Table of Contents


```
clear all; close all; clc;
```

Run the dynamics script and controller gains designed from it

```
rrbot_dyn;
```

```
EOM =

I1*q1dd - u1 + I2*q1dd + I2*q2dd + l1^2*m2*q1dd + m1*q1dd*r1^2 + m2*q1dd*r2^2
 + m2*q2dd*r2^2 - g*m2*r2*sin(q1 + q2) - g*l1*m2*sin(q1) - g*m1*r1*sin(q1) -
 l1*m2*q2d^2*r2*sin(q2) + 2*l1*m2*q1dd*r2*cos(q2) + l1*m2*q2dd*r2*cos(q2) -
 2*l1*m2*q1d*q2d*r2*sin(q2)

                                                                  I2*q1dd - u2 + I2*q2dd +
 m2*q1dd*r2^2 + m2*q2dd*r2^2 - g*m2*r2*sin(q1 + q2) + l1*m2*q1d^2*r2*sin(q2) +
 l1*m2*q1dd*r2*cos(q2)


EOM_gravity_term =

-g*(l1*m2*sin(q1) + m1*r1*sin(q1) + m2*r2*sin(q1 + q2))
                                  -g*m2*r2*sin(q1 + q2)


EOM_Mass_term =

I1*q1dd + I2*q1dd + I2*q2dd + l1^2*m2*q1dd + m1*q1dd*r1^2 + m2*q1dd*r2^2 +
 m2*q2dd*r2^2 + 2*l1*m2*q1dd*r2*cos(q2) + l1*m2*q2dd*r2*cos(q2)
                                                                  I2*q1dd +
 I2*q2dd + m2*q1dd*r2^2 + m2*q2dd*r2^2 + l1*m2*q1dd*r2*cos(q2)


M =

[m2*l1^2 + 2*m2*cos(q2)*l1*r2 + m1*r1^2 + m2*r2^2 + I1 + I2, m2*r2^2 +
 l1*m2*cos(q2)*r2 + I2]
[                           m2*r2^2 + l1*m2*cos(q2)*r2 + I2,
  m2*r2^2 + I2]
```

```
EOM_Coriolis_term =

-l1*m2*q2d*r2*sin(q2)*(2*q1d + q2d)
              l1*m2*q1d^2*r2*sin(q2)


EOM_tau_term =

u1
u2


EOM_manipForm =

q1dd*(m2*l1^2 + 2*m2*cos(q2)*l1*r2 + m1*r1^2 + m2*r2^2 + I1 + I2) - u1 -
 g*(l1*m2*sin(q1) + m1*r1*sin(q1) + m2*r2*sin(q1 + q2)) + q2dd*(m2*r2^2 +
 l1*m2*cos(q2)*r2 + I2) - l1*m2*q2d*r2*sin(q2)*(2*q1d + q2d)

                      l1*m2*r2*sin(q2)*q1d^2 - u2 + q2dd*(m2*r2^2 + I2) +
 q1dd*(m2*r2^2 + l1*m2*cos(q2)*r2 + I2) - g*m2*r2*sin(q1 + q2)


ans =

(I2*u1 - I2*u2 + m2*r2^2*u1 - m2*r2^2*u2 + l1*m2^2*q1d^2*r2^3*sin(q2) +
 l1*m2^2*q2d^2*r2^3*sin(q2) + g*l1*m2^2*r2^2*sin(q1) + I2*g*l1*m2*sin(q1) +
 I2*g*m1*r1*sin(q1) - l1*m2*r2*u2*cos(q2) + 2*l1*m2^2*q1d*q2d*r2^3*sin(q2)
 + l1^2*m2^2*q1d^2*r2^2*cos(q2)*sin(q2) - g*l1*m2^2*r2^2*sin(q1 +
 q2)*cos(q2) + I2*l1*m2*q1d^2*r2*sin(q2) + I2*l1*m2*q2d^2*r2*sin(q2)
 + g*m1*m2*r1*r2^2*sin(q1) + 2*I2*l1*m2*q1d*q2d*r2*sin(q2))/(-
 l1^2*m2^2*r2^2*cos(q2)^2 + l1^2*m2^2*r2^2 + I2*l1^2*m2 + m1*m2*r1^2*r2^2 +
 I1*m2*r2^2 + I2*m1*r1^2 + I1*I2)


ans =

-(I2*u1 - I1*u2 - I2*u2 - l1^2*m2*u2 - m1*r1^2*u2 + m2*r2^2*u1 -
 m2*r2^2*u2 + l1*m2^2*q1d^2*r2^3*sin(q2) + l1^3*m2^2*q1d^2*r2*sin(q2)
 + l1*m2^2*q2d^2*r2^3*sin(q2) - g*l1^2*m2^2*r2*sin(q1 + q2) -
 I1*g*m2*r2*sin(q1 + q2) + g*l1*m2^2*r2^2*sin(q1) + I2*g*l1*m2*sin(q1)
 + I2*g*m1*r1*sin(q1) + l1*m2*r2*u1*cos(q2) - 2*l1*m2*r2*u2*cos(q2) +
 2*l1*m2^2*q1d*q2d*r2^3*sin(q2) + 2*l1^2*m2^2*q1d^2*r2^2*cos(q2)*sin(q2)
 + l1^2*m2^2*q2d^2*r2^2*cos(q2)*sin(q2) - g*l1*m2^2*r2^2*sin(q1 +
 q2)*cos(q2) + g*l1^2*m2^2*r2*cos(q2)*sin(q1) - g*m1*m2*r1^2*r2*sin(q1
 + q2) + I1*l1*m2*q1d^2*r2*sin(q2) + I2*l1*m2*q1d^2*r2*sin(q2)
 + I2*l1*m2*q2d^2*r2*sin(q2) + g*m1*m2*r1*r2^2*sin(q1) +
 2*l1^2*m2^2*q1d*q2d*r2^2*cos(q2)*sin(q2) + l1*m1*m2*q1d^2*r1^2*r2*sin(q2)
 + 2*I2*l1*m2*q1d*q2d*r2*sin(q2) + g*l1*m1*m2*r1*r2*cos(q2)*sin(q1))/(-
 l1^2*m2^2*r2^2*cos(q2)^2 + l1^2*m2^2*r2^2 + I2*l1^2*m2 + m1*m2*r1^2*r2^2 +
 I1*m2*r2^2 + I2*m1*r1^2 + I1*I2)

-----State Space Representation-----
```

```
Xd =

                                                            q1d

                                                            q2d

       (I2*u1 - I2*u2 + m2*r2^2*u1 - m2*r2^2*u2 + l1*m2^2*q1d^2*r2^3*sin(q2)
 + l1*m2^2*q2d^2*r2^3*sin(q2) + g*l1*m2^2*r2^2*sin(q1) + I2*g*l1*m2*sin(q1)
 + I2*g*m1*r1*sin(q1) - l1*m2*r2*u2*cos(q2) + 2*l1*m2^2*q1d*q2d*r2^3*sin(q2)
 + l1^2*m2^2*q1d^2*r2^2*cos(q2)*sin(q2) - g*l1*m2^2*r2^2*sin(q1 +
 q2)*cos(q2) + I2*l1*m2*q1d^2*r2*sin(q2) + I2*l1*m2*q2d^2*r2*sin(q2)
 + g*m1*m2*r1*r2^2*sin(q1) + 2*I2*l1*m2*q1d*q2d*r2*sin(q2))/(-
 l1^2*m2^2*r2^2*cos(q2)^2 + l1^2*m2^2*r2^2 + I2*l1^2*m2 + m1*m2*r1^2*r2^2 +
 I1*m2*r2^2 + I2*m1*r1^2 + I1*I2)
-(I2*u1 - I1*u2 - I2*u2 - l1^2*m2*u2 - m1*r1^2*u2 + m2*r2^2*u1 -
 m2*r2^2*u2 + l1*m2^2*q1d^2*r2^3*sin(q2) + l1^3*m2^2*q1d^2*r2*sin(q2)
 + l1*m2^2*q2d^2*r2^3*sin(q2) - g*l1^2*m2^2*r2*sin(q1 + q2) -
 I1*g*m2*r2*sin(q1 + q2) + g*l1*m2^2*r2^2*sin(q1) + I2*g*l1*m2*sin(q1)
 + I2*g*m1*r1*sin(q1) + l1*m2*r2*u1*cos(q2) - 2*l1*m2*r2*u2*cos(q2) +
