PS 3

Keyi Zhang

Issued: Thu 10/09/2025

Due: Thu 10/23/2025

Problem 1:

We designed a P.D. controller to achieve a point-to-point task with a planar two-link manipulator before.

For impedance control of the manipulator in this problem, we proposed a control law for the joint torque that can achieve in the task space a restoring force F corresponding to the virtual programmable cartesian spring Kp and damper Kd.

With gravity compensation, the control law was formulated in terms of the end-effector cartesian displacement error x~ = x - xd as:

$$\tau = g(q) - J^T \left(K_p \tilde{x} + K_d \dot{\tilde{x}} \right)$$

Consider now the case of impedance control of a planar four-link manipulator for achieving a point-to-point task.

In addition to the virtual cartesian spring and damper at the end-effector, let us say we also have a virtual cartesian spring and damper at joint-2 (the joint adjacent to the \shoulder" of the manipulator).

Propose a control law similar to the one shown above for the torque for the case of this four-link manipulator.

- The control law must be written in terms of the end-effector cartesian displacement error ~xe and the joint-2 cartesian displacement error ~x2.
- Neglect the gravity compensation term g(q). (You can find this term easily, we have done a similar exercise as part of Problem 3(b) in Problem Set#2.)
- No simulations are required (i.e. just propose the control law: NO coding is required in this problem!)

[Hint: Your answer should contain two different (why?) JT matrices respectively premultiplying the restoring force vectors corresponding to the mechanical impedances at the end-effector and the joint-2. Explicitly derive expressions for the Jacobian matrices in terms of joint angles and link lengths of the manipulator.]

Answer:

Known:

Neglect the gravity compensation term g(q):

$$\tau = J^T \left(K_p \tilde{x} + K_d \dot{\tilde{x}} \right)$$

$$F = \left(K_p \tilde{x} + K_d \dot{\tilde{x}} \right)$$

The displacement error of endpoint and the joint 2:

$$\tilde{x_e} = x - x_d$$

$$\tilde{x_2} = x_2 - x_2 d$$

The forces of endpoint and the joint 2:

$$F_e = \left(K_p \tilde{x_e} + K_d \dot{\tilde{x_e}}\right)$$

$$F_2 = \left(K_p \tilde{x_2} + K_d \dot{\tilde{x_2}}\right)$$

The Jacobian of endpoint and the joint 2:

1. The end point position:

   $x_e = l1c1 + l2c12 + l3c123 + l4c1234$

   $y_e = l1s1 + l2s12 + l3s123 + l4s1234$

   The Jacobian matrix:

   $$J_e = \begin{bmatrix} \frac{\partial x_e}{\partial q_1} & \frac{\partial x_e}{\partial q_2} & \frac{\partial x_e}{\partial q_3} & \frac{\partial x_e}{\partial q_4} \\ \frac{\partial y_e}{\partial q_1} & \frac{\partial y_e}{\partial q_2} & \frac{\partial y_e}{\partial q_3} & \frac{\partial y_e}{\partial q_4} \end{bmatrix}$$

   The elements:

   $$\begin{aligned} \frac{\partial x_{e,x}}{\partial q_1} &= -l_1 s_1 - l_2 s_{12} - l_3 s_{123} - l_4 s_{1234} \\ \frac{\partial x_{e,x}}{\partial q_2} &= -l_2 s_{12} - l_3 s_{123} - l_4 s_{1234} \\ \frac{\partial x_{e,x}}{\partial q_3} &= -l_3 s_{123} - l_4 s_{1234} \\ \frac{\partial x_{e,x}}{\partial q_4} &= -l_4 s_{1234} \\ \frac{\partial x_{e,y}}{\partial q_1} &= l_1 c_1 + l_2 c_{12} + l_3 c_{123} + l_4 c_{1234} \\ \frac{\partial x_{e,y}}{\partial q_2} &= l_2 c_{12} + l_3 c_{123} + l_4 c_{1234} \\ \frac{\partial x_{e,y}}{\partial q_3} &= l_3 c_{123} + l_4 c_{1234} \\ \frac{\partial x_{e,y}}{\partial q_4} &= l_4 c_{1234} \end{aligned}$$

