Solutions_Problem1.m

```
m1 = 7.848;
m2 = 4.49;

l1 = 0.3;
lc1 = 0.1554;
lc2 = 0.0341;

I1 = 0.176;
I2 = 0.0411;

x0 = [0.05; 0.05; 0; 0];
tspan = [0,2];
[t,x] = ode45(@(t,x) EoM(t,x,1), tspan, x0); % third argument is the problem number

for a = 1:length(t)
    if t(a) < 1
        q_d(a) = pi/2; %#ok<SAGROW>
    else
        q_d(a) = 0; %#ok<SAGROW>
    end
end

% % First Plot
figure(1); clf(1)
plot(t, q_d, 'k')
hold on
plot(t, x(:,1),'b')
plot(t, x(:,2),'r')
xlabel('Time (s)'); ylabel('Position (rad)')
title('Link Responses')
legend('q_d(t)','q_1(t)','q_2(t)')

% % Second Plot
```

```
tau1 = limit_vec(100 * (q_d' - x(:,1)) - 20 * x(:,3));
tau2 = limit_vec(100 * (q_d' - x(:,2)) - 20 * x(:,4));

figure(2); clf(2)
hold on
plot(t, tau1, 'b')
plot(t, tau2, 'r')
legend('\tau_1(t)','\tau_2(t)')
xlabel('Time (s)'); ylabel('Torque (Nm)')
title('Joint Torque')

% % Third Plot
figure(3); clf(3)
hold on
plot(t, x(:,1) - q_d', 'b')
plot(t, x(:,2) - q_d', 'r')
yline(0)
legend([char([double('q'),771]),'_1(t)'],[char([double('q'),771]),'_2(t)'])
xlabel('Time (s)'); ylabel('Error (rad)')
title('Position Error')


%% Helper Function
function tau = limit_vec(tau)
    tau(tau>10) = 10;
    tau(tau<-10) = -10;
end
```

Solutions_Problem2.m

```
clear
m1 = 7.848;
m2 = 4.49;

l1 = 0.3;
lc1 = 0.1554;
lc2 = 0.0341;

I1 = 0.176;
I2 = 0.0411;

kp = 100;
kd = 20;

x0 = [0.05;0.05;0;0];
tspan = [0,2];
[t,x] = ode45(@(t,x) EoM(t,x,2), tspan, x0); % third argument is the problem number

[q_d, qd_d, qdd_d] = cubicTraj([0, pi/2, 0], [0, 1, 2], [0, 0, 0], t);

tau1 = limit_vec( qdd_d' + kp*(q_d' - x(:,1)) + kd*(qd_d' - x(:,3)) );
tau2 = limit_vec( qdd_d' + kp*(q_d' - x(:,2)) + kd*(qd_d' - x(:,4)) );

% First Plot
figure(1); clf(1)
plot(t, q_d, 'k')
hold on
plot(t, x(:,1),'b')
plot(t, x(:,2),'r')
xlabel('Time (s)'); ylabel('Position (rad)')
title('Link Responses')
legend('q_d(t)','q_1(t)','q_2(t)')
```

```matlab
% Second Plot
figure(2); clf(2)
hold on
plot(t, tau1, 'b')
plot(t, tau2, 'r')
legend('\tau_1(t)','\tau_2(t)')
xlabel('Time (s)'); ylabel('Torque (Nm)')
title('Joint Torque')

% Third Plot
figure(3); clf(3)
hold on
plot(t, x(:,1) - q_d', 'b')
plot(t, x(:,2) - q_d', 'r')
yline(0)
legend([char([double('q'),771]),'_1(t)'],[char([double('q'),771]),'_2(t)'])
xlabel('Time (s)'); ylabel('Error (rad)')
title('Position Error')


%% Helper Function
function tau = limit_vec(tau)
    tau(tau>10) = 10;
    tau(tau<-10) = -10;
end
```

Solutions_Problem3.m

```
clear
m1 = 7.848;
m2 = 4.49;

l1 = 0.3;
lc1 = 0.1554;
lc2 = 0.0341;

I1 = 0.176;
I2 = 0.0411;

kp = 100;
kd = 20;

