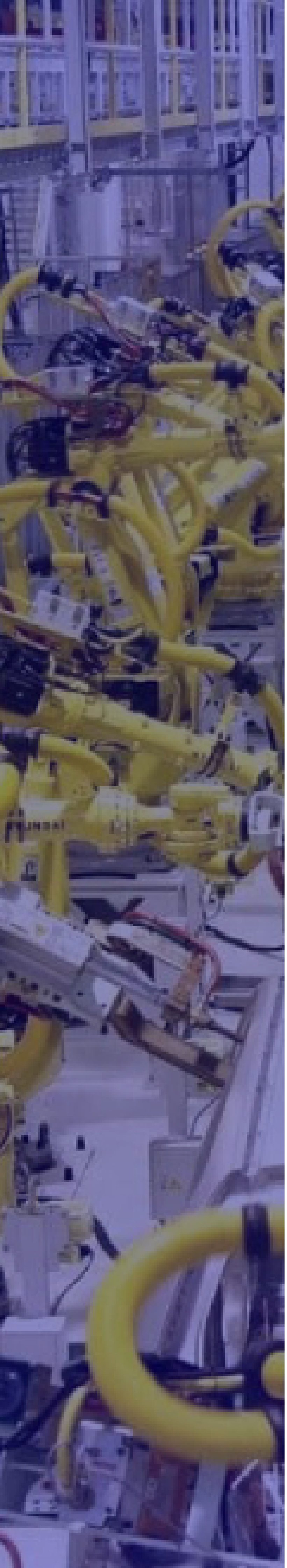


Dep. of Mechanical Eng.

Robotics

Dr. Saeed Behzadipour

Homework 4

Yashar Zafari
99106209

November 23, 2023

Robotics

Homework 4

Yashar Zafari

99106209

Questions

Question 1

In MATLAB, create a function that performs direct kinematics (H) for a spatial robot. It should take as input the Denavit-Hartenberg table and joint variable vector ($\vec{Q}$) symbolically, and return the translational ($\mathbf{J}_v$) and rotational ($\mathbf{J}_\omega$) Jacobian matrices symbolically. Using this function, generate and report the Jacobians for the robots in Homeworks 2 and 3, along with a report of the solution process in MATLAB code.

(In this case the second revolute joint is at zero position, $L_1 = 31cm,\ L_2 = 30cm$.)

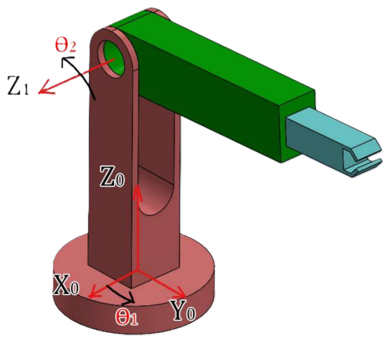


Figure 1: Isometric view of the robot.

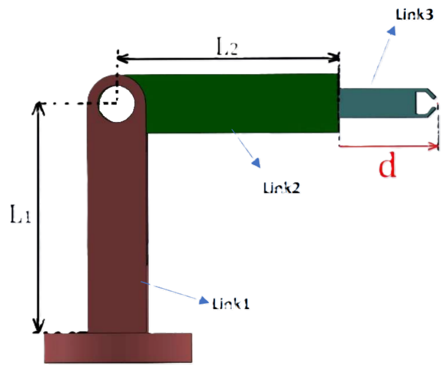
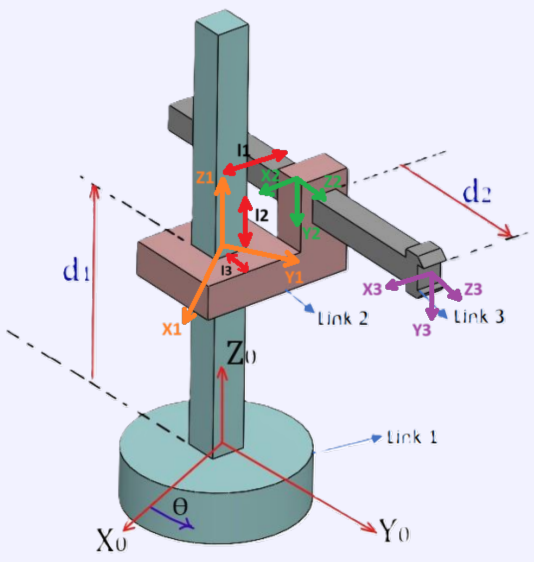


Figure 2: Front view of the robot.