2. The joint 2 position:

   $x_2 = l1c1$

   $y_2 = l1s1$

   The joint 2 Jacobian:

   $$J_2 = \begin{bmatrix} \frac{\partial x_e}{\partial q_1} & \frac{\partial x_e}{\partial q_2} & \frac{\partial x_e}{\partial q_3} & \frac{\partial x_e}{\partial q_4} \\ \frac{\partial y_e}{\partial q_1} & \frac{\partial y_e}{\partial q_2} & \frac{\partial y_e}{\partial q_3} & \frac{\partial y_e}{\partial q_4} \end{bmatrix} = \begin{bmatrix} -l_1 s1 & 0 & 0 & 0 \\ l1c1 & 0 & 0 & 0 \end{bmatrix}$$

Thus, the control law:

$$\tau = -J_e^T\left(K_{p,e}\tilde{x}_e + K_{d,e}\dot{\tilde{x}}_e\right) - J_2^T\left(K_{p,2}\tilde{x}_2 + K_{d,2}\dot{\tilde{x}}_2\right)$$

Problem 2:

Consider the dynamic equations in the vertical plane for a two degree-of-freedom planar manipulator with two rotational joints

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

Where:

$$q = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}, \quad H = \begin{bmatrix}H_{11} & H_{12}\\ H_{12} & H_{22}\end{bmatrix}, \quad C = \begin{bmatrix}-h\dot{\theta}_2 & -h\dot{\theta}_1 - h\dot{\theta}_2\\ h\dot{\theta}_1 & 0\end{bmatrix}, \quad g = \begin{bmatrix}G_1\\ G_2\end{bmatrix}$$

$$\begin{aligned}
H_{11} &= m_1 l_{c1}^2 + I_1 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos\theta_2) + I_2\\
H_{22} &= m_2 l_{c2}^2 + I_2\\
H_{12} &= m_2 l_1 l_{c2}\cos\theta_2 + m_2 l_{c2}^2 + I_2\\
h &= m_2 l_1 l_{c2}\sin\theta_2\\
G_1 &= m_1 l_{c1} g\cos\theta_1 + m_2 l_{c2} g\cos(\theta_1+\theta_2) + m_2 l_1 g\cos\theta_1\\
G_2 &= m_2 l_{c2} g\cos(\theta_1+\theta_2)
\end{aligned}$$

Design and simulate an adaptive controller without any initial knowledge of the constant parameters. The desired trajectory is:

$$\theta_{d1} = 2(1-e^{-t}), \quad \theta_{d2} = 3(1-e^{-2t})$$

(In the simulation, you can choose the real values of the parameters and the initial conditions by yourself.)

Answer:

Known:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

$$q = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}, \quad H = \begin{bmatrix}H_{11} & H_{12}\\ H_{12} & H_{22}\end{bmatrix}, \quad C = \begin{bmatrix}-h\dot{\theta}_2 & -h\dot{\theta}_1 - h\dot{\theta}_2\\ h\dot{\theta}_1 & 0\end{bmatrix}, \quad g = \begin{bmatrix}G_1\\ G_2\end{bmatrix}$$

$$\begin{aligned}
H_{11} &= m_1 l_{c1}^2 + I_1 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos\theta_2) + I_2\\
H_{22} &= m_2 l_{c2}^2 + I_2\\
H_{12} &= m_2 l_1 l_{c2}\cos\theta_2 + m_2 l_{c2}^2 + I_2\\
h &= m_2 l_1 l_{c2}\sin\theta_2\\
G_1 &= m_1 l_{c1} g\cos\theta_1 + m_2 l_{c2} g\cos(\theta_1+\theta_2) + m_2 l_1 g\cos\theta_1\\
G_2 &= m_2 l_{c2} g\cos(\theta_1+\theta_2)
\end{aligned}$$

