Robotics HW6

Yunfan Gao

October 2020

Problem 1

Direct method: Use the exponential product method from hw5 to label the joint axes and link vectors. Calculate the rotation matrix such as $R_{0,1}$,...,$R_{n-1,n}$ so that the link vectors and joint axes can be represented in the world frame. Calculate the link vector from each joint to the end effector such as P_{0T},P_{1T},P_{nT} in zero frame. The Jacobian will be a 6 by n matrix, where the first three rows are the joint axes in zero frame, and the last three rows are the cross products between the joint axes in zero frame and the vector from current link to the end effector in zero frame.
Iterative method: create a 6 row Jacobian matrix with the number of column equal to the number of joint. Start with the first column, denote it as H_1, which will be $h_1; zeros(3 \times 1)$ for revolute joint or $zeros(3 \times 1); h_1$ for prismatic joint. For each iteration, obtain the rotation matrix and the link vector. The next iteration is to multiply the Jacobian with the phi matrix $\Phi_{2,1}$, where the $\Phi_{2,1}$ matrix needs a rotation matrix and the link vector. The rotation matrix is from the current joint to the previous joint in phi function, so the rotation obtained in every iteration needs to be transposed first and then pass to the Phi function, which will return: Φ = [R zeros $(3,3); -\mathrm{R}^*$ hat (p) R]. Then the newly obtained Jacobian matrix will have its second column (all zeros) to be replaced with the H_2 to form the Jacobian matrix denoting as JP_2. After n iterations for all joints, a Jacobian JP_n will be generated. This Jacobian will multiply with $\Phi_{T,n}$, which needs a identity matrix and $P_{n,T}$ for translation only, to get the end effector Jacobian JT, which is represented in the T frame. Convert the JT_T into the zero frame with the matrix $[R_{0T}$ zeros$(3,3)$; zeros$(3,3)$ $R_{0T}]$, and thus the Jacobian is obtained.
Recursive method: Same procedure in computing the rotation matrix and link vectors and Jp_i, with the same Φ function. Instead of using a loop, the function will call itself when it has not reach the terminated condition. Similar starting with the last method, find the rotation matrix and use the transpose of the matrix with as well as the link vector to pass in Phi function to get Phi matrix, and multiply it with the Jacobian, and then replace the next zero column with H_i, corresponding to its joint type. Then the function will call itself with the appended index to do the same process until it repeats n times, $i = n+1$ (i starts at 1), which will be the terminated condition. In the terminated condition, the

JP_n converts to the Jp_T using $\Phi_{T,n}$, and then represent it in the world frame to get the JT_0, same procedure as the procedure after the loop in iterative method.

Problem 2

From the previous homework:
ABBirb 120:
Joint axes:
$H = [h_1 \mid h_2 \mid h_3 \mid h_4 \mid h_5 \mid h_6] = [e_z \quad e_y \quad e_y \quad e_x \quad e_y \quad e_x]$
Link vectors:
$P = [P_{01} \mid P_{12} \mid P_{23} \mid P_{34} \mid P_{45} \mid P_{56} \mid P_{6T}] = [z_v \quad 290e_z \quad 270e_z \quad 302e_x + 70e_z \quad 0 \quad 0 \quad 72e_x]$
Standford:
Joint axes:
$H = [h_1 \mid h_2 \mid h_3 \mid h_4 \mid h_5 \mid h_6] = [e_z \quad e_x \quad e_y \quad e_y \quad e_z \quad e_y]$
Link vectors:
$P = [P_{01} \mid P_{12} \mid P_{23} \mid P_{34} \mid P_{45} \mid P_{56} \mid P_{6T}] = [z_v \quad l_0e_z \quad l_1e_x \quad l_3e_y \quad 0 \quad 0 \quad l_4e_y]$
b) For the ABBirb 120:

