EENG 428 Laboratory --- Lab Session 6

Description:

In this session, modelling fundamental matrices representing differential motion for robotic manipulators using Matlab is to be introduced.

Prerequisites:

Attending students are expected to know:

- Different forms of transformation matrices (Rotation about / Translation along an axis) and their Matlab implementations using the robotic toolbox.
- D-H convention and the modelling D-H parameters with their matrix representation on Matlab.
- The basics of differential motion.

Contents:

1- Fundamental matrices representing differential motion.

2- The Jacobian of serial manipulators w.r.t the base frame (Vector Cross Product).

3- The Jacobian of serial manipulators w.r.t the hand frame (Paul's method).

4- The relation between the Jacobian w.r.t the base and the Jacobian w.r.t the hand.

1- Introduction:

The Jacobian or Jacobian matrix is one of the most important quantities in the analysis and control of robot motion. It arises in virtually every aspect of robotic manipulation: in the planning and execution of smooth trajectories, in the determination of singular configurations, in the execution of coordinated anthropomorphic motion, in the derivation of the dynamic equations of motion, and in the transformation of forces and torques from the end-effector to the manipulator joints [1].

In this session, deriving the Jacobian is our aim. This Jacobian matrix can be derived w.r.t any coordinate frame for the same manipulator, e.g. w.r.t the base frame, w.r.t the end-effector frame or w.r.t any median coordinate frame (any link in between).

However, these Jacobians can be related to each other with a single transformation matrix, we can think about the Jacobian as a phenomena that travels along the links and changes some of its characteristics <u>linearly</u> from one link to another.

2- Fundamental matrices representing differential motion:

- Two Important mathematical notation:

1- All transformation matrices involves the calculation of sinusoids of an angle or more. In the case of differential motion, the following notation is usually adopted:

$$\cos(dt) \sim 1 \qquad \sin(dt) \sim dt$$

we can see this result mathematically by calculating the sin and cos of a very small angle as follows:

```
>> cos(0.001)
ans =
   1.0000
>> sin(0.001)
ans =
  1.0000e-03
```

Consider a transformation matrix represents a rotation about the x-axis:

```
>> syms tx
>> trotx(tx)
ans =
[ 1,       0,        0,      0]
[ 0, cos(tx), -sin(tx), 0]
[ 0, sin(tx),  cos(tx),  0]
[ 0,       0,        0,       1]

>> trotx(0.0011)
ans =
   1.0000         0         0         0
        0    1.0000   -0.0011     0
        0    0.0011    1.0000     0
        0         0         0            1.0000
```

2- Sequential rotations are not commutative in general, but they are

commutative in differential motion:

The multiplication of two differential variables is negligible.

```
>> sin(0.001)*sin(0.001)
ans =
1.0000e-06
```

Which is 1000 times less than the original differential motion.

Consider the following:

```
>> syms tx tz
>> trotz(tz)*trotx(tx)
ans =
[ cos(tz), -cos(tx)*sin(tz),  sin(tx)*sin(tz), 0]
[ sin(tz),  cos(tx)*cos(tz), -cos(tz)*sin(tx), 0]
[       0,          sin(tx),          cos(tx), 0]
[       0,                0,                0, 1]
>> trotx(tx)*trotz(tz)
ans =
[         cos(tz),        -sin(tz),        0, 0]
[ cos(tx)*sin(tz), cos(tx)*cos(tz), -sin(tx), 0]
[ sin(tx)*sin(tz), cos(tz)*sin(tx),  cos(tx), 0]
[               0,               0,        0, 1]
```

While if we rotate with differential angles we get:

```
>> trotx(0.0011)*trotz(0.0015)
ans =
    1.0000   -0.0015         0         0
    0.0015    1.0000   -0.0011         0
    0.0000    0.0011    1.0000         0
         0         0         0    1.0000
>> trotz(0.0015)*trotx(0.0011)
ans =
    1.0000   -0.0015    0.0000         0
    0.0015    1.0000   -0.0011         0
         0    0.0011    1.0000         0
         0         0         0    1.0000
```

3- The Calculation of the Jacobian Matrix:

Finding the Jacobian of a manipulator is not easy without a systematic approach in general.

