## Table of Contents

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

## STEP 1 to 5: Running previous hw script to get EOM

```
rrbot_dyn
```

```
ans =

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)


ans =

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)


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)
```

## STEP 6: Finding equillibrium point

Use the original second-order ODE instead of state space rep for this fn At equilibrium point, the vel and acc is zero Also, here we are trying to find eq point when there is no input

```
EOM_eq = subs(EOM, [q1d, q1dd, q2d, q2dd, u1, u2], [0,0,0,0,0,0]);
sol = solve(EOM_eq == 0, [q1, q2]);

% We get 3 equilibirum points and just printing the equilibrium points
sol_print = [sol.q1, sol.q2];
for itr = 1:height(sol_print)     % height(A) gives row
    fprintf(['-----Equilibirum Point ', num2str(itr), '-----\n', '[q1, q2] -->
 ']);
    disp(sol_print(itr,:));
end
```

```
-----Equilibirum Point 1-----
[q1, q2] --> [0, 0]

-----Equilibirum Point 2-----
[q1, q2] --> [pi, 0]

-----Equilibirum Point 3-----
[q1, q2] --> [0, pi]
```

## STEP 7: Linearize our system to continue our analysis

```
A = jacobian(Xd, X);
B = jacobian(Xd, u);
% Substitute robot diemension and other constants from HW to A and B
A = subs(A, [g,m1,m2,I1,I2,l1,l2,r1,r2],
 [9.81,1,1,0.084,0.084,1,1,0.45,0.45]);
B = subs(B, [g,m1,m2,I1,I2,l1,l2,r1,r2],
 [9.81,1,1,0.084,0.084,1,1,0.45,0.45]);
fprintf("-----Linearizing our system gives the following A and B
 matrcies-----")
```

```
fprintf("\nA: \n");
disp(A);
fprintf("\nB: \n");
disp(B);

-----Linearizing our system gives the following A and B matrcies-----
A:
[

                0,











                        0,

                         1,

                         0]
[

                0,











                        0,

                         0,

                         1]
[
  -((16301277*cos(q1))/4000000 - (79461*cos(q1 + q2)*cos(q2))/40000)/
((81*cos(q2)^2)/400 - 1474329/4000000),





                  - ((5157*q1d^2*cos(q2))/40000 + (5157*q2d^2*cos(q2))/40000
 + (81*q1d^2*cos(q2)^2)/400 - (81*q1d^2*sin(q2)^2)/400 +
 (9*u2*sin(q2))/20 - (79461*cos(q1 + q2)*cos(q2))/40000 + (79461*sin(q1
 + q2)*sin(q2))/40000 + (5157*q1d*q2d*cos(q2))/20000)/((81*cos(q2)^2)/400
 - 1474329/4000000) - (81*cos(q2)*sin(q2)*((573*u1)/2000 - (573*u2)/2000
 + (16301277*sin(q1))/4000000 + (5157*q1d^2*sin(q2))/40000 +
 (5157*q2d^2*sin(q2))/40000 - (9*u2*cos(q2))/20 - (79461*sin(q1
 + q2)*cos(q2))/40000 + (5157*q1d*q2d*sin(q2))/20000 +
 (81*q1d^2*cos(q2)*sin(q2))/400))/(200*((81*cos(q2)^2)/400
 - 1474329/4000000)^2),
 -((5157*q1d*sin(q2))/20000 + (5157*q2d*sin(q2))/20000 +
 (81*q1d*cos(q2)*sin(q2))/200)/((81*cos(q2)^2)/400 - 1474329/4000000),
                                                                          -
((5157*q1d*sin(q2))/20000 + (5157*q2d*sin(q2))/20000)/((81*cos(q2)^2)/400 -
 1474329/4000000)]
[-((22717017*cos(q1 + q2))/4000000 - (16301277*cos(q1))/4000000 -
 (256041*cos(q1)*cos(q2))/40000 + (79461*cos(q1 + q2)*cos(q2))/40000)/
((81*cos(q2)^2)/400 - 1474329/4000000), ((14157*q1d^2*cos(q2))/20000
 - (22717017*cos(q1 + q2))/4000000 + (5157*q2d^2*cos(q2))/40000
 - (256041*sin(q1)*sin(q2))/40000 + (81*q1d^2*cos(q2)^2)/200
 + (81*q2d^2*cos(q2)^2)/400 - (81*q1d^2*sin(q2)^2)/200 -
 (81*q2d^2*sin(q2)^2)/400 - (9*u1*sin(q2))/20 + (9*u2*sin(q2))/10 -
 (79461*cos(q1 + q2)*cos(q2))/40000 + (79461*sin(q1 + q2)*sin(q2))/40000
 + (5157*q1d*q2d*cos(q2))/20000 + (81*q1d*q2d*cos(q2)^2)/200 -
 (81*q1d*q2d*sin(q2)^2)/200)/((81*cos(q2)^2)/400 - 1474329/4000000) +
 (81*cos(q2)*sin(q2)*((573*u1)/2000 - (1573*u2)/1000 - (22717017*sin(q1 +
 q2))/4000000 + (16301277*sin(q1))/4000000 + (14157*q1d^2*sin(q2))/20000
 + (5157*q2d^2*sin(q2))/40000 + (256041*cos(q2)*sin(q1))/40000 +
 (9*u1*cos(q2))/20 - (9*u2*cos(q2))/10 - (79461*sin(q1 + q2)*cos(q2))/40000
 + (5157*q1d*q2d*sin(q2))/20000 + (81*q1d^2*cos(q2)*sin(q2))/200 +
 (81*q2d^2*cos(q2)*sin(q2))/400 + (81*q1d*q2d*cos(q2)*sin(q2))/200))/
(200*((81*cos(q2)^2)/400 - 1474329/4000000)^2), ((14157*q1d*sin(q2))/10000
 + (5157*q2d*sin(q2))/20000 + (81*q1d*cos(q2)*sin(q2))/100 +
 (81*q2d*cos(q2)*sin(q2))/200)/((81*cos(q2)^2)/400 - 1474329/4000000),
 ((5157*q1d*sin(q2))/20000 + (5157*q2d*sin(q2))/20000 +
 (81*q1d*cos(q2)*sin(q2))/200 + (81*q2d*cos(q2)*sin(q2))/200)/
((81*cos(q2)^2)/400 - 1474329/4000000)]


B:
[                                                                   0,
                                                            0]
[                                                                   0,
                                                            0]
[                -573/(2000*((81*cos(q2)^2)/400 - 1474329/4000000)),
 ((9*cos(q2))/20 + 573/2000)/((81*cos(q2)^2)/400 - 1474329/4000000)]
[((9*cos(q2))/20 + 573/2000)/((81*cos(q2)^2)/400 - 1474329/4000000), -
((9*cos(q2))/10 + 1573/1000)/((81*cos(q2)^2)/400 - 1474329/4000000)]
```

