} \\ \dot{\vec{P}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ -0.3 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 3 \\ -0.3 \\ 0.9 \\ 1.1 \end{bmatrix}$$

This means point P has an angular velocity of $\omega = [1, -1, 3]$ rad/s and a linear velocity of $v = [-0.3, 0.9, 1.1]$ m/s, with respect to fixed coordinate system of (x_1, y_1, z_1).

III. Problem 2: Jacobian and Singularity

The D-H parameters of this robot were obtained from the configurations marked up in Figure 2 which was first sketched in the previous homework. The Q_i and $\vec{a_i}$ parameters were also derived which will be used here to form the general Jacobian matrix. Here's the previous results:

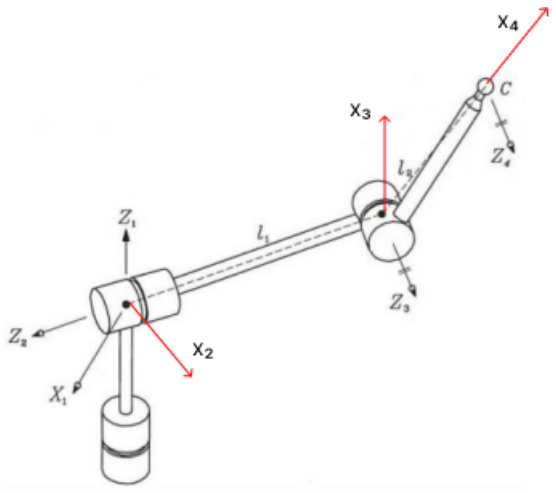


Fig. 2. Schematic view of robot marked up with D-H vectors

TABLE I
THE D-H PARAMETERS OF 3-DOF ROBOT

i	a_i	b_i	α_i	θ_i
1	0	0	$\pi/2$	θ_1
2	0	$-l_1$	$\pi/2$	θ_2
3	l_2	0	0	θ_3

$$[Q_1] = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \vec{\mathbf{a_1}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$[Q_2] = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ \sin\theta_2 & 0 & -\cos\theta_2 \\ 0 & 1 & 0 \end{bmatrix} \qquad \vec{\mathbf{a_2}} = \begin{bmatrix} 0 \\ 0 \\ -l_1 \end{bmatrix}$$

$$[Q_3] = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \vec{\mathbf{a_3}} = \begin{bmatrix} l_2\cos\theta_3 \\ l_2\sin\theta_3 \\ 0 \end{bmatrix}$$

$$[\vec{e_1}]_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad [\vec{e_2}]_1 = \underset{\sim}{Q}_1 [\vec{e_2}]_2 \qquad [\vec{e_3}]_1 = \underset{\sim}{Q}_1\underset{\sim}{Q}_2 [\vec{e_3}]_3 \quad (3)$$

$$[\vec{a_1}]_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad [\vec{a_2}]_1 = \underset{\sim}{Q}_1 [\vec{a_2}]_2 \qquad [\vec{a_3}]_1 = \underset{\sim}{Q}_1\underset{\sim}{Q}_2 [\vec{a_3}]_3 \quad (4)$$

$$[\vec{r_1}]_1 = [\vec{a_1}]_1 + [\vec{a_2}]_1 + [\vec{a_3}]_1 \qquad [\vec{r_2}]_1 = [\vec{a_2}]_1 + [\vec{a_3}]_1$$
$$[\vec{r_3}]_1 = [\vec{a_3}]_1 \quad (5)$$

Now we have all of the required equations; so, substituting gives:

$$\vec{e_1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \vec{e_2} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \sin\theta_1 \\ -\cos\theta_1 \\ 0 \end{bmatrix}$$

$$\vec{e_3} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ \sin\theta_2 & 0 & -\cos\theta_2 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1\sin\theta_2 \\ \sin\theta_1\sin\theta_2 \\ -\cos\theta_2 \end{bmatrix}$$

$$\vec{a_1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \vec{a_2} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -l_1 \end{bmatrix} = \begin{bmatrix} -l_1\sin\theta_1 \\ l_1\cos\theta_1 \\ 0 \end{bmatrix}$$

$$\vec{a_3} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ \sin\theta_2 & 0 & -\cos\theta_2 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} l_2\cos\theta_3 \\ l_2\sin\theta_3 \\ 0 \end{bmatrix} = \begin{bmatrix} l_2\cos\theta_3\cos\theta_1\cos\theta_2 + l_2\sin\theta_3\sin\theta_1 \\ l_2\cos\theta_3\sin\theta_1\cos\theta_2 - l_2\sin\theta_3\cos\theta_1 \\ l_2\cos\theta_3\sin\theta_2 \end{bmatrix}$$

So, putting in equation 5 gives:

$$\vec{r_1} = \vec{r_2} = \begin{bmatrix} l_2\cos\theta_3\cos\theta_1\cos\theta_2 + l_2\sin\theta_3\sin\theta_1 - l_1\sin\theta_1 \\ l_2\cos\theta_3\sin\theta_1\cos\theta_2 - l_2\sin\theta_3\cos\theta_1 + l_1\cos\theta_1 \\ l_2\cos\theta_3\sin\theta_2 \end{bmatrix}$$

$$\vec{r_3} = \begin{bmatrix} l_2\cos\theta_3\cos\theta_1\cos\theta_2 + l_2\sin\theta_3\sin\theta_1 \\ l_2\cos\theta_3\sin\theta_1\cos\theta_2 - l_2\sin\theta_3\cos\theta_1 \\ l_2\cos\theta_3\sin\theta_2 \end{bmatrix}$$

Finally, we can achieve the Jacobian matrix of this 3-DOF robot like:

$$\underset{\sim}{\mathcal{J}} = \begin{bmatrix} \vec{e_1} & \vec{e_2} & \vec{e_3} \\ \vec{e_1} \times \vec{r_1} & \vec{e_2} \times \vec{r_2} & \vec{e_3} \times \vec{r_3} \end{bmatrix} \quad (6)$$

Taking $l_1 = l_2 = 1$,

$$\rightarrow \underset{\sim}{J} = \begin{bmatrix} 0 & \sin\theta_1 & \cos\theta_1 \sin\theta_2 \\ 0 & -\cos\theta_1 & \sin\theta_1 \sin\theta_2 \\ 1 & 0 & -\cos\theta_2 \\ \sigma_1 & -\cos\theta_1 \cos\theta_3 \sin\theta_2 & \sigma_3 \\ \sigma_2 & -\cos\theta_3 \sin\theta_1 \sin\theta_2 & \sigma_4 \\ 0 & \cos\theta_2 \cos\theta_3 & -\sin\theta_2 \sin\theta_3 \end{bmatrix}$$

where

$$\begin{aligned} \sigma_1 &= \cos\theta_1 \sin\theta_3 - \cos\theta_1 - \cos\theta_2 \cos\theta_3 \sin\theta_1 \\ \sigma_2 &= \sin\theta_1 \sin\theta_3 - \sin\theta_1 + \cos\theta_1 \cos\theta_2 \cos\theta_3 \\ \sigma_3 &= \cos\theta_3 \sin\theta_1 - \cos\theta_1 \cos\theta_2 \sin\theta_3 \\ \sigma_4 &= -\cos\theta_1 \cos\theta_3 - \cos\theta_2 \sin\theta_1 \sin\theta_3 \end{aligned}$$

Generally, the spatial velocity vector has got 6 elements - 3 for translational velocity and 3 for angular velocity. That means the Jacobian matrix always has six rows. So, the robot manipulator Jacobian matrix is a 6 by N matrix, overall. But in this problem, the robot only has 3 joints and we call it an **under actuated robot**; that means its task space doesn't cover all of the set SE3. Although the end effector can achieve all positions within the limits of its work space, it's not able to achieve all possible **orientations**.

