MIDDLE EAST TECHNICAL UNIVERSITY

Department of Electrical and Electronics Engineering

EE587 Introduction to Robotics

Term Project - Report 1

Hunting Control of an Amphibious Creature

28.11.2018

Prepared by Ufuk EFE

Submitted to: Prof. Aydan Erkmen

1. INTRODUCTION

In this project, hunting control system of an amphibious creature which lives in water will be designed. The control mechanism will get measurements from eyes of the creature, then the creature will go towards to insect in order to catch it with its spiraling tongue.

In this project report, assumptions are stated, Denavit-Hartenberg Parameters of the creature are determined, kinematic model developed, and dynamical model is indicated with dynamical equations. The later part of project, Jacobians are going to find, and a visual tracking system will be developed. At the end with the help of the capturing control of insects the overall model will be simulated and animated.

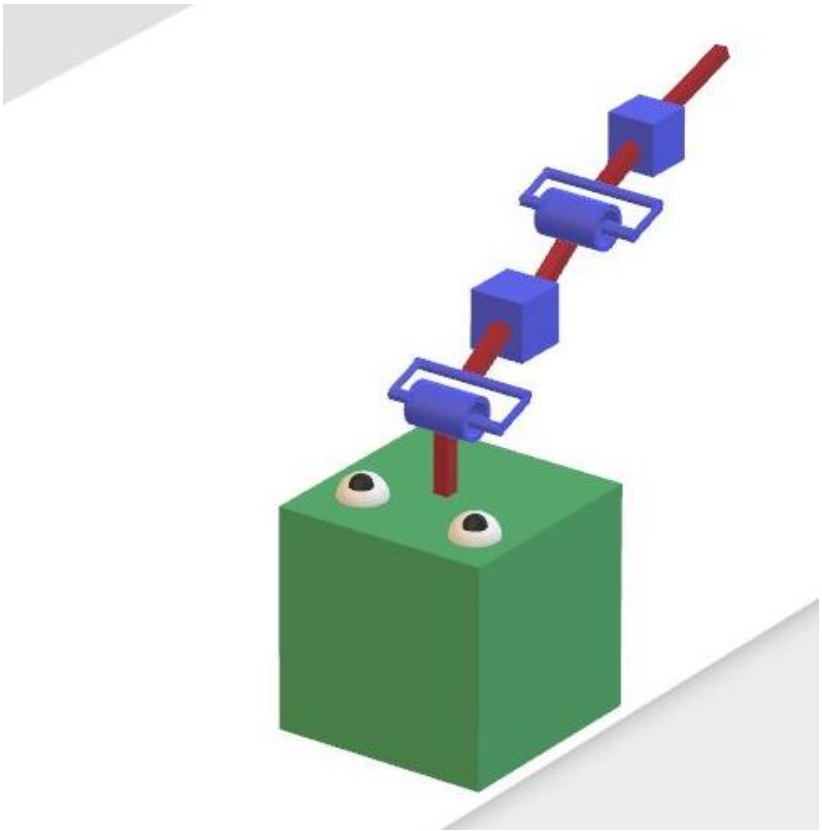

Figure 1: 3D view of the creature

1.1 ASSUMPTIONS

Mechanical Assumptions

1. The creature consists of two moving parts, body and tongue.
2. Body can move in 3 DoF while tongue can only go in one direction.
3. Tongue will have a specific maximum length, for example 15 cm.
4. Tongue is made of sticky material, when it touches an insect it catches it.

Assumptions for visual system

5. The creature has stereo vision, thanks to its two symmetric eyes, therefore it knows the distance to insects, i.e. creature knows depth of the scene.
6. Eyes of the creature look upward, compatibly with assumption 2.
7. The creature lives in a narrow range of the water so that the effect of diffraction is constant.
8. There may be more than one insect in the scene and they also may move, but the movement of an insect will be constant in order to estimate motion well i.e. the optical flow of an insect in image plane will be constant, since the visual controller performs to bring the insect in the middle of the image plane to hunt it, compatibly with assumption 2.
9. The creature will have intelligence to catch the optimal insect i.e. it will find the nearest insect and go towards it.

