Assignment 5

By- Aishwary Jagetia

Contents

testing algorithm with a set of initial and final states.

```
clc
clear all
close all
%Example to demostrate the robust control with planar 2d arm.
theta10=-0.5;
dtheta10 =0;
theta1f = 0.8;
dtheta1f=0;
tf=30;

% plan a trajectory to reach target postion given by theta1f, dot theta1f,
% theta2f, dot theta2f.
theta20=-1;
dtheta20= 0.1;
theta2f = 0.5;
dtheta2f=0;

robustControl(theta10,theta20,dtheta10, dtheta20,theta1f, theta2f,dtheta1f,dtheta2f,tf)%plan_control(theta10,theta20,dtheta10, dtheta20,theta1f, theta2f,dtheta1f,dt

function []= robustControl(theta10,theta20,dtheta10, dtheta20,theta1f, theta2f,dtheta1f,dtheta2f,tf)
```

Robust control design for 2-D planar arm.

input: initial and final state. output: Demostrate the performance of robust controller with parameter uncertainty.

```
% the nominal model parameter:
m1 =10; m2=5; l1=1; l2=1; r1=0.5; r2 =.5; I1=10/12; I2=5/12; % parameters in the paper.
% the nominal parameter vector b0 is
b0 = [ m1* r1^2 + m2*l1^2 + I1; m2*r2^2 + I2; m2*l1*r2];

% generating candidate for bound
gama1 = 0.06;
gama2 = 0.01;
gama3 = 0.03;

global torque
torque=[];
```

Trajectory planning block

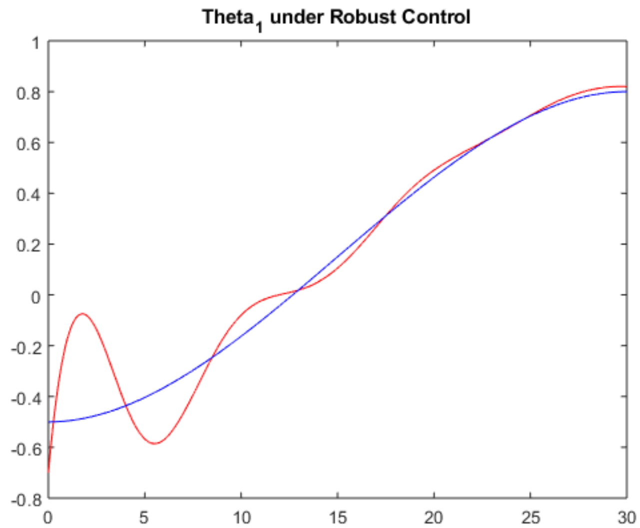
Initial condition (TODO: CHANGE DIFFERENT INITIAL AND FINAL STATES)

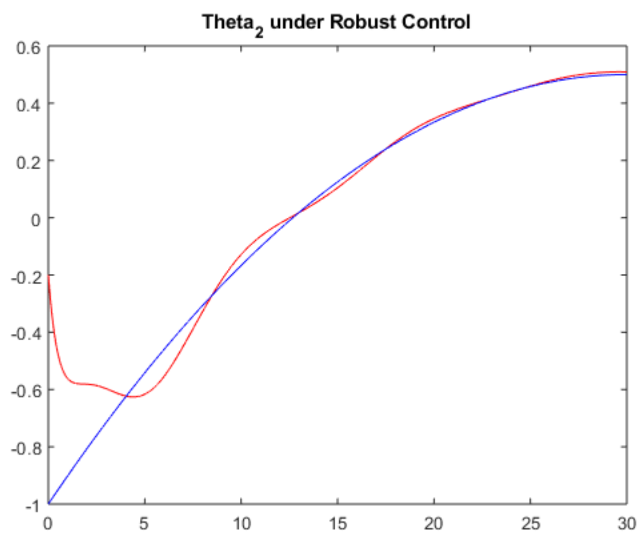
```
x0=[-0.5,-1,0,0.1];
x0e = [-0.7,-0.2,0.5,0]; % an error in the initial state.
xf=[0.8,0.5, 0, 0];
% The parameter for planned joint trajectory 1 and 2.
global a1 a2 % two polynomial trajectory for the robot joint
nofigure=1;
% Traj generation.
a1 = planarArmTraj(theta10,dtheta10, theta1f, dtheta1f,tf, nofigure);
a2 = planarArmTraj(theta20,dtheta20, theta2f, dtheta2f,tf, nofigure);

options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
```