Since the obtained Jacobian is not square, it can't be inverted. Therefore, as discussed why, we must omit the ω term in end-effector *twist* and its corresponding rows in *Jacobian* matrix. This gives a square (3×3) matrix whose determinant can be found easily thereafter.

$$\rightarrow \underset{\sim}{J} = \begin{bmatrix} \sigma_1 & -\cos\theta_1 \cos\theta_3 \sin\theta_2 & \sigma_3 \\ \sigma_2 & -\cos\theta_3 \sin\theta_1 \sin\theta_2 & \sigma_4 \\ 0 & \cos\theta_2 \cos\theta_3 & -\sin\theta_2 \sin\theta_3 \end{bmatrix}$$
$$\rightarrow \left|\underset{\sim}{J}\right| = -\cos\theta_2 \cos^2\theta_3$$

To find the singularities of this robot, we need to find the values of $\theta_1, \theta_2,$ andθ_3 which make the determinant of Jacobian zero. This whole process is done by MATLAB to simplify the algebraic maths. This code is appended along this report and its final answers are:

$$|\mathcal{J}| = 0 \rightarrow \begin{cases} \theta_2 = \frac{\pi}{2}, \frac{3\pi}{2}, \cdots \rightarrow \begin{cases} \theta_2 = (2k+1)\frac{\pi}{2} \\ k \in \mathbb{Z} \end{cases} \\ \\ \theta_3 = \frac{\pi}{2}, \frac{3\pi}{2}, \cdots \rightarrow \begin{cases} \theta_2 = (2k+1)\frac{\pi}{2} \\ k \in \mathbb{Z} \end{cases} \end{cases}$$

IV. Problem 3: Static Analysis

A. Obtaining and Verifying Jacobian Matrix

The general form of Jacobian matrix for a 6-DOF wrist robot is derived in source book's [1] chapter 5.

$$\mathbf{t} = \mathbf{J}\dot{\theta} \tag{7}$$

For decoupled manipulators it is more convenient to deal with the velocity of the center *C* of the wrist than with that of the operation point *P*. Thus, one has

$$\mathbf{t}_c = \mathbf{J}\dot{\theta} \tag{8}$$

where the twist of wrist point (t_c) is defined as

$$\mathbf{t}_c = \begin{bmatrix} \omega \\ \dot{\mathbf{c}} \end{bmatrix} \tag{9}$$

and thus, the Jacobian takes on the following simple form

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_{11}} & \mathbf{J_{12}} \\ \mathbf{J_{21}} & O \end{bmatrix} \tag{10}$$

where *O* denotes the 3×3 zero matrix, the other 3×3 blocks being given below, *for manipulators with revolute pairs only*, as

$$\mathbf{J_{11}} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \tag{11}$$
$$\mathbf{J_{12}} = [\mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6] \tag{12}$$
$$\mathbf{J_{21}} = [\mathbf{e}_1 \times \mathbf{r}_1, \mathbf{e}_2 \times \mathbf{r}_2, \mathbf{e}_3 \times \mathbf{r}_3] \tag{13}$$

Further, vector $\dot{\theta}$ is partitioned accordingly:

$$\dot{\theta} = \begin{bmatrix} \dot{\theta}_a \\ \dot{\theta}_w \end{bmatrix} \tag{14}$$

where

$$\dot{\theta}_a = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \qquad \dot{\theta}_w = \begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \tag{15}$$

In this problem, we deal with the three links (joints) of the wrist robot which handle the **orientation** of the robot, not its **position**. Therefore, the second row of introduced Jacobian $[\mathbf{J_{21}}, \mathbf{O}]$ must be omitted, giving:

$$\begin{pmatrix} \omega \\ \dot{\mathbf{c}} \end{pmatrix} = \begin{pmatrix} \mathbf{J_{11}} & \mathbf{J_{12}} \\ \mathbf{J_{21}} & O \end{pmatrix} \begin{pmatrix} \dot{\theta}_a \\ \dot{\theta}_w \end{pmatrix} \tag{16}$$

In other words, we have

$$\omega = \mathbf{J_{11}}\dot{\theta}_a + \mathbf{J_{12}}\dot{\theta}_w \tag{17}$$

Since we only have the first three links, the $\mathbf{J_{12}}$ term can be canceled out and the result is:

$$[\omega] = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{18}$$

In this problem, the given Jacobian matrix is defined for the

2nd joint frame; thus, for better comparison of our matrix with the given one, we need to find the *Rotation Matrices* for the transfer between frames. That requires finding the D-H parameters of this robot. Using Figure 3, The D-H parameters can be derived as below, resulting in the Q_i and then $\vec{e_i}$ parameters following it:

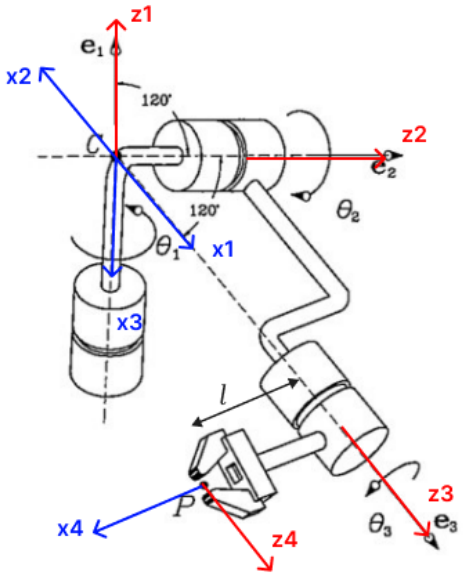


Fig. 3. Schematic view of the twist robot with its D-H vectors

TABLE II
THE D-H PARAMETERS OF 3-DOF WRIST ROBOT

i	a_i	b_i	α_i	θ_i
1	0	0	$2\pi/2$	θ_1
2	0	0	$2\pi/3$	θ_2
3	l	0	0	θ_3

$$[Q_1] = \begin{bmatrix} \cos\theta_1 & \frac{1}{2}\sin\theta_1 & \frac{\sqrt{3}}{2}\sin\theta_1 \\ \sin\theta_1 & -\frac{1}{2}\cos\theta_1 & -\frac{\sqrt{3}}{2}\cos\theta_1 \\ 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$$

$$[Q_2] = \begin{bmatrix} \cos\theta_2 & \frac{1}{2}\sin\theta_2 & \frac{\sqrt{3}}{2}\sin\theta_2 \\ \sin\theta_2 & -\frac{1}{2}\cos\theta_2 & -\frac{\sqrt{3}}{2}\cos\theta_2 \\ 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$$

$$[Q_3] = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So we can find $\vec{e_i}$ vectors and then, Jacobian matrix in the first joint frame, using equation 18.

$$J_1 = [\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}] \qquad \text{where,} \qquad [\mathbf{e}_1]_1 = [0,0,1]^T$$
$$[\mathbf{e}_2]_1 = [Q_1][0,0,1]^T \qquad [\mathbf{e}_3]_1 = [Q_1][Q_2][0,0,1]^T$$

$$\rightarrow J_1 = \begin{pmatrix} 0 & \frac{\sqrt{3}\sin t_1}{2} & \frac{\sqrt{3}\cos t_1 \sin t_2}{2} - \frac{\sqrt{3}\sin t_1}{4} - \frac{\sqrt{3}\cos t_2 \sin t_1}{4} \\ 0 & -\frac{\sqrt{3}\cos t_1}{2} & \frac{\sqrt{3}\cos t_1}{4} + \frac{\sqrt{3}\cos t_1 \cos t_2}{4} + \frac{\sqrt{3}\sin t_1 \sin t_2}{2} \\ 1 & -\frac{1}{2} & \frac{1}{4} - \frac{3\cos t_2}{4} \end{pmatrix}$$

We can also write

$$[J]_1 = Q_1[J]_2 \qquad \rightarrow \qquad [J]_2 = Q_1^T[J]_1 \tag{19}$$

So, transferring our attained Jacobian to the second joint frame and substituting $\theta_1 \,\&\, \theta_2$ with $0 \,\&\, \pi/2$ gives [1]:

$$[J]_2 = \begin{pmatrix} 0 & 0 & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{pmatrix}$$

comparing it with the normalized version of given Jacobian matrix shows that they are totally similar. Then it's proved that the given Matrix is, indeed, the Jacobian of this robot.

B. Calculating Force

Using equation 5.50 0f the textbook [1], we have:

$$\mathbf{J}^T\mathbf{W} = \boldsymbol{\tau} \tag{20}$$
$$\rightarrow \mathbf{W} = \mathbf{J}^{-T}\boldsymbol{\tau} \tag{21}$$

where $\boldsymbol{\tau}$ is the 3-dimensional vector of joint forces and torques, whose i-th component is τ_i, whereas $\mathbf{W} = [\mathbf{n}^T, \mathbf{f}^T]$ denotes the wrench exerted by the environment on the EE, with $\mathbf{n}$ denoting the resultant moment and $\mathbf{f}$ the resultant force applied at point P of the end-effector of the manipulator on Figure 3. So,

$$\begin{bmatrix} \mathbf{n} \\ \mathbf{f} \end{bmatrix} = \begin{pmatrix} 0 & 0 & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{pmatrix}^{-T} \begin{bmatrix} 20 \\ 20 \\ -10 \end{bmatrix} = \begin{bmatrix} 0 \\ 20\sqrt{3} \\ 20 \end{bmatrix}$$
$$\rightarrow \mathbf{n} = \begin{bmatrix} 0 \\ 34.641 \\ 20 \end{bmatrix}$$

V. PROBLEM 4: CONCEPTUAL ANALYSIS

A. Applications of Trajectory Planning

1) **Autonomous Vehicles (Self-Driving Cars)**: For autonomous vehicles to navigate safely and efficiently, trajectory planning is essential. It involves generating paths that avoid obstacles, follow traffic rules, and optimize fuel efficiency.
   - *Role and Purpose*: To ensure smooth and collision-free movement, allowing the vehicle to reach its destination while adhering to predefined constraints.