Solution

First I begin by placing the link frames according to the Denavit-Hartenberg convention for RPP robot (Homework 2) and RRP robot (Homework 3), as shown in the figures.
Following the DH convention, $Frame_0$ or respectively our World Frame is contacted with $Frame_1$ through a revolute joint along z-axis and a prismatic joint with the displacement of d_1 along z-axis. Next, $Frame_2$ is placed in between $Link_2$ and $Link_3$ along the z-axis performing as prismatic joint which its z-axis is along the prismatic joint path and $Link_3$ and also its x-axis is placed on the common perpendicular of z_1 and z_2.

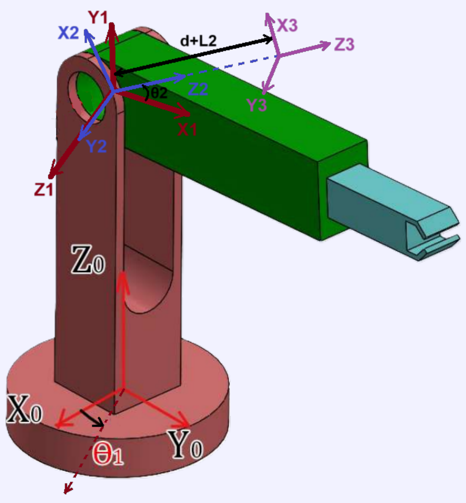


Note that following the DH convention forced us to place $Frame_2$ $L_3 = 30mm$ behind the start point of prismatic joint and d_2, thus the end-effector frame, $Frame_3$, is placed in a way that its z-axis is along the z-axis of $Frame_2$ and its x-axis is parallel with the x-axis of $Frame_2$, which apparently represents the displacement of end-effector via prismatic joint. As said before, this displacement is actually $d_2 + 30$ due to frames' placement based on DH convention. Thus the variable vector $\vec{Q}$ and DH table for the RPP robot is as below:

$$\vec{Q}_{RPP} = \begin{bmatrix} \theta \\ d_1 \\ d_2 \end{bmatrix}, \quad \text{DH}_{RPP} =$$

	θ	d	α	l
1	$\underline{\theta}$	$\underline{d_1}$	0	0
2	0	l_2	$-90°$	$-l_1$
3	0	$\underline{d_2}+l_3$	0	0

Now for the RRP robot, $Frame_0$ is referred as World Frame. Hence, $Frame_1$ is defined in a manner that its z-axis is placed along the axis of the revolute joint related to θ_2 and its y-axis is along z-axis of $Frame_0$ performing the revolute joint rotation θ_1. Afterwards, $Frame_2$ is placed in the revolute joint, its y-axis being along z-axis of $Frame_1$ and its z-axis being along the $Link_2$ which perform the revolute joint rotation θ_2 between x-axis of $Frame_1$ and z-axis of $Frame_2$.



Finally, $Frame_3$, the end-effector frame, is placed in the middle of the gripper and its z-axis is along the z-axis of $Frame_2$ and the other axes are parallel with the respective ones in $Frame_2$. Thus the variable vector $\vec{Q}$ and DH table for the RRP robot is as below:

$$\vec{Q}_{RRP} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ d \end{bmatrix}, \quad \text{DH}_{RRP} =$$

	θ	d	α	l
1	$\underline{\theta_1}+90°$	l_1	$90°$	0
2	$\underline{\theta_2}+90°$	0	$90°$	0
3	0	$\underline{d}+l_2$	0	0

Now I calculate the $\mathbf{J}_v$ and $\mathbf{J}_\omega$ using the vector method which can be summarized as below:

$$\mathbf{J}_v = \begin{bmatrix} \vec{C}_1 & \vec{C}_2 & \cdots & \vec{C}_n \end{bmatrix}, \vec{C}_i = \begin{cases} R^0_{i-1} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T & \text{if joint } i \text{ is revolute.} \\ (R^0_{i-1} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T) \times (\vec{O}^0_n - \vec{O}^0_{i-1}) & \text{if joint } i \text{ is prismatic.} \end{cases}$$

$$\mathbf{J}_\omega = \begin{bmatrix} \alpha_1 \hat{k} & \alpha_2 R^0_1 \hat{k} & \cdots & \alpha_n R^0_{n-1} \hat{k} \end{bmatrix}, \hat{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \alpha_i = \begin{cases} 1 \text{ if joint } i \text{ is revolute.} \\ 0 \text{ if joint } i \text{ is prismatic.} \end{cases}$$

