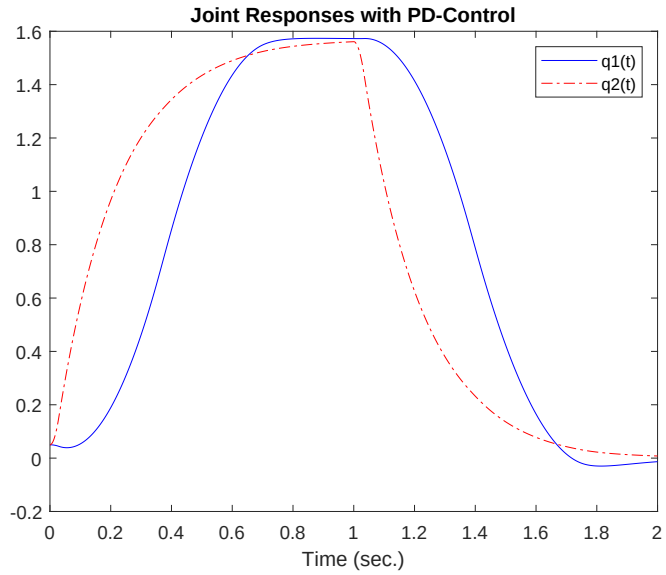
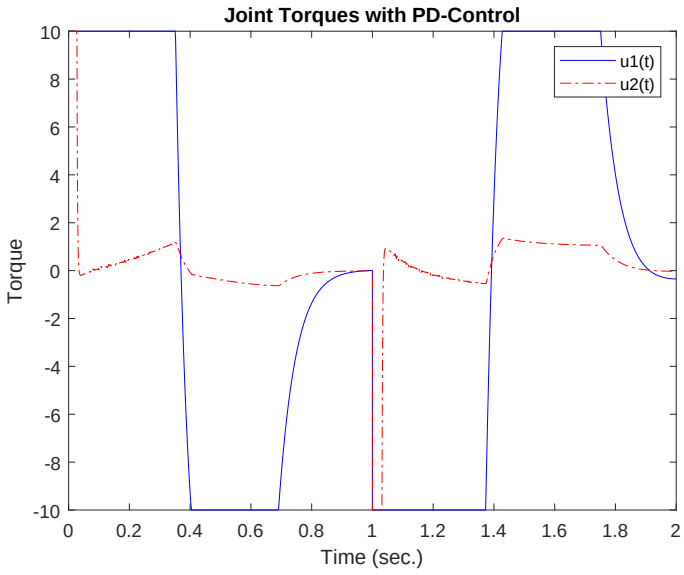
Course: Mech 567: Robot Kinematics and Dynamics
Instructor: Robert Gregg, PhD

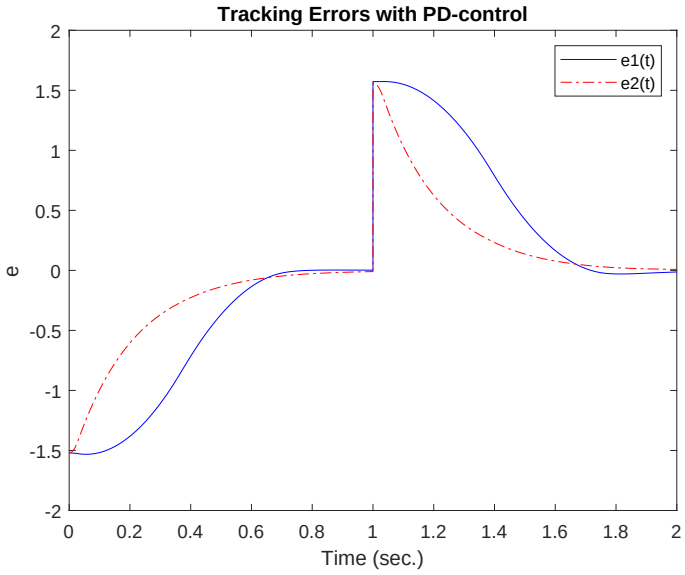
Solution 6

Problem 1

(a)