2. DENAVIT-HERTANBERG PARAMETERS OF THE CREATURE

In this section, DH parameters of the creature is determined. The body and tongue are hand drawn to better visualization.

3. KINEMATIC MODEL OF THE CREATURE

We can obtain forward kinematic model of the creature by using following transformation matrix. To use this matrix, we will get the DH parameters from previous section.

$$^{i-1}A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} = \begin{bmatrix} c_{\theta_i} & -c_{\alpha_i} s_{\theta_i} & s_{\alpha_i} s_{\theta_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\alpha_i} c_{\theta_i} & -s_{\alpha_i} c_{\theta_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By using MATLAB 2017b, one can easily compute the A matrices. Corresponding A matrices are computed as follows; (matrices are named as Ai_(i-1) in MATLAB)

Note that; q = [q1 q2 q3 q4 q5 q6 q7 q8] where each element is referred to generalized coordinates. q = [d1 d2 d3 a4 theta5 d6 theta7 d8]

Also note that, a4 (which defines the distance between the body center and first joint of tongue) is taken as 10.

```
A1_0 =

[ 0,  0, -1,  0]
[ 1,  0,  0,  0]
[ 0, -1,  0, q1]
[ 0,  0,  0,  1]
```

```
A2_1 =

[  0,  0, 1,  0]
[ -1,  0, 0,  0]
[  0, -1, 0, q2]
[  0,  0, 0,  1]


A3_2 =

[ 1, 0, 0,  0]
[ 0, 1, 0,  0]
[ 0, 0, 1, q3]
[ 0, 0, 0,  1]


A4_3 =

[ 1, 0, 0, 10]
[ 0, 1, 0,  0]
[ 0, 0, 1,  0]
[ 0, 0, 0,  1]


A5_4 =

[ cos(q5), 0,  sin(q5), 0]
[ sin(q5), 0, -cos(q5), 0]
[       0, 1,        0, 0]
[       0, 0,        0, 1]
```

```
A6_5 =

[ 1,  0, 0,  0]
[ 0,  0, 1,  0]
[ 0, -1, 0, q6]
[ 0,  0, 0,  1]


A7_6 =

[ cos(q7), 0,  sin(q7), 0]
[ sin(q7), 0, -cos(q7), 0]
[       0, 1,        0, 0]
[       0, 0,        0, 1]


A8_7 =

[ 1, 0, 0,  0]
[ 0, 1, 0,  0]
[ 0, 0, 1, q8]
[ 0, 0, 0,  1]
```

We can also determine the matrix A frame 0 to 8 by multiplying all the matrices above. Which is resulting with A8_0 matrix.

```
A8_0 =

[ sin(q5 + q7), 0, -cos(q5 + q7),    - q2 - q8*cos(q5 + q7) - q6*cos(q5)]
[            0, 1,             0,                                   q3]
[ cos(q5 + q7), 0,  sin(q5 + q7), q1 + q8*sin(q5 + q7) + q6*sin(q5) + 10]
[            0, 0,             0,                                    1]
```

4. DYNAMIC MODEL OF THE CREATURE

In this section, dynamic model of the creature is developed based on the approach from the book “Robotics: Control, Sensing, Vision, and Intelligence” by K. S. Fu, R. C. Gonzalez and C. S. G. Lee. Here, first note that the system is holonomic since the body can move all directions in 3D space. It is required to use Lagrange-Euler equations.

According to the book, the derivation of the dynamic equations of an n degrees of freedom manipulator is based on the understanding of:

1. The 4 x 4 homogeneous coordinate transformation matrix ${}^{i-1}A_i$, which describes the spatial relationships between the ith and (i-1)th link coordinate frames. It relates a point fixed in link I expressed in homogeneous coordinates with respect to the ith coordinate system to the (i-1)th coordinate system.