As it's apparent, $\vec{O}^0_n$ is calculated from H^0_n's last column and first three rows, and respectively $\vec{O}^0_{i-1}$ from H^0_{i-1}. Also notice that $H^0_i = \prod_{i=0}^{i-1} H^i_{i+1}$ in which every H^i_{i+1} is calculated by the homogeneous transformation defined in the i-th row of DH table. Now the following MATLAB function return the $\mathbf{J}_v$ and $\mathbf{J}_\omega$ for a given variable vector $\vec{Q}$ and Denavit-Hartenberg table:

```
function [Jv, Jw] = DH2Jacob(DH, Q)
len = size(DH, 1);
Hh = sym('Hh', [4 4 len+1 1]);
DH=[0 0 0 0; DH];
syms thetaz dz alphx lx real
S=[0 -alphx dz; alphx 0 -thetaz; -dz thetaz 0];
H=Rot('z',thetaz)*Trans('z',dz)*Rot('x',alphx)*Trans('x',lx);
rev=[1 0 1 0];
tmp=sym(eye(4));
a=has(DH,Q);
for i = 2:len+1
    Hh(:,:,i-1)=tmp*subs(H,[thetaz dz alphx lx],DH(i-1,:));
    tmp=Hh(:,:,i-1);
    Jw(:,i-1)=simplify(any(a(i,:)&rev)*tmp(1:3,1:3)*[0;0;1]);
    if i==len+1
        Hh(:,:,i)=tmp*subs(H,[thetaz dz alphx lx],DH(i,:));
    end
end
for i = 2:len+1
    if ~any(a(i,:)&rev)
        Jv(:,i-1)=simplify(Hh(1:3,1:3,i-1)*[0;0;1]);
    else
        Jv(:,i-1)=simplify(subs(S,[thetaz dz alphx],(Hh
           (1:3,1:3,i-1)*[0;0;1])')*(Hh(1:3,end,4)-Hh(1:3,end,
           i-1)));
    end
end
end
```

First by calculating the length of DH table, I found the number of the links that robot has. Then I define a 3 dimensional symbolic matrix "Hh" in which i-th page represents H^0_{i-1}. Also for easier and faster calculation, I added another page at the start of the matrix, which for $i = 0$ returns $H^0_0 = I$ and a zero row at the start of DH table. Hence I defined matrix "S" that performs the cross multiplication between two vectors. Later function "H" is defined as the row operator in the DH table that return the H matrix for the respective link. Here is the tricky part. In order to find the Jacobians through the vector method, I need to determine that which joint is revolute joint or prismatic joint. Thus, I first use the variable vector $\vec{Q}$ to find where the variables are located in the DH table and return 1 if one exists in a cell of table and return 0 otherwise. Now if joint i is revolute, I will have at least one "1" in the i-th row of the DH table. Now by defining a pattern of 1s and 0s representing revolute joint, call it "rev" and comparing it with every row of the matrix representing the locations of variables in DH table, I can find that whether a joint is revolute or prismatic.The rest of function is just simply the implementation of vector method definition of $\mathbf{J}_v$ and $\mathbf{J}_\omega$.

Now with the DH tables and variable vectors defined for the RPP and RRP robots, I get the following Jacobians:

RPP:

$$\mathbf{J}_v = \begin{bmatrix} l_1\sin(\theta) - (d_2 + l_3)\cos(\theta) & 0 & -\sin(\theta) \\ -l_1\cos(\theta) - (d_2 + l_3)\sin(\theta) & 0 & \cos(\theta) \\ 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{J}_\omega = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

RRP:

$$\mathbf{J}_v = \begin{bmatrix} -(d + l_2)\cos(\theta_1)\cos(\theta_2) & (d + l_2)\sin(\theta_1)\sin(\theta_2) & -\cos(\theta_2)\sin(\theta_1) \\ -(d + l_2)\cos(\theta_2)\sin(\theta_1) & -(d + l_2)\cos(\theta_1)\sin(\theta_2) & \cos(\theta_1)\cos(\theta_2) \\ 0 & (d + l_2)\cos(\theta_2) & -\cos(\theta_2 + 90^\circ) \end{bmatrix}$$

$$\mathbf{J}_\omega = \begin{bmatrix} 0 & \sin(\theta_1 + 90^\circ) & 0 \\ 0 & -\cos(\theta_1 + 90^\circ) & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Question 2