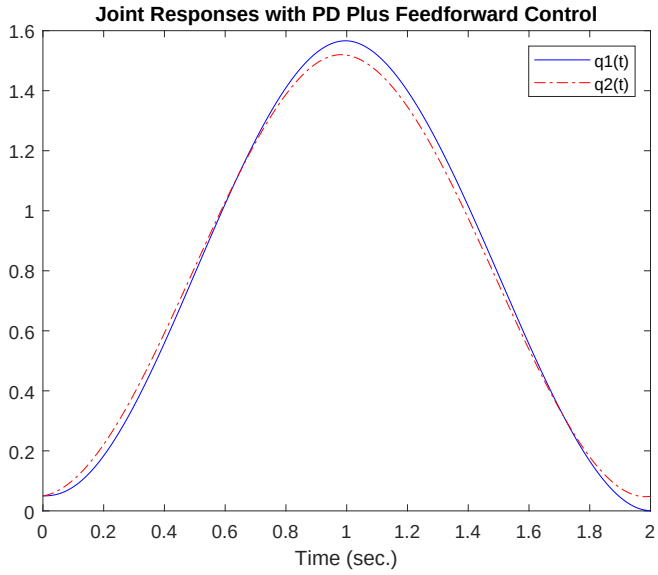


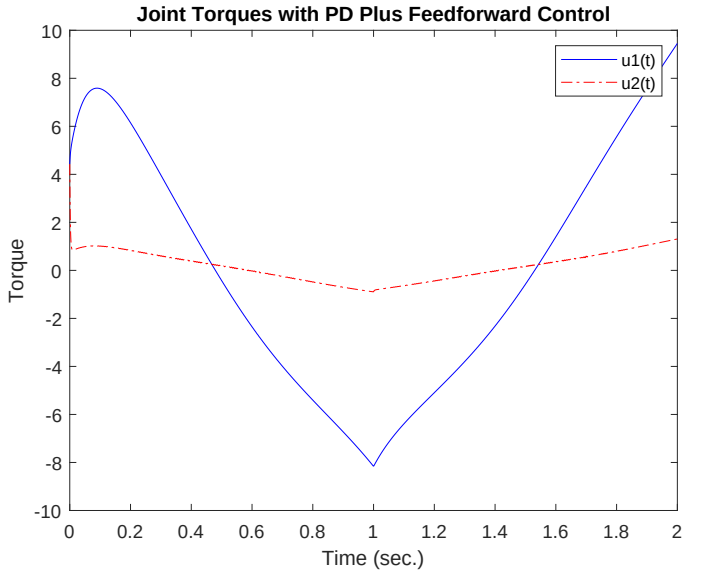


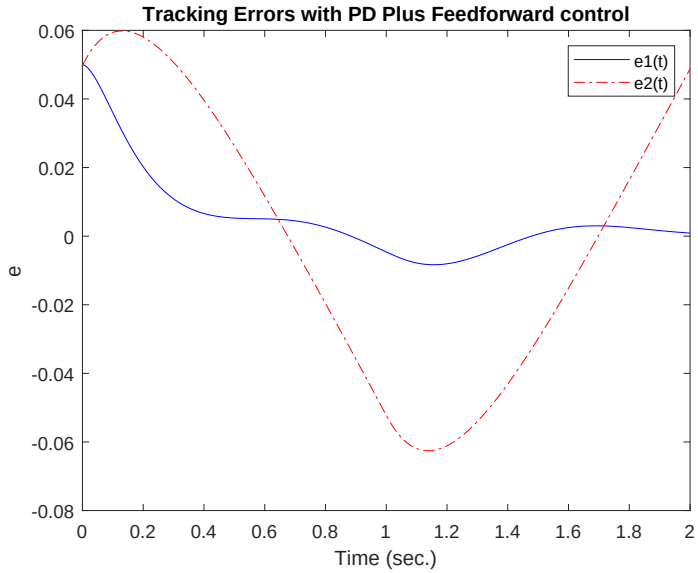
(b)
at $t = 1$, $e_1 \approx 0.0018$, $e_2 \approx -0.0098$, and
at $t = 2$, $e_1 \approx -0.0132$, $e_2 \approx 0.0080$.