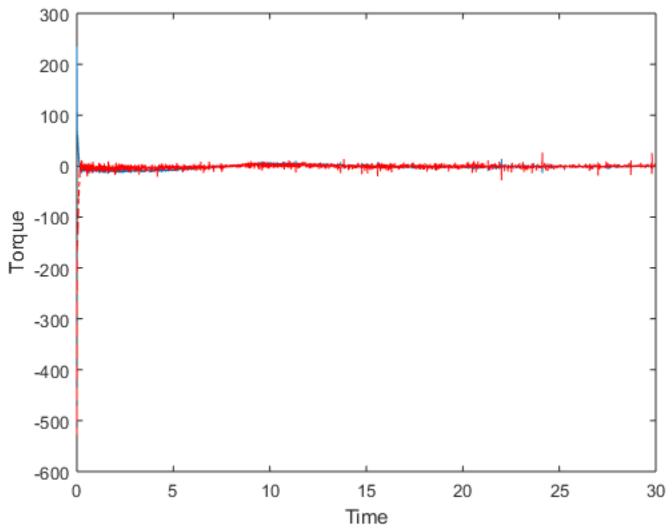
TODO: IMPLEMENT THE CONTROLLER

```
[T,X] = ode45(@(t,x)planarArmODERobust(t,x),[0 tf],x0e,options);

figure('Name','theta1');
plot(T, X(:,1),'r-');
hold on
```

```
plot(T, a1(1)+a1(2)*T+ a1(3)*T.^2+a1(4)*T.^3,'b-');
title('Theta_1 under Robust Control');
figure('Name','theta2');
plot(T, X(:,2),'r-');
hold on
plot(T, a2(1)+a2(2)*T+ a2(3)*T.^2+a2(4)*T.^3, 'b-');
title('Theta_2 under Robust Control');

%TODO: PLOT THE INPUT TRAJECTORY
figure('Name','Torque: Robust Control');
plot(T, torque(1,1:size(T,1)),'-');
hold on
plot(T, torque(2,1:size(T,1)),'r--');
hold on
xlabel('Time');
ylabel('Torque');
hold off
torque=[];
```