x0 = [0.05;0.05;0;0];
tspan = [0,2];
[t,x] = ode45(@(t,x) EoM(t,x,3), tspan, x0); % third argument is the problem number

[q_d, qd_d, qdd_d] = cubicTraj([0, pi/2, 0], [0 1 2], [0 0 0], t);


for a = 1:length(q_d)

    % Calculate the Matrices
    M11 = m1 * lc1^2 + m2 * (l1^2 + lc2^2 + 2 * l1 * lc2 * cos(x(a,2))) + I1 + I2;
    M12 = m2 * (lc2^2 + l1 * lc2 * cos(x(a,2))) + I2;
    M22 = m2 * lc2^2 + I2;

    % M11d1 = 0;
    M11d2 = -2 * m2 * l1 * lc2 * sin(x(a,2));
    % M12d1 = 0;
    M12d2 = -m2 * l1 * lc2 * sin(x(a,2));
    % M22d1 = 0;
```

```
    % M22d2 = 0;

    derivs(2,2,2) = 0;
    derivs(1,1,2) = M11d2;
    derivs(1,2,2) = M12d2;
    derivs(2,1,2) = M12d2;

    for i = 1:2
        for j = 1:2
            for k = 1:2
                chris(i,j,k) = 0.5 * (derivs(i,j,k) + derivs(i,k,j) - derivs(k,j,i));  %#ok<SAGROW>
            end
        end
    end
    C(:,:,1) = chris(:,:,1)*x(a,3);
    C(:,:,2) = chris(:,:,2)*x(a,4);
    C = sum(C,3);

    aq1(a) = qdd_d(a) + kp*(q_d(a) - x(a,1)) + kd*(qd_d(a) - x(a,3)); %#ok<SAGROW>
    aq2(a) = qdd_d(a) + kp*(q_d(a) - x(a,2)) + kd*(qd_d(a) - x(a,4)); %#ok<SAGROW>
    tau1(a) = limit_vec([M11, M12]*[aq1(a); aq2(a)] + C(1,:)*[x(a,3); x(a,4)]); %#ok<SAGROW>
    tau2(a) = limit_vec([M12, M22]*[aq1(a); aq2(a)] + C(2,:)*[x(a,3); x(a,4)]); %#ok<SAGROW>

end

% First Plot
figure(1); clf(1)
plot(t, q_d, 'k')
hold on
plot(t, x(:,1),'b')
plot(t, x(:,2),'r')
xlabel('Time (s)'); ylabel('Position (rad)')
title('Link Responses')
legend('q_d(t)','q_1(t)','q_2(t)')

% Second Plot
```

```
figure(2); clf(2)
hold on
plot(t, tau1, 'b')
plot(t, tau2, 'r')
legend('\tau_1(t)','\tau_2(t)')
xlabel('Time (s)'); ylabel('Torque (Nm)')
title('Joint Torque')

% Third Plot
figure(3); clf(3)
hold on
plot(t, x(:,1) - q_d', 'b')
plot(t, x(:,2) - q_d', 'r')
yline(0)
legend([char([double('q'),771]),'_1(t)'],[char([double('q'),771]),'_2(t)'])
xlabel('Time (s)'); ylabel('Error (rad)')
title('Position Error')


%% Helper Function
function tau = limit_vec(tau)
    tau(tau>10) = 10;
    tau(tau<-10) = -10;
end
```

EoM.m

```
function xdot = EoM(t, x, prob)
% Inputs:   x:      a vector of the state variables
%           t:      time
%           prob:   the problem number of the hw (used to define tau)
%
% Output:   xdot: the derivative of the state variable x

% % Given Variables
m1 = 7.848;
m2 = 4.49;

l1 = 0.3;
lc1 = 0.1554;
lc2 = 0.0341;

I1 = 0.176;
I2 = 0.0411;

kp = 100;
kd = 20;


% % Calculate the Matrices - from Mathematica
M11 = m1 * lc1^2 + m2 * (l1^2 + lc2^2 + 2 * l1 * lc2 * cos(x(2))) + I1 + I2;