Problem 2

(a)



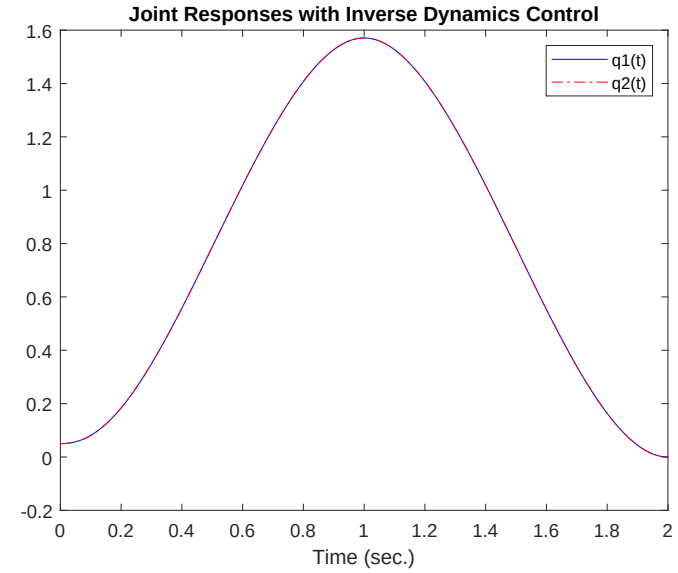


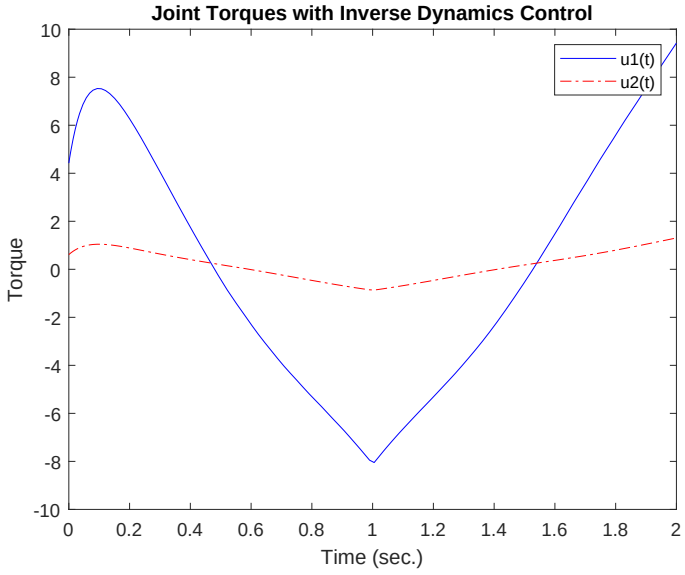


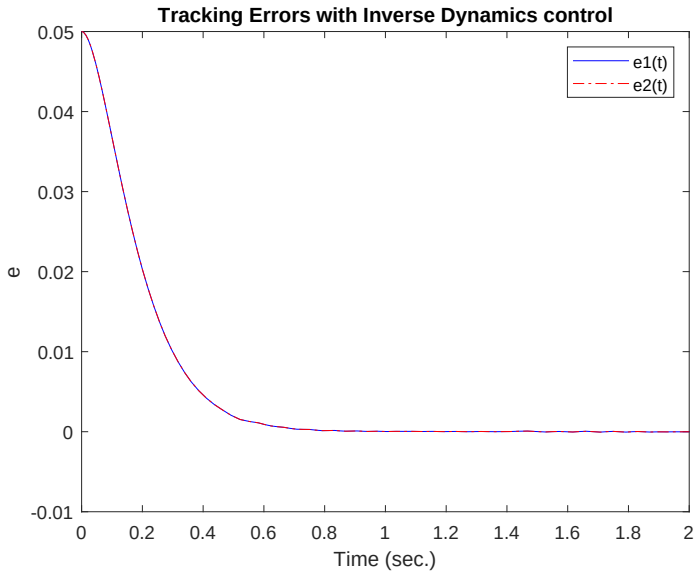
(b)
at $t = 1$, $e_1 \approx -0.0046$, $e_2 \approx -0.0521$, and
at $t = 2$, $e_1 \approx 0.0009$, $e_2 \approx 0.0487$.
The joint tracking errors are drastically smaller for PD+Feedforward Control. The joints are intelligently following smooth, interpolated paths between desired set-points, whereas the PD control law clumsily follows non-smooth step functions. Additionally, the PD+Feedforward torques are smaller in magnitude and never saturate the actuators.

Problem 3

(a)


Joint Responses with Inverse Dynamics Control
q1(t)
q2(t)
Time (sec.)


Joint Torques with Inverse Dynamics Control
u1(t)
u2(t)
Torque
Time (sec.)



(b)
at $t = 1$, $e_1 \approx 2.17e-5$, $e_2 \approx 2.17e-5$, and
at $t = 2$, $e_1 \approx -8.90e-6$, $e_2 \approx -8.90e-6$.
The initial error is about the same magnitude as the overall tracking error for the PD+feedforward controllers. As for the joint torques, the PD+Feedforwards and Inverse Dynamics torque plots are very similar in both magnitude and shape. Thus, the Inverse Dynamics controller has smaller torques than the pure PD controller.

Problem 4

The pure PD controller is a rather clumsy method, but it is by far the simplest strategy of the three. Moreover, the PD-controlled system approaches the desired set-points quicker than the other methods. PD+feedforward is a drastic improvement over the pure PD controller in terms of overall error tracking and torque magnitudes. On the other hand, the pure PD controller asymptotically converges to zero error, whereas the PD+feedforward just stays bounded close to zero. The Inverse Dynamics controller uses the PD+feedforwad controller as its outer loop, having an inverse dynamics/linearization control law as its inner loop. Inverse Dynamics has asymptotic convergence to zero error since it linearizes the error dynamics, which is an improvement over the PD+feedforward error tracking that does not converge to zero but rather stays bounded. The Inverse Dynamics controller has very similar torques to the PD+feedforward controller, thus having smaller torques than the pure PD controller. On the other hand, the Inverse Dynamics controller is completely wiping out the natural dynamics of the robot, which may be excessive and in principle could be demanding on the actuators during more complicated tasks.