2. The Lagrange-Euler equation

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}\imath}\right] - \frac{\partial L}{qi} = \tau i$$

Where;

L = Lagrangian function = kinetic energy K – potential Energy P

K = total kinetic energy of the robot arm

P = total potential energy of the robot arm

q_i = generalized coordinates of the robot arm

$\dot{q}$ = first time derivative of the generalized coordinate, q_i

τ_i = generized force (or torque) applied to the system at joint I to drive link i

The final equation will be in the form $D(q)\ddot{q}+c(q,\dot{q})+g(q)=\tau$

or equivalently $D(q)\ddot{q}+\dot{q}^T C(q)\dot{q}+g(q)=\tau$

We need to find matrices D(q), C(q) and g(q) in order to determine the dynamic model of the creature. We can start with D(q). The position of a point at link i, with respect to the ith link coordinate frame is:

$$ {}^{i}r_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} $$

Related base coordinate frame can be written as;

$$ {}^{0}r_i = {}^{0}A_i\, {}^{i}r_i = {}^{0}A_1\, {}^{1}A_2 \ldots {}^{i-1}A_i\, {}^{i}r_i $$

Also, the speed of that point with respect to the base coordinate frame can be written as;

$$ \begin{aligned} {}^{0}v_i = v_i &= \frac{d({}^{0}r_i)}{dt} = \frac{d({}^{0}A_i\, {}^{i}r_i)}{dt} \\ &= {}^{0}\dot{A}_1\, {}^{1}A_2 \ldots {}^{i-1}A_i\, {}^{i}r_i \\ &+ {}^{0}A_1\, {}^{1}\dot{A}_2 \ldots {}^{i-1}A_i\, {}^{i}r_i \\ &+ \ldots \\ &+ {}^{0}A_1\, {}^{1}A_2 \ldots {}^{i-1}A_i\, {}^{i}\dot{r}_i = \left(\sum_{j=1}^{i} \frac{\partial {}^{0}A_i}{\partial q_j} \dot{q}_j \right) {}^{i}r_i \end{aligned} $$

Then we can define following Q_i matrices, for revolute and prismatic joints respectively;

Revolute:

$$Q_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Prismatic:

$$Q_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

It results with the derivative;

$$\frac{\partial^0 A_i}{\partial q_i} = Q_i \, {}^{i-1}A_i$$

Now, we can define the matrix

$$U_{ij} \triangleq \partial \, {}^0A_i / \partial q_j$$

$$U_{ij} = \frac{\partial^0 A_i}{\partial q_j} = \begin{cases} {}^0A_{j-1} Q_j \, {}^{j-1}A_i & \text{for} j \leq i \\ 0, & \text{otherwise} \end{cases}$$

Then the speed of each joint can be written as

$$v_i = \left(\sum_{j=1}^{i} U_{ij} \dot{q}_j \right) {}^i r_i$$

After all, we can define differential kinetic energy K;

$$\begin{aligned} dK_i &= \frac{1}{2}(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) dm \\ &= \frac{1}{2}\text{trace}(v_i v_i^T) dm = \frac{1}{2}\text{Tr}(v_i v_i^T) dm \\ &= \frac{1}{2}\text{Tr}\left[\sum_{p=1}^{i} U_{ip} \dot{q}_p \, {}^i r_i \left(\sum_{r=1}^{i} U_{ir} \dot{q}_r \, {}^i r_i \right)^T \right] dm \\ &= \frac{1}{2}\text{Tr}\left[\sum_{p=1}^{i} \sum_{r=1}^{i} U_{ip} ({}^i r_i dm \, {}^i r_i^T) U_{ir}^T \dot{q}_r \dot{q}_p \right] \end{aligned}$$