Definition of Tracking Error and Reference Trajectory:

$$
\begin{gathered}
\tilde{q} = q_d - q \\
\dot{q}_r = \dot{q}_d - \Lambda e \\
s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \Lambda \tilde{q}
\end{gathered}
$$

Linear Parameterization:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

$$H(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + g(q) = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)a$$

For a:

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\ m_2 l_{c2}^2 + I_2 \\ m_2 l_1 l_{c2} \\ m_1 l_{c1} g + m_2 l_1 g \\ m_2 l_{c2} g \end{bmatrix}$$

For Y:

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} & Y_{15} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} & Y_{25} \end{bmatrix} = \left[\begin{matrix} \ddot{q}_{r1} & \ddot{q}_{r1} + \ddot{q}_{r2} & 2\cos\theta_2 \ddot{q}_{r1} + \cos\theta_2 \ddot{q}_{r2} - \sin\theta_2(\dot{\theta}_2 \dot{q}_{r1} + (\dot{\theta}_1 + \dot{\theta}_2)\dot{q}_{r2}) & \cos\theta_1 \\ 0 & \ddot{q}_{r1} + \ddot{q}_{r2} & \cos\theta_2 \ddot{q}_{r1} + \sin\theta_2 \dot{\theta}_1 \dot{q}_{r1} & 0 \end{matrix}\right.$$

Control and Adaption:

$$\tau = Y\hat{a} - K_d s$$

$$\dot{\hat{a}} = -PY^\top s = \dot{\hat{a}} = -PY(q,\dot{q},\dot{q}_r,\ddot{q}_r)^\top s$$

Lyapunov stability analysis：

$$V = 1/2(S^T H S) + 1/2(\tilde{a}^T P^{-1} \tilde{a})$$

$$\dot{V} = s^T H \dot{s} + \frac{1}{2} s^T \dot{H} s + \tilde{a}^T P^{-1} \dot{\tilde{a}} = -s^T K_d s \leq 0$$

Since

$$V \geq 0, \quad \dot{V} \leq 0, \quad V \text{ is radially unbounded}$$

according to Lyapunov stability theory, the system is globally uniformly stable. Moreover,

$$s \to 0 \quad \text{as} \quad t \to \infty$$

Because

$$s = \dot{\tilde{q}} + \Lambda \tilde{q}$$

it follows that

$$\tilde{q} \to 0, \quad \dot{\tilde{q}} \to 0 \text{ as } t \to \infty$$

meaning both the tracking error and its derivative converge to zero.

Simulation with:

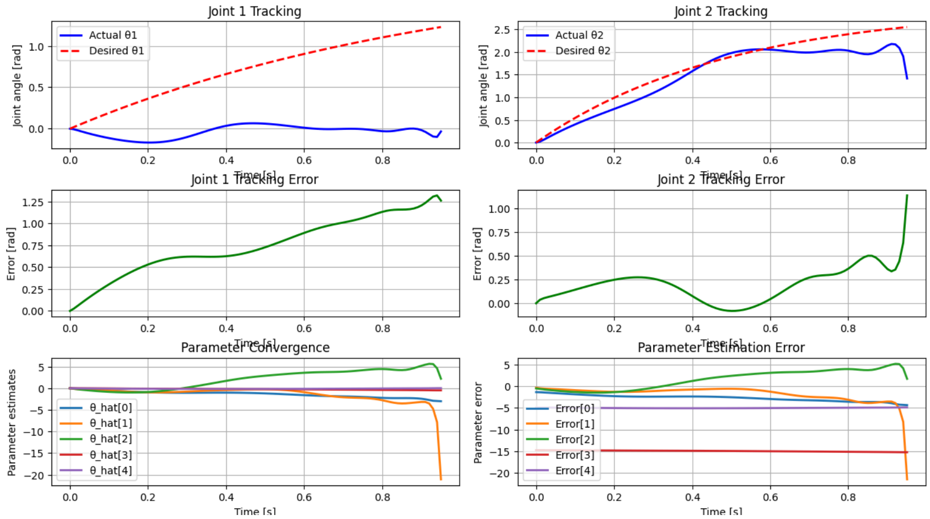
$$\theta_{d1} = 2(1 - e^{-t}), \quad \theta_{d2} = 3(1 - e^{-2t})$$

