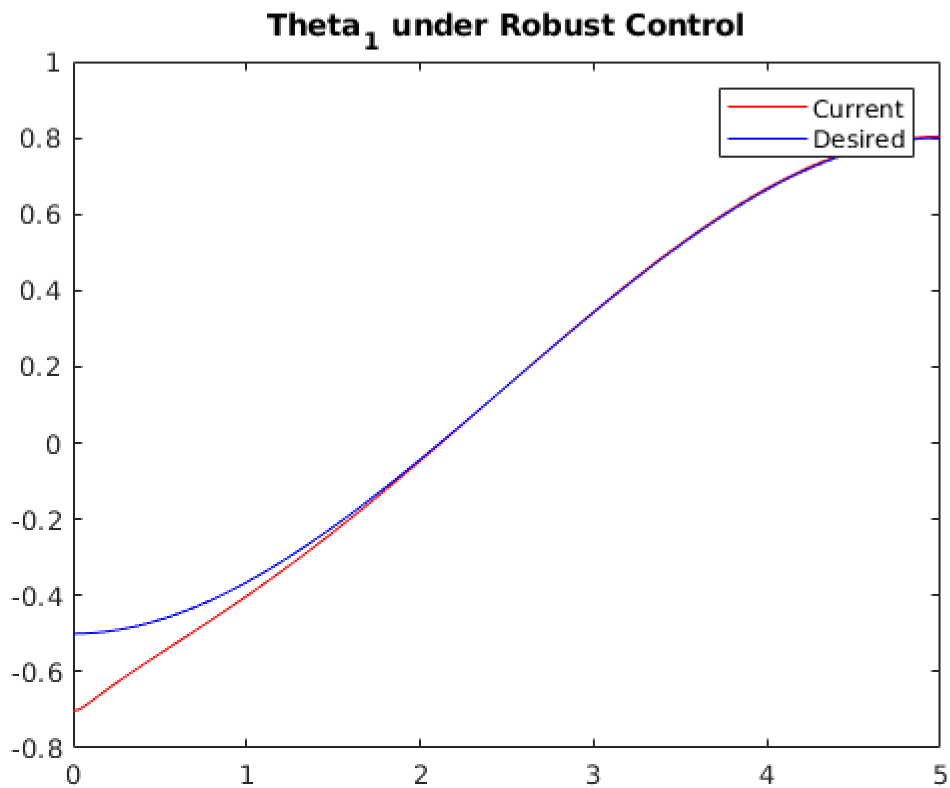
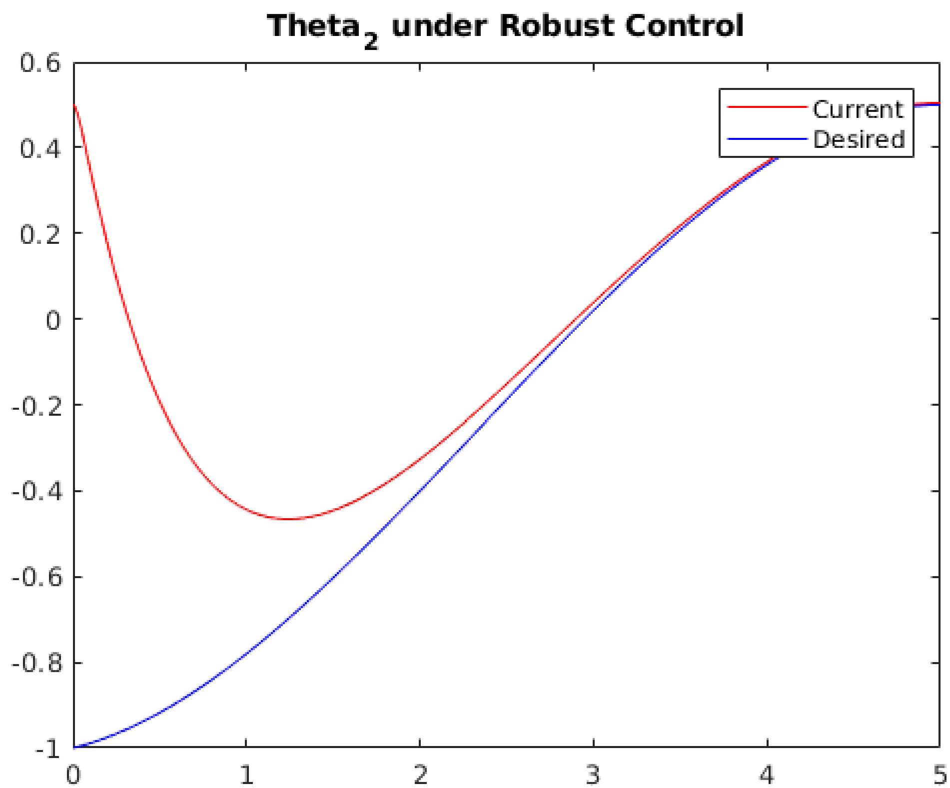
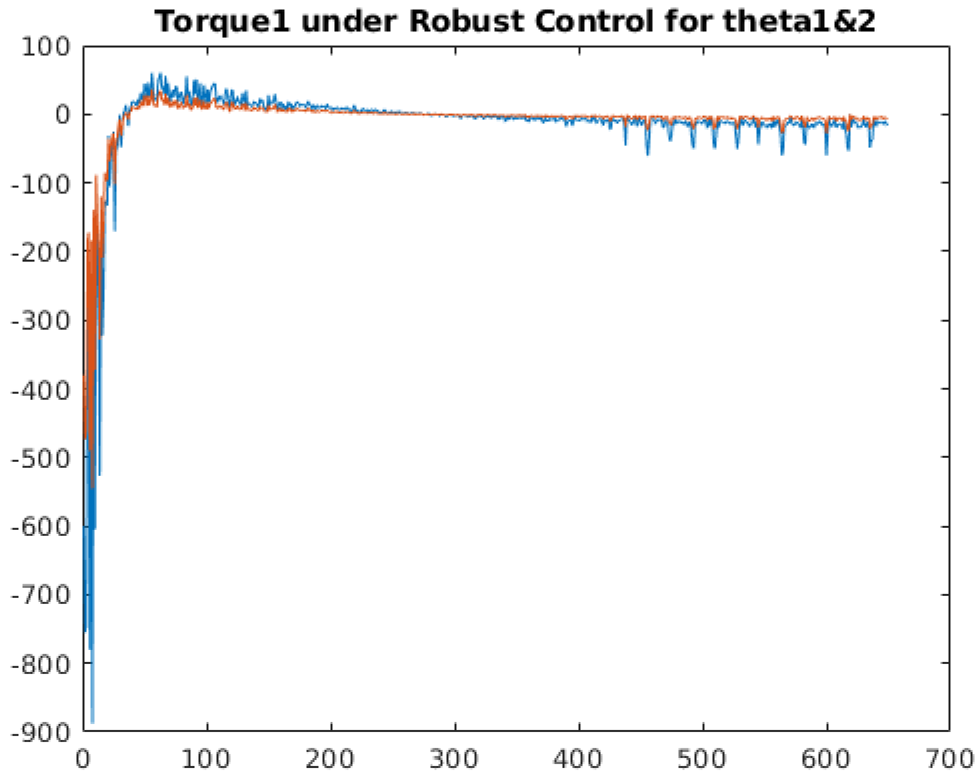
testing algorithm with a set of initial and final states.