We need kinetic energy K, then we can simply integrate the differential kinetic energy, it gives us the relation with inertia tensor as;

$$J_i = \int {}^i r_i \, {}^i r_i^T dm = \begin{bmatrix} \int x_i^2 dm & \int x_i y_i dm & \int x_i z_i dm & \int x_i dm \\ \int x_i y_i dm & \int y_i^2 dm & \int y_i z_i dm & \int y_i dm \\ \int x_i z_i dm & \int y_i z_i dm & \int z_i^2 dm & \int z_i dm \\ \int x_i dm & \int y_i dm & \int y_i dm & \int dm \end{bmatrix}$$

$$= \begin{bmatrix} \frac{-I_{xx}+I_{yy}+I_{zz}}{2} & I_{xy} & I_{xz} & m_i\overline{x_i} \\ I_{xy} & \frac{I_{xx}-I_{yy}+I_{zz}}{2} & I_{yz} & m_i\overline{y_i} \\ I_{xz} & I_{yz} & \frac{I_{xx}+I_{yy}-I_{zz}}{2} & m_i\overline{z_i} \\ m_i\overline{x_i} & m_i\overline{y_i} & m_i\overline{z_i} & m_i \end{bmatrix}$$

Using these equations, one can calculate the kinetic energy as;

$$K = \sum_{i=1}^{n} K_i = \frac{1}{2} \sum_{i=1}^{n} \mathrm{Tr} \left[\sum_{p=1}^{i} \sum_{r=1}^{i} U_{ip} J_i U_{ir}^T \dot{q}_r \dot{q}_p \right]$$

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{p=1}^{i} \sum_{r=1}^{i} [\mathrm{Tr}(U_{ip} J_i U_{ir}^T) \dot{q}_r \dot{q}_p]$$

And note that; $K = \frac{1}{2} \dot{q}^T D(q) \dot{q}$

By using MATLAB, D matrix (8 x 8) found as;

```
[                            m1 + m2,                                                  0,
[                                   0,                                         m1 + m2,
[                                   0,                                                0,
[ m2*(q8*cos(q5 + q7) + q6*cos(q5)), -m2*(q8*sin(q5 + q7) + q6*sin(q5) + 10),
[ m2*(q8*cos(q5 + q7) + q6*cos(q5)),      -m2*(q8*sin(q5 + q7) + q6*sin(q5)),
[                      m2*sin(q5),                                  m2*cos(q5),
[               m2*q8*cos(q5 + q7),                        -m2*q8*sin(q5 + q7),
[                  m2*sin(q5 + q7),                            m2*cos(q5 + q7),
```

1st and 2nd columns of D matrix

```
      0,                                                                m2*(q8*cos(q5 + q7) + q6*cos(q5)),
      0,                                                              -m2*(q8*sin(q5 + q7) + q6*sin(q5) + 10),
m1 + m2,                                                                                                  0,
      0, m2*q6^2 + 2*m2*cos(q7)*q6*q8 + 20*m2*sin(q5)*q6 + m2*q8^2 + 20*m2*sin(q5 + q7)*q8 + I2 + 100*m2,
      0,          m2*q6^2 + 2*m2*cos(q7)*q6*q8 + 10*m2*sin(q5)*q6 + m2*q8^2 + 10*m2*sin(q5 + q7)*q8 + I2,
      0,                                                                -m2*(10*cos(q5) + q8*sin(q7)),
      0,                                          I2 + m2*q8^2 + 10*m2*q8*sin(q5 + q7) + m2*q6*q8*cos(q7),
      0,                                                              m2*q6*sin(q7) - 10*m2*cos(q5 + q7),
```

3rd and 4th columns of D matrix

```
                                                                m2*(q8*cos(q5 + q7) + q6*cos(q5)),                  m2*sin(q5),
                                                                -m2*(q8*sin(q5 + q7) + q6*sin(q5)),                  m2*cos(q5),
                                                                                                 0,                           0,
m2*q6^2 + 2*m2*cos(q7)*q6*q8 + 10*m2*sin(q5)*q6 + m2*q8^2 + 10*m2*sin(q5 + q7)*q8 + I2, -m2*(10*cos(q5) + q8*sin(q7)),
                                                m2*q6^2 + 2*m2*cos(q7)*q6*q8 + m2*q8^2 + I2,              -m2*q8*sin(q7),
                                                                             -m2*q8*sin(q7),                          m2,
                                                          m2*q8^2 + m2*q6*cos(q7)*q8 + I2,              -m2*q8*sin(q7),
                                                                              m2*q6*sin(q7),                  m2*cos(q7),
```