[1]This result is achieved using MATLAB code. The code is also attached to this report file.

- *Importance and Impact*: Proper trajectory planning avoids accidents, reduces travel time, improves passenger comfort, and enhances safety, especially in complex scenarios like lane changes, merging, and parking. Lack of planning means choosing simplest and fastest controlling routs which usually could lead to abrupt maneuvers, endangering passengers and other road users.
- *Challenges*: Autonomous vehicles operate in dynamic environments with uncertainties (like sudden obstacles, road conditions, etc), requiring rapid trajectory adjustments. Real-time computation is challenging due to the need for quick decisions.

2) **Industrial Robots and Manufacturing**: In manufacturing environments, robots perform tasks like assembling, welding, painting, and material handling. Trajectory planning ensures smooth and accurate movements.
   - *Role and Purpose*: To ensure precise and efficient movements of industrial robots in manufacturing processes, avoiding collisions, optimizing cycle time, and maintaining product quality.
   - *Importance and Impact*: Optimized trajectories reduce production time, improve product quality, and minimize wear on robot components, while preventing collisions with workpieces, fixtures, or other robots. Without proper planning, robots may collide with objects, leading to damage and waste of energy.
   - *Challenges*: Industrial robots operate within confined spaces. Trajectory planning must consider workspace limitations. Avoiding collisions with static and moving obstacles (like other robots, conveyors, etc) is critical too. Robots can encounter singular configurations where certain joints lose motion capability. Handling these situations is challenging as well.
3) **Aerial Drones and Surveillance**: Drones rely on trajectory planning to navigate through airspace, capture images, and perform surveillance.
   - *Role and Purpose*: To avoid obstacles, maintain stable flight, and execute complex maneuvers (like search patterns, tracking moving targets, etc).
   - *Importance and Impact*: Precise trajectories enhance image quality during surveillance missions. Quadcopters are also sometimes used for irrigation, manuring, or spraying pesticides; to cover the entire agricultural land, a thorough trajectory must be designed to minimize the costs and energy.
   - *Challenges*: Balancing trajectory efficiency with battery life is challenging. Detecting and avoiding dynamic obstacles (like birds, other drones, etc) and adapting to the ever-changing climate and wind requires robust algorithms.

B. Trajectory Planning Methods

- **Autonomous Vehicles (Self-Driving Cars)**
  1) *Polynomial Trajectory Planning*: To follow a smooth path while driving a car, this method can be used, enhancing the passenger comfort and avoiding sudden accelerations.
  2) *Point-to-Point Trajectory Planning*: To reach from a starting point to the destination point on the road, the fastest way, this method might be used. Since it causes sudden steering and rapid speed changes, it's not appropriate for on-road travels. It might be of use in special cases like race cars or military vehicles.
- **Industrial Robots and Manufacturing**
  1) *Minimum Jerk Trajectory*: This method must be used for applications where not only must no force be applied at the beginning or ending of a robot's motion, but also there must be no jerk at these moments.
  2) *Cycloidal Motion*: This method is quite much same as *Polynomial Interpolation* methods; meaning it has zero velocity and acceleration (and thus, force) at the start and end of the motion. But it applies high jerks at the start and stop moments. So it is a simple and practical method for applications with less sensitivity and affectivity.
- **Aerial Drones and Surveillance**
  1) *Trapezoidal Velocity Profile*: This method creates a simple velocity profile which starts at zero, reaches a respectively long constant maximum and ends at zero. This creates a respectively smooth motion, although not as much as what 3-4-5 or 4-7 polynomials do.
  2) *Rectangular Velocity Profile*: The simplest way to plan a motion is to hold the required speed from beginning to the end, with no alternation. This is a not practical motion for most of the applications, since they need a smooth and jerk-less motion to avoid harming the objects or the robot itself. However, in military applications like armed drones, this method might be of use, since it's able to perform the fastest motions and satisfy the rapid maneuvers expected from a military vehicle. Since these drones are equipped with highly durable materials and devices, this method can cause no harm to their actuators and systems [7].

C. Optimization and Metrics

I) Trajectory Optimization Methods:

1) **Model Predictive Path Integral (MPPI)**:
   - **Overview**: MPPI combines path integral control with model predictive control (MPC).
   - **Working Principle**: MPPI leverages path integral control theory to compute solutions for stochastic optimal control. It optimizes a finite-horizon trajectory at each time step, sampling trajectories and evaluating them using a cost function. MPPI

adapts to changing environments and disturbances, providing real-time solutions.

- **Impacts**:
  - **Smooth Trajectories**: MPPI minimizes jerk (rate of acceleration change), resulting in smoother motion profiles.
  - **Adaptability**: It handles dynamic environments effectively.
  - **Real-Time Performance**: MPPI provides timely trajectory adjustments [3].

2) **Optimal Trajectory Generation with Jerk Minimization**:

- **Objective**: Minimize jerk along the trajectory.
- **How It Works**:
  - **Cost Function**: Formulate a cost function that penalizes jerk.
  - **Optimization**: Solve an optimization problem to find a trajectory with minimized jerk.
  - **Constraints**: Consider collision avoidance, velocity limits, and dynamic feasibility.
- **Impacts**:
  - **Smooth Motion**: Reduced jerk leads to smoother profiles.
  - **Energy Efficiency**: Lower jerk reduces energy consumption.
  - **Comfort**: Enhances passenger comfort.
  - **Safety**: Avoids sudden acceleration changes [4].

II) Metrics for Evaluating Trajectory Performance:

- **Smoothness**:
  - **Jerk**: Evaluate acceleration change rate. Smoother trajectories have lower jerk.
  - **Curvature**: Assess directional changes. Smaller curvature indicates smoother paths.
- **Efficiency**:
  - **Energy Consumption**: Measure energy required to follow the trajectory.
  - **Time**: Consider execution time; faster trajectories are more efficient [6].
- **Safety**:
  - **Collision Avoidance**: Ensure trajectory avoids obstacles.
  - **Dynamic Feasibility**: Adherence to system dynamics and constraints.
- **Accuracy**:
  - **Goal Achievement**: Evaluate how well the trajectory reaches desired goals.
  - **Tracking Error**: Measure deviation from the desired path.
- **Robustness**:
  - **Sensitivity Analysis**: Assess sensitivity to disturbances or uncertainties.
  - **Adaptability**: Evaluate handling of unexpected changes.

VI. Problem 5: Trajectory Generation

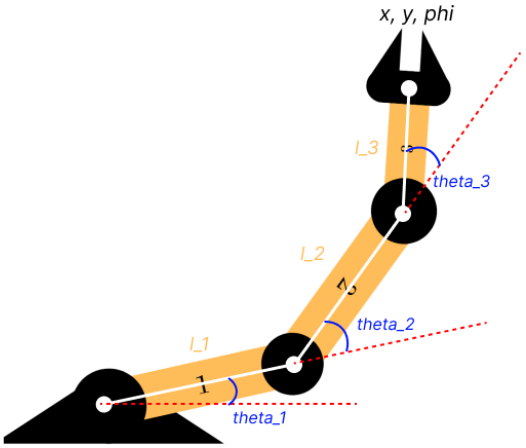


Fig. 4. Planar 3-DOF robot with its angles marked up

A. Forward and inverse kinematics

FKP Equations: Having θ_1, θ_2 and θ_3, we want to find x, y and ϕ of end-effector.

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad (22)$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad (23)$$

$$\phi = \theta_1 + \theta_2 + \theta_3 \quad (24)$$

IKP Equations: Having x, y and ϕ of end-effector, we want to find θ_1, θ_2 and θ_3.