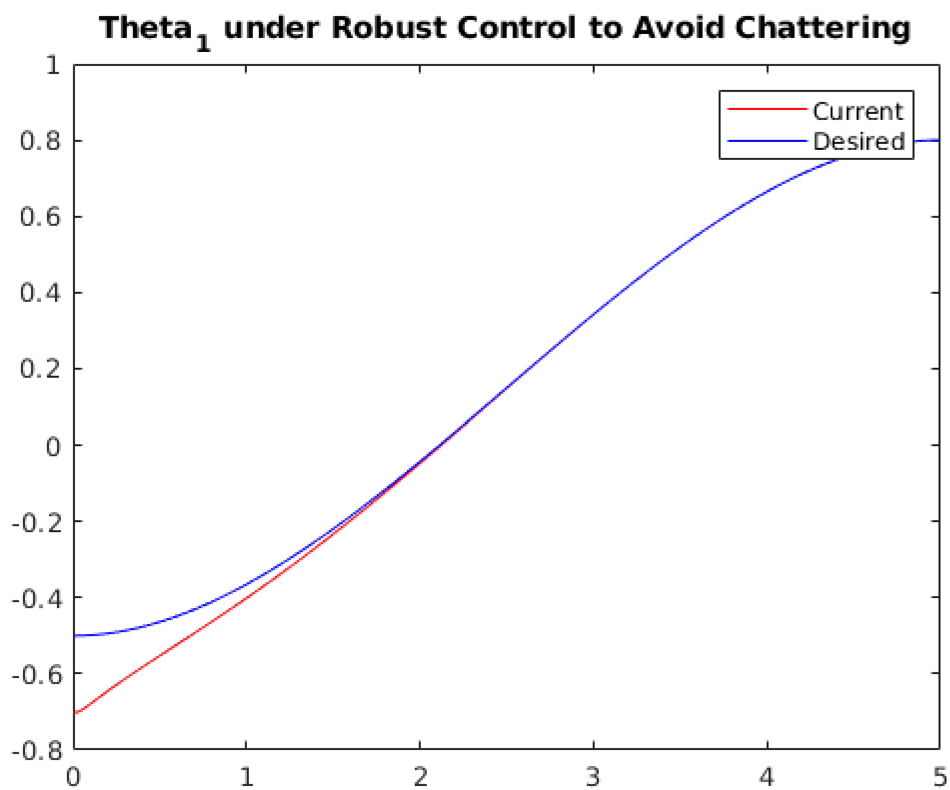
```
clc
clear all
close all
%Example to demostrate the robust control with planar 2d arm.
theta10=-0.5;
dtheta10 =0;
theta1f = 0.8;
dtheta1f=0;
tf=5;

% 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,dtheta2f,tf)
```