5th and 6th columns of D matrix

```
                                                                m2*q8*cos(q5 + q7),                  m2*sin(q5 + q7)]
                                                                -m2*q8*sin(q5 + q7),                 m2*cos(q5 + q7)]
                                                                                  0,                                0]
I2 + m2*q8^2 + 10*m2*q8*sin(q5 + q7) + m2*q6*q8*cos(q7), m2*q6*sin(q7) - 10*m2*cos(q5 + q7)]
                                                m2*q8^2 + m2*q6*cos(q7)*q8 + I2,                  m2*q6*sin(q7)]
                                                                             -m2*q8*sin(q7),                 m2*cos(q7)]
                                                                               m2*q8^2 + I2,                         0]
                                                                                          0,                        m2]
```

7th and 8th columns of D matrix

Assumptions and conclusions: For the sake of simplicity I consider the creature consists of two independent mass, m1 is the mass of the body and m2 is the mass of the tongue. m1 is placed the geometric center of the body and m2 is placed to at the end of the tongue.

We can here, conclude that D(1,1) = D(2,2) = D(3,3) = m1 + m2 as expected, since first three joints are prismatic. And also D(8,8) should be m2, since it is the mass at the end of the tongue. Some of other terms also verified by hand, and I conclude that D matrix is satisfying.

Referring to main equation $D(q)\ddot{q}+\dot{q}^T C(q)\dot{q}+g(q)=\tau$ we have calculated D matrix, we need to find C and g in order to find generalized force. Here C refers to Coriolis and centrifugal force matrix, and finally g is the gravity loading force.

Benefiting from derivations of C and g on the book, I simply run a MATLAB code to calculate them. We can do some conclusions in this step also, let begin with g vector. It simply calculated as;

```
                    (981*m1)/100 + (981*m2)/100
                                              0
                                              0
(981*m2*(q8*cos(q5 + q7) + q6*cos(q5)))/100
(981*m2*(q8*cos(q5 + q7) + q6*cos(q5)))/100
                          (981*m2*sin(q5))/100
                 (981*m2*q8*cos(q5 + q7))/100
                   (981*m2*sin(q5 + q7))/100
```

We are expected to see terms related with m1 and m2 in the first element. Also, since second and third joints represent the motion in x and y axes it is expected to see 0 values for 2nd and 3rd elements. Let's look at the last term, since q5 and q7 represents revolute joints they are affecting the end effector, also think about when q5 and q7 are both equal to 0, then the potential will also be 0 since there is no distance on global z axis.

Now we can concentrate on C matrices. They are the most complicated matrices, but still we can do some conclusions. For example, since the centrifugal and Coriolis forces appear on nonlinear systems, we expect 0 matrices for first three C matrices since the first three joints are prismatic joints. Thanks to MATLAB I can verify it is correct.

Since we do not find Jacobians, another approach followed to developed dynamical model of the creature. After Jacobians founded, I am expecting the same results with this dynamical model. Then I can use either this model or Jacobian for simulations and animations.

5. CONCLUSIONS

In this report, DH parameters, forward kinematic model and dynamical model of the creature have obtained. Also, some verifications are done to understand if models are correct or not. This report will be a helpful guide for the further works on this project.

