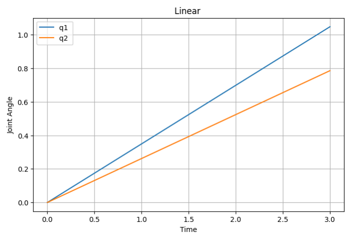
```python
import numpy as np
import matplotlib.pyplot as plt
```

```python
T = 3.0
dt = 0.1
t = np.arange(0, T+dt, dt)
q1i, q2i = 0.0, 0.0
q1f, q2f = np.pi/3, np.pi/4
```

```python
q1_linear = q1i + (q1f - q1i) * (t / T)
q2_linear = q2i + (q2f - q2i) * (t / T)
```

```python
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 ')
plt.legend()
plt.grid(True)
plt.show()
```

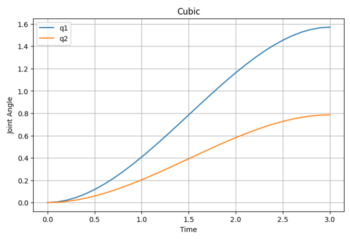


```python
def cubic_trajectory(q_start, q_end, T, t):
    a0 = q_start
    a1 = 0
    a2 = 3*(q_end - q_start)/(T**2)
    a3 = -2*(q_end - q_start)/(T**3)

    q = a0 + a1*t + a2*t**2 + a3*t**3
    return q
```

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

```python
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('Cubic')
plt.legend()
plt.grid(True)
plt.show()
```



```
linear trajectory was derived simply using the equation of a
straight line. For cubic we used dqi/dt=0 at t=0 and t=T and qi(0)
=0 and qi(T)=Qi. The difference is that in cubic trajectory the
velocity of the end point is zero during the initial and final
positions. This property is of importance as it minimizes jerks
on the mechanism due to repeated use and improves precision.
```

linear trajectory was derived simply using the equation of a straight line. For cubic we used dqi/dt=0 at t=0 and t=T and qi(0)=0 and qi(T)=Qi. The difference is that in cubic trajectory the velocity of the end point is zero during the initial and final positions. This property is of importance as it minimizes jerks on the mechanism due to repeated use and improves precision.