Contents


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

testing algorithm with a set of initial and final states.

```
clc
clear all
close all
%link1 initial and final position and velocities.
theta10=-0.6;
dtheta10 =0;
theta1f = 0.9;
dtheta1f=0;

%time
tf= 60;

%link2 initial and final position and velocities.
theta20=-0.8;
dtheta20= 0.1;
theta2f = 0.6;
dtheta2f=0;

% 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];
```

Trajectory planning block

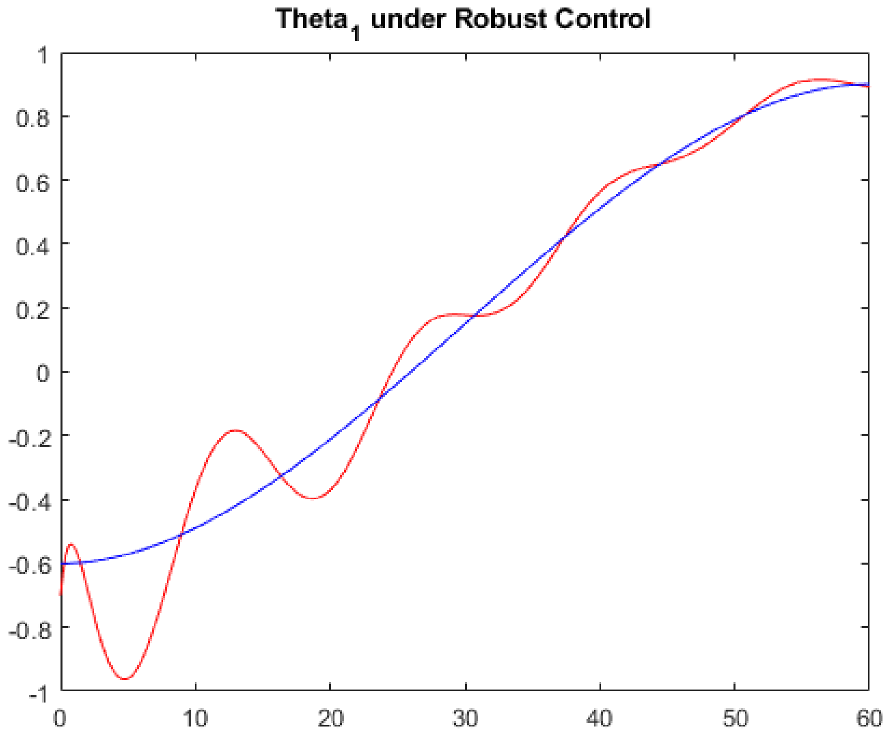
Initial condition (TODO: CHANGE DIFFERENT INITIAL AND FINAL STATES)