6. APPENDICES

MATLAB script for calculating corresponding matrices (the code is tested in MATLAB 2017b)

```
%Kinematic model of the system
clear; clc;
%define generalized coordinates as symbolic variables
syms q1 q2 q3 q4 q5 q6 q7 q8;
q = [q1 q2 q3 q4 q5 q6 q7 q8];

%a values
a1=0; a2=0; a3=0; a5=0; a6=0; a7=0; a8=0; a4=10;

%alpha values, in degrees, otherwise cos(-pi/2) gives errors in machine
%epsilon
alpha = [-90; -90; 0; 0; 90; -90; 90; 0];

%d values
d = [q1; q2; q3; 0; 0; q6; 0; q8];

%theta values

%non-symbolic theta values, in degrees as in the case of alpha
theta1 = 90; theta2 = -90; theta3 = 0; theta4 = 0; theta6 = 0; theta8 = 0;

%symbolic assignments for thetas

theta5 = q5;
theta7 = q7

% Body of the creature
A1_0 = [ cosd(theta1)    -cosd(alpha(1))*sind(theta1)
sind(alpha(1))*sind(theta1)      a1*cosd(theta1)
         sind(theta1)    cosd(alpha(1))*cosd(theta1)      -
sind(alpha(1))*cosd(theta1)     a1*sind(theta1)
         0               sind(alpha(1))                   cosd(alpha(1))
d(1)
         0               0                                0
1];

A2_1 = [ cosd(theta2)    -cosd(alpha(2))*sind(theta2)
sind(alpha(2))*sind(theta2)      a2*cosd(theta2)
         sind(theta2)    cosd(alpha(2))*cosd(theta2)      -
sind(alpha(2))*cosd(theta2)     a2*sind(theta2)
         0               sind(alpha(2))                   cosd(alpha(2))
d(2)
         0               0                                0
1];

A3_2 = [ cosd(theta3)    -cosd(alpha(3))*sind(theta3)
sind(alpha(3))*sind(theta3)     a3*cosd(theta3)
```

```
         sind(theta3)    cosd(alpha(3))*cosd(theta3)    -
sind(alpha(3))*cosd(theta3)    a3*sind(theta3)
         0               sind(alpha(3))                 cosd(alpha(3))
d(3)
         0               0                              0
1];

% Tongue
A4_3 = [ cosd(theta4)    -cosd(alpha(4))*sind(theta4)
sind(alpha(4))*sind(theta4)      a4*cosd(theta4)
         sind(theta4)    cosd(alpha(4))*cosd(theta4)      -
sind(alpha(4))*cosd(theta4)     a4*sind(theta4)
         0              sind(alpha(4))                 cosd(alpha(4))
d(4)
         0              0                              0
1];

A5_4 = [ cos(theta5)    -cosd(alpha(5))*sin(theta5)
sind(alpha(5))*sin(theta5)      a5*cos(theta5)
         sin(theta5)    cosd(alpha(5))*cos(theta5)      -
sind(alpha(5))*cos(theta5)     a5*sin(theta5)
         0              sind(alpha(5))                 cosd(alpha(5))
d(5)
         0              0                              0
1];

A6_5 = [ cosd(theta6)    -cosd(alpha(6))*sind(theta6)
sind(alpha(6))*sind(theta6)     a6*cosd(theta6)
         sind(theta6)    cosd(alpha(6))*cosd(theta6)    -
sind(alpha(6))*cosd(theta6)    a6*sind(theta6)
         0               sind(alpha(6))                 cosd(alpha(6))
d(6)
         0               0                              0
1];

A7_6 = [ cos(theta7)    -cosd(alpha(7))*sin(theta7)
sind(alpha(7))*sin(theta7)      a7*cos(theta7)
         sin(theta7)    cosd(alpha(7))*cos(theta7)      -
sind(alpha(7))*cos(theta7)     a7*sin(theta7)
         0              sind(alpha(7))                 cosd(alpha(7))
d(7)
         0              0                              0