Theta$_1$ under Robust Control
Current
Desired
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1
2
3
4
5

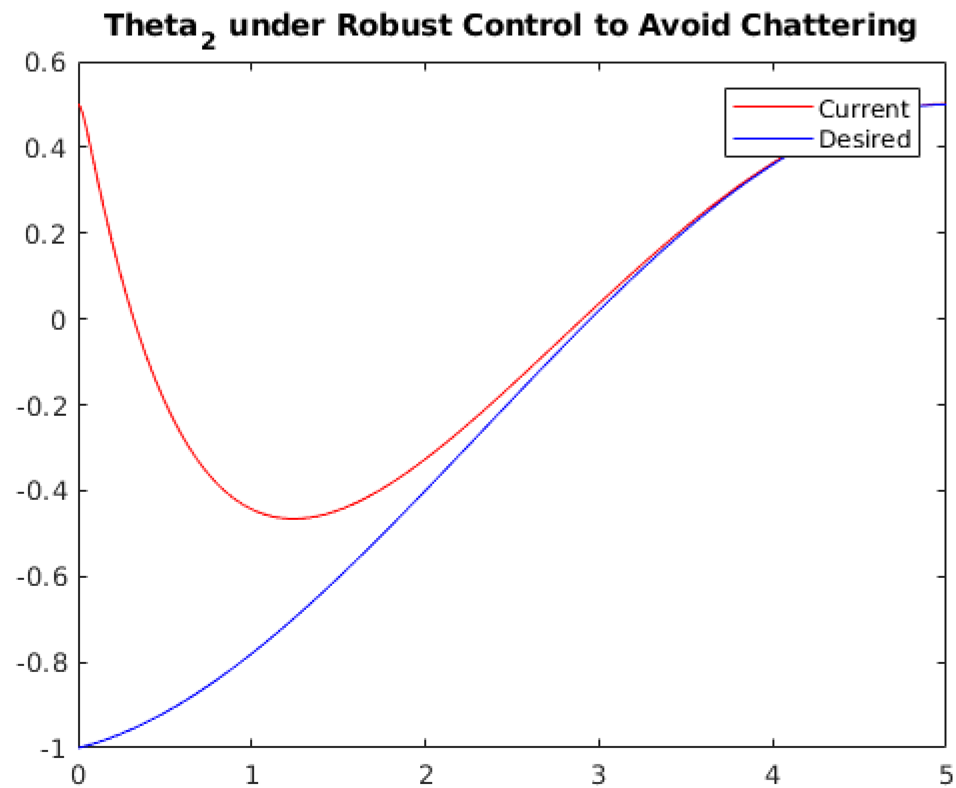

Theta$_2$ under Robust Control
Current
Desired
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5