 2*l1*m2^2*q1d*q2d*r2^3*sin(q2) + 2*l1^2*m2^2*q1d^2*r2^2*cos(q2)*sin(q2)
 + l1^2*m2^2*q2d^2*r2^2*cos(q2)*sin(q2) - g*l1*m2^2*r2^2*sin(q1 +
 q2)*cos(q2) + g*l1^2*m2^2*r2*cos(q2)*sin(q1) - g*m1*m2*r1^2*r2*sin(q1
 + q2) + I1*l1*m2*q1d^2*r2*sin(q2) + I2*l1*m2*q1d^2*r2*sin(q2)
 + I2*l1*m2*q2d^2*r2*sin(q2) + g*m1*m2*r1*r2^2*sin(q1) +
 2*l1^2*m2^2*q1d*q2d*r2^2*cos(q2)*sin(q2) + l1*m1*m2*q1d^2*r1^2*r2*sin(q2)
 + 2*I2*l1*m2*q1d*q2d*r2*sin(q2) + g*l1*m1*m2*r1*r2*cos(q2)*sin(q1))/(-
 l1^2*m2^2*r2^2*cos(q2)^2 + l1^2*m2^2*r2^2 + I2*l1^2*m2 + m1*m2*r1^2*r2^2 +
 I1*m2*r2^2 + I2*m1*r1^2 + I1*I2)
```

```
----Controllabiltiy test for virtual control input?: YES----
-----State Feedback Control-----
lambda =
   -13   -10   -12    -6

K =
   72.0000         0   18.0000         0
         0  130.0000         0   23.0000
```

Defining trajectory

```
q0 = [180;  90];    q0 = deg2rad(q0);  % denotes q_init of joint1 and joint2
qf = [0;    0];     qf = deg2rad(qf);
qd0 = [0;    0];    qd0 = deg2rad(qd0);
qdf = [0;    0];    qdf = deg2rad(qdf);
t0 = 0;
tf = 10;

% A*a = b... From lecture, we know A and b.. solving for co-efficients
global a     % each col has coefficients for each joint's trajectory
a_j1 = traj_cubic_solve(q0(1), qf(1), qd0(1), qdf(1), t0, tf);  % Trajectory
 Co-efficients for joint1
a_j2 = traj_cubic_solve(q0(2), qf(2), qd0(2), qdf(2), t0, tf);  % Trajectory
 Co-efficients for joint2
a = [a_j1 a_j2];
fprintf("-----Cubic Polynomial co-efficients for given
 q0,qf,qd0,qdf,t0,tf-----\n");
fprintf("For joint1:  a0, a1, a2, a3 \n");
disp(a_j1');
fprintf("For joint2:  a0, a1, a2, a3 \n");
disp(a_j2');
```

```
-----Cubic Polynomial co-efficients for given q0,qf,qd0,qdf,t0,tf-----
For joint1:  a0, a1, a2, a3
    3.1416   -0.0000   -0.0942    0.0063

For joint2:  a0, a1, a2, a3
    1.5708   -0.0000   -0.0471    0.0031
```

Solving the State Space Equations

```
T = tf;                                     % Simulating for 10 seconds
y0 = [deg2rad(200),deg2rad(125),0,0];       % Initial conditions
[t,y] = ode45(@rrbot_ode, [0,T], y0);
```

Reconstructing trajectories

```
X_desired = zeros(4,height(t));     % 4 rows with [q1,q2,q1d,q2d]' ..each col
