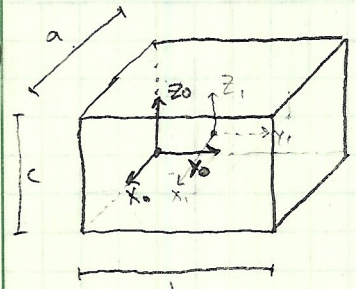
HW 5 | MEEN 537 | Dustan Kraus

1a: 7-3)



By direct calculation: around O_0

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

O_1 is at c.o.m

$$I_{xx} = \rho \int_0^c \int_0^b \int_0^a y^2 + z^2 \, dx\, dy\, dz = \rho \int_0^c \int_0^b (y^2+z^2)x\Big|_0^a dy\, dz = a\rho \int_0^c \int_0^b y^2 + z^2 \, dy\, dz$$

$$= a\rho \int_0^c \tfrac{1}{3}y^3 + z^2 y \Big|_0^b dz = ab\rho \int_0^c \tfrac{1}{3}b^2 + z^2 \, dz = ab\rho\left(\tfrac{1}{3}b^2 z + \tfrac{1}{3}z^3 \Big|_0^c\right)$$

$$= ab\rho\left(\tfrac{1}{3}b^2 c + \tfrac{1}{3}c^3\right) \Rightarrow \quad I_{xx} = \tfrac{1}{3}abc\rho\left(b^2 + c^2\right) \quad \& \quad \underline{m = abc\rho}$$

$$\boxed{I_{xx} = \tfrac{1}{3}m\left(b^2 + c^2\right)} \leftarrow$$

$$I_{xy} = I_{yx} = -\rho \int_0^c \int_0^b \int_0^a xy \, dx\, dy\, dz = -\rho \int_0^c \int_0^b \tfrac{1}{2}x^2 y\Big|_0^a dy\, dz = -\tfrac{1}{2}a^2\rho \int_0^c \int_0^b y \, dy\, dz$$

$$= -\tfrac{1}{2}a^2\rho \int_0^c \tfrac{1}{2}y^2\Big|_0^b dz = -\tfrac{1}{4}a^2 b^2 \rho \int_0^c dz = -\tfrac{1}{4}a^2 b^2 \rho\left(z\Big|_0^c\right) = -\tfrac{1}{4}abc\rho(ab)$$

$$\Rightarrow \boxed{I_{xy} = I_{yx} = -\tfrac{1}{4}mab} \leftarrow$$

using matlab for the rest:

$$\boxed{I_{yy} = \tfrac{1}{3}m\left(a^2 + c^2\right)} \leftarrow \qquad \boxed{I_{zz} = \tfrac{1}{3}m\left(a^2 + b^2\right)} \leftarrow$$

$$\boxed{I_{zx} = I_{xz} = -\tfrac{1}{4}mac} \leftarrow \qquad \boxed{I_{zy} = I_{yz} = -\tfrac{1}{4}mbc} \leftarrow$$

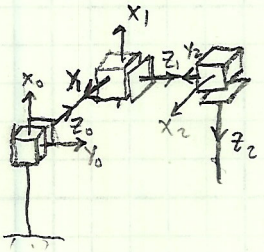
1b: 7-5) Show $D(q)$ for an n Link robot is always positive definite

$$D(q) = \sum_{i=1}^{n}\left(m_i J_{vi}^T J_{vi} + J_{wi}^T R_i I_i R_i^T J_{wi}\right) \text{ - is}$$

$D(q)$ is positive definite if $z^T D(q) z$ is positive for all z (z is a non-zero column vector of real numbers)

$K = \tfrac{1}{2}\dot{q}^T D(q)\dot{q}$ $\Leftarrow$ in this case $\dot{q} = z$ $\therefore$ Kinetic energy is always positive so $D(q)$ is positive definite

1c: 7-7)



(a) links are len. 1, wr $\frac{1}{4}$, hr $\frac{1}{4}$, mass 1
find J_i for $i = 1, 2, 3$ J_i is the inertia tensor

$$J_1 = J_2 = J_3 = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \Rightarrow \begin{matrix} I_{xx} = \frac{1}{12}\left(1^2 + (\frac{1}{4})^2\right) \\ I_{yy} = I_{xx} \\ I_{zz} = \frac{1}{12}\left((\frac{1}{4})^2 + (\frac{1}{4})^2\right) \end{matrix}$$

$$\boxed{J_1 = J_2 = J_3 = \begin{bmatrix} 0.0885 & 0 & 0 \\ 0 & 0.0885 & 0 \\ 0 & 0 & 0.0104 \end{bmatrix}} \leftarrow$$

(b) since no rotation $J_w = 0$ so don't even need J_i...