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos\phi$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin\phi$$

$$\rightarrow (x - l_1 \cos\theta_1 - l_3 \cos\phi)^2 + (y - l_1 \sin\theta_1 - l_3 \sin\phi)^2 = l_2^2$$

$$A \cos\theta_1 + B \sin\theta_1 = C \quad (25)$$

$$A = 2l_1(x - l_3 \cos\phi) \quad (26)$$

$$B = 2l_1(y - l_3 \sin\phi) \quad (27)$$

$$C = x^2 + y^2 + l_1^2 - l_2^2 + l_3^2 - 2l_3(x \cos\phi + y \sin\phi) \quad (28)$$

$$Answers \longrightarrow \begin{cases} \theta_1^+ \longrightarrow \theta_2^+ \longrightarrow \theta_3^+ \\ \\ \theta_1^- \longrightarrow \theta_2^- \longrightarrow \theta_3^- \end{cases}$$

B. Angular pose, velocity and acceleration equations

General form of equations for angle, angular velocity, acceleration and jerk comes from:

$$\vec{\theta}(t) = \vec{\theta}_i + \left(\vec{\theta}_f - \vec{\theta}_i\right) s(\tau) \tag{29}$$

$$\dot{\vec{\theta}}(t) = \frac{1}{T}\left(\vec{\theta}_f - \vec{\theta}_i\right) s'(\tau) \tag{30}$$

$$\ddot{\vec{\theta}}(t) = \frac{1}{T^2}\left(\vec{\theta}_f - \vec{\theta}_i\right) s''(\tau) \tag{31}$$

$$\dddot{\vec{\theta}}(t) = \frac{1}{T^3}\left(\vec{\theta}_f - \vec{\theta}_i\right) s'''(\tau) \tag{32}$$

where, for each of the trajectory planning methods, the linear slope function $s(\tau)$ is derived according to the course pamphlet as below.

*1) **Trapezoidal Polynomial**:*

- conditions: $s(0) = s'(0) = 0; s(1) = 1, s'(1) = 0$
- maximum order of linear polynomial: 3
- achieved polynomial: $\underline{\mathbf{s}(\tau) = -\mathbf{2}\tau^{\mathbf{3}} + \mathbf{3}\tau^{\mathbf{2}}}$

*2) **3-4-5 Interpolating Polynomial**:*

- conditions: $s(0) = s'(0) = s''(0) = 0; s(1) = 1, s'(1) = s''(1) = 0$
- maximum order of linear polynomial: 5
- achieved polynomial: $\underline{\mathbf{s}(\tau) = \mathbf{6}\tau^{\mathbf{5}} - \mathbf{15}\tau^{\mathbf{4}} + \mathbf{10}\tau^{\mathbf{3}}}$

*3) **4-5-6-7 Interpolating Polynomial**:*

- conditions: $s(0) = s'(0) = s''(0) = s'''(0) = 0; s(1) = 1, s'(1) = s''(1) = s'''(0) = 0$
- maximum order of linear polynomial: 7
- achieved polynomial: $\underline{\mathbf{s}(\tau) = -\mathbf{20}\tau^{\mathbf{7}} + \mathbf{70}\tau^{\mathbf{6}} - \mathbf{84}\tau^{\mathbf{5}} + \mathbf{35}\tau^{\mathbf{4}}}$

*4) **Polynomial-Based 3-point Trajectory**:* Assuming the robot is to meet the intermediate point *B* at instant: $t = 1(s) \rightarrow \tau = 1/2$

- conditions: $s(0) = s'(0) = s''(0) = 0; s(0.5) = S; s(1) = 1, s'(1) = s''(1) = 0$ *where we can find **S** from:* $\theta(0.5) = \theta^I + \left(\theta^F - \theta^I\right) S$
- maximum order of linear polynomial: 6
- achieved polynomial: $\underline{\mathbf{s}(\tau) = \mathbf{27.13}\tau^{\mathbf{6}} - \mathbf{75.40}\tau^{\mathbf{5}} + \mathbf{66.40}\tau^{\mathbf{4}} - \mathbf{17.13}\tau^{\mathbf{3}}}$

Note that for the last method, to meet all of the three required points, we need a somehow different prospect. The polynomial trajectories discussed for the first three methods do not allow the specification of intermediate Cartesian poses of the EE. All they guarantee is that the Cartesian trajectories prescribed at the initial and final instants are met [1]. One way of verifying the feasibility of the Cartesian trajectories is with what are called **via poses**, i.e., poses of the EE in the Cartesian space that lie between the initial and the final poses, and are determined so as to avoid collisions [8]. Using *via pose*, requires finding the joint space of the robot to access the intermediate point (B in this case), via IKP. The following shows how this is done, using equation 25 and its relative sub-equations.

pt.B : $\quad x_{EE} = 1.0774L, \quad y_{EE} = 0.4673L, \quad \phi_{EE} = 30^\circ$

$$l_1 = l_2 = \frac{3}{2} l_3 = L \rightarrow$$

$$A = 2L(1.0774L - \frac{2}{3}L \times \frac{\sqrt{3}}{2}) = L^2$$

$$B = 2L(0.4673L - \frac{2}{3}L \times \frac{1}{2}) = 0.2679L^2$$

$$C = (1.0774L)^2 + (0.4673L)^2 + (\frac{2}{3}L)^2 - 2 \cdot \frac{2}{3}L \cdot (1.0774L\frac{\sqrt{3}}{2} + 0.4673L\frac{1}{2}) = 0.2679L^2$$

$$\rightarrow \cos(\theta_1) + 0.2679 \sin(\theta_1) = 0.2679$$

$$\rightarrow \theta_1^+ = 1.5708rad = 90^\circ \qquad \theta_1^- = -1.0473rad = -60^\circ$$
$$\rightarrow \theta_2^+ = -2.6179rad = -150^\circ \qquad \theta_2^- = 2.6179rad = 150^\circ$$
$$\rightarrow \theta_3^+ = 1.5707rad = 90^\circ \qquad \theta_3^- = -1.0471rad = -60^\circ$$

---

pt.A : $\quad x_{EE} = 1.3660L, \quad y_{EE} = 2.0327L, \quad \phi_{EE} = 90^\circ$

$$A = 1.3987L^2 \quad B = 2.7321L^2 \quad C = 3.7320L^2$$

$$\rightarrow 1.3987\cos(\theta_1) + 2.7321 \sin(\theta_1) = 3.7320$$

$$\rightarrow \theta_1^+ = 83.14^\circ \qquad \theta_1^- = -8.96^\circ$$
$$\rightarrow \theta_2^+ = -66.47^\circ \qquad \theta_2^- = 85.0^\circ$$
$$\rightarrow \theta_3^+ = 73.33^\circ \qquad \theta_3^- = 13.96^\circ$$

---

pt.C : $\quad x_{EE} = -0.6994L, \quad y_{EE} = 1.9434L, \quad \phi_{EE} = 120^\circ$

$$A = -0.3988L^2 \quad B = 2.1547L^2 \quad C = 0.6449L^2$$

$$\rightarrow -0.3988\cos(\theta_1) + 2.1547 \sin(\theta_1) = 0.6449$$

$$\rightarrow \theta_1^+ = 173.37^\circ \qquad \theta_1^- = 27.60^\circ$$
$$\rightarrow \theta_2^+ = -110.00^\circ \qquad \theta_2^- = 116.61^\circ$$
$$\rightarrow \theta_3^+ = 56.64^\circ \qquad \theta_3^- = -24.21^\circ$$

According to the source book [1], *Chapter 6.5, p267-269*, for one intermediate point, *B*, we must consider one joint, for example first joint (θ_1), and use its value, (*attained from IKP*), in a scalar version of equation 29. So,

$$@t = 1(s) \rightarrow \tau = 1/2 \rightarrow s(1/2) = S$$
$$\theta(1/2) = \theta^I + \left(\theta^F - \theta^I\right) S$$
$$\theta(1/2) = \theta_1^B \qquad \theta^I = \theta_1^A \qquad \theta^F = \theta_1^C \rightarrow$$
$$90 = 83.14 + (173.37 - 83.14)s(\tau = 1/2)$$

This adds a new constraint to the $s(\tau)$ which is: $s(\tau = 0.5) = S$. We already had 6 other conditions as we had in *3-4-5 Polynomial* method, which were: $s(0) = s'(0) = s''(0) = 0; s(1) = 1, s'(1) = s''(1) = 0$. Overall, there are 7 conditions our Interpolating Polynomial must satisfy, so it can be a linear polynomial of maximum order of 6. So,

$$s(\tau) = a\tau^6 + b\tau^5 + c\tau^4 + d\tau^3 + e\tau^2 + f\tau + g$$

Applying these conditions gives:

$$g = f = e = 0; \qquad a + b + c + d = 1;$$
$$6a + 5b + 4c + 3d = 0; \qquad 30a + 20b + 12c + 6d = 0;$$
$$a(\frac{1}{2})^6 + b(\frac{1}{2})^5 + c(\frac{1}{2})^4 + d(\frac{1}{2})^3 = 0.0760279;$$

Solving this system of linear equations gives:

$$a = 27.13 \quad b = -75.40 \quad c = 66.40 \quad d = -17.13$$
$$\mathbf{s(\tau) = 27.13\,\tau^6 - 75.40\,\tau^5 + 66.40\,\tau^4 - 17.13\,\tau^3}$$

C. Plotting the velocity profiles

In this section, MATLAB is used to plot multiple diagrams. First, $s(\tau)$ and its derivatives are plotted for each of the 4 methods discussed. The first derivative indicates the *velocity* profile, the second derivative indicates the *acceleration* profile and the third one, corresponds to *jerk* parameter. Note that the first method, *Trapezoidal* profile, has two figures: first one models a trapezoid with a Polynomial of third degree, while the second one creates a precise sharp trapezoid, using *heaviside* function.[2]

The **Heaviside step function**, denoted as $H(x)$, is a mathematical function. It behaves like a switch, returning different values based on its input:

- For $x < 0$, $H(x)$ is 0.
- For $x = 0$, $H(x)$ is $\frac{1}{2}$.
- For $x > 0$, $H(x)$ is 1.