 denotes values of it at each time instants
```

```
U_desired = zeros(2,height(t));      % 2 rows with [u1,u2]' ..each col denotes
 values of it at each time instants

for i = 1:height(t)
    t_ = t(i);

    % Trajectory Generation using cubic equation for joint1
    a0=a_j1(1); a1=a_j1(2); a2=a_j1(3); a3=a_j1(4);
    X_desired(1,i) = a0 + a1*t_ + a2*t_^2 + a3*t_^3;
    X_desired(3,i) = a1 + 2*a2*t_ + 3*a3*t_^2;
    U_desired(1,i) = 2*a2 + 6*a3*t_;

    % Trajectory Generation using cubic equation for joint2
    a0=a_j2(1); a1=a_j2(2); a2=a_j2(3); a3=a_j2(4);
    X_desired(2,i) = a0 + a1*t_ + a2*t_^2 + a3*t_^3;
    X_desired(4,i) = a1 + 2*a2*t_ + 3*a3*t_^2;
    U_desired(2,i) = 2*a2 + 6*a3*t_;
end
```

Reconstruct control inputs

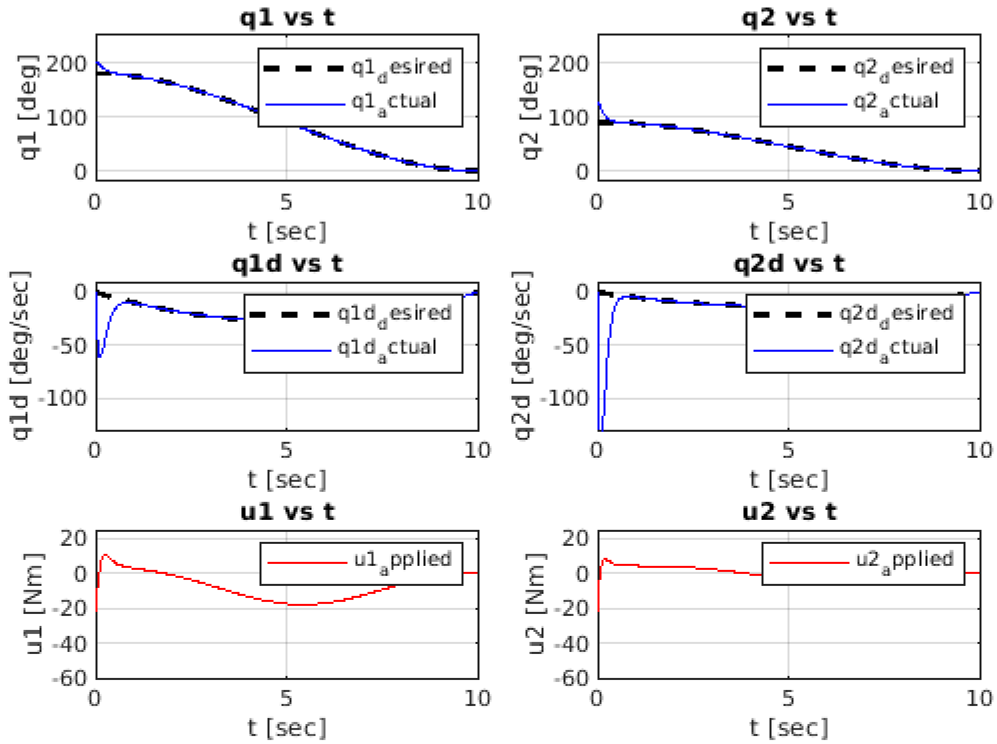
Constants from assignment

```
g=9.81;
m1=1; m2=1;
I1=0.084; I2=0.084;
l1=1; l2=1; r1=0.45; r2=0.45;

global K
u = zeros(2,height(t));      % 2 rows with [u1,u2]' ..each col denotes values
 of it at each time instants
for i = 1:height(t)

    % [a] Virtual Control Input
    v = -K*(y(i,:)' - X_desired(:,i)) + U_desired(:,i);

    % [b] Final feedback linearized control law
    q1 = y(i,1);    q1d = y(i,3);
    q2 = y(i,2);    q2d = y(i,4);

    M = [m2*l1^2 + 2*m2*cos(q2)*l1*r2 + m1*r1^2 + m2*r2^2 + I1 + I2,
 m2*r2^2 + l1*m2*cos(q2)*r2 + I2
         m2*r2^2 + l1*m2*cos(q2)*r2 + I2,
 m2*r2^2 + I2];
    EOM_Coriolis_term = [-l1*m2*q2d*r2*sin(q2)*(2*q1d + q2d)
                          l1*m2*q1d^2*r2*sin(q2)];
    EOM_gravity_term = [-g*(l1*m2*sin(q1) + m1*r1*sin(q1) + m2*r2*sin(q1 +
 q2))
                        -g*m2*r2*sin(q1 + q2)];
    u_ = M*v + EOM_Coriolis_term + EOM_gravity_term;

    u(:,i) = u_;
end
```

Plotting the results

```
figure;

% [1] Trajectory tracking of q1
subplot(3,2,1);
% Plotting q1_desired trajectory
wrapTo2Pi(X_desired(1,:));
plot(t,rad2deg(X_desired(1,:)),'k--', 'linewidth',2);  % ref line in black
% Plotting q1_actual trajectory from ODE solver
hold on
wrapTo2Pi(y(:,1));
plot(t,rad2deg(y(:,1)),'b');
% Graph Styling
title('q1 vs t');
xlabel('t [sec]');
