MeEn 537 Homework #5 Solution

7-3:

There are two approaches. For the first, we look up in a table that the principal moments of inertia for a rectangular solid are the following:

$$\begin{aligned}
I_{xx} &= \frac{1}{12}m(b^2+c^2)\\
I_{yy} &= \frac{1}{12}m(a^2+c^2)\\
I_{zz} &= \frac{1}{12}m(a^2+b^2)
\end{aligned}$$

Then we can use the fact that the inertia tensor at a new location of distance r from the center of mass is defined as:

$$I_r = I_{COM} - S(r)S(r)$$

where

$$I_{COM} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

and

$$S([a,b,c]) = \begin{bmatrix} 0 & -\frac{c}{2} & \frac{b}{2} \\ \frac{c}{2} & 0 & -\frac{a}{2} \\ -\frac{b}{2} & \frac{a}{2} & 0 \end{bmatrix}$$

The solution then is:

$$I_r = \begin{bmatrix} \frac{m}{3}(b^2+c^2) & -\frac{abm}{4} & -\frac{acm}{4} \\ -\frac{abm}{4} & \frac{m}{3}(a^2+c^2) & -\frac{bcm}{4} \\ -\frac{acm}{4} & -\frac{bcm}{4} & \frac{m}{3}(a^2+b^2) \end{bmatrix}$$

For the second approach, we can use the integrals as shown from the book solution on the next page. This solution is identical except it uses ρ for density and introduces additional terms for volume ($V = abc$) instead of using mass:

7-3 Referring to Figure 7.6 we have

$$\begin{aligned}\int (y^2+z^2)dm &= \int_0^c \int_0^b \int_0^a (y^2+z^2)\rho dx dy dz \\ &= \frac{1}{3}\rho abc(b^2+c^2)\end{aligned}$$

Computing the remaining terms similarly we have

$$\begin{aligned} I &= \begin{bmatrix} \int (y^2+z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int (x^2+z^2)dm & -\int yzdm \\ -\int xzdm & -\int yzdm & -\int (x^2+y^2)dm \end{bmatrix} \\ &= \rho \begin{bmatrix} \frac{1}{3}abc(b^2+c^2) & -a^2b^2c/4 & -a^2bc^2/4 \\ -a^2b^2c/4 & \frac{1}{3}abc(a^2+c^2) & -ab^2c^2/4 \\ -a^2bc^2/4 & -ab^2c^2/4 & \frac{1}{3}abc(a^2+b^2) \end{bmatrix}\end{aligned}$$

7-5:

The easiest way to show this is to use the definition of Kinetic energy which is as follows:

$$\dot{q}^T D(q)\dot{q}$$

Since the definition of kinetic energy is that it must always be greater than zero unless the velocity terms are zero. In fact, this is the definition of a positive definite matrix. For any arbitrary vector x if a matrix P is positive definite, then the following must be true for any non-zero x:

$$x^T Px > 0$$

Therefore, by inspection, $D(q)$ must be positive definite for any value of q.

7-7:

a) We use the same equation from a table for a rectangular solid $I = \frac{1}{12}m(l_1^2 + l_2^2)$ where l_1 and l_2 are the dimensions of the solid and I is at the center of mass. Assume that the rotational inertia is in the link frame so that the z-direction is always aligned with the length of the link (you could instead define them such that the length goes along the z-axis, then the y-axis, then the x-axis in a global frame):

$$J_1 \quad = J_2 = J_3 = \begin{bmatrix} \frac{(1)(\frac{1}{4}^2+\frac{1}{4}^2)}{12} & 0 & 0 \\ 0 & \frac{(1)(\frac{1}{4}^2+\frac{1}{4}^2)}{12} & 0 \\ 0 & 0 & \frac{(1)(\frac{1}{4}^2+1^2)}{12} \end{bmatrix} = \begin{bmatrix} 0.0104 & 0 & 0 \\ 0 & 0.0104 & 0 \\ 0 & 0 & 0.0885 \end{bmatrix}$$

b) To compute $D(q)$, we can look at the equation we used in class, however in this case there is no rotational velocity so:

$$D(q) = \sum_{i=1}^{3} [m_i J_{v_c,i}^T J_{v_c,i}]$$

in this case, because it is a Cartesian manipulator, the jacobians take on the following form:

$$
\begin{aligned}
J_{v_c,1} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\
J_{v_c,2} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\
J_{v_c,3} &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}
\end{aligned}
$$

such that $D(q)$ is

$$
D(q) = \begin{bmatrix} m_1 + m_2 + m_3 & 0 & 0 \\ 0 & m_2 + m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix}
$$

c) Because $D(q)$ is not actually a funciton of q in this case, and the Christoffel coeficients all involve partial derviatives of $D(q)$, the Christoffel coefficients must all be zero for this robot.