M12 = m2 * (lc2^2 + l1 * lc2 * cos(x(2))) + I2;
M22 = m2 * lc2^2 + I2;

% % Calculate the derivs by hand and then hard code
% M11d1 = 0;
M11d2 = -2 * m2 * l1 * lc2 * sin(x(2));
% M12d1 = 0;
M12d2 = -m2 * l1 * lc2 * sin(x(2));
% M22d1 = 0;
```

```
% M22d2 = 0;

derivs(2,2,2) = 0;
derivs(1,1,2) = M11d2;
derivs(1,2,2) = M12d2;
derivs(2,1,2) = M12d2;

for i = 1:2
    for j = 1:2
        for k = 1:2
            chris(i,j,k) = 0.5 * (derivs(i,j,k) + derivs(i,k,j) - derivs(k,j,i)); %#ok<AGROW>
        end
    end
end
C(:,:,1) = chris(:,:,1)*x(3);
C(:,:,2) = chris(:,:,2)*x(4);
C = sum(C,3);


% % Define Tau based on problem number
switch prob
    case 1
        if t < 1
            qd = pi/2;
        else
            qd = 0;
        end
        tau1 = limit(kp * (qd - x(1)) - kd * x(3));
        tau2 = limit(kp * (qd - x(2)) - kd * x(4));
    case 2
        [q_d, qd_d, qdd_d] = cubicTraj([0, pi/2, 0], [0 1 2], [0 0 0], t);
        tau1 = limit( qdd_d + kp*(q_d - x(1)) + kd*(qd_d - x(3)) );
        tau2 = limit( qdd_d + kp*(q_d - x(2)) + kd*(qd_d - x(4)) );
    case 3
        [q_d, qd_d, qdd_d] = cubicTraj([0, pi/2, 0], [0 1 2], [0 0 0], t);
        aq1 = qdd_d + kp*(q_d - x(1)) + kd*(qd_d - x(3));
```

```
        aq2 = qdd_d + kp*(q_d - x(2)) + kd*(qd_d - x(4));
        tau1 = limit([M11, M12]*[aq1; aq2] + C(1,:)*[x(3); x(4)]);
        tau2 = limit([M12, M22]*[aq1; aq2] + C(2,:)*[x(3); x(4)]);
    otherwise
        error('Invalid problem number')
end


% % Making the Derivative
a1 = tau1 - chris(1,2,1)*x(3)*x(4) - chris(2,1,1)*x(4)*x(3) - chris(2,2,1)*x(4)^2;
a2 = tau2 - chris(1,1,2)*x(3)^2;

delta = M11*M22 - M12*M12;

qddot1 = (1/delta) * (M22*a1 - M12*a2);
qddot2 = (1/delta) * (M11*a2 - M12*a1);

xdot = [x(3); x(4); qddot1; qddot2];

end


function tauLimited = limit(tau)
    if tau > 10
        tauLimited = 10;
    elseif tau < -10
        tauLimited = -10;
    else
        tauLimited = tau;
    end
end
```

cubicTraj.m

```
function [q,qd,qdd] = cubicTraj(y, t, v, t_i)
% Function is written specifically for this HW
% Inputs:   t:      the time points (1x3)
%           y:      the positions at time points (1x3)
%           v:      the velocity at time points (1x3)
%           t_i:    (optional) the time index you want to return, otherwise
%                   the entire vector for t is returned
%
% Outputs:  q:      the position
%           qd:     the velocity
%           qdd:    the acceleration

if ~exist('t_i','var')
    t_i = t(1):0.01:t(end);
end

[q,qd,qdd,~] = cubicpolytraj(y, t, t_i, 'VelocityBoundaryCondition', v); % pre-made matlab function

end
```