MatLab codes are shown below

```
function xdot = manipulator(t, x, controller)
% x =[q1;q1dot;q2;q2dot];
q1 = x(1); q1dot = x(2); q2 = x(3); q2dot = x(4);
m1 = 7.848; m2 = 4.49; l1 = 0.3;
lc1 = 0.1554; lc2 = 0.0341; I1 = 0.176; I2 = 0.0411;
D = zeros(2,2);
D(1,1) = m1*lc1^2 + m2*(l1^2 + lc2^2 + 2*l1*lc2*cos(q2))+I1+I2;
D(1,2) = m2*(lc2^2+l1*lc2*cos(q2))+I2;
D(2,1) = D(1,2);
D(2,2) = m2*lc2^2+I2;
C121 = -m2*l1*lc2*sin(q2);
C211 = C121;
C221 = C121;
C112 = -C121;

[tau, ~] = controller(t, x);

a1 = tau(1) - C121*q1dot.*q2dot-C211*q2dot*q1dot-C221*q2dot.^2;
a2 = tau(2) - C112*q1dot.^2;
xdot = [x(2);
        1./(D(1,1).*D(2,2) - D(2,1).*D(1,2)).*(D(2,2).*a1 - D(1,2).*a2)
           ;
        x(4);
        1./(D(1,1).*D(2,2) - D(2,1).*D(1,2)).*(-D(2,1).*a1 + D(1,1).*a2
           );];
```

```
function [qd, vd, ad] = cubicpoly(t)
if t<=1 && t>=0
    a = [1 0 0 0;
         0 1 0 0;
         1 1 1 1;
         0 1 2 3;]\[0;0;pi/2;0];
elseif t>1
    a = [1 1 1 1;
         0 1 2 3;
         1 2 4 8;
         0 1 4 12;]\[pi/2;0;0;0];
end
qd = [a'*[1;t;t.^2;t.^3];
      a'*[1;t;t.^2;t.^3];];
vd = [a'*[0;1;2*t;3*t.^2];
      a'*[0;1;2*t;3*t.^2];];
ad = [a'*[0;0;2;6*t];
      a'*[0;0;2;6*t];];
```

```
function [tau, e] = PDcontrol(t, x)
% x =[q1;q1dot;q2;q2dot];
```

```
q1 = x(1); q1dot = x(2); q2 = x(3); q2dot = x(4);
kp1 = 100; kp2 = 100;
kd1 = 20; kd2 = 20;

if t<=1 && t>=0
    q1d = pi/2;
    q2d = pi/2;
elseif t>1
    q1d = 0;
    q2d = 0;
end
tau = [min(max(-10, kp1*(q1d-q1)-kd1*q1dot), 10);
       min(max(-10, kp2*(q2d-q2)-kd2*q2dot), 10)];
e = [q1 - q1d;
     q2 - q2d;];
```

```
function [tau, e] = PDPFcontrol(t, x)
% x =[q1;q1dot;q2;q2dot];
q1 = x(1); q1dot = x(2); q2 = x(3); q2dot = x(4);
kp1 = 100; kp2 = 100;
kd1 = 20; kd2 = 20;

[qd, vd, ad] = cubicpoly(t);

tau = [min(max(-10, ad(1)+kp1*(qd(1)-q1)+kd1*(vd(1)-q1dot)), 10);
       min(max(-10, ad(2)+kp2*(qd(2)-q2)+kd2*(vd(2)-q2dot)), 10)];
e = [q1 - qd(1);
     q2 - qd(2);];
```

```
function [tau, e] = InverseDynamicscontrol(t, x)
% x =[q1;q1dot;q2;q2dot];
q1 = x(1); q1dot = x(2); q2 = x(3); q2dot = x(4);
kp1 = 100; kp2 = 100;
kd1 = 20; kd2 = 20;
m1 = 7.848; m2 = 4.49; l1 = 0.3;
lc1 = 0.1554; lc2 = 0.0341; I1 = 0.176; I2 = 0.0411;
D = zeros(2,2);
D(1,1) = m1*lc1^2 + m2*(l1^2 + lc2^2 + 2*l1*lc2*cos(q2))+I1+I2;
D(1,2) = m2*(lc2^2+l1*lc2*cos(q2))+I2;
D(2,1) = D(1,2);
D(2,2) = m2*lc2^2+I2;
C121 = -m2*l1*lc2*sin(q2);
C211 = C121;
C221 = C121;
C112 = -C121;
[qd, vd, ad] = cubicpoly(t);
aq = ad + [kp1;kp2]*(qd(1)-q1)+[kd1;kd2]*(vd(1)-q1dot);
tau = D*aq + [C121*q1dot.*q2dot+C211*q2dot*q1dot+C221*q2dot.^2;C112*
```

```
    q1dot.^2;];
tau = [min(max(-10, tau(1)), 10);
       min(max(-10, tau(2)), 10)];
e = [q1 - qd(1);
     q2 - qd(2);];
```

```
clear;clc;close all