d) To get the complete equations of motion, we need to also define the potential energy P and take its partial derivative with respect to each joint. If we let the reference datum for potential energy be the resting position of the first link, then we can define the potential energy for each link as follows (where the 2nd vector in each expression is just the forward kinematics to the center of mass):

$$
\begin{aligned}
P_1 &= m_1 \begin{bmatrix} 0 & 0 & 9.81 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ q_1 \end{bmatrix} \\
P_2 &= m_2 \begin{bmatrix} 0 & 0 & 9.81 \end{bmatrix} \begin{bmatrix} 0 \\ q_2 + offset_2 \\ q_1 + offset_1 \end{bmatrix} \\
P_3 &= m_3 \begin{bmatrix} 0 & 0 & 9.81 \end{bmatrix} \begin{bmatrix} q_3 + offset_3 \\ q_2 + offset_2 \\ q_1 + offset_1 \end{bmatrix}
\end{aligned}
$$

now, if $P = \sum P_i$, then the partial of P with respect to each q_i will give us the torque or force terms due to gravity:

$$G(q) = \begin{bmatrix} 9.81(m_1 + m_2 + m_3) \\ 0 \\ 0 \end{bmatrix}$$

then the equations of motion (if we neglect friction and assume the joint forces are F) are:

$$\begin{aligned} D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) &= F \\ \begin{bmatrix} m_1+m_2+m_3 & 0 & 0 \\ 0 & m_2+m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} g(m_1+m_2+m_3) \\ 0 \\ 0 \end{bmatrix} &= \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \end{aligned}$$

7-10: We first include the actual code that calculates symbolic versions of $D(q)$, $C(q,\dot{q}$, and G. Then we also include three very simple helper scripts that defined the kinematics to the center of mass of each link.

```
clear all;
close all;

syms m1 m2 m3 Ixx Iyy Izz q1 q2 q3 q1_d q2_d q3_d g;

%defining three different robots, to easily find the kinematics to the
%center of mass (COM) for each link
three_link_first_com;
three_link_second_com;
three_link_third_com;

%just assigning the robots simpler names
tl1 = three_link_1_com;
tl2 = three_link_2_com;
tl3 = three_link_3_com;

%assigning vectors of symbolic variables to make calculations easier
jt_angles = [q1 q2 q3];
jt_vels = [q1_d, q2_d, q3_d];
ms = [m1 m2 m3];
robots = {tl1; tl2; tl3};

%I'll assume that the inertia tensor has no off-diagongal terms (i.e.
 the
%links are symmetric) and that each link is identical.
I = diag([Ixx, Iyy, Izz]);
Is = {I, I, I};

%code before this point is not very general. In python it would be
 easy to
%make it general for any number links. In Matlab I'm sure it's
 possible but
%it was a little harder

%define the number of links
n = 3;

%make a way to store jacobians if we need them for later
Js = {};

%start the mass matrix and potential energy as a bunch of zeros
D = vpa(zeros(n,n));
P = vpa(0);

%the mass matrix and the potential energy are calculated in this loop
 for
%each link
for i=1:1:n
   Js{i} = robots{i}.jacob0(jt_angles(1:i)');

   %this is the symbolic version of the forward kinematics,
 progressively
```

```
   %including more links for each iteration
   T = robots{i}.fkine(jt_angles(1:i));

   %pulling off the rotation matrix
   R = T(1:3,1:3);

   %calculating the contribution of that ith link to the mass matrix
   D(1:i,1:i) = D(1:i,1:i) + ms(i)*Js{i}(1:3,:)'*Js{i}(1:3,:) + ...
       Js{i}(4:6,:)'*R*Is{i}*R'*Js{i}(4:6,:);

   %adding in the potential energy for the ith link as well
   P = P + ms(i)*[0, g, 0]*T(1:3,4);
end

%calculating the gravity torques/forces now
G = [];
for i=1:1:n
    G = [G; diff(P, jt_angles(i))];
end


%calculating the Coriolis and Centripetal terms
C = vpa(zeros(n,n));
for k=1:1:n
   for j=1:1:n
       for i =1:1:n
          C(k,j) = C(k,j) + 0.5*(diff(D(k,j),jt_angles(i)) + ...
                                 diff(D(k,i),jt_angles(j)) + ...

 diff(D(i,j),jt_angles(k)))*jt_vels(i);
       end
   end
end


three_link_1_com =

three_link (1 axis, R, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|          q|          0|        0.2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0         0  1  0  0           0  1  0  0
       9.81         0  0  1  0           0  0  1  0
                    0  0  0  1           0  0  0  1


three_link_2_com =

three_link (2 axis, RR, stdDH, fastRNE)
```

```
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|        0.4|          0|          0|
|  2|         q2|          0|        0.2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0         0  1  0  0           0  1  0  0
       9.81         0  0  1  0           0  0  1  0
                    0  0  0  1           0  0  0  1


three_link_3_com = 

three_link (3 axis, RRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|        0.4|          0|          0|
|  2|         q2|          0|        0.4|          0|          0|
|  3|         q3|          0|        0.2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0         0  1  0  0           0  1  0  0
       9.81         0  0  1  0           0  0  1  0
                    0  0  0  1           0  0  0  1
```

Published with MATLAB® R2015a

```
L(1) = Link ([0, 0., 0.2, 0], 'standard');
three_link_1_com = SerialLink(L, 'name', 'three_link')
```

```
three_link_1_com =

three_link (1 axis, R, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|          q|          0|        0.2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0         0  1  0  0           0  1  0  0
       9.81         0  0  1  0           0  0  1  0
                    0  0  0  1           0  0  0  1
```

