Kathia Coronado

Robot Dynamics Hw4

Contents

Problem 1- Derive Total Kinetic Energy and foarm Inertia matrix D(q)

```
clear ; clc;
syms m1 m2 m3
syms l1 l2 l3
syms q1 q2 q3 real
syms r1 r2 r3
syms I1 I2 I3 real

Slist=[[0;0;1;0;0;0],[0;-1;0;l1;0;0],[0 ;-1;0;l1;0;-l2]];
thetalist= [q1;q2;q3];
I = eye(3);
T={};

for i = 1 : length(thetalist)
    w = vector_2_skew(Slist(1:3,i));
    v = Slist(4:6,i);
    theta = thetalist(i);
    R = I + sin(theta)* w + (1-cos(theta))*w^2;
    star = (I * theta+(1-cos(theta))*w+(theta-sin(theta))*w^2)*v;
    T{i} = [R star; 0 0 0 1];
end

M01=[1 0 0 0;0 0 -1 0; 0 1 0 l1;0 0 0 1];
T01= T{1}*M01;

M02=[1 0 0 l2;0 0 -1 0; 0 1 0 l1;0 0 0 1];
T02= T{1}*T{2}*M02;

M03= [1 0 0 l2+l3;0 0 -1 0;0 1 0 l1; 0 0 0 1];
T03= simplify(T{1}*T{2}*T{3}*M03);

% Now need T0c1 T0c2 T0c3

T1_c1=[1 0 0 0; 0 1 0 -l1+r1;0 0 1 0 ; 0 0 0 1];
T0_c1= simplify ( T01*T1_c1);

T2_c2= [1 0 0 -l2+r2;0 1 0 0;0 0 1 0;0 0 0 1];
T0_c2= simplify( T02*T2_c2);

T3_c3= [1 0 0 -l3+r3;0 1 0 0; 0 0 1 0; 0 0 0 1];
T0_c3= simplify (T03*T3_c3);

% now need to find jacobian at all of the masses
zero= zeros([3,1]);

Oc1=T0_c1(1:3,4);
Jv_c1 = jacobian(Oc1, [q1 q2 q3]);
z0= T01(1:3,3);
Jw_c1= [z0 zero zero];

Oc2=T0_c2(1:3,4);
Jv_c2 = jacobian(Oc2, [q1 q2 q3]);
z1= T02(1:3,3);
Jw_c2= [z0 z1 zero];

Oc3=T0_c3(1:3,4);
Jv_c3 = jacobian(Oc3, [q1 q2 q3]);
z2= T03(1:3,3);
Jw_c3= [z0 z1 z2];

Dv = m1*Jv_c1.'*Jv_c1 + m2*Jv_c2.'*Jv_c2 + m3*Jv_c3.'*Jv_c3;

%Rotation matrices at the weights
Rc1= T0_c1(1:3,1:3);
Rc2= T0_c2(1:3,1:3);
Rc3= T0_c3(1:3,1:3);

Dw = Jw_c1.'*Rc1*I1*Rc1.'*Jw_c1 + Jw_c2.'*Rc2*I2*Rc2.'*Jw_c2 + Jw_c3.'*Rc3*I3*Rc3.'*Jw_c3;

Dw_simplify= simplify(Dw);
D= simplify(Dv+Dw)
```

Part 2 - derive coriolis centripetal matrix C(q,qd)

```
syms q_dot_1 q_dot_2 q_dot_3 q_dot_4 real

% Derivative of D wrt to q1 q2 q3 q4
dDdq = zeros(3,3,3)*sym(1);
dDdq(:,:,1) = simplify(diff(D, q1));
dDdq(:,:,2) = simplify(diff(D, q2));
dDdq(:,:,3) = simplify(diff(D, q3));

%Here we are using 3 loops to get all cijk cristofell symbols
Cs = zeros(3,3,3)*sym(1);
for i = 1:3
    for j = 1:3
        for k = 1:3
            Cs(i,j,k) = simplify(0.5*(dDdq(k,j,i) + dDdq(k,i,j) - dDdq(i,j,k)));
        end
    end
end

C = zeros(3,3)*sym(1);
dq = [q_dot_1, q_dot_2, q_dot_3];
for k = 1:3
    for j = 1:3
        C(k,j) = simplify(squeeze(dq*Cs(:,j,k)));
    end
end
C
```

```
C =

[- q_dot_2*((m3*sin(2*q2)*l2^2)/2 + m3*sin(2*q2 + q3)*l2*r3 + (m2*sin(2*q2)*r2^2)/2 + (m3*sin(2*q2 + 2*q3)*r3^2)/2) - (m3*q_dot_3*r3*(r3*sin(2*q2 + 2*q3) + l2*sin(q
[                                                                             q_dot_1*((m3*sin(2*q2)*l2^2)/2 + m3*sin(2*q2 + q3)*l2*r3 + (m2*sin(2*q2)*r2^2)/2 + (m3
[                                                                                                                         (m3*q_dot_1*r3*(r3*sin(2*q2 + 2*q3) + l2*sin(q
```