$$D(q) = \sum_{i=1}^{n}\left(m_i J_{vi}(q)^T J_{vi}(q)\right) \qquad J_{v1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad J_{v2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad J_{v3} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$D(q) = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(c) $c_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}$ but D is constant, so

$$\underline{c_{ijk} = 0}$$

means that there are no centripedal or coriolis terms

(d) $D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$ ($C(q,\dot{q})\dot{q} \to 0$)

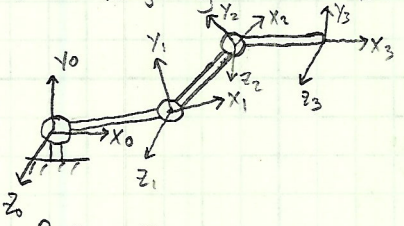
$P_i = m_i g^T r_{ci}$ $\quad g(q) = \frac{\partial P}{\partial q}$

$$P = \sum_{i=1}^{3} m_i g^T r_{ci} = 1\left([9.81\ 0\ 0]\begin{bmatrix}0\\0\\q_1+0.5\end{bmatrix} + [9.81\ 0\ 0]\begin{bmatrix}0\\0.5+q_2\\1+q_1\end{bmatrix} + [9.81\ 0\ 0]\begin{bmatrix}-q_3-0.5\\q_2+1\\q_1+1\end{bmatrix}\right)$$

$P = -9.81(q_3 + 0.5)$ $\quad g_k(q) = \frac{\partial P}{\partial q_k} \Rightarrow$ $\quad g(q) = \begin{bmatrix}0\\0\\-9.81\end{bmatrix}$

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\ddot{q} + \begin{bmatrix}0\\0\\-9.81\end{bmatrix} = \begin{bmatrix}F_1\\F_2\\F_3\end{bmatrix} \longleftarrow$$

1d: 7-10) find Euler-Lagrange EOM for fig. 3.32 manipulator



	θ	d	a	α
1	q_1	0	0.4	0
2	q_2	0	0.4	0
3	q_3	0	0.4	0

$$D(q) = \sum_{i=1}^{n}\left(m_i J_{vi}(q)^T J_{vi}(q) + J_{wi}(q)^T R_i(q) I_i R_i(q)^T J_{wi}(q)\right)$$

rest on matlab

Table of Contents


```
clear all
close all

% School Linux
%run ~/Desktop/rvctools/startup_rvc.m
% Personal laptop
%run C:\Users\Dustan\Desktop\rvctools\startup_rvc.m
```

Problem 1a) 7-3

```
syms x y z a b c rho
% note: a*b*c*rho = m
% using direct calculation
Ixx = rho*int(int(int(y^2 + z^2,x,0,a),y,0,b),z,0,c);
Iyy = rho*int(int(int(x^2 + z^2,x,0,a),y,0,b),z,0,c);
Izz = rho*int(int(int(x^2 + y^2,x,0,a),y,0,b),z,0,c);
Ixy = -rho*int(int(int(x*y,x,0,a),y,0,b),z,0,c);
Ixz = -rho*int(int(int(x*z,x,0,a),y,0,b),z,0,c);
Iyz = -rho*int(int(int(y*z,x,0,a),y,0,b),z,0,c);
Iyx = Ixy;
Izx = Ixz;
Izy = Iyz;
I_1 = [Ixx Ixy Ixz; Iyx Iyy Iyz; Izx Izy Izz];

% using parallel axis theorem (Tensor generalization from wikipedia)
Ixx = rho*int(int(int(y^2 + z^2,x,-a/2,a/2),y,-b/2,b/2),z,-c/2,c/2);
Iyy = rho*int(int(int(x^2 + z^2,x,-a/2,a/2),y,-b/2,b/2),z,-c/2,c/2);
Izz = rho*int(int(int(x^2 + y^2,x,-a/2,a/2),y,-b/2,b/2),z,-c/2,c/2);
Ixy = -rho*int(int(int(x*y,x,-a/2,a/2),y,-b/2,b/2),z,-c/2,c/2);
Ixz = -rho*int(int(int(x*z,x,-a/2,a/2),y,-b/2,b/2),z,-c/2,c/2);
Iyz = -rho*int(int(int(y*z,x,-a/2,a/2),y,-b/2,b/2),z,-c/2,c/2);
Iyx = Ixy;
Izx = Ixz;
Izy = Iyz;
Icom = [Ixx Ixy Ixz; Iyx Iyy Iyz; Izx Izy Izz];
R = [-a/2 -b/2 -c/2];
I_2 = simplify(Icom + a*b*c*rho*((R*R.')*eye(3) - R.'*R));
```

Problem 1c) 7-7

```
% Part a
Ixx = (1/12)*(1+0.25^2);
```

```
Iyy = Ixx;
Izz = (1/12)*(0.25^2 + 0.25^2);

% Part b
Jv1 = [0 0 0; 0 0 0; 1 0 0];
Jv2 = [0 0 0; 0 1 0; 1 0 0];
Jv3 = [0 0 -1; 0 1 0; 1 0 0];

Dq = Jv1.'*Jv1 + Jv2.'*Jv2 + Jv3.'*Jv3;
```

Problem 1d) 7-10

```
clear all
L = 0.4; %link length
m = 1; %link mass
b = 0.1;
h = 0.1;
rho = m/(L*b*h);
% Make robots
Lk_1 = Link('revolute', 'offset', 0, 'd', 0, 'a', L/2, 'alpha', 0);

Lk_2(1) = Link('revolute', 'offset', 0, 'd', 0, 'a', L, 'alpha', 0);
Lk_2(2) = Link('revolute', 'offset', 0, 'd', 0, 'a', L/2, 'alpha', 0);

Lk_3(1) = Link('revolute', 'offset', 0, 'd', 0, 'a', L, 'alpha', 0);
Lk_3(2) = Link('revolute', 'offset', 0, 'd', 0, 'a', L, 'alpha', 0);
Lk_3(3) = Link('revolute', 'offset', 0, 'd', 0, 'a', L/2, 'alpha', 0);

% Make the first robot to get Jacobian for the c.o.m. of the first link
bot1 = SerialLink(Lk_1, 'name', 'com1');
% Make the 2nd robot to get Jacobian for the c.o.m. of the 2nd link
bot2 = SerialLink(Lk_2, 'name', 'com2');
% Make the 3rd robot to get Jacobian for the c.o.m. of the 3rd link
bot3 = SerialLink(Lk_3, 'name', 'com3');

syms q q1 q2 q3
q = [q1, q2, q3];
% Jacobian to c.o.m. of 1st link
J1 = [bot1.jacob0([q(1)]), [0;0;0;0;0;0], [0;0;0;0;0;0]];
Jv1 = J1(1:3,:);
Jw1 = J1(4:6,:);
% Jacobian to c.o.m. of 2nd link
J2 = [bot2.jacob0([q(1) q(2)]), [0;0;0;0;0;0]];
Jv2 = J2(1:3,:);
Jw2 = J2(4:6,:);
% Jacobian to c.o.m. of 3rd link
J3 = bot3.jacob0([q(1) q(2) q(3)]);
Jv3 = J3(1:3,:);
Jw3 = J3(4:6,:);

%Inertia tensor for each link at the c.o.m.
Ixx = (1/12)*(b^2 + h^2);
Iyy = (1/12)*(L^2 + b^2);
```

```
Izz = Iyy;
I1 = [Ixx 0 0; 0 Iyy 0; 0 0 Izz];
I2 = I1;
I3 = I1;

%Get rotation matrices for each c.o.m. tensor to the base frame
T1 = bot1.fkine([q(1)]);
R1 = T1(1:3,1:3);

T2 = bot2.fkine([q(1) q(2)]);
R2 = T2(1:3,1:3);

T3 = bot3.fkine([q(1) q(2) q(3)]);
R3 = T3(1:3,1:3);

% Calculate D(q)
D1q = m*Jv1.'*Jv1 + Jw1.'*R1*I1*R1.'*Jw1;
D2q = m*Jv2.'*Jv2 + Jw2.'*R2*I2*R2.'*Jw2;
D3q = m*Jv3.'*Jv3 + Jw3.'*R3*I3*R3.'*Jw3;
Dq = simplify(D1q + D2q + D3q)
C = sym(zeros(3));

% Calculate C(q, qdot)
for k = 1:3,
    for j = 1:3,
        for i = 1:3,
            C(k,j) = C(k,j) + (1/2)*(diff(Dq(k,i),q(j)) + ...
                diff(Dq(k,i),q(j)) - diff(Dq(i,j),q(k)));
        end
    end
end
C = simplify(C)

% Calculate g(q)
g = [0; 9.81; 0];
rc = [T1(1:3,4), T2(1:3,4), T3(1:3,4)];
P = sym(0);
for i = 1:3,
    P = P + m*g.'*rc(:,i);
end
P = simplify(P);
gq = sym(zeros(3,1));
for i = 1:3,
    gq(i) = diff(P,q(i));
    gq(i) = simplify(gq(i));
end

gq

syms Tau Tau1 Tau2 Tau3
Tau = [Tau1; Tau2; Tau3];


Dq =
```

```
[ (4*cos(q2 + q3))/25 + (12*cos(q2))/25 + (4*cos(q3))/25 + 257/400, (2*cos(q2 + q3
[  (2*cos(q2 + q3))/25 + (6*cos(q2))/25 + (4*cos(q3))/25 + 161/600,
[                    (2*cos(q2 + q3))/25 + (2*cos(q3))/25 + 13/240,


C =

[                                      0, - (8*sin(q2 + q3))/25 - (18*sin(q2))/25, -
[ (4*sin(q2 + q3))/25 + (9*sin(q2))/25,       - sin(q2 + q3)/25 - (3*sin(q2))/25,
[      (4*sin(q2 + q3))/25 + sin(q3)/5,              sin(q3)/5 - sin(q2 + q3)/25,


gq =

 (981*cos(q1 + q2 + q3))/500 + (2943*cos(q1 + q2))/500 + (981*cos(q1))/100
                     (981*cos(q1 + q2 + q3))/500 + (2943*cos(q1 + q2))/500
                                           (981*cos(q1 + q2 + q3))/500
```