Published with MATLAB® R2015a

```
L(1) = Link ([0, 0., 0.4, 0], 'standard');
L(2) = Link ([0, 0., 0.2, 0], 'standard');

three_link_2_com = SerialLink(L, 'name', 'three_link')
```

```

three_link_2_com =

three_link (2 axis, RR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|        0.4|          0|          0|
|  2|         q2|          0|        0.2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  1  0  0           0  1  0  0
        9.81         0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1

```

Published with MATLAB® R2015a

```
L(1) = Link ([0, 0., 0.4, 0], 'standard');
L(2) = Link ([0, 0., 0.4, 0], 'standard');
L(3) = Link ([0, 0., 0.2, 0], 'standard');

three_link_3_com = SerialLink(L, 'name', 'three_link')
```

```
three_link_3_com =

three_link (3 axis, RRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|        0.4|          0|          0|
|  2|         q2|          0|        0.4|          0|          0|
|  3|         q3|          0|        0.2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  1  0  0           0  1  0  0
        9.81         0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```

Published with MATLAB® R2015a

2.b)

See attached code and plots

```
run D:\Dropbox\rvctools_new\startup_rvc.m;
load ./desired_accel.mat;

[left, right] = mdl_baxter('sim');

%first have to find the approximate velocity and acceleration from the
%joint angles
q_d = [(zeros(1,7)); (q(2:end,:)-q(1:end-1, :))/0.01];
q_dd = [zeros(1,7); (q_d(2:end,:)-q_d(1:end-1, :))/0.01];

%now just evaluate the dynamic equations in order to find the torque
tau = left.rne(q, q_d, q_dd);

fig = figure()
for i=1:1:7
    subplot(7,1,i);
    hold on;
    plot(t,tau(:,i));
end
xlabel('time (s)');
subplot(7,1,4)
ylabel('torque (N-m)');
```

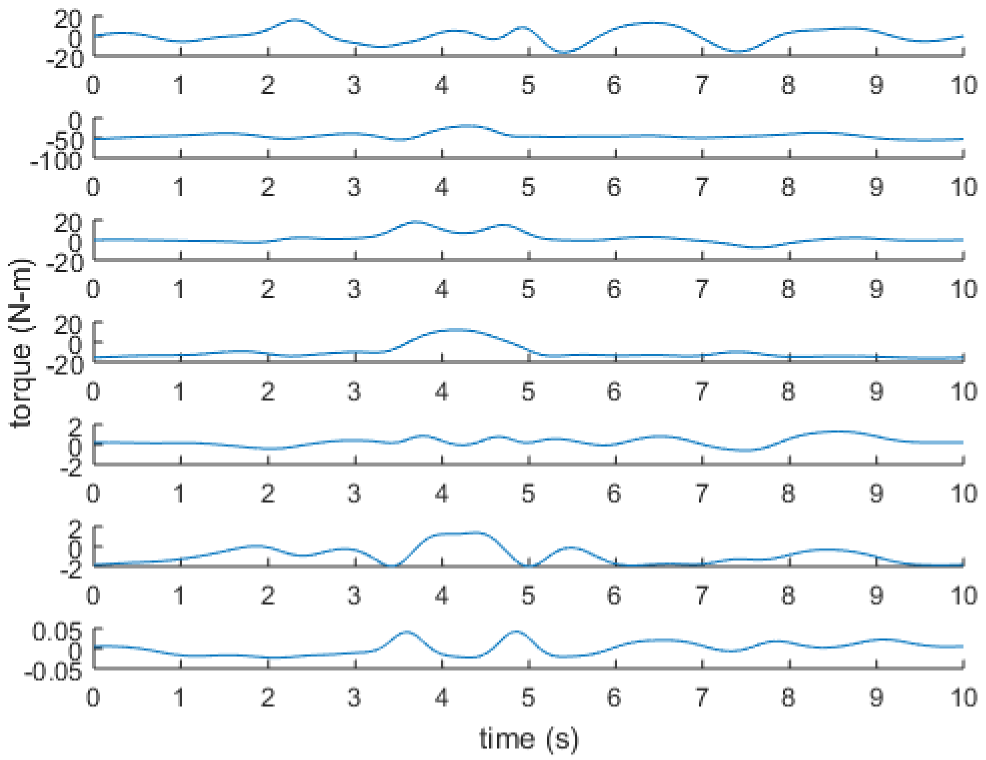
```
Robotics, Vision & Control: (c) Peter Corke 1992-2011 http://
www.petercorke.com
- Robotics Toolbox for Matlab (release 9.10)
 - pHRIWARE (release 1.1): pHRIWARE is Copyrighted by Bryan Moutrie
 (2013-2016) (c)
Run rtbdemo to explore the toolbox
Loaded Baxter Model in Simulation Mode (urdf-data)

fig =

  Figure (3) with properties:

      Number: 3
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [680 558 560 420]
       Units: 'pixels'

  Use GET to show all properties
```



Published with MATLAB® R2015a