Using heaviside function, we can create a trapezoidal shape for velocity relation like: *(S = maximum speed)*

$$s'(\tau) = \begin{cases} S \cdot \tau & 0 < \tau < \frac{1}{3} \\ S & \frac{1}{3} < \tau < \frac{2}{3} \\ -S \cdot \tau & \frac{1}{3} < \tau < 1 \end{cases}$$

The results are exactly as expected, satisfying the velocity, acceleration and jerk values at the beginning and ending of each trajectory. After that, the joint space information is thoroughly plotted using θ_is and their derivatives for each method. Again the Trapezoidal Method is divided to *sharp* and *soft* methods. The results are represented in Figures 5 to 10 in the Appendix.

[2]Assuming the trajectory is divided to three equal sections: accelerating to maximum speed of $S = 1rad/s$, keeping the constant speed and decelerating to stop at the end point of the path.

D. Comparison of velocity profiles

1) 3-4-5 Polynomial:
   - This trajectory method uses a polynomial curve with coefficients corresponding to the degrees 3, 4, and 5.
   - The velocity profile exhibits smooth transitions, but it may not handle high speeds well.
   - The maximum velocity during the rise period is moderate1.
2) 4-5-6-7 Polynomial:
   - This method extends the polynomial curve to include coefficients for degrees 4, 5, 6, and 7.
   - It achieves zero jerk (rate of change of acceleration) and finite fourth-order characteristics.
   - However, note that the maximum velocity and acceleration within the rise period have increased compared to the 3-4-5 polynomial12.
3) Trapezoidal Method:
   - The trapezoidal method combines constant acceleration and deceleration.
   - It ensures zero velocity at both ends of the rise period.
   - The resulting motion curve consists of two parabolas, providing smoother transitions.
   - However, the maximum acceleration remains discontinuous at the start and end of the rise portion1.
4) Via Pose Method:
   - The via pose method involves specifying intermediate poses (waypoints) for the robot.
   - It constructs a trajectory that passes through these waypoints.
   - While it allows flexibility in defining desired poses, the velocity profile depends on the specific waypoints chosen13. [9]

E. Practicality of last method

As discussed in the classroom, the way we handled the last method in order to cross a mid-point in the trajectory is straight-forward, but very limited. In fact, this method uses system of linear equations that might seem easy for few mid-points on the trajectory, but as this number increases, the complexity and volume of computations get exponentially huge and huger. So no driver or PLC would be able to handle it for real-world multi-point applications. That's why so many have been introducing via-pose methods to ease this process. These methods are beyond the syllabus of this course and this homework report.

VII. Problem 6: MATLAB Coding

In this problem, pre-written MATLAB commands are used to generate different trajectories for passing through provided path points. To use these functions, some additional libraries must have been installed on MATLAB like *Robotics System Toolbox*. [10]

For installing the mentioned toolbox, the following procedure has been followed.

A. Installation from shared MATLAB Drive

This will work for MATLAB Online or MATLAB Desktop provided you have MATLAB drive setup.

- Click on the appropriate link below and an invitation to share will be emailed to the address associated with your MATLAB account:
  - RVC 2nd edition RTB10+MVTB4 (2017)
  - RVC 1st edition: RTB9+MVTB3 (2011)
- Accept the invitation.
- A folder named RVC1 or RVC2 will appear in your MATLAB drive folder.
- Use the MATLAB file browser and navigate to the folder RVCx/rvctools and double-click the script named startup_rvc.m
- Note that this is a combo-installation that includes the Machine Vision Toolbox (MVTB) as well.
- Type `rtbdemo` in command prompt to see if the library has been successfully installed and is running well. [11]

B. Implementation of programs

Two series of codes are provided for this problem. First one is a thorough unit code named prob6.m which includes the whole program. The other is a combination of *Matlab Live Script* files[3] which provide the implementation of each of the required methods, separately and in a different way, requiring correct installation of mentioned toolboxes which might not be accessible in some devices. Both series are appended along this report. The results can be found in Figures 11 to 14 in the Appendix.

C. Discussion on peak velocities and accelerations

Peak velocity and acceleration are key parameters in trajectory planning methods, influencing the efficiency, stability, and safety of motion profiles. In the Multi-Point Trapezoidal method, peak velocity represents the maximum speed attained between waypoints, occurring at the midpoint of each segment, while peak acceleration denotes the highest rate of change of velocity, typically at the segment boundaries. These parameters help ensure smooth transitions and adherence to system constraints. In Multi-Point Cubic Polynomials, peak velocity and acceleration similarly characterize maximum speed and rate of change of velocity, but with smoother trajectories due to cubic polynomial interpolation. Quintic polynomials further refine trajectory smoothness, potentially lowering peak velocity and acceleration while maintaining precision. Understanding and optimizing peak velocity and acceleration enable engineers to design motion profiles that meet performance requirements, minimize stress on mechanical components, and ensure stable, efficient motion.

[3]suffixed as .mlx file

D. Discussion on trajectory planning methods

The first three methods of trajectory planning —*Multi-Point Trapezoidal*, *Multi-Point Cubic Polynomials*, and *Multi-Point Quintic Polynomials*— differ primarily in their mathematical representation of the trajectory and the resulting characteristics of the generated paths. Let's discuss each method's key features:

1) **Multi-Point Trapezoidal:**
   - **Description:** This method divides the trajectory into segments with trapezoidal velocity profiles. Each segment consists of a ramp-up, constant velocity, and ramp-down phase, resembling the shape of a trapezoid.
   - **Mathematical Representation:** The trajectory is represented by a series of straight-line segments with constant acceleration and deceleration.
   - **Characteristics:**
     - Simple to implement and computationally efficient.
     - Generates trajectories with constant acceleration and deceleration, resulting in smooth transitions between waypoints.
     - Suitable for applications where simplicity and predictable motion are preferred over higher-order path planning.
2) **Multi-Point Cubic Polynomials:**
   - **Description:** This method generates trajectories using cubic polynomial functions. These functions smoothly interpolate between waypoints by fitting a cubic polynomial to each segment of the trajectory.
   - **Mathematical Representation:** The trajectory is represented by a set of cubic polynomial functions, where each function defines a segment between consecutive waypoints.
   - **Characteristics:**
     - Provides more flexibility in shaping the trajectory compared to trapezoidal profiles.
     - Produces trajectories with continuous velocity and acceleration profiles, resulting in smoother motion.
     - Requires solving systems of linear equations to compute polynomial coefficients, which may introduce computational overhead compared to trapezoidal profiles.
3) **Multi-Point Quintic Polynomials:**
   - **Description:** This method extends the cubic polynomial approach by using quintic (degree-five) polynomial functions to generate trajectories. Quintic polynomials offer additional degrees of freedom, allowing for even smoother path transitions and better handling of constraints.
   - **Mathematical Representation:** The trajectory is represented by a set of quintic polynomial functions, providing higher-order continuity between waypoints.

- **Characteristics:**
  - Offers superior smoothness and continuity compared to cubic polynomials, particularly in terms of higher-order derivatives such as jerk.
  - Enables more precise control over trajectory shaping and adherence to constraints such as velocity and acceleration limits.
  - Requires solving more complex systems of equations compared to cubic polynomials, potentially leading to increased computational complexity.

E. Discussion on differences of methods

The primary distinction between the B-spline method and the other three methods (Multi-Point Trapezoidal, Multi-Point Cubic Polynomials, and Multi-Point Quintic Polynomials) lies in their trajectory representation. While the former employs piecewise polynomial curves of higher degrees for interpolation, the latter methods use simpler functions like straight lines or lower-degree polynomials. This difference impacts robot performance by offering B-splines smoother paths and better adaptability to complex constraints and environments. However, B-splines entail higher computational complexity, potentially affecting real-time performance in dynamic settings. Therefore, the choice depends on balancing the need for trajectory flexibility and smoothness with computational resources and real-time constraints in robot applications.