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

#design parameters
m1, m2 = 1.0, 1.0
l1, l2 = 1.0, 1.0
lc1, lc2 = 0.5, 0.5
I1, I2 = 0.1, 0.1
g = 9.81

phi_true = np.array([
    m1*lc1**2 + m2*l1**2 + I1,
    m2*lc2**2 + I2,
    m2*l1*lc2,
    m1*lc1*g + m2*l1*g,
    m2*lc2*g
])

print("True parameters:")
for i in range(5):
    print(f"π{i+1} = {phi_true[i]:.3f}")

def qd(t):
    theta1 = 2*(1 - np.exp(-t))
    theta2 = 3*(1 - np.exp(-2*t))
    return np.array([theta1, theta2])

def qd_dot(t):
    return np.array([2*np.exp(-t), 6*np.exp(-2*t)])

def qd_ddot(t):
    return np.array([-2*np.exp(-t), -12*np.exp(-2*t)])

def H_matrix(q):
    th1, th2 = q
    H11 = m1*lc1**2 + I1 + m2*(l1**2 + lc2**2 + 2*l1*lc2*np.cos(th2)) + I2
    H12 = m2*l1*lc2*np.cos(th2) + m2*lc2**2 + I2
    H22 = m2*lc2**2 + I2
    return np.array([[H11, H12],[H12, H22]])

def C_matrix(q, q_dot):
    th1, th2 = q
    dth1, dth2 = q_dot
    h = m2*l1*lc2*np.sin(th2)
    return np.array([[-h*dth2, -h*(dth1 + dth2)],
                     [h*dth1, 0]])

def g_vector(q):
    th1, th2 = q
    G1 = m1*lc1*g*np.cos(th1) + m2*lc2*g*np.cos(th1 + th2) + m2*l1*g*np.cos(th1)
```

```
    G2 = m2*lc2*g*np.cos(th1 + th2)
    return np.array([G1, G2])

Lambda = np.diag([5.0, 5.0])
K = np.diag([10.0, 10.0])
Gamma = np.diag([0.1, 0.1, 0.1, 0.1, 0.1])

def Y_matrix(q, q_dot, q_r_dot, q_r_ddot):
    th1, th2 = q
    dth1, dth2 = q_dot
    dqr1, dqr2 = q_r_dot
    ddqr1, ddqr2 = q_r_ddot

    Y = np.zeros((2, 5))

    Y[0, 0] = ddqr1
    Y[0, 1] = ddqr1 + ddqr2
    Y[0, 2] = 2*np.cos(th2)*ddqr1 + np.cos(th2)*ddqr2 - np.sin(th2)*(dth2*dqr1 + (dth1 + dth2)*dqr2)
    Y[0, 3] = np.cos(th1)
    Y[0, 4] = np.cos(th1 + th2)

    Y[1, 0] = 0
    Y[1, 1] = ddqr1 + ddqr2
    Y[1, 2] = np.cos(th2)*ddqr1 + np.sin(th2)*dth1*dqr1
    Y[1, 3] = 0
    Y[1, 4] = np.cos(th1 + th2)

    return Y

def dynamics(t, state):
    q = state[0:2]
    q_dot = state[2:4]
    theta_hat = state[4:9]

    q_d = qd(t)
    q_d_dot = qd_dot(t)
    q_d_ddot = qd_ddot(t)

    q_tilde = q_d - q

    q_r_dot = q_d_dot - Lambda @ q_tilde

    q_tilde_dot = q_d_dot - q_dot
    q_r_ddot = q_d_ddot - Lambda @ q_tilde_dot

    s = q_dot - q_r_dot

    Y = Y_matrix(q, q_dot, q_r_dot, q_r_ddot)
    tau = Y @ theta_hat - K @ s

    H = H_matrix(q)
    C = C_matrix(q, q_dot)
    g_vec = g_vector(q)
    q_ddot = np.linalg.inv(H) @ (tau - C @ q_dot - g_vec)
```