Problem 3- Derive total potential enrgy of the robot 3x1 gravity g(q)

```
syms g real
Pc1 = m1 * g * T0_c1(3,4);
Pc2 = m2 * g * T0_c2(3,4);
Pc3 = m3 * g * T0_c3(3,4);

P= Pc1+Pc2+Pc3;

g1 = diff(P,q1);
g2 = diff(P,q2);
g3 = diff(P,q3);

G = [g1; g2; g3]
```

```
G =

                                                    0
g*m3*(r3*cos(q2 + q3) + l2*cos(q2)) + g*m2*r2*cos(q2)
                                 g*m3*r3*cos(q2 + q3)
```

Problem 4- Form dynamical model of robot in compact form

```
syms q_ddot_1 q_ddot_2 q_ddot_3 real

Tau = simplify(expand(D*[q_ddot_1; q_ddot_2; q_ddot_3] + C*[q_dot_1; q_dot_2; q_dot_3] + G))
```

```
Tau =

I1*q_ddot_1 + I2*q_ddot_1 + I2*q_ddot_2 + I3*q_ddot_1 + I3*q_ddot_2 + I3*q_ddot_3 + (l2^2*m3*q_ddot_1)/2 + (m2*q_ddot_1*r2^2)/2 + (m3*q_ddot_1*r3^2)/2 + (l2^2*m3*q_
                                                                                                                                 I2*q_ddot_1 + I2*q_ddot_2 + I3*q_ddot
```

Problem 5- rewrite dynamical model using Newtons method

Unit Vector directions

```
i = [1; 0; 0];
j = [0; 1; 0];
k = [0; 0; 1];
```

```
%Forward Recursion
%initial params
w0 = [0;0;0];
alpha0 = [0;0;0];
ac0 = [0;0;0];
ae0 = [0;0;0];

R0_1 = T01(1:3,1:3);
R0_2 = T02(1:3,1:3);
R0_3 = T03(1:3,1:3);
R1_2 = R0_1.'*R0_2;
R2_3 = R0_2.'*R0_3;

%z0=k;
%z1=R0_1(:,3);
%z2=R0_2(:,3);

%z0= R0_1(:,3)
%z1= R0_2(:,3)
%z2= R0_3(:,3)
z0 = k
z1 = T01(1:3,3)
z2 = T02(1:3,3)

b1= R0_1.'*z0;
b2= R0_2.'*z1;
b3= R0_3.'*z2;

w1 = (R0_1.'*w0)+(b1*q_dot_1);
w2 = (R1_2.'*w1)+(b2*q_dot_2);
w3 = (R2_3.'*w2)+(b3*q_dot_3);

alpha1 = (R0_1.'*alpha0)+(b1*q_ddot_1)+(cross(w1,b1*q_dot_1));
alpha2 = (R1_2.'*alpha1)+(b2*q_ddot_2)+(cross(w2,b2*q_dot_2));
alpha3 = (R2_3.'*alpha2)+(b3*q_ddot_3)+(cross(w3,b3*q_dot_3));

%
r1c1 = r1*k;
r2c1 = (r1-l1)*k;
r12 = l1*k;

r2c2 = r2*i; %lc2*i;
r3c2 = (r2-l2)*i;%(lc2-l2)*i;
r23 = l2*i;

r3c3 =  r3*i; %lc3*i;
r4c3 = (r3-l3)*i;%(lc3-l3)*i;
r34 = l3*i;

ac1 = (R0_1.'*ae0) + (cross(alpha1,r1c1)) + (cross(w1,cross(w1,r1c1)));
ae1 = (R0_1.'*ae0) + (cross(alpha1,r12))  + (cross(w1,cross(w1,r12)));

ac2 = (R1_2.'*ae1) + (cross(alpha2,r2c2)) + (cross(w2,cross(w2,r2c2)));
ae2 = (R1_2.'*ae1) + (cross(alpha2,r23))  + (cross(w2,cross(w2,r23)));

ac3 = (R2_3.'*ae2) + (cross(alpha3,r3c3)) + (cross(w3,cross(w3,r3c3)));
ae3 = (R2_3.'*ae2) + (cross(alpha3,r34))  + (cross(w3,cross(w3,r34)));

newton_g1 = -R0_1.'*g*k
newton_g2 = -R0_2.'*g*k
newton_g3 = -R0_3.'*g*k

% Backwards recursion

f4 = [0;0;0];
tau4 = [0;0;0];
R3_4=[0 0 0;0 0 0;0 0 0];

%f3 = R3_4*f4 + m3*ac3 - m3*newton_g3;
%tau3= R3_4*tau4 - cross(f3,r3c3) + cross(R3_4*f4,r4c3) + I3*alpha3 + cross(w3,I3*w3);
%f2 = R2_3*f3 + m2*ac2 - m2*newton_g2;
%tau2 = R2_3*tau3 - cross(f2,r2c2) + cross(R2_3*f3,r3c2) + I2*alpha2 + cross(w2,I2*w2);
%f1 = R1_2*f2 + m1*ac1 - m1*newton_g1;
%tau1 = R1_2 *tau2 - cross(f1,r1c1) + cross(R1_2*f2,r2c1) + I1*alpha1 + cross(w1,I1*w1);

f3 = R2_3*f4 + m3*ac3 - m3*newton_g3;
tau3= R2_3*tau4 - cross(f3,r3c3) + cross(R2_3*f4,r4c3) + I3*alpha3 + cross(w3,I3*w3);

f2 = R1_2*f3 + m2*ac2 - m2*newton_g2;
tau2 = R1_2*tau3 - cross(f2,r2c2) + cross(R1_2*f3,r3c2) + I2*alpha2 + cross(w2,I2*w2);

f1 = R0_1*f2 + m1*ac1 - m1*newton_g1;
tau1 = R0_1 *tau2 - cross(f1,r1c1) + cross(R0_1*f2,r2c1) + I1*alpha1 + cross(w1,I1*w1);

Newton_Tau = simplify(expand([tau1(3); tau2(3); tau3(3)]))

simplify(Tau-Newton_Tau)  % not zero ... I get an extra term somewher

% when q= [0 0 0] or when both q2 and q3 are 0 I get the same expression
% but when q2 and q3 are not zero i get different results for lagrange and
% newton method Idk
```

```
% below i tried to test this out with different values
% still not zero dont know why
lagrange= (subs(Tau, [q1 q2 q3 q_ddot_1 q_ddot_2 q_ddot_3 q_dot_1 q_dot_2 q_dot_3 ], [pi/2 pi 3*pi/2 0 0 0 0 0 0 ]))
newton= (subs(Newton_Tau, [q1 q2 q3 q_ddot_1 q_ddot_2 q_ddot_3 q_dot_1 q_dot_2 q_dot_3 ], [pi/2 pi 3*pi/2 0 0 0 0 0 0 ]))

simplify(lagrange-newton)
```

```
z0 =

     0
     0
     1


z1 =

 sin(q1)
-cos(q1)
       0


z2 =

 sin(q1)
-cos(q1)
       0


newton_g1 =

 0
-g
 0


newton_g2 =

-g*sin(q2)
-g*cos(q2)
         0


newton_g3 =

-g*sin(q2 + q3)
-g*cos(q2 + q3)
              0


Newton_Tau =

                                                        I2*q_ddot_1*cos(q2) + I3*q_ddot_1*cos(q3) + (m3*q_ddot_1*r3^2*cos(2*q2 + q3))/2 + (l2*m3*q_ddot_1*r3)/2 - I2
I2*q_ddot_2 + I3*q_ddot_2 + I3*q_ddot_3 + m2*q_ddot_2*r2^2 + m3*q_ddot_2*r3^2 + m3*q_ddot_3*r3^2 + l2^2*m3*q_ddot_2*cos(q2 - q3) + (l2^2*m3*q_dot_1^2*sin(q2 + q3))/


ans =

I1*q_ddot_1 + I2*q_ddot_1 + I2*q_ddot_2 + I3*q_ddot_1 + I3*q_ddot_2 + I3*q_ddot_3 + (l2^2*m3*q_ddot_1)/2 + (m2*q_ddot_1*r2^2)/2 + (m3*q_ddot_1*r3^2)/2 - I2*q_ddot_1



lagrange =

                  0
- g*l2*m3 - g*m2*r2
                  0


newton =

       0
-g*m2*r2
       0


ans =

       0
-g*l2*m3
       0
```