ylabel('q1 [deg]');
axis([0 10 -20 250]);  % Setting limits on the output
grid on;
legend('q1_desired', 'q1_actual');


% [2] Trajectory tracking of q2
subplot(3,2,2);
% Plotting q2_desired trajectory
wrapTo2Pi(X_desired(2,:));
plot(t,rad2deg(X_desired(2,:)),'k--', 'linewidth',2);  % ref line in black
% Plotting q2_actual trajectory from ODE solver
hold on
wrapTo2Pi(y(:,2));
plot(t,rad2deg(y(:,2)),'b');
% Graph Styling
title('q2 vs t');
xlabel('t [sec]');
ylabel('q2 [deg]');
axis([0 10 -20 250]);  % Setting limits on the output
grid on;
legend('q2_desired', 'q2_actual');


% [3] Trajectory tracking of q1d
subplot(3,2,3);
% Plotting q1d_desired trajectory
wrapTo2Pi(X_desired(3,:));
plot(t,rad2deg(X_desired(3,:)),'k--', 'linewidth',2);  % ref line in black
% Plotting q1d_actual trajectory from ODE solver
hold on
wrapTo2Pi(y(:,3));
plot(t,rad2deg(y(:,3)),'b');
% Graph Styling
title('q1d vs t');
xlabel('t [sec]');
ylabel('q1d [deg/sec]');
```

```
axis([0 10 -130 10]);  % Setting limits on the output for better view
grid on;
legend('q1d_desired', 'q1d_actual');



% [4] Trajectory tracking of q2d
subplot(3,2,4);
% Plotting q2d_desired trajectory
wrapTo2Pi(X_desired(4,:));
plot(t,rad2deg(X_desired(4,:)),'k--', 'linewidth',2);  % ref line in black
% Plotting q2d_actual trajectory from ODE solver
hold on
wrapTo2Pi(y(:,4));
plot(t,rad2deg(y(:,4)),'b');
% Graph Styling
title('q2d vs t');
xlabel('t [sec]');
ylabel('q2d [deg/sec]');
axis([0 10 -130 10]);  % Setting limits on the output for better view
grid on;
legend('q2d_desired', 'q2d_actual');



% [5] Plotting u1 generated and applied to joints with our control law
subplot(3,2,5);
plot(t,u(1,:),'r');
% Graph Styling
title('u1 vs t');
xlabel('t [sec]');
ylabel('u1 [Nm]');
axis([0 10 -60 25]);  % Setting limits on the output
grid on;
legend('u1_applied');



% [6] Plotting u2 generated and applied to joints with our control law
subplot(3,2,6);
plot(t,u(2,:),'r');
% Graph Styling
title('u2 vs t');
xlabel('t [sec]');
ylabel('u2 [Nm]');
axis([0 10 -60 25]);  % Setting limits on the output
grid on;
legend('u2_applied');
```



Published with MATLAB® R2021b