```
    theta_hat_dot = -Gamma @ Y.T @ s

    return np.concatenate([q_dot, q_ddot, theta_hat_dot])

q0 = np.array([0.0, 0.0])
q_dot0 = np.array([0.0, 0.0])
theta_hat0 = np.zeros(5)
state0 = np.concatenate([q0, q_dot0, theta_hat0])

print("Starting adaptive control simulation...")
t_span = (0, 10)
t_eval = np.linspace(0, 10, 2000)
sol = solve_ivp(dynamics, t_span, state0, t_eval=t_eval, method='RK45', rtol=1e-6)

print("Simulation completed!")

t_sol = sol.t
q_sol = sol.y[0:2, :]
q_dot_sol = sol.y[2:4, :]
theta_hat_sol = sol.y[4:9, :]

q_d_sol = np.array([qd(t) for t in t_sol]).T
q_d_dot_sol = np.array([qd_dot(t) for t in t_sol]).T

q_tilde_sol = q_d_sol - q_sol
s_sol = q_dot_sol - (q_d_dot_sol - Lambda @ q_tilde_sol)

plt.figure(figsize=(15, 12))

plt.subplot(3, 2, 1)
plt.plot(t_sol, q_sol[0, :], 'b-', linewidth=2, label='Actual θ1')
plt.plot(t_sol, q_d_sol[0, :], 'r--', linewidth=2, label='Desired θ1')
plt.xlabel('Time [s]')
plt.ylabel('Joint angle [rad]')
plt.title('Joint 1 Tracking')
plt.legend()
plt.grid(True)

plt.subplot(3, 2, 2)
plt.plot(t_sol, q_sol[1, :], 'b-', linewidth=2, label='Actual θ2')
plt.plot(t_sol, q_d_sol[1, :], 'r--', linewidth=2, label='Desired θ2')
plt.xlabel('Time [s]')
plt.ylabel('Joint angle [rad]')
plt.title('Joint 2 Tracking')
plt.legend()
plt.grid(True)

plt.subplot(3, 2, 3)
plt.plot(t_sol, q_tilde_sol[0, :], 'g-', linewidth=2)
plt.xlabel('Time [s]')
plt.ylabel('Error [rad]')
plt.title('Joint 1 Tracking Error (q̃₁)')
plt.grid(True)
```

```
plt.subplot(3, 2, 4)
plt.plot(t_sol, q_tilde_sol[1, :], 'g-', linewidth=2)
plt.xlabel('Time [s]')
plt.ylabel('Error [rad]')
plt.title('Joint 2 Tracking Error (q̃₂)')
plt.grid(True)

plt.subplot(3, 2, 5)
plt.plot(t_sol, s_sol[0, :], 'm-', linewidth=2, label='s₁')
plt.plot(t_sol, s_sol[1, :], 'c-', linewidth=2, label='s₂')
plt.xlabel('Time [s]')
plt.ylabel('Sliding variable')
plt.title('Sliding Variables')
plt.legend()
plt.grid(True)

plt.subplot(3, 2, 6)
for i in range(5):
    plt.plot(t_sol, theta_hat_sol[i, :], linewidth=2, label=f'â[{i+1}]')
    plt.axhline(y=phi_true[i], color=plt.gca().lines[-1].get_color(), linestyle='--', alpha=0.5)
plt.xlabel('Time [s]')
plt.ylabel('Parameter estimates')
plt.title('Parameter Convergence (dashed: true values)')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