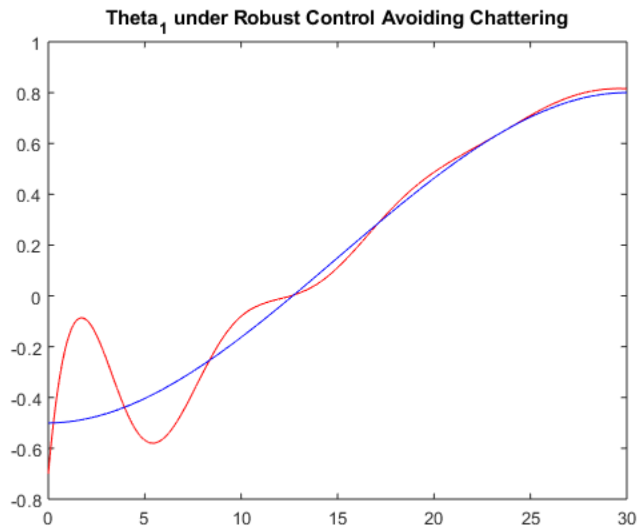


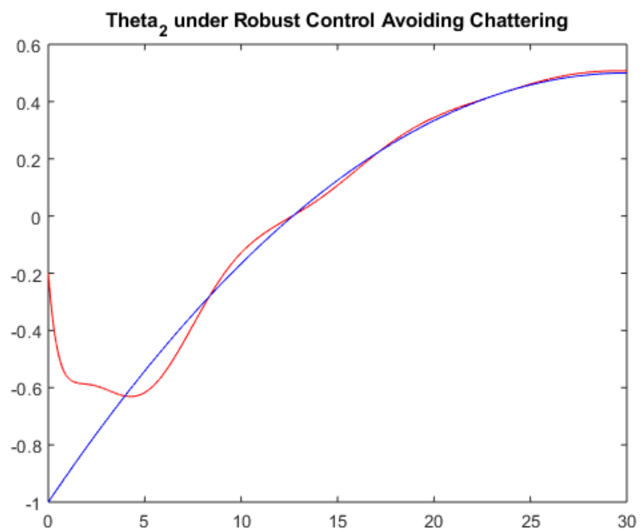


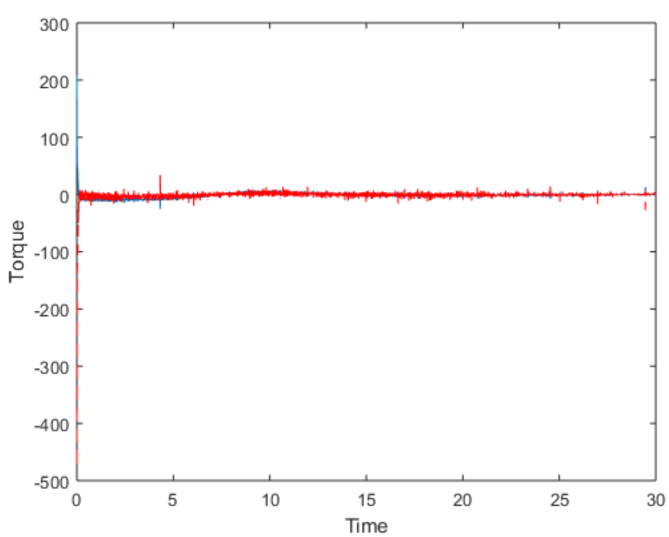


TODO: IMPLEMENT THE CONTROLLER TO AVOID CHATTERING.

```
[T,X] = ode45(@(t,x)planarArmODERobustApprx(t,x),[0 tf],x0e,options);

figure('Name','theta1 Avoiding Chattering');
plot(T, X(:,1),'r-');
hold on
plot(T, a1(1)+a1(2)*T+ a1(3)*T.^2+a1(4)*T.^3,'b-');
title('Theta_1 under Robust Control Avoiding Chattering');
figure('Name','theta2');
plot(T, X(:,2),'r-');
hold on
plot(T, a2(1)+a2(2)*T+ a2(3)*T.^2+a2(4)*T.^3, 'b-');
title('Theta_2 under Robust Control Avoiding Chattering');

%TODO: PLOT THE INPUT TRAJECTORY
figure('Name','Torque: Robust Control Avoiding Chattering');
plot(T, torque(1,1:size(T,1)),'-');
hold on
plot(T, torque(2,1:size(T,1)),'r--');
hold on
xlabel('Time');
ylabel('Torque');
hold off
torque=[];
```







IMPLEMENTING THE CONTROLLER

```
function [dx ] = planarArmODERobust(t,x)

    % Compute the desired state and their time derivatives from planned
    % trajectory.
    vec_t = [1; t; t^2; t^3]; % cubic polynomials
    %ref = [ref,theta_d];
```

```
    % compute the velocity and acceleration in both theta 1 and theta2.
    a1_vel = [a1(2), 2*a1(3), 3*a1(4), 0];
    a1_acc = [2*a1(3), 6*a1(4),0,0 ];
    a2_vel = [a2(2), 2*a2(3), 3*a2(4), 0];
    a2_acc = [2*a2(3), 6*a2(4),0,0 ];

    theta_d= [a1'*vec_t; a2'*vec_t]; % [x1d;x2d] Desired Position
    dtheta_d =[a1_vel*vec_t; a2_vel* vec_t]; % [x1d_dot;x2d_dot]
    ddtheta_d =[a1_acc*vec_t; a2_acc* vec_t]; % [x1d_ddot;x2d_ddot]
    theta= x(1:2,1); % [x1;x2]=[x(1);x(2)]
    dtheta= x(3:4,1); % [x1_dot;x2_dot]=[x(3);x(4)]

    %the true model
    m2t = m2+ 10*rand(1);% m1 true value is in [m1, m1+epsilon_m1] and epsilon_m1 a random number in [0,10];
    r2t = r2 + 0.5*rand(1);
    I2t = I2 + (15/12)*rand(1);

    at = I1+I2t+m1*r1^2+ m2t*(l1^2+ r2t^2);
    bt = m2t*l1*r2t;
    dt = I2t+ m2t*r2t^2;
    % the actual dynamic model of the system is characterized by M and
    % C

    global Mmat Cmat
    Mmat = [at+2*bt*cos(x(2)), dt+bt*cos(x(2));  dt+bt*cos(x(2)), dt];
    Cmat = [-bt*sin(x(2))*x(4), -bt*sin(x(2))*(x(3)+x(4)); bt*sin(x(2))*x(3),0];
    invM = inv(Mmat);
    invMC = invM*Cmat;

    global Mn Cn
    [Mn,Cn] = MCbarmatrix(x);

    % norm(invM*Mn - eye(2))

    v1 = addedinput(theta_d,dtheta_d,ddtheta_d,theta,dtheta,1);

    %TODO: compute the robust controller
    tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta,v1);
    torque = [torque, tau];

    %TODO: update the system state, compute dx
    dx=zeros(4,1);
    dx(1) = x(3);
    dx(2) = x(4);
    dx(3:4) = -invMC* x(3:4) +invM*tau; % because ddot theta = -M^{-1}(C \dot Theta) + M^{-1} tau
end
```