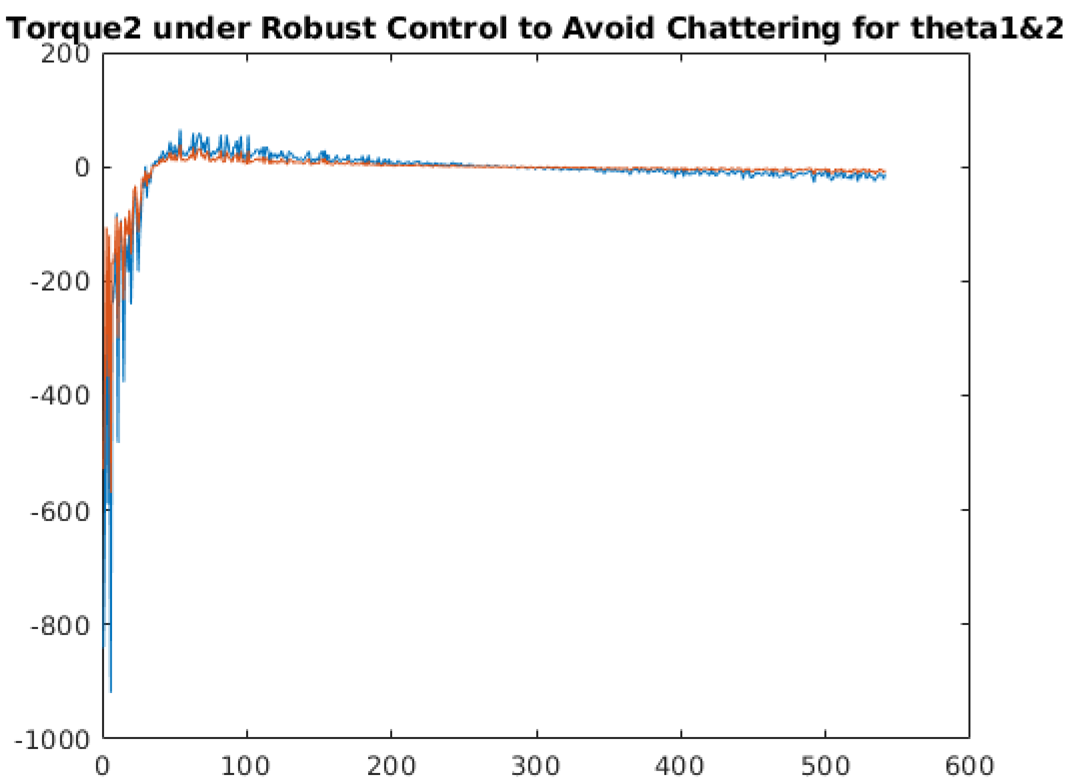
Torque1 under Robust Control for theta1&2
100
0
-100
-200
-300
-400
-500
-600
-700
-800
-900
0
100
200
300
400
500
600
700



Theta$_1$ under Robust Control to Avoid Chattering
Current
Desired
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1
2
3
4
5



Theta$_2$ under Robust Control to Avoid Chattering
Current
Desired
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5



Torque2 under Robust Control to Avoid Chattering for theta1&2
200
0
-200
-400
-600
-800
-1000
0
100
200
300
400
500
600

INFERENCE/REMARKS

To select the constants: Gammas and alpha were selected at randam starting from 0.01. As seen from the plots the second controller converges similar to the one without chattering compensation. But if the Torque plot i.e the control input in both cases is compared we find that the first controller oscillates after saturation as the del_a parameter change is discontinuous. When we provide a smoother/continuous del_a the controller is stable.