print("\nPerformance Analysis:")
print(f"Final Joint 1 Error: {q_tilde_sol[0, -1]:.6f} rad")
print(f"Final Joint 2 Error: {q_tilde_sol[1, -1]:.6f} rad")
print(f"RMS Joint 1 Error: {np.sqrt(np.mean(q_tilde_sol[0, :]**2)):.6f} rad")
print(f"RMS Joint 2 Error: {np.sqrt(np.mean(q_tilde_sol[1, :]**2)):.6f} rad")

print("\nFinal Parameter Estimates vs True Values:")
for i in range(5):
    error = theta_hat_sol[i, -1] - phi_true[i]
    print(f"â[{i+1}] = {theta_hat_sol[i, -1]:.4f} (True: {phi_true[i]:.4f}, Error: {error:.4f})")
```



Problem 3:

Consider the dynamic equations of a robot manipulator

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau$$

where the matrix D is constant symmetric positive definite. Consider an adaptive P.D. controller for such a robot with the control law

$$\tau = Y\hat{a} - K_d\dot{\tilde{q}} - K_p\tilde{q}$$

and the adaptation law

$$\dot{\hat{a}} = -PY^T s$$

where

$$Ya = H(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + D\dot{q}_r + g(q)$$

$$s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \lambda\tilde{q}$$

$$\tilde{q} = q - q_d$$

The adaptation and controller gain matrices $P, K_d and K_p$ are all symmetric positive definite.

Show that this controller will force the tracking error $\tilde{q}$ to converge to zero.

[Hint: You may want to use the Lyapunov function candidate

$$V = \frac{1}{2}s^T H s + \frac{1}{2}\tilde{a}^T P^{-1}\tilde{a} + \frac{1}{2}\tilde{q}^T(K_p + \lambda K_d)\tilde{q}$$

Answer:

Known:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau$$

$$\tau = Y\hat{a} - K_d\dot{\tilde{q}} - K_p\tilde{q}$$

$$Ya = H(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + D\dot{q}_r + g(q)$$

Definition of Tracking Error and Reference Trajectory:

$$\tilde{q} = q - q_d$$
$$s = \dot{\tilde{q}} + \lambda\tilde{q} = \dot{q} - \dot{q}_r$$
$$\dot{q}_r = \dot{q}_d - \lambda\tilde{q}$$

Adaptive:

$$\dot{\hat{a}} = -PY^T s$$

Known Lyapunov function candidate:

$$V = \frac{1}{2}s^T H s + \frac{1}{2}\tilde{a}^T P^{-1}\tilde{a} + \frac{1}{2}\tilde{q}^T(K_p + \lambda K_d)\tilde{q}$$

Thus:

$$\dot{V} = s^T H\dot{s} + \frac{1}{2}s^T\dot{H}s - \tilde{a}^T P^{-1}\dot{\hat{a}} + \tilde{q}^T(K_p + \lambda K_d)\dot{\tilde{q}}$$

Fort the $s^T H\dot{s}$:

$$\tau = Y\hat{a} - K_d\dot{\tilde{q}} - K_p\tilde{q} = H\dot{s} + Cs + Ds$$

$$s^T H\dot{s} = s^T(Y\tilde{a} - Cs - Ds - K_d\dot{\tilde{q}} - K_p\tilde{q})$$

For the $\tilde{a}^T P^{-1}\dot{\hat{a}}$:

$$\dot{\hat{a}} = -PY^T s$$

$$\tilde{a}^T P^{-1}\dot{\hat{a}} = -\tilde{a}^T Y^T s = -s^T Y\tilde{a}$$

Because $H^{\cdot}(q) - 2C(q, q^{\cdot})$ is skew-symmetric matrix:

$$\frac{1}{2}s^T\dot{H}s - s^T Cs = 0$$

Thus:

$$\dot{V} = -s^T Ds - s^T K_d\dot{\tilde{q}} - s^T K_p\tilde{q} + \tilde{q}^T(K_p + \lambda K_d)\dot{\tilde{q}}$$