x0 = [0.05;0;0.05;0];
[T, X] = ode45(@(t, x)manipulator(t, x, @PDcontrol), [0, 2], x0);

figure(1)
plot(T, X(:,1), 'b', T, X(:,3), 'r-.')
legend('q1(t)', 'q2(t)')
title('Joint Responses with PD-Control')
xlabel('Time (sec.)')

tau = zeros(2,length(T));
e = zeros(2,length(T));
for i = 1:length(T)
    [tau(:,i), e(:,i)] =PDcontrol(T(i), X(i,:));
end
figure(2)
plot(T, tau(1,:), 'b', T, tau(2,:), 'r-.')
legend('u1(t)', 'u2(t)')
title('Joint Torques with PD-Control')
xlabel('Time (sec.)')
ylabel('Torque')

figure(3)
plot(T, e(1,:), 'b', T, e(2,:), 'r-.')
legend('e1(t)', 'e2(t)')
title('Tracking Errors with PD-control')
xlabel('Time (sec.)')
ylabel('e')
```

```
clear;clc;close all

t = 0:0.01:2;
qd = zeros(2,length(t));
vd = zeros(2,length(t));
ad = zeros(2,length(t));
for i = 1:length(t)
    [qd(:,i), vd(:,i), ad(:,i)] = cubicpoly(t(i));
end
figure(1)
plot(t, qd(1,:), 'b', t, qd(2,:), 'r-.')
legend('qd1(t)', 'qd2(t)')
title('Cubic Polynomial Reference Trajectory')
```

```
xlabel('Time (sec.)')
figure(2)
plot(t, vd(1,:), 'b', t, vd(2,:), 'r-.')
legend('vd1(t)', 'vd2(t)')
title('Reference Velocity')
xlabel('Time (sec.)')
figure(3)
plot(t, ad(1,:), 'b', t, ad(2,:), 'r-.')
legend('ad1(t)', 'ad2(t)')
title('Reference Acceleration')
xlabel('Time (sec.)')

x0 = [0.05;0;0.05;0];
[T, X] = ode45(@(t, x)manipulator(t, x, @PDPFcontrol), [0, 2], x0);

figure(4)
plot(T, X(:,1), 'b', T, X(:,3), 'r-.')
legend('q1(t)', 'q2(t)')
title('Joint Responses with PD Plus Feedforward Control')
xlabel('Time (sec.)')

tau = zeros(2,length(T));
e = zeros(2,length(T));
for i = 1:length(T)
    [tau(:,i), e(:,i)] =PDPFcontrol(T(i), X(i,:));
end
figure(5)
plot(T, tau(1,:), 'b', T, tau(2,:), 'r-.')
legend('u1(t)', 'u2(t)')
title('Joint Torques with PD Plus Feedforward Control')
xlabel('Time (sec.)')
ylabel('Torque')

figure(6)
plot(T, e(1,:), 'b', T, e(2,:), 'r-.')
legend('e1(t)', 'e2(t)')
title('Tracking Errors with PD Plus Feedforward control')
xlabel('Time (sec.)')
ylabel('e')
```

```
clear;clc;close all

x0 = [0.0;0;0.0;0];
[T, X] = ode45(@(t, x)manipulator(t, x, @InverseDynamicscontrol), [0,
    2], x0);

figure(1)
plot(T, X(:,1), 'b', T, X(:,3), 'r-.')
```

```
legend('q1(t)', 'q2(t)')
title('Joint Responses with Inverse Dynamics Control')
xlabel('Time (sec.)')

tau = zeros(2,length(T));
e = zeros(2,length(T));
for i = 1:length(T)
    [tau(:,i), e(:,i)] =InverseDynamicscontrol(T(i), X(i,:));
end
figure(2)
plot(T, tau(1,:), 'b', T, tau(2,:), 'r-.')
legend('u1(t)', 'u2(t)')
title('Joint Torques with Inverse Dynamics Control')
xlabel('Time (sec.)')
ylabel('Torque')

figure(3)
plot(T, e(1,:), 'b', T, e(2,:), 'r-.')
legend('e1(t)', 'e2(t)')
title('Tracking Errors with Inverse Dynamics control')
xlabel('Time (sec.)')
ylabel('e')
```