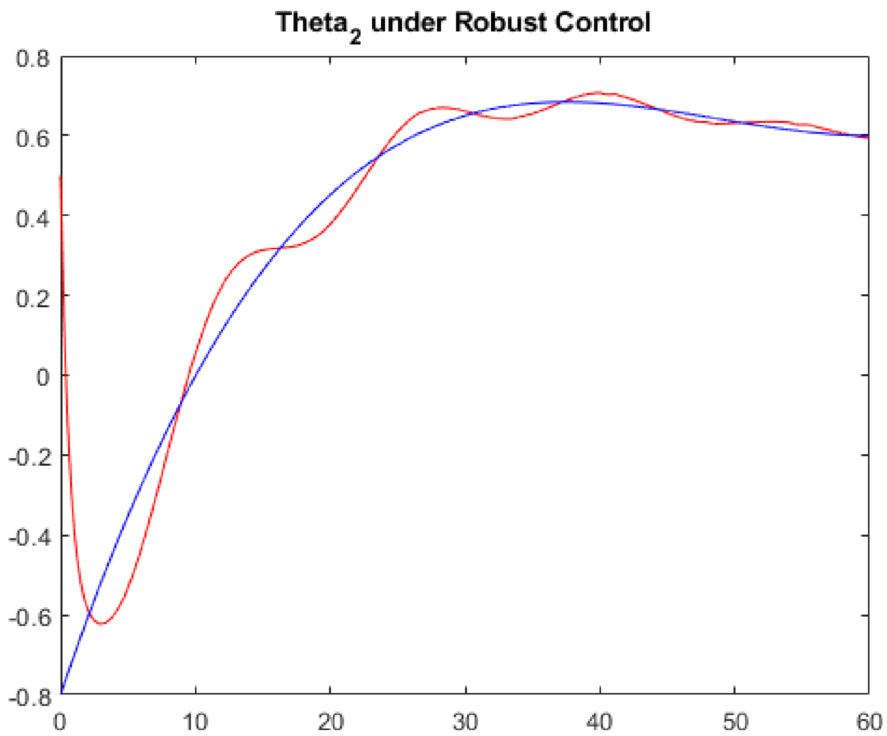
```
x0=[-0.6,-0.8,0,0.1];
x0e = [-0.7,0.5,-0.2,0]; % an error in the initial state.
xf=[0.9,0.6, 0, 0];
% The parameter for planned joint trajectory 1 and 2.
global a1 a2 % two polynomial trajectory for the robot joint
nofigure=1;
% Traj generation.
a1 = planarArmTraj(theta10,dtheta10, theta1f, dtheta1f,tf, nofigure);
a2 = planarArmTraj(theta20,dtheta20, theta2f, dtheta2f,tf, nofigure);

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

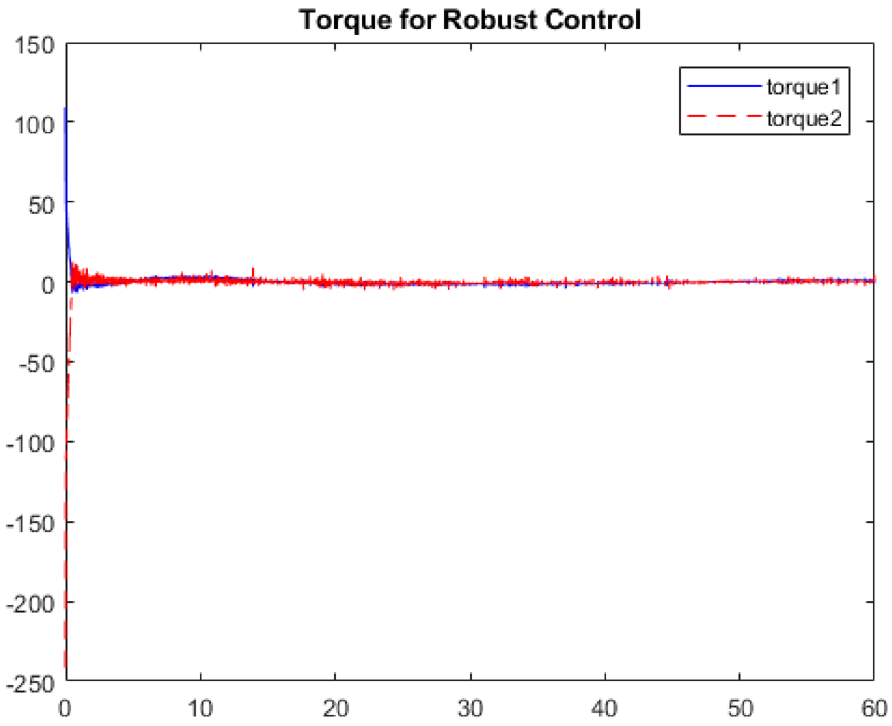
TODO: IMPLEMENT THE CONTROLLER

```
[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');