IMPLEMENTING THE CONTROLLER TO AVOID CHATTERING.

```
function [dx ] = planarArmODERobustApprx(t,x)
    % Compute the desired state and their time derivatives from planned
    % trajectory.
    vec_t = [1; t; t^2; t^3]; % cubic polynomials
    theta_d= [a1'*vec_t; a2'*vec_t];
    %ref = [ref,theta_d];
    % compute the velocity and acceleration in both theta 1 and theta2.
    a1_vel = [a1(2), 2*a1(3), 3*a1(4), 0];
    a1_acc = [2*a1(3), 6*a1(4),0,0 ];
    a2_vel = [a2(2), 2*a2(3), 3*a2(4), 0];
    a2_acc = [2*a2(3), 6*a2(4),0,0 ];
    dtheta_d =[a1_vel*vec_t; a2_vel* vec_t];
    ddtheta_d =[a1_acc*vec_t; a2_acc* vec_t];
    theta= x(1:2,1);
    dtheta= x(3:4,1);

    %the true model
    m2t = m2+ 10*rand(1);% m1 true value is in [m1, m1+epsilon_m1] and epsilon_m1 a random number in [0,10];
    r2t = r2 + 0.5*rand(1);
    I2t = I2 + (15/12)*rand(1);

    at = I1+I2+m1*r1^2+ m2t*(l1^2+ r2t^2);
    bt = m2t*l1*r2t;
    dt = I2t+ m2t*r2t^2;
    % the actual dynamic model of the system is characterized by M and
    % C
    global Mmat Cmat
    Mmat = [at+2*bt*cos(x(2)), dt+bt*cos(x(2));  dt+bt*cos(x(2)), dt];
    Cmat = [-bt*sin(x(2))*x(4), -bt*sin(x(2))*(x(3)+x(4)); bt*sin(x(2))*x(3),0];
    invM = inv(Mmat);
    invMC = invM*Cmat;

    global Mn Cn
    [Mn,Cn] = MCbarmatrix(x);

    v2 = addedinput(theta_d,dtheta_d,ddtheta_d,theta,dtheta,2);

    %TODO: compute the robust controller
    tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta,v2);
    torque = [torque, tau];
```

```
    %TODO: update the system state, compute dx
    dx=zeros(4,1);
    dx(1) = x(3);
    dx(2) = x(4);
    dx(3:4) = -invMC* x(3:4) +invM*tau; % because ddot theta = -M^{-1}(C \dot Theta) + M^{-1} tau
end
```

To calculate the MC matrix for Control law

```
function [Mn,Cn] = MCbarmatrix(x)
    % we compute the parameters in the dynamic model
    m2u = m2+ 10;% m1 true value is in [m1, m1+epsilon_m1] and epsilon_m1 a random number in [0,10];
    r2u = r2 + 0.5;
    I2u = I2 + (15/12);

    m2l = m2;% m1 true value is in [m1, m1+epsilon_m1] and epsilon_m1 a random number in [0,10];
    r2l = r2;
    I2l = I2;

    au = I1+I2u+m1*r1^2+ m2u*(l1^2+ r2u^2);
    bu = m2u*l1*r2u;
    du = I2u+ m2u*r2u^2;

    al = I1+I2l+m1*r1^2+ m2l*(l1^2+ r2l^2);
    bl = m2l*l1*r2l;
    dl = I2l+ m2l*r2l^2;

    Mu = [au+2*bu*cos(x(2)), du+bu*cos(x(2));
        du+bu*cos(x(2)), du];
    Ml = [al+2*bl*cos(x(2)), dl+bl*cos(x(2));
        dl+bl*cos(x(2)), dl];
    Mn = inv((Mu+Ml)/2);
    dn = Mn(2,2);
    bn = (Mn(2,1)-dn)/cos(x(2));
    Cn = [-bn*sin(x(2))*x(4), -bn*sin(x(2))*(x(3)+x(4)); bn*sin(x(2))*x(3),0];
end
```