F. Discussion on use cases of methods

Multi-Point Trapezoidal trajectories are ideal for operations needing efficient and predictable motion, like manufacturing assembly lines, due to their simplicity and computational efficiency, ensuring rapid execution while maintaining stability. Conversely, B-spline trajectories suit tasks requiring precise and smooth motion, such as robotic surgeries or autonomous vehicle navigation, offering superior flexibility and adaptability to complex environments, enhancing performance and safety in challenging scenarios.

G. Overall conclusion of trajectory planning

Trajectory generation in robotics is a crucial aspect influencing performance and efficiency. Different methods, such as Multi-Point Trapezoidal and B-splines, offer unique advantages based on task complexity and computational resources. While simpler methods suffice for straightforward operations, complex techniques like B-splines are preferred for precise motion in intricate environments. Understanding these methods' strengths and limitations is key to designing effective robotic systems tailored to specific applications.

VIII. Conclusion

In conclusion, this report delves into the significance of Jacobians in robotics and control theory. These mathematical tools elucidate the relationship between joint velocities and end-effector velocities. Additionally, our analysis encompasses trajectory planning, a crucial aspect of determining feasible paths for robots. MATLAB plays a pivotal role in implementing these concepts, allowing us to create smooth trajectories, explore polynomial paths, and optimize motion profiles. We will indeed keep exploring these fascinating areas to enhance robotic control and automation!

References

[1] J. Angeles, "Fundamentals of Robotic Mechanical Systems", Theory, Methods, and Algorithms, 4th edition, Springer.

[2] J. J. Craig, "Introduction to Robotics", 3d edition, Pearson Education, Inc.

[3] M. Kazim, J. Hong, M. Kim, K. K. Kim, (2023). "Recent advances in path integral control for trajectory optimization: An overview in theoretical and algorithmic perspectives". Annual Reviews in Control, 57, 100931. [Online]. Available: https://doi.org/10.1016/j.arcontrol.2023.100931

[4] B. Nadir, O. Mohammed, N. Minh-Tuan, et al. "Optimal trajectory generation method to find a smooth robot joint trajectory based on multiquadric radial basis functions". Int J Adv Manuf Technol 120, 297-312 (2022). [Online]. Available: https://doi.org/10.1007/s00170-022-08696-1

[5] M. Ceccarelli, M. Russo, (2020). "Parallel Architectures for Humanoid Robots". Robotics, 9(4). [Online]. Available: https://doi.org/10.3390/robotics9040075

[6] T. Stouraitis, L. Yan, J. Moura, M. Gienger and S. Vijayakumar, "Multi-mode Trajectory Optimization for Impact-aware Manipulation," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 9425-9432, doi: 10.1109/IROS45743.2020.9341246.

[7] R. Liebhart, "Navigating the World of Robotics: Rectangular vs. Trapezoidal Motion Profiles", MAB Robotics. [Online]. Available: http://bit.ly/3U8IkDu

[8] Gosselin, C.M., and Hadj-Messaoud, A., 1993, "Automatic planning of smooth trajectories for pick-and-place operations", ASME Journal of Mechanical Design, Vol. 115, no. 3, pp. 450-456.

[9] T. Kirana, S. K. Srivastavab, "Analysis and Simulation of Cam Follower Mechanism Using Polynomial Cam Profile", International Journal of Multidisciplinary and Current Research. [Online]. Available: http://ijmcr.com/wp-content/uploads/2013/11/Paper15211-2151.pdf

[10] MATLAB documentation, "Trajectory Generation", MathWorks.com, https://www.mathworks.com/help/robotics/trajectory-generation.html

[11] P. Corke, "robotics-toolbox-matlab", Github, https://github.com/petercorke/robotics-toolbox-matlab?tab=readme-ov-file#install-from-shared-matlab-drive-folder

IX. Appendix

A. Resulted Plots of Problem 5: six figures

B. Resulted Plots of Problem 6: four figures

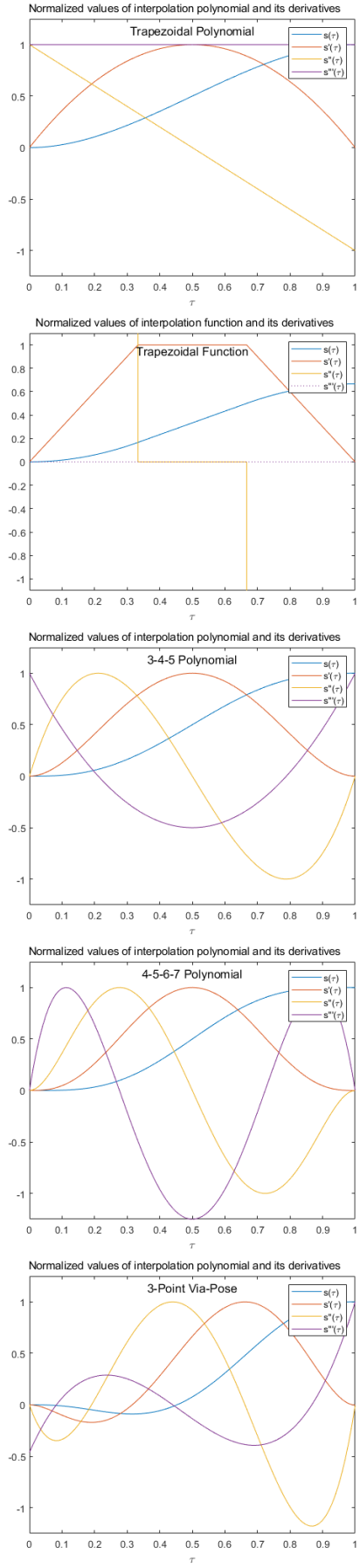


Fig. 5. Interpolating functions ($s(\tau)$) and their derivatives based on operation time for 5 pre-described methods

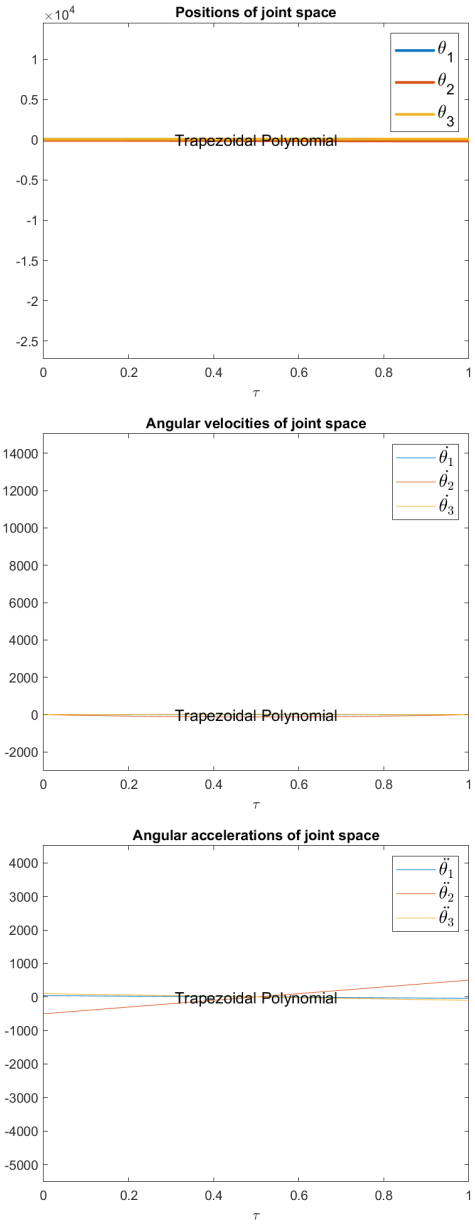


Fig. 6. Angular position, velocity and acceleration for Semi-Trapezoidal trajectory based on operation time

