# Assignment 2

Ayush Dwivedi

December 25, 2025

## 1 Problem Statement

A 2-link planar robotic arm moves from:

$$(q_1^{start}, q_2^{start})$$

to:

$$(q_1^{end}, q_2^{end})$$

over a fixed duration $T$. We generate and compare:

1. Linear joint-space trajectory
2. Smooth cubic polynomial trajectory (zero velocity at start/stop)

## 2 Linear Joint-Space Trajectory

$$q_1(t) = q_1^{start} + \frac{q_1^{end} - q_1^{start}}{T} t \tag{1}$$

$$q_2(t) = q_2^{start} + \frac{q_2^{end} - q_2^{start}}{T} t \tag{2}$$

### 2.1 Python Implementation

```
import numpy as np
import matplotlib.pyplot as plt

T = 3.0
dt = 0.1
t = np.arange(0, T+dt, dt)

q1_start, q2_start = 0.0, 0.0
q1_end, q2_end = np.pi/3, np.pi/4

q1_linear = q1_start + (q1_end - q1_start) * (t / T)
q2_linear = q2_start + (q2_end - q2_start) * (t / T)

plt.figure(figsize=(8,5))
plt.plot(t, q1_linear, label='q1')
plt.plot(t, q2_linear, label='q2')
plt.xlabel('Time')
plt.ylabel('Joint Angle')
plt.title('Linear Joint-Space Trajectory')
plt.legend()
plt.grid(True)
plt.savefig("linear_traj.png")
plt.show()
```

## 3 Smooth Cubic Polynomial Trajectory

Cubic form:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Boundary conditions:

$$q(0) = q^{start},\ q(T) = q^{end},\ \dot{q}(0) = 0,\ \dot{q}(T) = 0$$

Solved coefficients:

$$a_0 = q^{start} \tag{3}$$

$$a_1 = 0 \tag{4}$$

$$a_2 = \frac{3(q^{end} - q^{start})}{T^2} \tag{5}$$

$$a_3 = \frac{-2(q^{end} - q^{start})}{T^3} \tag{6}$$

Joint trajectories:

$$q_1(t) = q_1^{start} + \frac{3(q_1^{end} - q_1^{start})}{T^2} t^2 - \frac{2(q_1^{end} - q_1^{start})}{T^3} t^3 \tag{7}$$

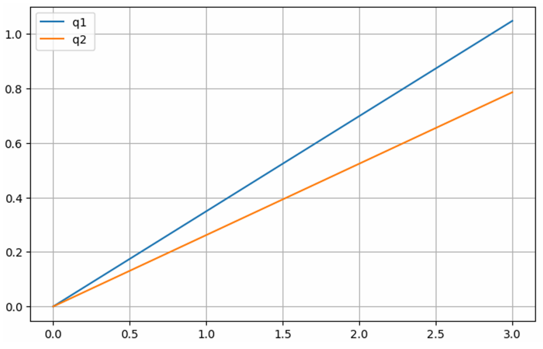
$$q_2(t) = q_2^{start} + \frac{3(q_2^{end} - q_2^{start})}{T^2} t^2 - \frac{2(q_2^{end} - q_2^{start})}{T^3} t^3 \tag{8}$$

### 3.1 Python Implementation

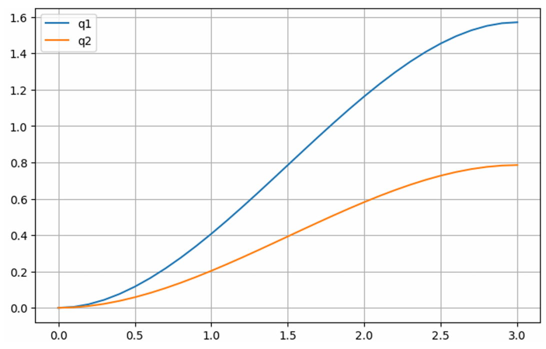
```
import numpy as np
import matplotlib.pyplot as plt

def cubic_trajectory(q_start, q_end, T, t):
    a0 = q_start
    a1 = 0
    a2 = 3*(q_end - q_start)/(T**2)
    a3 = -2*(q_end - q_start)/(T**3)
    q = a0 + a1*t + a2*t**2 + a3*t**3
    return q

T = 3.0
dt = 0.1
t = np.arange(0, T+dt, dt)

q1_start, q2_start = 0.0, 0.0
q1_end, q2_end = np.pi/3, np.pi/4

q1_cubic = cubic_trajectory(q1_start, q1_end, T, t)
q2_cubic = cubic_trajectory(q2_start, q2_end, T, t)

plt.figure(figsize=(8,5))
plt.plot(t, q1_cubic, label='q1')
plt.plot(t, q2_cubic, label='q2')
plt.xlabel('Time')
plt.ylabel('Joint Angle')
plt.title('Smooth Cubic Polynomial Trajectory')
plt.legend()
plt.grid(True)
plt.savefig("cubic_traj.png")
plt.show()

# Optional: Velocity plots
```

```
v1 = np.gradient(q1_cubic, t)
v2 = np.gradient(q2_cubic, t)

plt.figure(figsize=(8,4))
plt.plot(t, v1, label='dq1/dt')
plt.plot(t, v2, label='dq2/dt')
plt.xlabel('Time')
plt.ylabel('Velocity')
plt.title('Joint Velocities (Optional)')
plt.legend()
plt.grid(True)
plt.show()
```

## 4 Trajectory Plots



(a) Linear Trajectory



(b) Smooth Cubic Trajectory

Figure 1: Joint angle vs time comparison

## 5 Discussion

Linear interpolation is easy to compute but causes non-zero velocity at motion boundaries, which can introduce jerks. producing smooth start/stop motion. Smoothness is crucial for real robots to reduce joint stress, avoid oscillations, improve precision, and prevent impulsive torque spikes.