Published with MATLAB® R2014b

```
clear all
close all

% School Linux
%run ~/Desktop/rvctools/startup_rvc.m
% Personal laptop
%run C:\Users\Dustan\Desktop\rvctools\startup_rvc.m
```

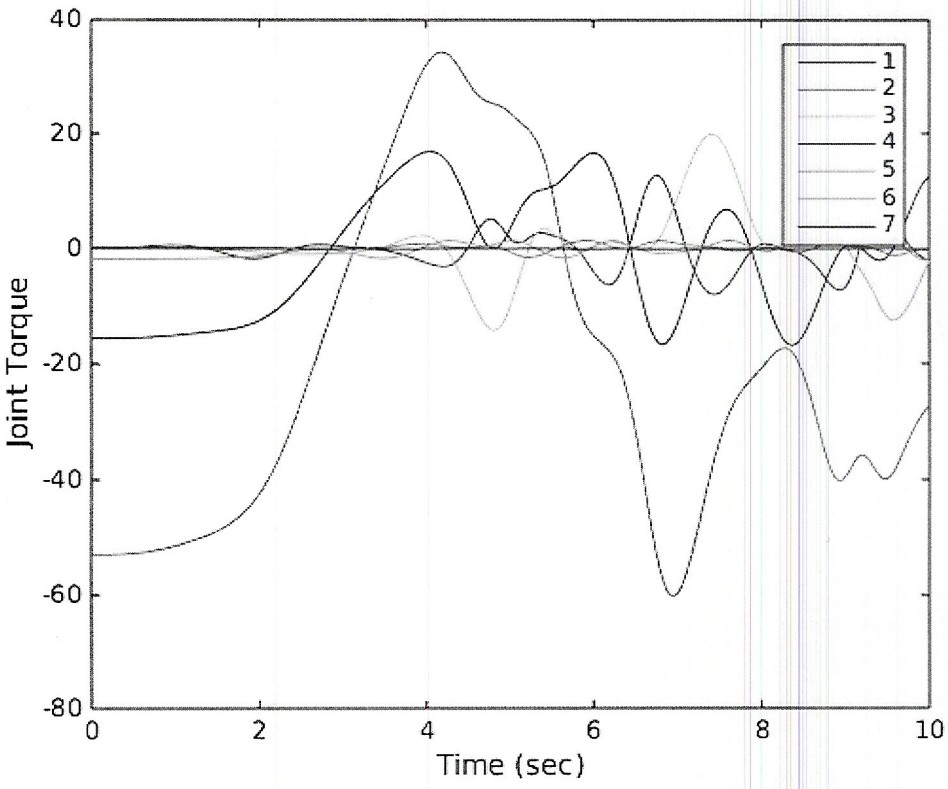
Problem 2b

```
load desired_accel.mat
qdd = q;
clear q
%initialize qd and q
qd = [0 0 0 0 0 0 0];
q = qd;
for i = 2:length(qdd),
   qd(i,:) =  qd(i-1,:) + qdd(i-1,:)*(t(i) - t(i-1));
   q(i,:) =  q(i-1,:) + qd(i-1,:)*(t(i) - t(i-1));
end

[left, right] = mdl_baxter('sim');
tau = zeros(size(qdd));
%calculating the mass, corilios and gravity terms
for i = 1:length(qdd),
    M = left.inertia(q(i,:));
    C = left.coriolis(q(i,:), qd(i,:));
    G = left.gravload(q(i,:));
    tau(i,:) = (M*qdd(i,:).' + C*qd(i,:).' + G.').';
end

figure()
plot(t,tau)
legend('1', '2', '3', '4', '5', '6', '7');
xlabel('Time (sec)')
ylabel('Joint Torque')
```

Loaded Baxter Model in Simulation Mode (urdf-data)



Published with MATLAB® R2014b