Published with MATLAB® R2017b

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];
```

Trajectory planning block

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

```
x0=[-0.5,0,-1,0.1];
x0e = [-0.7,0.5,-0.2,0]; % an error in the initial state.
xf=[0.9,0.7, 0, 0]; %Changed
% 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);

global torque1 torque2
torque1=[]; torque2 = [];
options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
```

Not enough input arguments.

Error in robustControl (line 22)
a1 = planarArmTraj(theta10,dtheta10, theta1f, dtheta1f,tf, nofigure);

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');
legend("Current","Desired");
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');
legend("Current","Desired");
figure('Name', 'Torque1');
plot(torque1');
title('Torque1 under Robust Control for theta1&2');
```

TODO: IMPLEMENT THE CONTROLLER TO AVOID CHATTERING.

```
[T2,X2] = ode45(@(t,x)planarArmODERobustApprx(t,x),[0
 tf],x0e,options);

figure('Name','theta1');
plot(T2, X2(:,1),'r-');
hold on
plot(T2, a1(1)+a1(2)*T2+ a1(3)*T2.^2+a1(4)*T2.^3,'b-');
title('Theta_1 under Robust Control to Avoid Chattering');
legend("Current","Desired");
figure('Name','theta2');
plot(T2, X2(:,2),'r-');
hold on
plot(T2, a2(1)+a2(2)*T2+ a2(3)*T2.^2+a2(4)*T2.^3, 'b-');
title('Theta_2 under Robust Control to Avoid Chattering');
legend("Current","Desired");
figure('Name', 'Torque2');
plot(torque2');
title('Torque2 under Robust Control to Avoid Chattering for
 theta1&2');

%TODO: PLOT THE INPUT TRAJECTORY
```

```
    function [dx ] = planarArmODERobust(t,x)
        %Todo: Select your feedback gain matrix Kp and Kd.
        Kp=[60,0;0,60];
        Kd=[40,0;0,40];
        % 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;
        % 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;


        %TODO: compute the robust controller
        %constants
        B = [0;0;1;1];alpha = 0.1;
        gamma1 = 0.11; gamma2 = 0.15; gamma3 = 0.20;
        P = eye(4);

        e = [(theta-theta_d); (dtheta-dtheta_d)];
        rho = (1/(1-
alpha))*(gamma1*norm(e)+gamma2*(norm(e)^2)+gamma3);
        if norm(B'*P*e)==0
            del_a = 0;
        else
            del_a = -rho*((B'*P*e)/norm(B'*P*e));
        end
        aq = ddtheta_d - Kp*e(1:2,1) - Kd*e(3:4,1) + del_a;
        tau = Mmat*aq +Cmat* dtheta;
        torque1 = [torque1,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



    function [dx ] = planarArmODERobustApprx(t,x)
        %Todo: Select your feedback gain matrix Kp and Kd.
        Kp=[60,0;0,60];
        Kd=[40,0;0,40];
        % 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;
        % 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;


        %TODO: compute the robust controller to avoid chattering.
        B = [0;0;1;1];alpha = 0.1;
        gamma1 = 0.11; gamma2 = 0.15; gamma3 = 0.20;
        P = eye(4);

        e = [(theta-theta_d); (dtheta-dtheta_d)];
        rho = (1/(1-
alpha))*(gamma1*norm(e)+gamma2*(norm(e)^2)+gamma3);
```

```
        epsilon = 0.2;
        if norm(B'*P*e)> epsilon
            del_a = -rho*((B'*P*e)/norm(B'*P*e));
        else
            del_a = -rho*((B'*P*e)/epsilon);
        end
        aq = ddtheta_d - Kp*e(1:2,1) - Kd*e(3:4,1) + del_a;
        tau = Mmat*aq + Cmat* dtheta;
        torque2 = [torque2,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

end
```

Published with MATLAB® R2017b

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
```

```
Not enough input arguments.

Error in planarArmTraj (line 10)
    1 tf tf^2 tf^3;
```

Published with MATLAB® R2017b