There are several ways to derive the Jacobian of serial manipulators, the most familiar ones are:

- Paul's Method (deriving the Jacobian w.r.t the base hand coordinate frame)
- Vector cross product method (deriving the Jacobian w.r.t the tool coordinate frame)
- Screw based Jacobian (deriving the Jacobian for any intermediate coordinate frame) [2].

The first two methods are covered in the scope of this course.

In [1], very important basics regarding the derivation of the Jacobian are presented (see the additional notes on the website).

3.1 Paul's Method:

This method is used to derive the Jacobian of a serial manipulator w.r.t the tool coordinate frame. The algorithm and a Matlab implementation are shown in the following:

Remember that the Jacobian has 6 rows, and n columns; where n is the DOF of the manipulator.

The calculation of the i^{th} column of the Jacobian is derived from ${}^{i-1}T_n$

For i between 1 and n do the following:

Evaluate ${}^{i-1}T_n$, say it is equal to:

$$ {}^{i-1}T_n = \begin{bmatrix} nx & ox & ax & px \\ ny & oy & ay & py \\ nz & oz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Then according the nature of the joint i, we evaluate J^i as follows:

If i is revolute:

$$ J^i = \begin{bmatrix} -nx.py + ny.px \\ -ox.py + oy.px \\ -ax.py + ay * px \\ nz \\ oz \\ az \end{bmatrix} $$

If i is prismatic:

$$ J^i = \begin{bmatrix} nz \\ oz \\ az \\ 0 \\ 0 \\ 0 \end{bmatrix} $$

Example: (Paul's Method)

Given the following DH parameters for a 4DOF, RRPR manipulator.

α	a	d	t
0	a1	d1	t1
180	a2	0	t2
0	0	d3	0
0	0	d4	t4

```
%% Jacobian wrt hand frame (Paul's method)
syms t d al a
syms a1 d1 t1 a2 t2 d3 d4 t4
A=trotz(t)*transl(a,0,d)*trotx(al);
A1=subs(A,[al a d t],[0 a1 d1 t1]);
A2=subs(A,[al a d t],[pi a2 0 t2]);
A3=subs(A,[al a d t],[0 0 d3 0]);
A4=subs(A,[al a d t],[0 0 d4 t4]);
A34=A3*A4;
A234=A2*A34;
A1234=A1*A234;
J1=[A1234(1,4)*A1234(2,1)-A1234(2,4)*A1234(1,1)
    A1234(1,4)*A1234(2,2)-A1234(2,4)*A1234(1,2)
    A1234(1,4)*A1234(2,3)-A1234(2,4)*A1234(1,3)
    A1234(3,1)
    A1234(3,2)
    A1234(3,3)];
J1=simplify(J1);
J2=[A234(1,4)*A234(2,1)-A234(2,4)*A234(1,1)
    A234(1,4)*A234(2,2)-A234(2,4)*A234(1,2)
    A234(1,4)*A234(2,3)-A234(2,4)*A234(1,3)
    A234(3,1)
```

```
    A234(3,2)
    A234(3,3)];
J2=simplify(J2);
J3=[A34(3,1)
    A34(3,2)
    A34(3,3)
    0
    0
    0];
J3=simplify(J3);
J4=[A4(1,4)*A4(2,1)-A4(2,4)*A4(1,1)
    A4(1,4)*A4(2,2)-A4(2,4)*A4(1,2)
    A4(1,4)*A4(2,3)-A4(2,4)*A4(1,3)
    A4(3,1)
    A4(3,2)
    A4(3,3)];
J4=simplify(J4);
Jh=[J1 J2 J3 J4];
```

3.2 Vector Cross Product Method:

This method is used to derive the Jacobian of a serial manipulator w.r.t the base coordinate frame. The algorithm and a Matlab implementation are shown in the following:

Remember that the Jacobian has 6 rows, and n columns; where n is the DOF of the manipulator.

