Contents


```
function [ dx ] = odepassivity(t,x,system_params)
```

```
% Extracting the system params
        I1=system_params(1);  I2 = system_params(2); m1=system_params(3); r1=system_params(4); m
2=system_params(5); r2=system_params(6); l1=system_params(7); l2=system_params(8);
        g=9.8;

        a = I1+I2+m1*r1^2+ m2*(l1^2+ r2^2);
        b = m2*l1*r2;
        d = I2+ m2*r2^2;

        % the actual dynamic model of the system:
        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];
        Gmat =  [m1*g*r1*cos(x(1))+m2*g*(l1*cos(x(1))+r2*cos(x(1)+x(2)));
    m2*g*r2*cos(x(1)+x(2))];
        invM = inv(Mmat);
        invMC = invM*Cmat;
```

```
Not enough input arguments.

Error in odepassivity (line 4)
        I1=system_params(1);  I2 = system_params(2); m1=system_params(3); r1=system_params(4); m
2=system_params(5); r2=system_params(6); l1=system_params(7); l2=system_params(8);
```

Forward Kinematics

```
        q1=x(1);
        q2=x(2);
        q1_dot=x(3);
        q2_dot=x(4);

        X=l1*cos(q1)+l2*cos(q1+q2);
        Y=l1*sin(q1)+l2*sin(q1+q2);
```

Jacobian

```
        J=[-l1*sin(q1)-l2*sin(q1+q2) -l2*sin(q1+q2);l1*cos(q1)+l2*cos(q1+q2) l2*cos(q1+q2)];

        Jdot = [-l1*cos(q1)*q1_dot-l2*cos(q1+q2)*(q1_dot+q2_dot) -l2*cos(q1+q2)*(q1_dot+q2_dot);
...
        -l1*sin(q1)*q1_dot-l2*sin(q1+q2)*(q1_dot+q2_dot) -l2*sin(q1+q2)*(q1_dot+q2_dot)];
```

Cartesian space velocity

```
        xdot = J*[q1_dot;q2_dot];
        KD=80;
        %Kp=100;
```

Desired Position in cartesian coordinate

```
        r=0.9;
    x_d = [r*sin(t);r*cos(t)];
        dx_d =[r*cos(t);-r*sin(t)];
        ddx_d = [-r*sin(t);-r*cos(t)];
        %dtheta = [x(3:4,1);t-l2*sin(q1+q2)*(q1_dot+q2_dot) -l2*sin(q1+q2)*(q1_dot+q2_dot)];
    dtheta=x(3:4,1);


        q_dot = [x(3);x(4)];
        e_x = [X;Y] - x_d;
        ed_x = dx_d - xdot;

        Lambda = 2;
        r_x = ed_x + Lambda*e_x;
        v_x = dx_d - Lambda*e_x;
        a_x = ddx_d - Lambda*ed_x;
        v_q = pinv(J)*v_x;
        a_q = pinv(J)*(a_x-Jdot*v_q);
        r_q = q_dot - v_q;
        Torque = Mmat*a_q + Cmat*v_q + Gmat - transpose(J)*KD*J*(r_q-v_q);



        dx = [x(3);x(4);invM*(Torque - Gmat) -  invM*Cmat*dtheta];
```

```
end
```

Published with MATLAB® R2018b