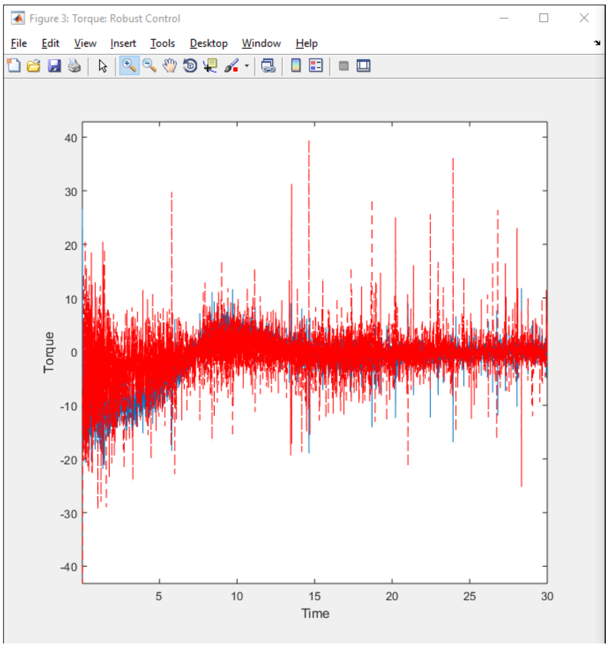
To calculate the value of added input v

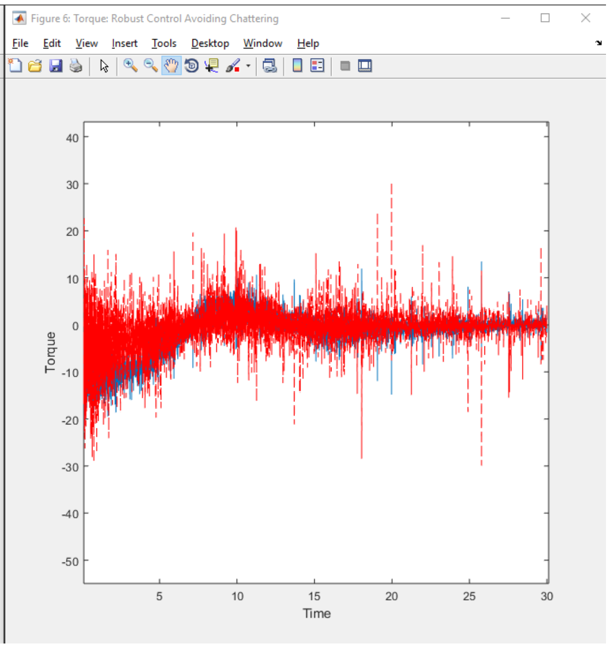
```
function v = addedinput(theta_d,dtheta_d,ddtheta_d,theta,dtheta,index)
    P_e = theta_d - theta;
    V_e = dtheta_d - dtheta;
    epsi = 0.001;
    B = [0;1];
    P = eye(2);
    E1 = [P_e(1);V_e(1)];
    E2 = [P_e(2);V_e(2)];
    ro1 = gama1*norm(E1) + gama2*norm(E1)*norm(E1) + gama3;
    ro2 = gama1*norm(E2) + gama2*norm(E2)*norm(E2) + gama3;

    if (index == 1)
      w1 = transpose(E1)*P*B;
      w2 = transpose(E2)*P*B;
        if (w1 ~= 0)
            v1 = -w1*ro1/norm(w1);
        else
            v1 = 0;
        end
        if (w2 ~= 0)
            v2 = -w2*ro2/norm(w2);
        else
            v2 = 0;
        end
      v = [v1;v2];
    end

    if (index == 2)
      wn1 = transpose(B)*P*E1;
      wn2 = transpose(B)*P*E2;
        if (norm(wn1) > epsi)
            v1 = -wn1*ro1/norm(wn1);
        else if (norm(wn1) <= epsi)
            v1 = -wn1*ro1/epsi;
            end
        end
        if (norm(wn2) > epsi)
            v2 = -wn2*ro2/norm(wn2);
        else if (norm(wn2) <= epsi)
            v2 = -wn2*ro2/epsi;
            end
        end
      v = [v1;v2];
    end

end
```

Roboust Control Law

```
    function tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta,v)
       P_e = theta_d - theta;
       V_e = dtheta_d - dtheta;

      %Todo: Select your feedback gain matrix Kp and Kd.
       Kp = 850*eye(2);
       Kv = 550*eye(2);

       global Mn Cn
       tau = Mn*(Kp*P_e + Kv*V_e) + Cn*dtheta + Mn*ddtheta_d + Mn*v;
    end
```

```
end
```





The sudden disruption observed in the normal Robust Control can be avoided by using a continuous formula for v (additional input in control law). Thus, we get a plot with less of spikes/disruption after avoiding chattering. Both the results are shown above.