figure('Name', 'I/p- Robust control')
plot(T, torque(1,1:size(T,1)), 'b-');
hold on
plot(T, torque(2,1:size(T,1)), 'r--');
legend('torque1', 'torque2')
title('Torque for Robust Control');
torque = [];
```


Theta$_1$ under Robust Control
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50
60


Theta$_2$ under Robust Control
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
10
20
30
40
50
60



TODO: IMPLEMENT THE CONTROLLER TO AVOID CHATTERING.

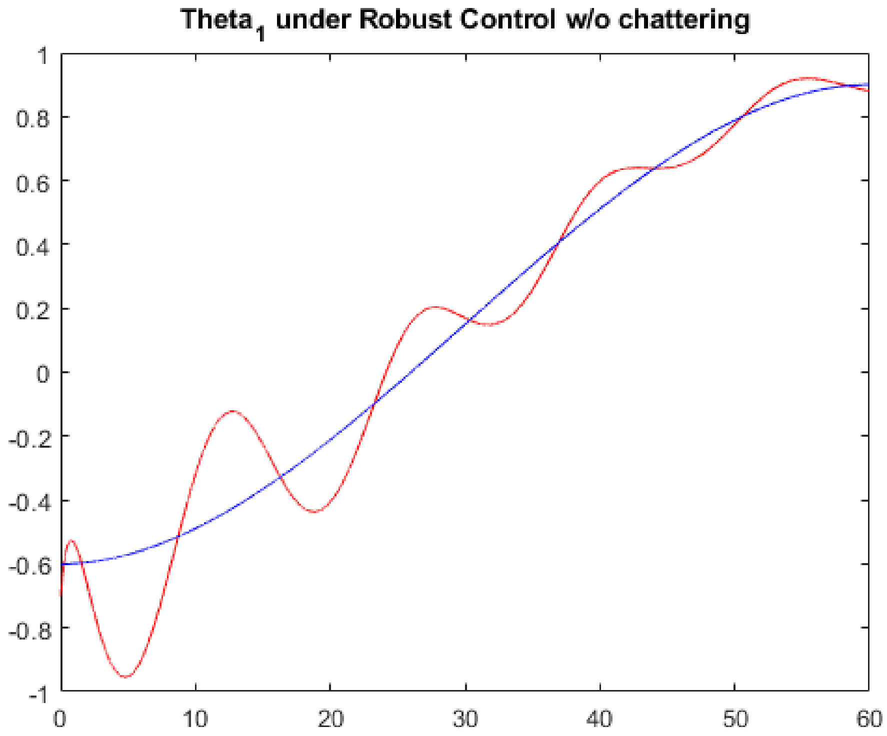
```
[T,X] = ode45(@(t,x)planarArmODERobustApprx(t,x),[0 tf],x0e,options);

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 w/o 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 w/o chattering');

figure('Name', 'I/p- Robust control w/o chattering')
plot(T, torque(1,1:size(T,1)), 'b-');
hold on
plot(T, torque(2,1:size(T,1)), 'r--');
legend('torque1', 'torque2')
title('Torque for Robust Control without chattering');
torque = [];

% Robust control with chattering
    function [dx ] = planarArmODERobust(t,x)
        %Todo: Select your feedback gain matrix Kp and Kd.

        % 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);
```

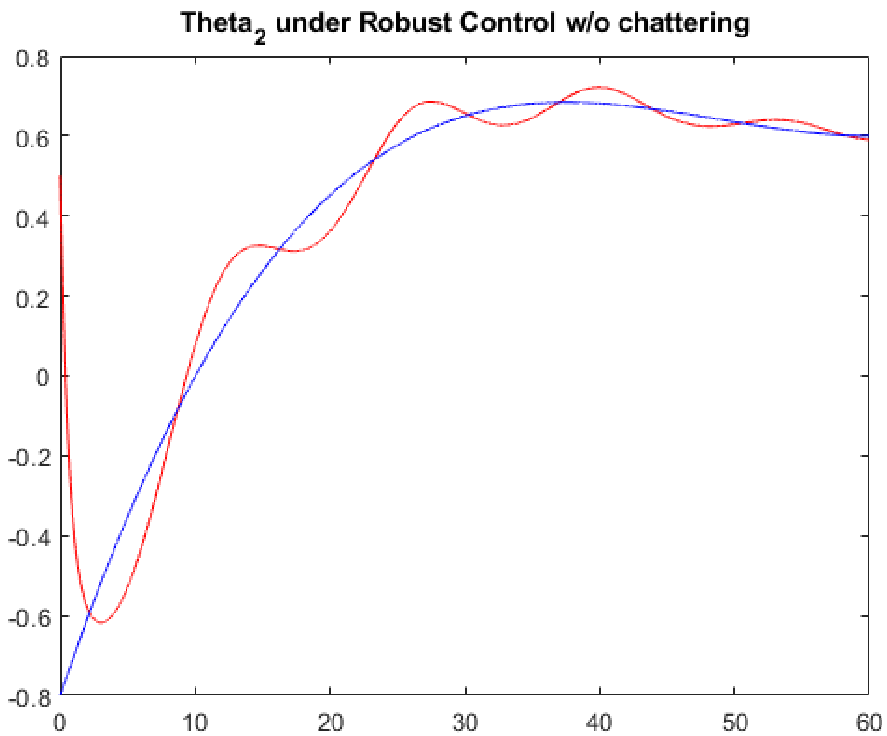
```
        %the true model
        m2t = m2+ 10*rand(1);% m1 true value is in [m1, m1+epsilon_m1] and epsilon_m1 a random number in [0,10];
        r2t = r2 + 0.5*rand(1);
        I2t = I2 + (15/12)*rand(1);