The calculation of the i^{th} column of the Jacobian is derived from ${}^{0}T_{i-1}$

1- Evaluate ${}^{0}T_n$ and find p_n

2- Write $a_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ and $P_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

3- For i between 1 and n do the following:

Evaluate ${}^{0}T_{i-1}$, say it is equal to:

$${}^{0}T_{i-1} = \begin{bmatrix} nx & ox & ax & px \\ ny & oy & ay & py \\ nz & oz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix}$$, and $R = [n \quad o \quad a]$, $P = \begin{bmatrix} px \\ py \\ pz \end{bmatrix}$

Then according the nature of the joint i, we evaluate J^i as follows:

If i is revolute:

$$J^i = \begin{bmatrix} a \times (Pn - P) \\ a \end{bmatrix}$$

If i is prismatic:

$$J^i = \begin{bmatrix} a \\ 0 \end{bmatrix}$$

Example: (Vector Cross Product Method)

Given the following DH parameters for a 4DOF, RRPR manipulator.

α	a	d	t
0	a1	d1	t1
180	a2	0	t2
0	0	d3	0
0	0	d4	t4

```
%% Matrix Representation of the Frames according to DH
convention:
syms t d al a
syms a1 d1 t1 a2 t2 d3 d4 t4
A=trotz(t)*transl(a,0,d)*trotx(al);
A1=subs(A,[al a d t],[0 a1 d1 t1]);
A2=subs(A,[al a d t],[pi a2 0 t2]);
A3=subs(A,[al a d t],[0 0 d3 0]);
A4=subs(A,[al a d t],[0 0 d4 t4]);
%% Jacobian wrt base frame (vector cross product
method)
a0=[0 0 1]';
p0=[0 0 0]';
A1;
A12=A1*A2;
A123=A1*A2*A3;
A1234=A123*A4;
p4=A1234(1:3,4);
J1=[cross(a0,p4-p0);a0];
J1=simplify(J1);
a1=A1(1:3,3);
p1=A1(1:3,4);
```

```
J2=[cross(a1,p4-p1);a1];
J2=simplify(J2);
z2=A12(1:3,3);
J3=[z2;[0;0;0]];
J3=simplify(J3);
p3=A123(1:3,4);
a3=A123(1:3,3);
J4=[cross(a3,p4-p3);a3];
J4=simplify(J4);
J0=[J1 J2 J3 J4];
```

3.3 Converting the Jacobian from Hand frame to Tool Frame and vice versa:

The conversion is given by the equation:

${}^{0}J = M * {}^{H}J \qquad {}^{H}J=M^{-1} * {}^{0}J$

Where M is defined as:

$M=\begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix}$, and R is the orientation matrix of ${}^{0}T_{n}$

For the previous example:

```
M=[A1234(1:3,1:3) zeros(3)
    zeros(3) A1234(1:3,1:3)];
M=simplify(M);
disp('Jacobian w.r.t hand Pauls Method')
Jh
disp('Jacobian w.r.t base VCP Method')
J0
disp('Jacobian w.r.t base using the conversion from
hand to base')
J0Fromconversion=M*Jh;
J0Fromconversion=simplify(J0Fromconversion)
disp('Jacobian w.r.t hand using the conversion from
base to hand')
JhFromconversion=inv(M)*J0;
JhFromconversion=simplify(JhFromconversion)
```

Homework:

Spatial robots with 6 DOF, are known as general purpose manipulators.

These manipulators are very commonly used in the industry.

1- Find a 6 DOF manipulator (of your choice) and draw it by hand (try to get a unique manipulator to avoid cheating).
2- Find the DH parameters of your manipulator.
3- Adapt the codes presented in the lab sheet to find the Jacobian w.r.t hand.
4- Adapt the codes presented in the lab sheet to find the Jacobian w.r.t base.
5- Adapt the codes presented in the lab sheet to show that the conversion between the Jacobians hold

   (Ready functions to find the Jacobian will be graded as zero if presented alone, they might be optionally used for verification).

Only Email submissions are accepted (you have exactly 6 days to submit)

Paper solutions are not accepted under any circumstances

- Take clear photos of your paper solutions and combine them all in a single pdf file.
- Send your Matlab codes in a separate file (m-file or word document).

  Submit to the Email: lab.eeng428@gmail.com

- In case of emergency, contact me on the Email: (Don’t visit me in the office before writing an Email explaining your problem)

Mohamad.Harastani@emu.edu.tr

4- References:

[1] Craig, J. J. (2005). *Introduction to robotics: mechanics and control* (Vol. 3, pp. 48-70). Upper Saddle River: Pearson Prentice Hall.

[2] Tsai, L. W. (1999). *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons.