Robot Parameters: 2-Link Planar Arm (l1 = 1, l2 = 1)

1. Problem Description

The goal is to move the robot from a start configuration to an end configuration over a fixed time duration T = 2 seconds. We will compare a linear trajectory with a smooth polynomial trajectory.

Joint Configurations used:

- **Start:** q1 = 0, q2 = 0 (Straight arm)
- **End:** q1 = pi/2, q2 = pi/2 (Bent arm)

2. Linear Trajectory Generation

In this approach, joint angles vary linearly with time.

- **Formula:** q(t) = q{start} +((q{end} - q{start})/T) *t
- **Observation:** This creates a constant velocity but results in abrupt changes (infinite acceleration) at the start and end of the motion.

3. Smooth Polynomial Trajectory

To ensure smooth motion, we use a Cubic Polynomial q(t) = a0 + a1t + a2t^2 + a3t^3.

We enforce zero velocity at t=0 and t=T.

Calculated Coefficients:

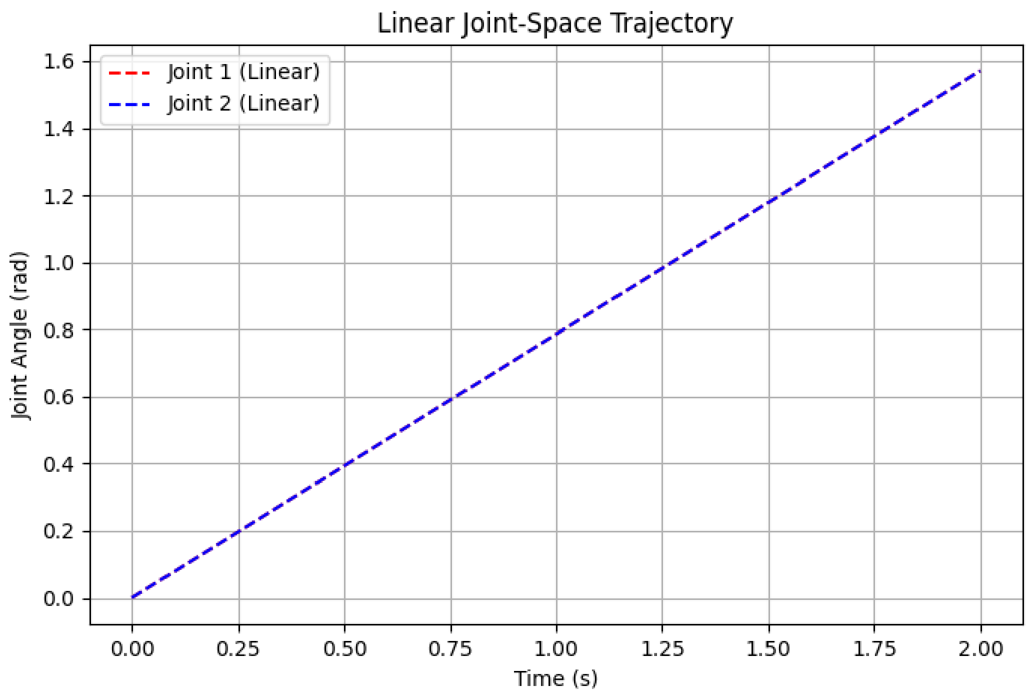
- a0 = q{start}
- a1 = 0
- a2 = 3/(T^2)*(q{end} - q{start})
- a3 = -(2/T^3)*(q{end} - q{start})