        a = I1+ I2t +m1*r1^2+ m2t*(l1^2+ r2t^2);
        b = m2t*l1*r2t;
        d = I2t+ m2t*r2t^2;

        %lower bound
        a_l = I1+I2+m1*r1^2+ m2*(l1^2+ r2^2);
        b_l = m2*l1*r2;
        d_l = I2 + m2*r2^2;

        %Upper bound
        m2t_u = m2+ 10;
        r2t_u = r2 + 0.5;
        I2t_u = I2 + (15/12);

        a_u = I1+I2t_u+m1*r1^2+ m2t_u*(l1^2+ r2t_u^2);
        b_u = m2t_u*l1*r2t_u;
        d_u = I2t_u+ m2t_u*r2t_u^2;

        % lower bound M
        M_l = [a_l + 2*b_l*cos(x(2)), d_l + b_l*cos(x(2));  d_l + b_l*cos(x(2)), d_l];
        %Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4)); b*sin(x(2))*x(3),0];

        % upper bound M
        M_u = [a_u + 2*b_u*cos(x(2)), d_u + b_u*cos(x(2)); d_u + b_u*cos(x(2)),d_u];
        %Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4)); b*sin(x(2))*x(3),0];

        % Control law M and C
        global M_bar C_bar
        M_bar = inv((M_u + M_l)/2);
        d_bar = M_bar(2,2);
        b_bar = (M_bar(2,1) - d_bar)*sec(x(2));
        C_bar = [-b_bar*sin(x(2))*x(4), -b_bar*sin(x(2))*(x(3)+x(4)); b_bar*sin(x(2))*x(3),0];

        % actual dynamic model of the system is characterized by M and C
        global Mmat Cmat
        Mmat = [a+2*b*cos(x(2)), d+b*cos(x(2));  d+b*cos(x(2)), d];
        Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4)); b*sin(x(2))*x(3),0];
        invM = inv(Mmat);
        invMC = invM*Cmat;

        %TODO: compute the robust controller
        tau = Rob_ctrl(theta_d, dtheta_d, ddtheta_d, theta, dtheta);
        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


% Robust control without chattering
    function [dx] = planarArmODERobustApprx(t,x)
        %Todo: Select your feedback gain matrix Kp and Kd.

