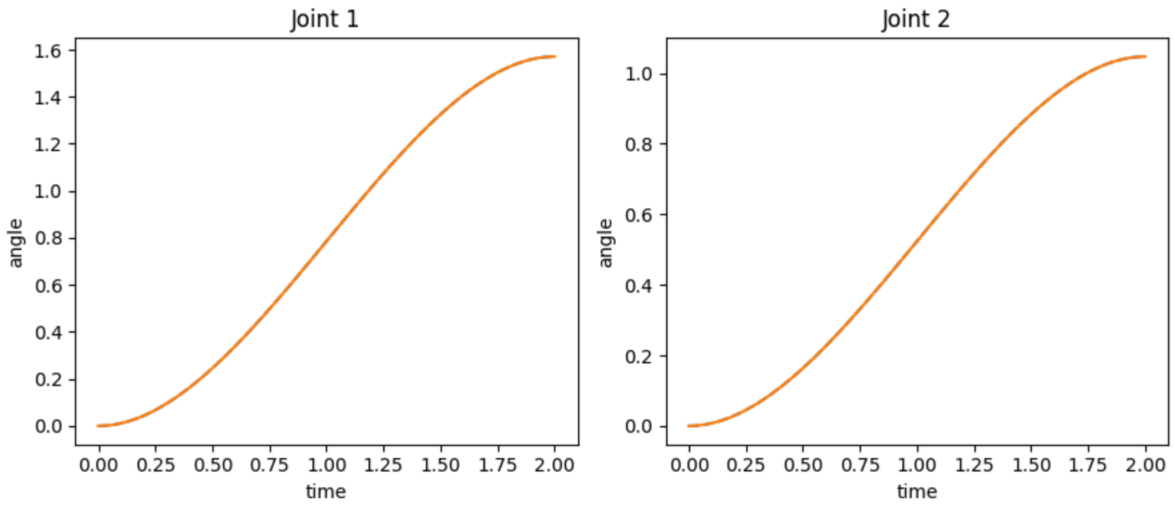
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

T = 2
N = 100
dt = T/(N-1)

q1s, q2s = 0, 0
q1e, q2e = np.pi/2, np.pi/3

a = 0
b = 1


def cubic(q0, qf):
    t = np.linspace(0, T, N)
    return q0 + 3*(qf-q0)*(t/T)**2 - 2*(qf-q0)*(t/T)**3


def cost(x):
    q1 = x[:N]
    q2 = x[N:]

    dq1 = (q1[1:] - q1[:-1]) / dt
    dq2 = (q2[1:] - q2[:-1]) / dt

    ddq1 = (q1[2:] - 2*q1[1:-1] + q1[:-2]) / dt**2
    ddq2 = (q2[2:] - 2*q2[1:-1] + q2[:-2]) / dt**2

    Jv = np.sum(dq1**2) + np.sum(dq2**2)
    Ja = np.sum(ddq1**2) + np.sum(ddq2**2)

    return a*Jv + b*Ja


cons = []

# position constraints
cons.append({'type': 'eq', 'fun': lambda x: x[0] - q1s})
cons.append({'type': 'eq', 'fun': lambda x: x[N-1] - q1e})
cons.append({'type': 'eq', 'fun': lambda x: x[N] - q2s})
cons.append({'type': 'eq', 'fun': lambda x: x[2*N-1] - q2e})

# zero velocity start
cons.append({'type': 'eq', 'fun': lambda x: x[1] - x[0]})
cons.append({'type': 'eq', 'fun': lambda x: x[N+1] - x[N]})

# zero velocity end
cons.append({'type': 'eq', 'fun': lambda x: x[N-1] - x[N-2]})
cons.append({'type': 'eq', 'fun': lambda x: x[2*N-1] - x[2*N-2]})


q1_0 = cubic(q1s, q1e)
q2_0 = cubic(q2s, q2e)
x0 = np.concatenate((q1_0, q2_0))


out = minimize(cost, x0, method='SLSQP', constraints=cons)

q1_opt = out.x[:N]
q2_opt = out.x[N:]


t = np.linspace(0, T, N)

plt.figure(figsize=(9,4))

plt.subplot(1,2,1)
plt.plot(t, q1_0, '--')
plt.plot(t, q1_opt)
plt.title('Joint 1')
plt.xlabel('time')
plt.ylabel('angle')

plt.subplot(1,2,2)
plt.plot(t, q2_0, '--')
plt.plot(t, q2_opt)
plt.title('Joint 2')
plt.xlabel('time')
plt.ylabel('angle')
plt.ylabel('angle')

plt.tight_layout()
plt.show()


print("initial cost:", cost(x0))
print("final cost:", cost(out.x))
```



```
initial cost: 256.6637147891111
final cost: 256.66431912503674
```

Close

I have taken the cost to a weighted sum of the energy cost and s
with initial conditions as zero velocities at start and end and
final joint angles. From last assignment i have plotted cubic tr
baseline. An interesting case occur when cost is just the smooth
turns out the solution for max smoothness for the given boundary
in fact a cubic polynomial exactly as in assignment 2 ! One can
using caluculus of variations. Because it is discretized its cos
slightly higher than the actual solution. For increasing high we
cost the trajectory shifts to a linear trajectory.

I have taken the cost to a weighted sum of the energy cost and smoothness cost with initial conditions as zero velocities at start and end and initail and final joint angles. From last assignment i have plotted cubic trajectory as my baseline. An interesting case occur when cost is just the smoothness cost. It turns out the solution for max smoothness for the given boundary conditions is in fact a cubic polynomial exactly as in assignment 2 ! One can prove this using caluculus of variations. Because it is discretized its cost will be slightly higher than the actual solution. For increasing high weight of energy cost the trajectory shifts to a linear trajectory.