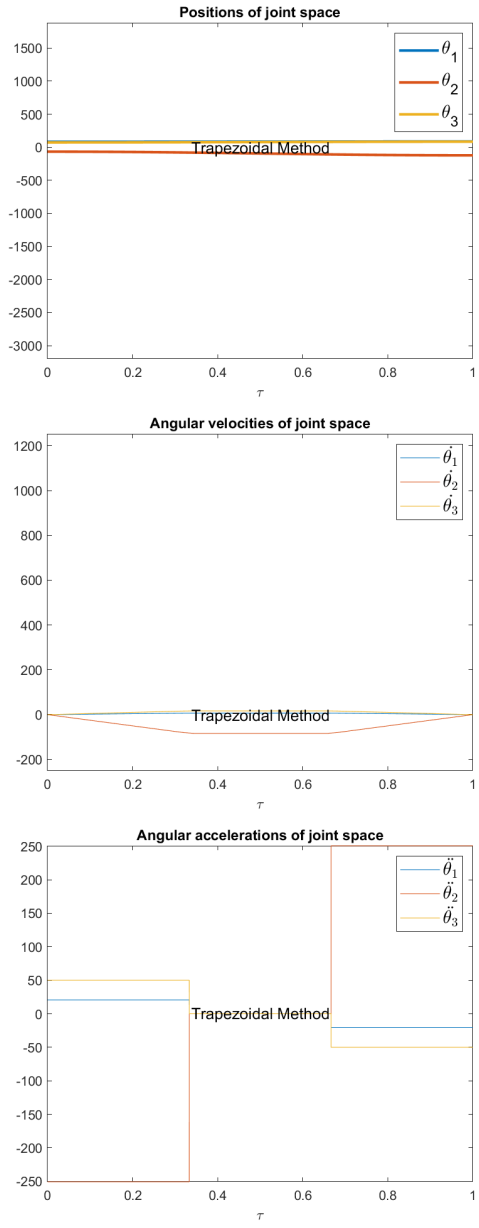


Fig. 7. Angular position, velocity and acceleration for Sharp Trapezoidal trajectory based on operation time

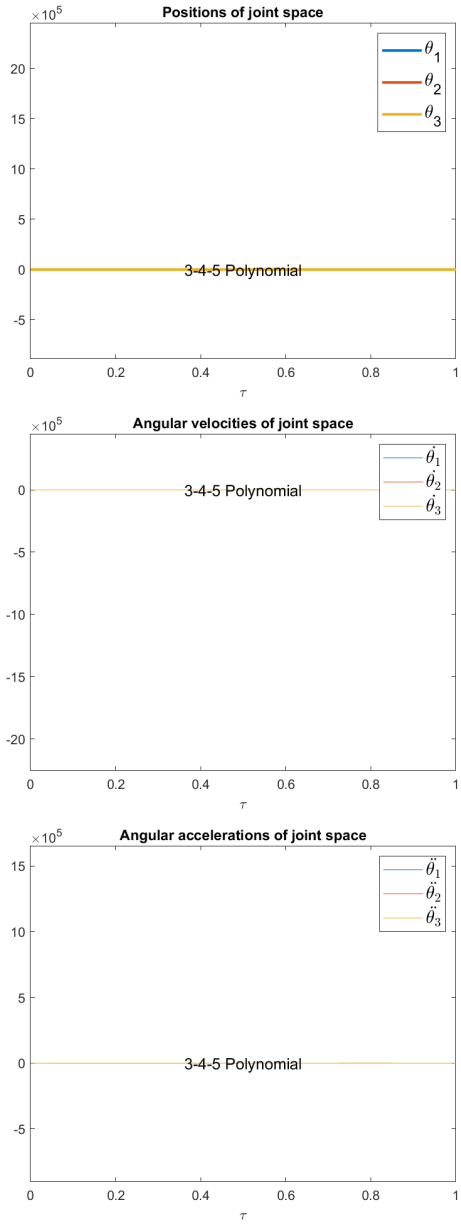
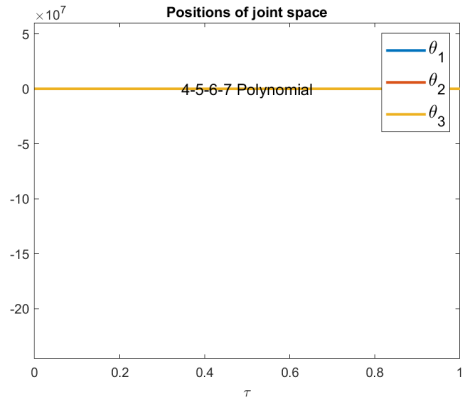
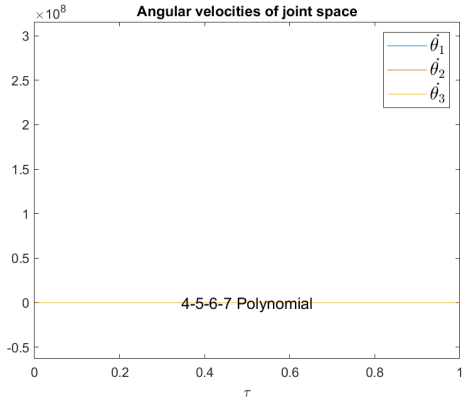


Fig. 8. Angular position, velocity and acceleration for 3-4-5 trajectory based on operation time





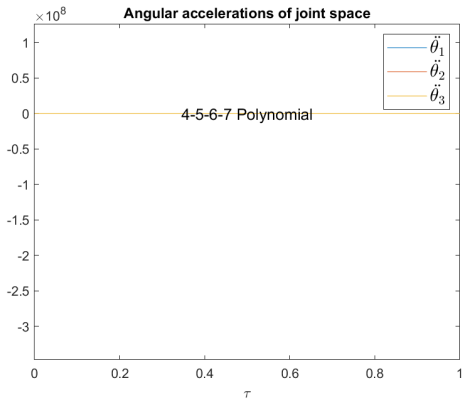
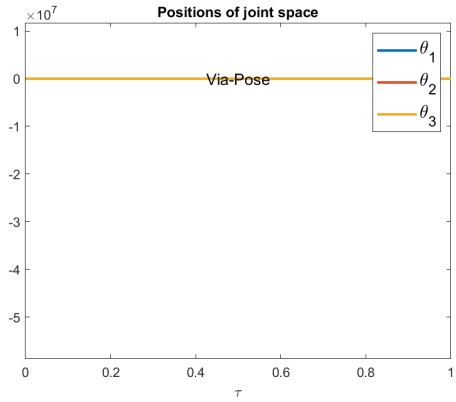
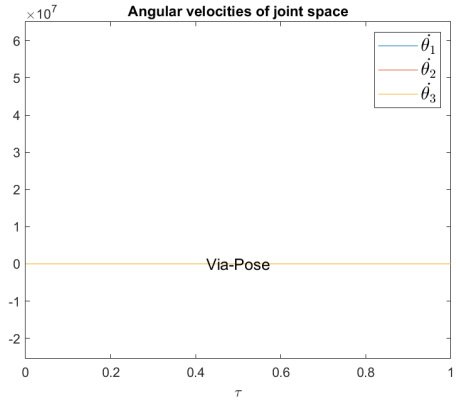


Fig. 9. Angular position, velocity and acceleration for 4-5-6-7 trajectory based on operation time





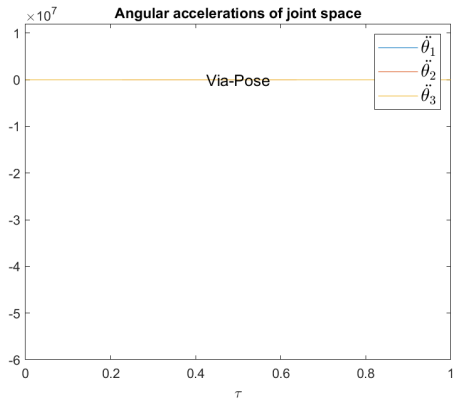


Fig. 10. Angular position, velocity and acceleration for 3-point via-pose trajectory based on operation time