        % 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];
```

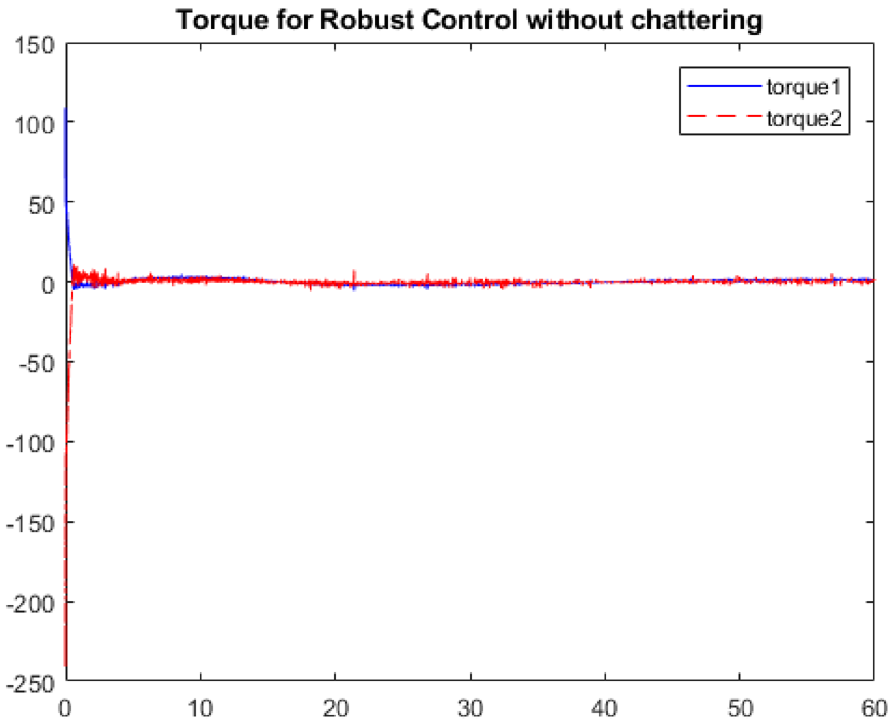
```
        a2_acc = [2*a2(3), 6*a2(4),0,0 ];
        dtheta_d =[a1_vel*vec_t; a2_vel* vec_t];
        ddtheta_d =[a1_acc*vec_t; a2_acc* vec_t];
        theta= x(1:2,1);
        dtheta= x(3:4,1);

        %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);

        a = I1+I2+m1*r1^2+ m2t*(l1^2+ r2t^2);
        b = m2t*l1*r2t;
        d = I2t+ m2t*r2t^2;

        %lower bound
        a_l = I1+I2+m1*r1^2+ m2*(l1^2+ r2^2);
        b_l = m2*l1*r2;
        d_l = I2 + m2*r2^2;

        %Upper bound
        m2t_u = m2+ 10;
        r2t_u = r2 + 0.5;
        I2t_u = I2 + (15/12);

        a_u = I1+I2t_u+m1*r1^2+ m2t_u*(l1^2+ r2t_u^2);
        b_u = m2t_u*l1*r2t_u;
        d_u = I2t_u+ m2t_u*r2t_u^2;

        % lower bound M
        M_l = [a_l + 2*b_l*cos(x(2)), d_l + b_l*cos(x(2));  d_l + b_l*cos(x(2)), d_l];
        %Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4)); b*sin(x(2))*x(3),0];

        % upper bound M
        M_u = [a_u + 2*b_u*cos(x(2)), d_u + b_u*cos(x(2)); d_u + b_u*cos(x(2)),d_u];
        %Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4)); b*sin(x(2))*x(3),0];

        % Control law M and C
        global M_bar C_bar
        M_bar = inv((M_u + M_l)/2);
        d_bar = M_bar(2,2);
        b_bar = (M_bar(2,1) - d_bar)*sec(x(2));
        C_bar = [-b_bar*sin(x(2))*x(4), -b_bar*sin(x(2))*(x(3)+x(4)); b_bar*sin(x(2))*x(3),0];

        % actual dynamic model of the system is characterized by M and C
        global Mmat Cmat
        Mmat = [a+2*b*cos(x(2)), d+b*cos(x(2));  d+b*cos(x(2)), d];
        Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4)); b*sin(x(2))*x(3),0];
        invM = inv(Mmat);
        invMC = invM*Cmat;

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

        %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*taur; % because ddot theta = -M^{-1}(C \dot Theta) + M^{-1} tau
    end
 % torque function for Robust Control
    function tau = Rob_ctrl(theta_d, dtheta_d, ddtheta_d, theta, dtheta);
        global M_bar C_bar Mmat Cmat
        Kp = 300*eye(2);
        Kv = 200*eye(2);
        e = theta_d - theta;
        de = dtheta_d -dtheta;
        y1 = 0.01;
        y2 = 0.02;
```