Listing 1: ABBirb 120 testing code

```
%This is for ABBirb 120
clear all;close all;
syms q1 q2 q3 q4 q5 q6 real;
ex=[1;0;0];ey=[0;1;0];ez=[0;0;1];zv=[0;0;0];
h1=ez;h2=ey;h3=ey;h4=ex;h5=ey;h6=ex;
p01=zv;p12=290*ez;p23=270*ez;
p34=302*ex+70*ez;p45=zv;p56=zv;p6T=72*ex;
type=[0 0 0 0 0 0]'; % 6R robot
n=6;
P=[p01 p12 p23 p34 p45 p56 p6T];
H=[h1 h2 h3 h4 h5 h6];
% direct computation
tstart1=tic;

for i=1:6;
    ii1=num2str(i-1);
    ii=num2str(i);
    if type(i)==0
        string=['R',ii1,ii,'=rot(h',ii,',q',ii,');'];
    else
        string=['R',ii1,ii,'=eye(3,3);'];
    end
    eval(string);
end
```

```
p6T_0=R01*R12*R23*R34*R45*R56*p6T;
p56_0=R01*R12*R23*R34*R45*p56;
p45_0=R01*R12*R23*R34*p45;
p34_0=R01*R12*R23*p34;
p23_0=R01*R12*p23;
p12_0=R01*p12;

p1T_0=p12_0+p23_0+p34_0+p45_0+p56_0+p6T_0;
p2T_0=p23_0+p34_0+p45_0+p56_0+p6T_0;
p3T_0=p34_0+p45_0+p56_0+p6T_0;
p4T_0=p45_0+p56_0+p6T_0;
p5T_0=p56_0+p6T_0;

h1_0=R01*h1;
h2_0=R01*R12*h2;
h3_0=R01*R12*R23*h3;
h4_0=R01*R12*R23*R34*h4;
h5_0=R01*R12*R23*R34*R45*h5;
h6_0=R01*R12*R23*R34*R45*R56*h6;

JT1=simplify([h1_0;cross(h1_0,p1T_0)]);
JT2=simplify([h2_0;cross(h2_0,p2T_0)]);
JT3=simplify([h3_0;cross(h3_0,p3T_0)]);
JT4=simplify([h4_0;cross(h4_0,p4T_0)]);
JT5=simplify([h5_0;cross(h5_0,p5T_0)]);
JT6=simplify([h6_0;cross(h6_0,p6T_0)]);

J1=simplify([JT1 JT2 JT3 JT4 JT5 JT6]);

telapse1=toc(tstart1)

% iterative method
tstart2=tic;
q=[q1;q2;q3;q4;q5;q6];
ABB120.H=sym(H(:,1:6));
ABB120.P=P;
ABB120.joint_type=type(1:6);
[T2,J2]=fwddiffkiniter(q,ABB120);
telapse2=toc(tstart2)
%disp('J2');disp(J2);

disp('J1-J2');disp(simplify(expand(J2)-J1));

% recursive method
tstart3=tic;
[T3,J3]=fwddiffkinrec(1,eye(4,4),zeros(6,6),q,ABB120);
```

```
telapse3=toc(tstart3)
disp('J1-J3');disp(simplify(expand(J3)-J1));
```

ABBirb 120 output result:

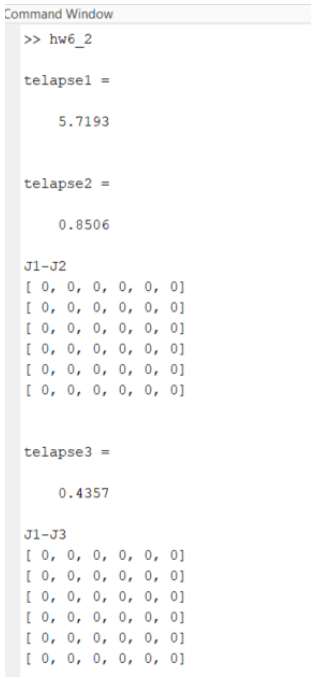

Figure 1: Running time and Jacobian difference under different methods for ABBirb 120

For the Standford:

Listing 2: Standford testing code

```
%This is for Standford
clear all;close all;
syms q1 q2 q3 q4 q5 q6 real;
syms l_0 l_1 l_3 l_4 positive
ex=[1;0;0];ey=[0;1;0];ez=[0;0;1];zv=[0;0;0];
h1=ez;h2=ex;h3=ey;h4=ey;h5=ez;h6=ey;
p01=zv;p12=l_0*ez;p23=l_1*ex;
p34=l_3*ey;p45=zv;p56=zv;p6T=l_4*ey;
type=[0 0 0 0 0 0]'; % 6R robot
n=6;
P=[p01 p12 p23 p34 p45 p56 p6T];
```

```
H=[h1 h2 h3 h4 h5 h6];
% direct computation
tstart1=tic;
for i=1:6;
    ii1=num2str(i-1);
    ii=num2str(i);
    if type(i)==0
        string=['R',ii1,ii,'=rot(h',ii,',q',ii,');'];
    else
        string=['R',ii1,ii,'=eye(3,3);'];
    end
    eval(string);
end
p6T_0=R01*R12*R23*R34*R45*R56*p6T;
p56_0=R01*R12*R23*R34*R45*p56;
p45_0=R01*R12*R23*R34*p45;
p34_0=R01*R12*R23*p34;
p23_0=R01*R12*p23;
p12_0=R01*p12;

p1T_0=p12_0+p23_0+p34_0+p45_0+p56_0+p6T_0;
p2T_0=p23_0+p34_0+p45_0+p56_0+p6T_0;
p3T_0=p34_0+p45_0+p56_0+p6T_0;
p4T_0=p45_0+p56_0+p6T_0;
p5T_0=p56_0+p6T_0;

h1_0=R01*h1;
h2_0=R01*R12*h2;
h3_0=R01*R12*R23*h3;
h4_0=R01*R12*R23*R34*h4;
h5_0=R01*R12*R23*R34*R45*h5;
h6_0=R01*R12*R23*R34*R45*R56*h6;

JT1=simplify([h1_0;cross(h1_0,p1T_0)]);
JT2=simplify([h2_0;cross(h2_0,p2T_0)]);
JT3=simplify([h3_0;cross(h3_0,p3T_0)]);
JT4=simplify([h4_0;cross(h4_0,p4T_0)]);
JT5=simplify([h5_0;cross(h5_0,p5T_0)]);
JT6=simplify([h6_0;cross(h6_0,p6T_0)]);

J1=simplify([JT1 JT2 JT3 JT4 JT5 JT6]);

telapse1=toc(tstart1)

% iterative method
tstart2=tic;
```

```
q=[q1;q2;q3;q4;q5;q6];
ABB120.H=sym(H(:,1:6));
ABB120.P=P;
ABB120.joint_type=type(1:6);
[T2,J2]=fwddiffkiniter(q,ABB120);
telapse2=toc(tstart2)
%disp('J2');disp(J2);

disp('J1-J2');disp(simplify(expand(J2)-J1));

% recursive method
tstart3=tic;
[T3,J3]=fwddiffkinrec(1,eye(4,4),zeros(6,6),q,ABB120);
telapse3=toc(tstart3)
disp('J1-J3');disp(simplify(expand(J3)-J1));
```

Standford output result:

```
>> hw6_2standford

telapse1 =

    6.1160


telapse2 =

    1.0648

J1-J2
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]


telapse3 =

    0.6934

J1-J3
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0]
```

Figure 2: Running time and Jacobian difference under different methods for Standford

Problem 3

Since it has a spherical joint at O_4
h_2 and h_3 are the same

$$J_T = \begin{bmatrix} \vec{h}_1 & \vec{h}_2 & \vec{h}_2 & \vec{h}_4 & \vec{h}_5 & \vec{h}_6 \\ \vec{h}_1 \times \vec{p}_{1T} & \vec{h}_2 \times \vec{p}_{2T} & \vec{h}_2 \times \vec{p}_{3T} & \vec{h}_4 \times \vec{p}_{3T} & \vec{h}_5 \times \vec{p}_{4T} & \vec{h}_6 \times \vec{p}_{4T} \end{bmatrix}$$

$$J_4 = \Phi_{4T} J_T \quad \Phi_{4T} = \begin{bmatrix} \mathcal{I} & 0 \\ -\vec{p}_{T4} & \mathcal{I} \end{bmatrix}$$

$$J_4 = \begin{bmatrix} \vec{h}_1 & \vec{h}_2 & \vec{h}_2 & \vec{h}_4 & \vec{h}_5 & \vec{h}_6 \\ \vec{h}_1 \times \vec{p}_{14} & \vec{h}_2 \times \vec{p}_{24} & \vec{h}_2 \times \vec{p}_{34} & 0 & 0 & 0 \end{bmatrix}$$

To make the Jacobian singular,
either $[\vec{h}_4 \quad \vec{h}_5 \quad \vec{h}_6]$ or $[\vec{h}_1 \times \vec{p}_{14} \quad \vec{h}_2 \times \vec{p}_{24} \quad \vec{h}_2 \times \vec{p}_{34}]$ loose rank
Geometric condition: $\vec{h}_4 = \pm \vec{h}_6$

$$[\vec{h}_4 \quad \vec{h}_5 \quad \vec{h}_6]_5 = [R_{45}^T h_4 \quad h_5 \quad R_{56} h_6] = [R_y(-q_5)e_x \quad e_y \quad R_x(q_6)e_x] = \begin{bmatrix} c_5 & 0 & 1 \\ 0 & 1 & 0 \\ s_5 & 0 & 0 \end{bmatrix}$$