Problem 6- Ic1=I1 Ic2=I2 Ic3= I3 I1=I2=I3=0 q=0 then l=0.3 m0 0.5 g=9.8

```
Is=0;
ls=0.3;
```

```
ms= 0.5;
gs= 9.8;

lagrange= double(subs(Tau, [q1 q2 q3 q_ddot_1 q_ddot_2 q_ddot_3 q_dot_1 q_dot_2 q_dot_3 m1 m2 m3 g l1 l2 l3 r1 r2 r3], [0 0 0 0 0 0 0 0 0 ms ms ms gs ls ls ls ls ls
newton= double(subs(Newton_Tau, [q1 q2 q3 q_ddot_1 q_ddot_2 q_ddot_3 q_dot_1 q_dot_2 q_dot_3 m1 m2 m3 g l1 l2 l3 r1 r2 r3], [0 0 0 0 0 0 0 0 0 ms ms ms gs ls ls ls
```

```
lagrange =

         0
    4.4100
    1.4700


newton =

         0
    4.4100
    1.4700
```

```
function X = vector_2_skew(x)

X=[0 -x(3) x(2) ; x(3) 0 -x(1) ; -x(2) x(1) 0 ];

end
```

```
D =

[I1 + I2 + I3 + (l2^2*m3)/2 + (m2*r2^2)/2 + (m3*r3^2)/2 + (m3*r3^2*cos(2*q2 + 2*q3))/2 + (l2^2*m3*cos(2*q2))/2 + (m2*r2^2*cos(2*q2))/2 + l2*m3*r3*cos(q3) + l2*m3*r3
[
[
```

Published with MATLAB® R2020b