Table of Contents


```
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];
b0 = [ m1* r1^2 + m2*l1^2 + I1, m2*r2^2 + I2, m2*l1*r2];
```

Trajectory planning block

Initial condition

```
x0=[-0.5,0,-1,0.1];
x0e = [-0.7,0.5,-0.2,0,b0]; % 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
%%%%%%%
global torqueTime % for keeping track of control inputs in ode45
%%%%%%%
nofigure=1;
a1 = planarArmTraj(theta10,dtheta10, theta1f, dtheta1f,tf, nofigure);
a2 = planarArmTraj(theta20,dtheta20, theta2f, dtheta2f,tf, nofigure);


torque=[];
%%%%%%%%%%
torqueTime = [];
%%%%%%%%%%
options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4,1e-4,
 1e-4, 1e-4]);
[T,X] = ode45(@(t,x)planarArmODEUncertain(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');
%%%%%%%%
uIdx = resampleTime(T,torqueTime);
figure
plot(T,torque(1,uIdx))
hold on
plot(T,torque(2,uIdx))
xlabel seconds
xlabel N-m
title 'Input Torque'
legend('\tau_1','\tau_1')
%%%%%%%%
```

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

        % feedback gain matrix K AND LAMBDA.
        K= 10*eye(2);
        Lambda= 5*eye(2);
        % 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
        m1t = 12;% m1 true value is in [m1, m1+epsilon_m1] and
 epsilon_m1 a random number in [0,10];
        r1t = 0.8;
        l1t = 1.2;
        I1t = 1;

        m2t = 4;% m1 true value is in [m1, m1+epsilon_m1] and
 epsilon_m1 a random number in [0,10];
        l2t = 1.2;
```

```
        r2t = 0.5;
        I2t = 0.5;

        a = I1t+I2t+m1t*r1t^2+ m2t*(l1t^2+ r2t^2);
        b = m2t*l1t*r2t;
        d = I2t+ m2t*r2t^2;
        % the actual dynamic model of the system is characterized by M
and
        % C
        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;


        % compute the robust controller
        e = theta - theta_d;
        de = dtheta - dtheta_d;
        r = de + Lambda*e;
        v = dtheta_d - Lambda*e;
        a = ddtheta_d - Lambda*de;

        Y2= [ a(1), a(1) + a(2), 2*a(1)*cos(theta(2))
+ a(2)*cos(theta(2)) - dtheta(2)*v(1)*sin(theta(2)) -
v(2)*sin(theta(2))*(dtheta(1) + dtheta(2));...
             0, a(1) + a(2), a(1)*cos(theta(2)) +
dtheta(1)*v(1)*sin(theta(2))];
```

this is new

```
        gamma = [ 1,0,0;
                  0, 1,0;
                  0 0 1];
        thetahat_dot = -inv(gamma)*Y2'*r;

        epsilon=0.2;
        rho = 13.46; % see eq(27) in the paper.

%         if norm(Y2'*r) > epsilon
%             u = -rho* Y2'*r/norm(Y2'*r);
%         else
%             u = - rho* Y2'*r/epsilon;
%         end

        %tau = Y2*(b0 + u)- K*r
```

new tau calculated

```
        tau = Y2*x(5:7) - K*r;
        torque =[torque, tau];
        %%%%%%%%
        torqueTime =[torqueTime, t];
```

```
        %%%%%%%%
        % feed back the thetaHat_dots
        dx=zeros(7,1);
        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
        dx(5:7) = thetahat_dot;

    end
```

```
Not enough input arguments.

Error in robustControl (line 25)
a1 = planarArmTraj(theta10,dtheta10, theta1f, dtheta1f,tf, nofigure);
```

```
end


function uIdx = resampleTime(T,inputTime)
% Input:
%   T - ode45 output time values
%   inputTime - time's recorded within ode45
% This first loop filters out torque indexes that are repeated as
% ode45 restarts iterations.
uIdx = zeros(length(T),1);
for i = 1:length(T)
    try
        uIdx(i) = find(T(i)==inputTime,1,'last');
    catch ME
        if strcmp(ME.identifier,'MATLAB:subsassignnumelmismatch')
            if i>1
                uIdx(i) = uIdx(i-1);
            else
                uIdx(i) = 1;
            end
        else
            rethrow ME
        end
    end
end
end
```

Published with MATLAB® R2016b