As $s = \dot{\tilde{q}} + \lambda\tilde{q} = \dot{q} - \dot{q}_r$

$$-s^T K_d\dot{\tilde{q}} = -\dot{\tilde{q}}^T K_d\dot{\tilde{q}} - \lambda\tilde{q}^T K_d\dot{\tilde{q}}$$

$$-s^T K_p\tilde{q} = -\dot{\tilde{q}}^T K_p\tilde{q} - \lambda\tilde{q}^T K_p\tilde{q}$$

$$\tilde{q}^T(K_p + \lambda K_d)\dot{\tilde{q}} = \tilde{q}^T K_p\dot{\tilde{q}} + \lambda\tilde{q}^T K_d\dot{\tilde{q}}$$

Thus:

$$\dot{V} = -s^T D s - \dot{\tilde{q}}^T K_d \dot{\tilde{q}} - \lambda \tilde{q}^T K_p \tilde{q} \leq 0$$

Because the the adaptation and controller gain matrices $P, K_d and K_p$ are all symmetric positive definite, so $V^{\cdot} \leq 0$。

Problem 4:

Consider the following dynamics equation from the lecture:

$$J\ddot{q} + b\dot{q}|\dot{q}| + mgl\sin q = \tau$$

$$a_1 = J$$

$$a_2 = b$$

$$a_3 = mgl$$

with unknown but constant parameters $a = [a_1, a_2, a_3]^\intercal$. From the lecture and the previous problems, you should be able to design a control law and an adaptation law for this system, assuming no prior knowledge of the actual values of a_1, a_2, a_3.

Now assume that we know that parameters a_1, a_2 satisfy $a_1, a_2 > 0$ and parameter a_3 satisfies $a_3 > c$ for some known constant $c > 0$. How can we exploit this knowledge in the adaptation law? State the adaptation law, and show that the controller will still force the tracking error $\tilde{q}$ to zero.

Answer:

Know:

$$J\ddot{q} + b\dot{q}|\dot{q}| + mgl\sin q = \tau$$

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad a_1, a_2 > 0, \quad a_3 > c > 0$$

Definition of Tracking Error:

$$\begin{aligned} \tilde{q} &= q - q_d \\ \dot{q}_r &= \dot{q}_d - \lambda \tilde{q} \\ s = \dot{\tilde{q}} + \lambda \tilde{q} &= \dot{q} - \dot{q}_r \\ \dot{s} &= \ddot{q} - \ddot{q}_r \\ \ddot{q}_r &= \ddot{q}_d - \lambda \dot{\tilde{q}} \end{aligned}$$

Option1, If Define:

$$V = \frac{1}{2} s^2$$

Thus:

$$\dot{V} = s\dot{s} = s(\ddot{q} - \ddot{q}_r)$$

$$\ddot{q} = J^{-1}(\tau - b\dot{q}|\dot{q}| - mgl\sin q)$$

$$\dot{V} = s\dot{s} = s(\tau - J\ddot{q}r - b\dot{q}|\dot{q}| - mgl\sin q)) = s(\tau - Ya)$$

Design control and adaptation law:

$$\tau = Y\hat{a} - ks$$

Thus:

$$\dot{V} = s(Y\hat{a} - ks - Ya) = sY(\hat{a} - a) - ks^2$$

$$\dot{V} = -ks^2 \leq 0$$

Option2, If Define:

$$V = \frac{1}{2}Js^2 + \frac{1}{2}\tilde{a}^T P^{-1}\tilde{a} \geq 0$$

Thus:

$$\dot{V} = sJ\dot{s} + \tilde{a}^T P^{-1}\dot{\hat{a}} = -ks^2 + \left(sY\tilde{a} + \tilde{a}^T P^{-1}\dot{\hat{a}}\right)$$

Design control and adaptation law:

$$\tau = Y\hat{a} - ks$$

$$\dot{\hat{a}} = -PY^T s$$

Thus:

$$\dot{V} = -ks2 \leq 0$$

The controller will still force the tracking error $\tilde{q}$ to zero.