Algebraic Condition: $s_5 = 0$, then $q_5 = 0$ or π

This is called wrist singularity, at this singularity: self-motion = span$\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \mp 1 \end{bmatrix} \}$

Unreachable subspace: span $\{\vec{h}_4 \times \vec{h}_5\}$
Now for the $[\vec{h}_1 \times \vec{p}_{14} \quad \vec{h}_2 \times \vec{p}_{24} \quad \vec{h}_2 \times \vec{p}_{34}]$ to loose rank
we can have geometric condition: $\vec{p}_{14} \| \vec{h}_1$, this is called interior singularity.
$(P_{14})_1 = P_{12} + R_{12}P_{23} + R_{12}R_{23}P_{34}$
$= 290e_z + R_y(q_2)270e_z + R_y(q_2)R_y(q_3)(70e_z + 302e_x)$
Since $h_1 = e_z$, the algebraic condition:
$e_z^\times (P_{14})_1 = e_z^\times (290e_z + R_y(q_2)270e_z + R_y(q_2)R_y(q_3)(70e_z + 302e_x) = 0$
This is equivalent to

$$\begin{bmatrix} 0 \\ 270\sin q_2 + 302\cos(q_2)\cos(q_3) + 70\cos(q_2)\sin(q_3) + 70\cos(q_3)\sin(q_2) - 302\sin(q_2)\sin(q_3) \\ 0 \end{bmatrix}$$

$= 0$
In this case, O_4 can only reach to the span of $\vec{h}_2 \times \vec{p}_{24}$ and $\vec{h}_2 \times \vec{p}_{34}$, thus the unreachable subspace will be span$\{\vec{h}_2\}$. The self-motion = span$\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \}$

Another case for the $[\vec{h}_1 \times \vec{p}_{14} \quad \vec{h}_2 \times \vec{p}_{24} \quad \vec{h}_2 \times \vec{p}_{34}]$ to loose rank:
Geometric condition: $\vec{p}_{23} \| \vec{p}_{34}$, this is called boundary singularity.

In this case: $(p_{23})_2 = (p_{34})_2$

$(p_{23})_2 = 270e_z$; $R_{23}P_{34} = R_y(q_3)(70e_z + 302e_x) = \begin{bmatrix} c_3 & 0 & s_3 \\ 0 & 1 & 0 \\ -s_3 & 0 & c_3 \end{bmatrix} \begin{bmatrix} 302 \\ 0 \\ 70 \end{bmatrix}$

$302C_3 + 70S_3 = 0$, thus the algebraic condition: $q_3 = tan^{-1}(\frac{-302}{70})$ In this case, the unreachable subspace is span$\{\vec{p}_{23}\}$, and the self-motion will be the span$\{\begin{bmatrix} 0 \\ \pm\sqrt{70^2 + 302^2} \\ 270 \\ 0 \\ 0 \\ 0 \end{bmatrix}\}$

The plot of wrist singularity from Matlab, here q_5 is chosen to be π with the rest of q chosen to be zero:

```
>> abb = loadrobot('ABBirb120');
c = setJoint(abb,[0;0;0;0;pi;0]);
show(abb,c)
```

Figure 3: Set q5 to satisfy wrist singularity

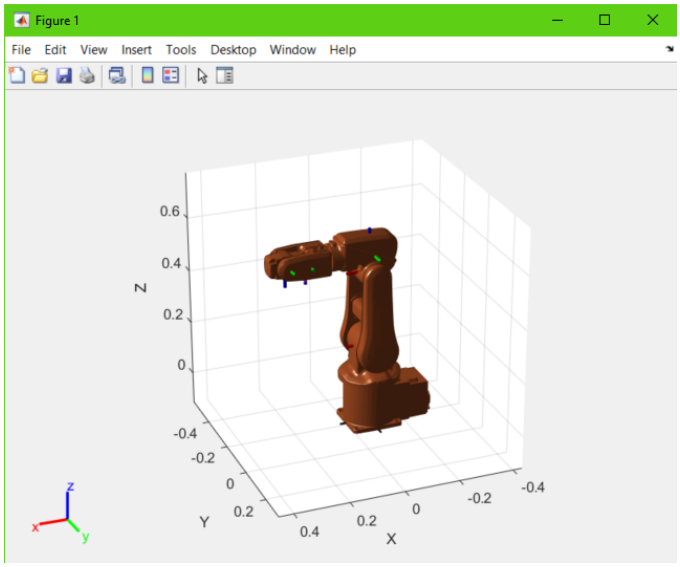

Figure 4: wrist singularity

For the plot of interior singularity, since there are many q_2 and q_3 capable of satisfying the algebraic condition:

$$270\sin q_2 + 302\cos(q_2)\cos(q_3) + 70\cos(q_2)\sin(q_3) + 70\cos(q_3)\sin(q_2) - 302\sin(q_2)\sin(q_3) = 0$$