```
        y3 = 1;
        P = eye(2);
        B = [0;1];
        %for link 1
        x_err1 = [e(1,1) ; de(1,1)];
        p1 = y1*norm(x_err1) + y2*(norm(x_err1)^2) + y3;
        w1 = transpose(x_err1)*P*B;
        if w1 == 0
            v1=0;
        else
            v1 = -((w1*p1)/ norm(w1));
        end

        %for link 2
        x_err2 = [e(2,1) ; de(2,1)];
        p2 = y1*norm(x_err2) + y2*(norm(x_err2)^2) + y3;
        w2 = transpose(x_err2)*P*B;
        E = norm((inv(Mmat)*M_bar) - eye(2));
        eta = norm(((inv(Mmat)*M_bar) - eye(2)) + inv(Mmat)*((C_bar-Cmat)*dtheta));
        if w2 ==0
            v2=0;
        else
            v2 = -((w2*p2)/ norm(w2));
        end
        % Added input
        v = [v1;v2];

        aq = (ddtheta_d + Kp*e + Kv*de + v);
        tau =  M_bar*aq + C_bar*dtheta ;
    end
  % torque function for Robust Control without Chattering
    function taur = Rob_ctrlapprx(theta_d, dtheta_d, ddtheta_d, theta, dtheta);
        global M_bar C_bar Mmat Cmat
        Kp = 300*eye(2);
        Kv = 200*eye(2);
        e = theta_d - theta;
        de = dtheta_d -dtheta;
        y1 = 0.01;
        y2 = 0.02;
        y3 = 1;
        P = eye(2);
        B = [0;1];
        %for link 1
        x_err1 = [e(1,1) ; de(1,1)];
        p1 = y1*norm(x_err1) + y2*(norm(x_err1)^2) + y3;
        w1 = transpose(B)*P*(x_err1);
        eps1 = 0.01;
        if w1 <= eps1
            v1= -((w1*p1)/eps1);
        else
            v1 = -((w1*p1)/ norm(w1));
        end

        %for link 2
        x_err2 = [e(2,1) ; de(2,1)];
        p2 = y1*norm(x_err2) + y2*(norm(x_err2)^2) + y3;
        w2 = transpose(B)*P*(x_err2);
        E = norm((inv(Mmat)*M_bar) - eye(2));
        eta = norm(((inv(Mmat)*M_bar) - eye(2)) + inv(Mmat)*((C_bar-Cmat)*dtheta));
        eps2 = 0.01;
        if w2 <= eps2
            v2= -((w2*p2)/eps2);
        else
            v2 = -((w2*p2)/ norm(w2));
        end
        % Added input
        v = [v1;v2];


        aq = (ddtheta_d + Kp*e + Kv*de + v);
```

```
        taur =  M_bar*aq + C_bar*dtheta ;
    end
```







In Robust control without chattering, the model performed slightly better in the sense that the sudden high spikes were reduced. It basically eliminated the discontinuity problem of previous case.

Trajectory planning using polynomial functions.

```
function [a] = planarArmTraj(theta10,dtheta10, theta1f, dtheta1f,tf, nofigure)
% Input: Initial and final position and velocities, planning horizon [0,tf]
% nofigure=1 then do not output the planned trajectory.
% Cubic polynomial trajectory.

% formulate the linear equation and solve.
M= [1 0 0 0;
    0 1 0 0;
    1 tf tf^2 tf^3;
    0 1 2*tf 3*tf^2];
b=[theta10; dtheta10;theta1f; dtheta1f];
a=M\b;
t=0:0.01:tf;

if nofigure==1
    return
else

figure('Name','Position (degree)');
plot(t,a(1)+a(2)*t+ a(3)*t.^2+a(4)*t.^3,'LineWidth',3);
title('Position (degree)')
grid

figure('Name','Velocity (degree/s)');
plot(t,a(2)*t+ 2*a(3)*t +3*a(4)*t.^2,'LineWidth',3);
title('Velocity (degree/s)')
grid

figure('Name','Acceleration (degree/s^2)');
plot(t, 2*a(3) +6*a(4)*t,'LineWidth',3);
title('Acceleration (degree/s^2)')
grid
end
end
```

```
end
```

Published with MATLAB® R2017b