## STEP 8: Stability Analysis at each equilibrium point

Substitute equilibirum points to A

```
for itr = 1:height(sol_print)
    tmp_q1 = double(sol_print(itr,1));
    tmp_q2 = double(sol_print(itr,2));
    tmp_A = A;                     % Dont change original A with variables
    tmp_A = subs(tmp_A, [X(1),X(2),X(3),X(4)], [tmp_q1,tmp_q2,0,0]);
    tmp_A = double(tmp_A);  % to convert symbols to decimal values
    eigA = eig(tmp_A);

    fprintf(['-----Eigen Values at equilibirum point ', num2str(itr), '-----
\n', '[q1, q2] --> ']);
    disp(sol_print(itr,:));

    fprintf('eigA = \n');
    disp(eigA);

    % check if all entries in eigA are -ve
    if ( min(round(real(eigA))<0) == 0 )
        fprintf("Asymptotically Stable?: NO\n");
    else
        fprintf("Asymptotically Stable?: YES\n");
    end
end
```

```
-----Eigen Values at equilibirum point 1-----
[q1, q2] --> [0, 0]

eigA =
    7.1676
    2.7129
   -7.1676
   -2.7129

Asymptotically Stable?: NO
-----Eigen Values at equilibirum point 2-----
[q1, q2] --> [pi, 0]

eigA =
  -0.0000 + 7.1676i
  -0.0000 - 7.1676i
  -0.0000 + 2.7129i
  -0.0000 - 2.7129i

Asymptotically Stable?: NO
-----Eigen Values at equilibirum point 3-----
[q1, q2] --> [0, pi]

eigA =
  -3.8995 + 0.0000i
   3.8995 + 0.0000i
   0.0000 + 4.9864i
   0.0000 - 4.9864i

Asymptotically Stable?: NO
```

## STEP 9: Controllability Analysis for upward configuration

Upward Config is [q1, q2] --> [0, 0] Let's change the original A and q1, q2 from here as it is what we consider going further

```
q1 = 0;
q2 = 0;
A = subs(A, [X(1),X(2),X(3),X(4)], [q1,q2,0,0]);
A = double(A);
B = subs(B, [X(1),X(2),X(3),X(4)], [q1,q2,0,0]);
B = double(B);
rankCO = rank(ctrb(A,B));
if (rankCO >= height(q))
    fprintf("----Controllable at upward config?: YES----\n");
else
    fprintf("----Controllable at upward config?: NO----\n");
end

% For reporting purpose
fprintf("-----Linearization at upward configuration gives the following A and
 B matrices-----")
fprintf("\nA: \n");
disp(A);
fprintf("\nB: \n");
disp(B);
```

```
----Controllable at upward config?: YES----
-----Linearization at upward configuration gives the following A and B
 matrices-----
A:
         0         0    1.0000         0
         0         0         0    1.0000
   12.5769  -11.9611         0         0
  -16.9227   46.1565         0         0


B:
         0         0
         0         0
    1.7250   -4.4345
   -4.4345   14.8902
```

## STEP 10: Design State-feedback Control

Let eigenvalues for state-feedback design be as follows

```
syms k1 k2 k3 k4 k5 k6 k7 k8 lambda
lambda = [-2, -5, -3, -4];
A = double(A);
B = double(B);
```

```
global K;                    % To access this in other scripts
K = place(A, B, lambda);
fprintf("-----State Feedback Control-----\n");
fprintf("lambda = \n");
disp(lambda);
fprintf("K = \n");
disp(K);
```

```
-----State Feedback Control-----
lambda =
    -2    -5    -3    -4

K =
   42.1307    6.0018   15.8495    3.2185
   11.1603    5.7187    4.6494    1.4472
```

*Published with MATLAB® R2021b*