1];

A8_7 = [ cosd(theta8)    -cosd(alpha(8))*sind(theta8)
sind(alpha(8))*sind(theta8)      a8*cosd(theta8)
         sind(theta8)    cosd(alpha(8))*cosd(theta8)      -
sind(alpha(8))*cosd(theta8)     a8*sind(theta8)
         0              sind(alpha(8))                 cosd(alpha(8))
d(8)
         0              0                              0
1];


A8_0 = simplify(A1_0*A2_1*A3_2*A4_3*A5_4*A6_5*A7_6*A8_7);

A(:,:,1) = A1_0;
```

```
A(:,:,2) = A2_1;
A(:,:,3) = A3_2;
A(:,:,4) = A4_3;
A(:,:,5) = A5_4;
A(:,:,6) = A6_5;
A(:,:,7) = A7_6;
A(:,:,8) = A8_7;

isPrismatic = [true; true; true; false; false; true; false; true];


U = sym(zeros(4,4,length(d),length(d)));
A_j_1_i = sym(zeros(4,4,length(d)+1,length(d)+1));

Qrev = [0 -1 0 0
        1 0  0 0
        0 0  0 0
        0 0  0 0];

Qpris =[0 0  0 0
        0 0  0 0
        0 0  0 1
        0 0  0 0];

for i = 1:(length(d)+1)
    A_j_1_i(:,:,i,i) = sym(eye(4,4));
end
%A_j_1_i(:,:,1,2) = A_j_1_i(:,:,1,1)*A(:,:,1);
%A_j_1_i(:,:,1,3) = A_j_1_i(:,:,1,2)*A(:,:,2);
%A_j_1_i(:,:,1,4) = A_j_1_i(:,:,1,3)*A(:,:,3);

%A_j_1_i(:,:,1,13) = A_j_1_i(:,:,1,12)*A(:,:,12);
for i=1:(length(d)+1)
    for j=(i+1):(length(d)+1)
        A_j_1_i(:,:,i,j) = (A_j_1_i(:,:,i,j-1)*A(:,:,j-1));
        %disp([i j]);
    end
end

%disp(A_0_12-A_j_1_i(:,:,1,13))
%disp(A(:,:,12)-A_j_1_i(:,:,12,13))
%disp(A(:,:,1)*A(:,:,2)-A_j_1_i(:,:,1,3))

for i=1:(length(d))
    for j=1:i
        if(isPrismatic(j))
            U(:,:,i,j) = A_j_1_i(:,:,1,j)*Qpris*A_j_1_i(:,:,j,i+1);
        else
            U(:,:,i,j) = A_j_1_i(:,:,1,j)*Qrev*A_j_1_i(:,:,j,i+1);
        end
    end
end

%disp(simplify(U(:,:,12,4)-diff(A_j_1_i(:,:,1,13),q4)))
%disp(simplify(U(:,:,10,5)-diff(A_j_1_i(:,:,1,11),q5)))

%for i=1:length(d)
```

```
%    disp(i)
%    disp(U(:,:,i,i)-diff(A_j_1_i(:,:,1,i+1),q(i)));
%end
%disp(U(:,:,3,1)-diff(A_j_1_i(:,:,1,4),q1))

syms m1 m2 I1 I2;
m = [0 0 m1 0 0 0 0 m2];
J = sym(zeros(4,4,length(d)));
J(:,:,3) = [I1/2 0 0 0;
            0 I1/2 0 0
            0 0 I1/2 0
            0 0 0 m1];
J(:,:,8) = [I2/2 0 0 0;
            0 I2/2 0 0
            0 0 I2/2 0
            0 0 0 m2];

D = sym(zeros(length(d),length(d)));
for i=1:length(d)
    for k=i:length(d)
        temp = sym(0);
        for j=max(i,k):length(d)
            temp=(temp+trace(U(:,:,j,k)*J(:,:,j)*U(:,:,j,i).'));
        end
        D(i,k) = temp;
        D(k,i) = D(i,k);
    end
end

D_simplified = simplify(D);


g = sym(zeros(length(d),1));
gravity = [0 0 -9.81 0];
for i=1:length(d)
    for j=1:length(d)
        g(i) = g(i)-m(j)*gravity*U(:,:,j,i)*[0 0 0 1]';
    end
end

g_simplified = simplify(g);



C = sym(zeros(length(d),length(d),length(d)));

for i=1:length(d)
    C(:,:,i) = jacobian(d_simplified(:,i),q)+jacobian(d_simplified(:,i),q).'-
diff(d_simplified,q(i));
end


C_simplified = simplify(C);
```

7. REFERENCES

1. Fu, K. S., González, R. C., Lee, C. S., & Freeman, H. (1987). Robotics: Control, sensing, vision and intelligence. New York: McGraw-Hill.
2. Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2012). *ROBOT MODELING AND CONTROL*. JOHN WILEY & Sons.
3. https://www.mathworks.com