$q_3 = \frac{\pi}{2}$ is chosen, and thus q_2 can be calculated to be: $q_2 = \tan^{-1}(\frac{70}{32})$. The rest is chosen to be random.

```
>> abb = loadrobot('ABBirb120'); q = rand(6,1)*2*pi;
q(2)= atan(70/32);q(3) = pi/2
c = setJoint(abb,q);show(abb,c);

q =

    1.2562
    1.1420
    1.5708
    4.7230
    2.3144
    5.9176
```

Figure 5: Set q2 and q3 to satisfy interior singularity

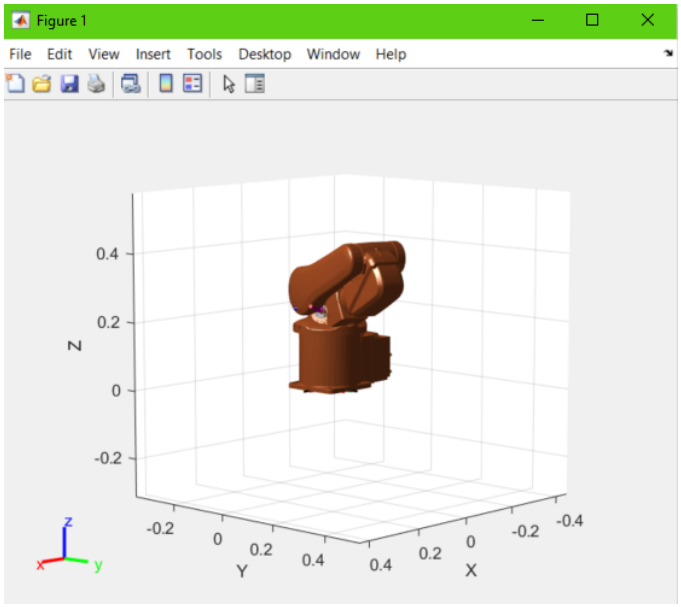

Figure 6: interior singularity

Last, for the boundary singularity, choose $q_3 = \tan^{-1}(\frac{-302}{70})$, and chosen the rest of q to be zero:

```
>> abb = loadrobot('ABBirb120'); q = zeros(6,1);
q(3) = atan(-302/70)
c = setJoint(abb,q);show(abb,c);

q =

         0
         0
   -1.3430
         0
         0
         0
```

Figure 7: Set q3 to satisfy boundary singularity

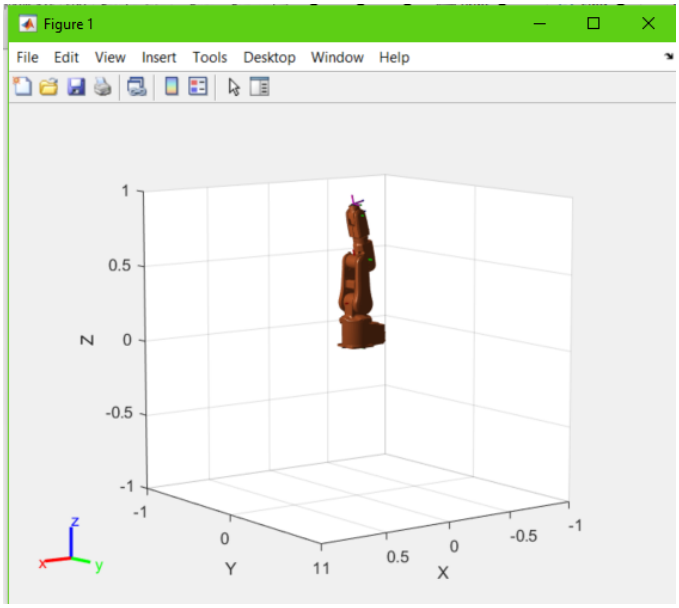

Figure 8: boundary singularity