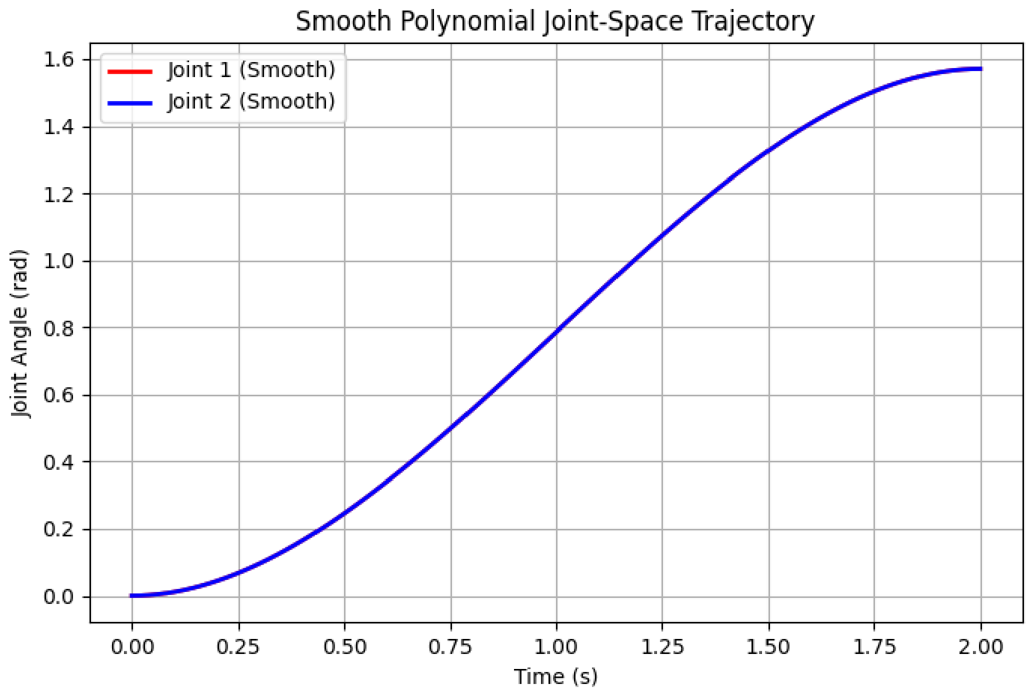
4. Python Implementation

```
import numpy as np

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt



T = 2.0
```

```
t = np.linspace(0, T, 100)

q_start = np.array([0, 0])

q_end = np.array([np.pi/2, np.pi/2])


q1_linear = q_start[0] + (q_end[0] - q_start[0]) * (t / T)

q2_linear = q_start[1] + (q_end[1] - q_start[1]) * (t / T)


plt.figure(figsize=(8, 5))

plt.plot(t, q1_linear, 'r--', label="Joint 1 (Linear)")

plt.plot(t, q2_linear, 'b--', label="Joint 2 (Linear)")

plt.title("Linear Joint-Space Trajectory")

plt.xlabel("Time (s)")

plt.ylabel("Joint Angle (rad)")

plt.legend()

plt.grid(True)

plt.savefig("plot_linear.png")

plt.close()


def get_cubic_coeffs(q0, qf, Tf):

    return q0, 0, (3/Tf**2)*(qf-q0), (-2/Tf**3)*(qf-q0)


a0_1, a1_1, a2_1, a3_1 = get_cubic_coeffs(q_start[0], q_end[0], T)

a0_2, a1_2, a2_2, a3_2 = get_cubic_coeffs(q_start[1], q_end[1], T)


q1_smooth = a0_1 + a1_1*t + a2_1*t**2 + a3_1*t**3
```

```
q2_smooth = a0_2 + a1_2*t + a2_2*t**2 + a3_2*t**3


plt.figure(figsize=(8, 5))
plt.plot(t, q1_smooth, 'r-', linewidth=2, label="Joint 1 (Smooth)")
plt.plot(t, q2_smooth, 'b-', linewidth=2, label="Joint 2 (Smooth)")
plt.title("Smooth Polynomial Joint-Space Trajectory")
plt.xlabel("Time (s)")
plt.ylabel("Joint Angle (rad)")
plt.legend()
plt.grid(True)
plt.savefig("plot_smooth.png")
plt.close()
```


Linear Joint-Space Trajectory
Joint 1 (Linear)
Joint 2 (Linear)
Joint Angle (rad)
Time (s)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Smooth Polynomial Joint-Space Trajectory
Joint 1 (Smooth)
Joint 2 (Smooth)
Joint Angle (rad)
Time (s)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

The linear trajectory shows a constant slope, meaning the robot's velocity changes instantly at the start and end, causing high "jerk". In contrast, the smooth polynomial trajectory creates an S-curve profile which ensures the robot starts and stops with zero velocity. This smoothness is critical for real robotic systems to prevent mechanical vibration and protect motor actuators from sudden torque spikes. While linear interpolation is easier to calculate, polynomial trajectories ensure hardware longevity and stability during motion. Therefore, for the 2-link planar arm, the smooth polynomial approach is the superior choice for real-world application.