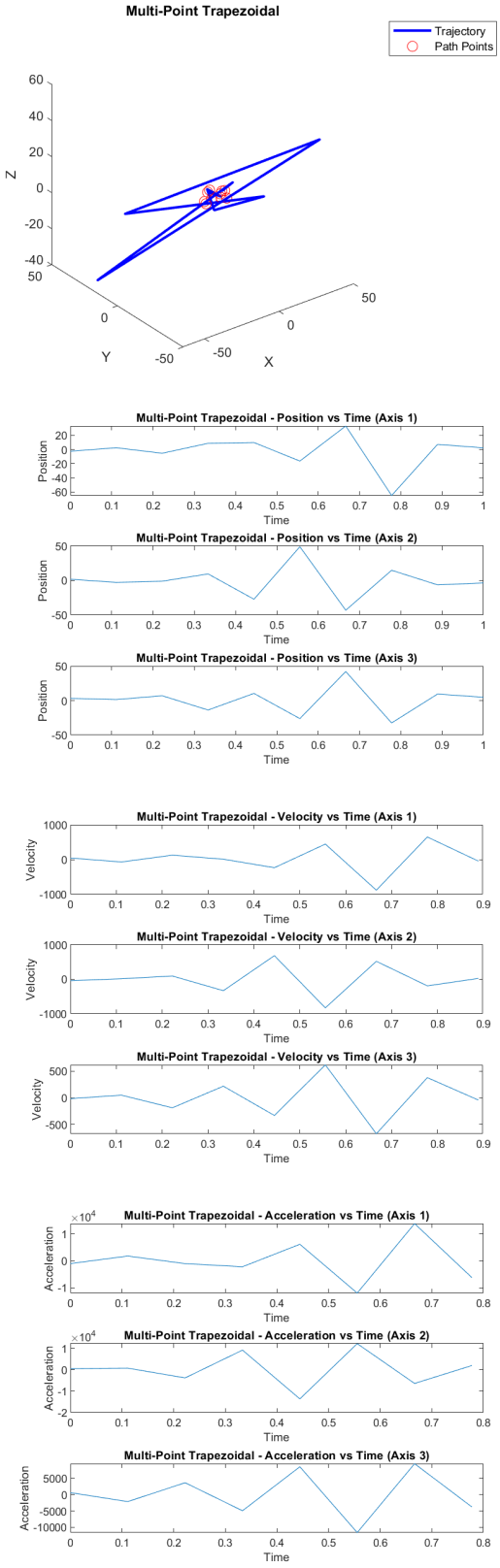


Fig. 11. First four results for problem 6

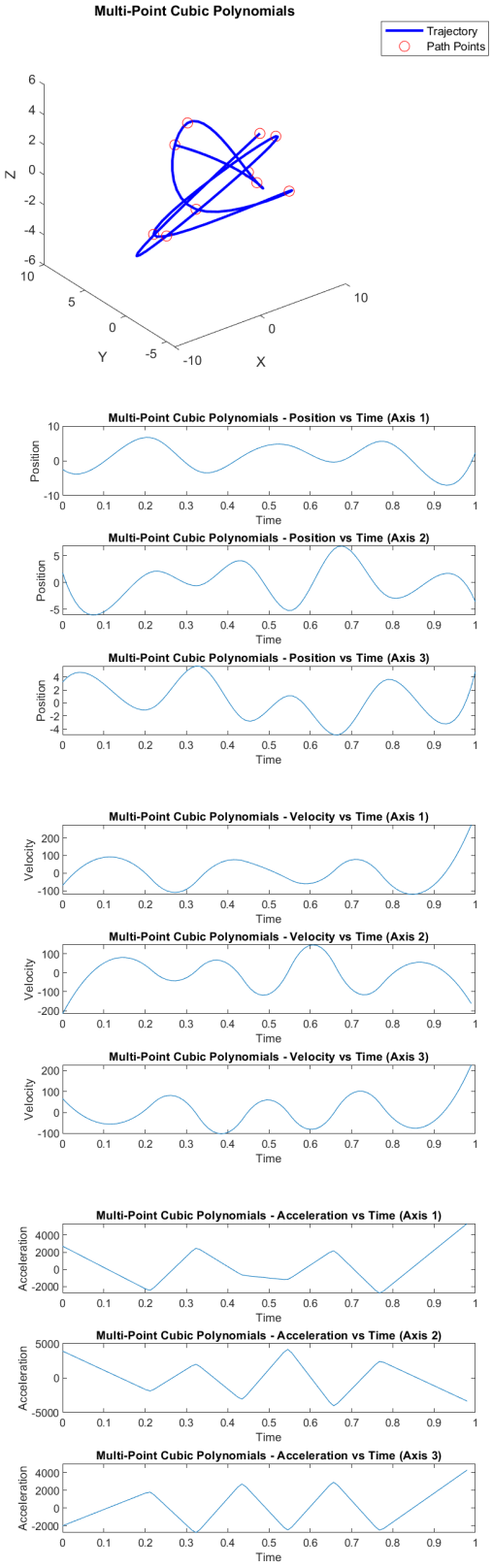


Fig. 12. Second four results for problem 6

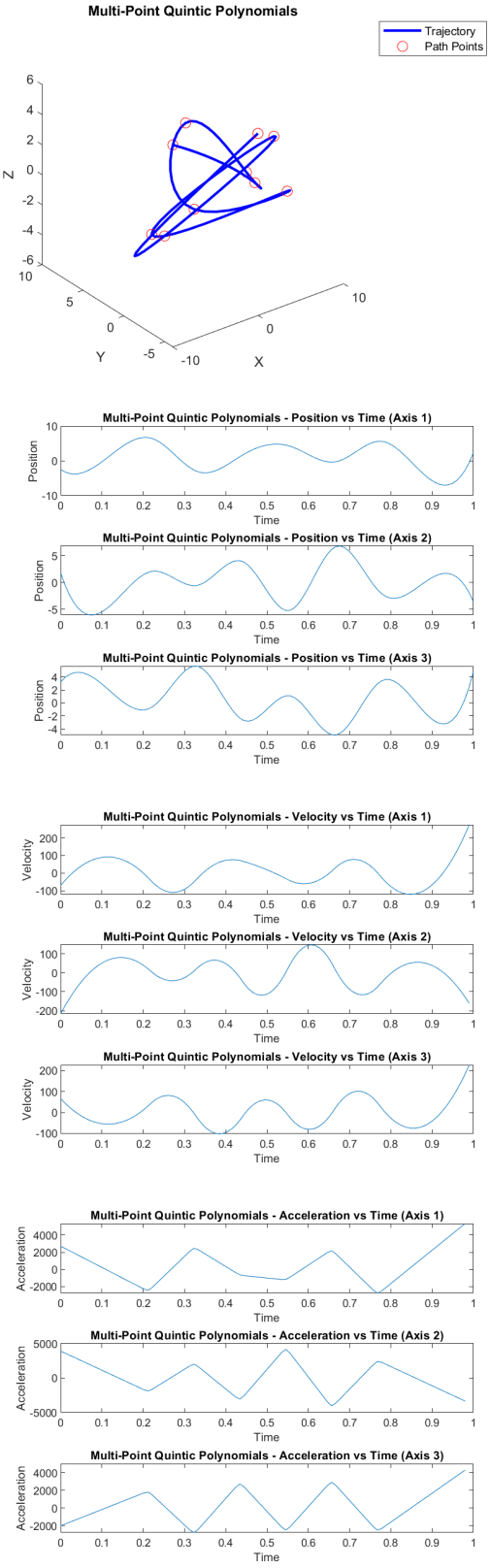


Fig. 13. Third four results for problem 6

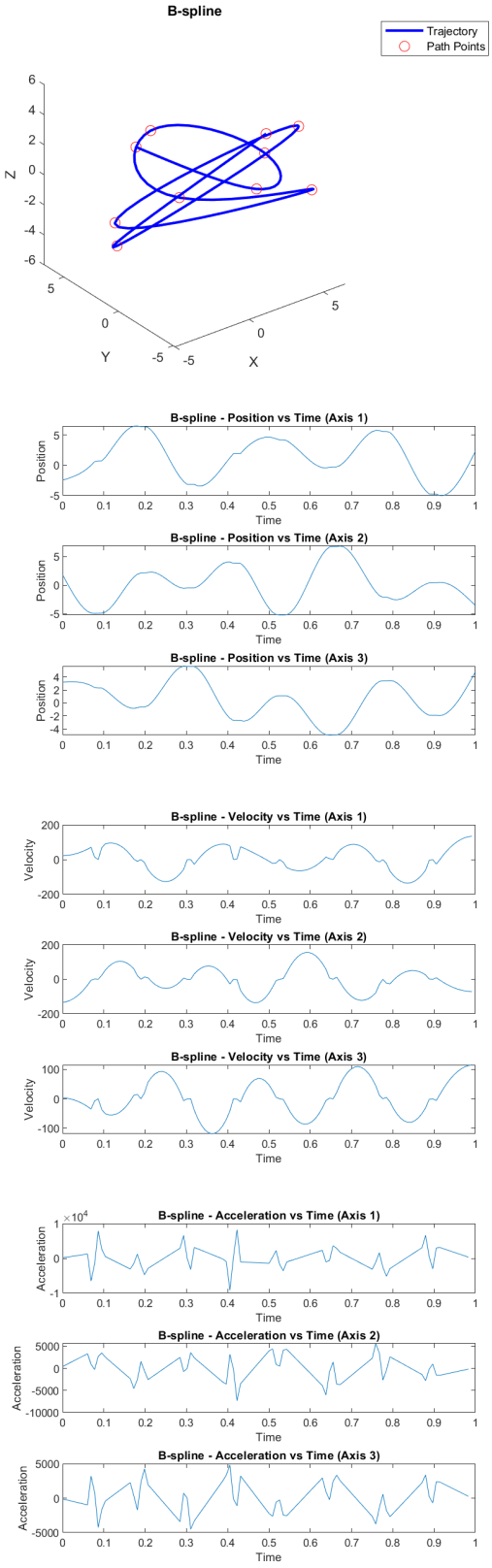


Fig. 14. Fourth four results for problem 6