Find and report any singular points, if they exist, for the robots in Homeworks 2 and 3, along with your results. (10% credits)

Solution

Now simply by the configuration of the robots, it's obvious that these are designed for the purpose of the positioning rather than orientation. Thus considiering the rotational velocities, robots are already singular and the equation $det(\mathbf{J}_\omega) = 0$ already suffices. Considering the singularities caused by positioning and translational velocities, I check the following equation for each of the robots:

RPP:

$$\begin{aligned} det(\mathbf{J}_v) = 0 \rightarrow \\ \rightarrow \begin{vmatrix} l_1\sin(\theta) - (d_2 + l_3)\cos(\theta) & 0 & -\sin(\theta) \\ -l_1\cos(\theta) - (d_2 + l_3)\sin(\theta) & 0 & \cos(\theta) \\ 0 & 1 & 0 \end{vmatrix} = 0 \rightarrow \\ \rightarrow d_2 + l_3 = 0 \rightarrow \\ \rightarrow d_2 = -l_3 \end{aligned}$$

Thus when $d_2 = -l_3$ or simply $Link_3$ doesn't produce any offset from $Link_1$ axis, it will be in singular position. Simply imagine the state in which $d_2 = -l_3$ and we want to produce the translation velocity that is radial to the z-axis of $Link_1$ and it's simply impossible in this position thus it's a singular position. The singularity space for this manner is cylindrical surface with the radius of l_1.

RRP:

$$det(\mathbf{J}_v) = 0 \rightarrow$$

$$\rightarrow \begin{vmatrix} -(d+l_2)\cos(\theta_1)\cos(\theta_2) & (d+l_2)\sin(\theta_1)\sin(\theta_2) & -\cos(\theta_2)\sin(\theta_1) \\ -(d+l_2)\cos(\theta_2)\sin(\theta_1) & -(d+l_2)\cos(\theta_1)\sin(\theta_2) & \cos(\theta_1)\cos(\theta_2) \\ 0 & (d+l_2)\cos(\theta_2) & -\cos(\theta_2+90^\circ) \end{vmatrix} = 0 \rightarrow$$

$$\rightarrow (d+l_2)\cos(\theta_2) = 0$$

Now we have the following cases:

$$\begin{cases} d = -l_2 \\ \theta_2 = \pm 90^\circ \end{cases}$$

Now considering the first case, it means that the end-effector is placed in revolute joint and the actuation of other joints doesn't have any effect on the position and velocity state of the end-effector. Or in other words, for producing any end-effector velocity vector we should have kind of a infinite velocities in other actuators. Thus it's one of singular cases.
For the second case, whole robot's body including its links is along $Link_1$ and thus the end-effector can only produce velocity along the $Link_1$ and other velocity vectors cause singularity.
Another singularity case is when $\theta_1 = 0, \pm 90^\circ, \pm 180^\circ$. In this case, robots body falls into one of the $x-z$ plane or $y-z$ plane of World Frame and thus if we want to produce a velocity vector that is perpendicular to these planes, we will end up having singularity case.

Question 2

Assume the end effector of the spherical robot (Homework 3) must be able to generate the velocity vector $\vec{V} = (150, 100, 50)$ mm/s at all points in its workspace. Consider the robot's workspace such that joint 1 (θ_1) moves from 0° to 80°, joint 2 (θ_2) from −45° to 45°, and joint 3 (d) from 100 to 200 mm. Find the maximum required velocity for each of the robot's three motors. Provide a summary report of the solution process along with MATLAB code and final results.

Solution

Using the velocity Jacobians found in Question 1, I use the following eqaution to find the joint velocities for the given end-effector velocity:

$$\vec{V} = \mathbf{J}_v\vec{\Theta} \rightarrow \ \vec{\Theta} = \mathbf{J}_v^{-1}\vec{V}$$

Here is the MATLAB function that I wrote to find the maximum velocity of each joint actuator:

```
function max = maxMotor(Jv,V,theta1range,theta2range,drange)
syms theta1 theta2 l2 d real
n=size(Jv,2);
Jv=subs(Jv,l2,300);
obj=inv(Jv)*V;
```

```
max=zeros(n,1);
Q=[theta1;theta2;d];
lowbound=[theta1range(1) theta2range(1) drange(1)];
upbound=[theta1range(2) theta2range(2) drange(2)];
options=optimset('fmincon');
options.Display='off';
ms=MultiStart('FunctionTolerance',1e-2,'UseParallel',true);
gs = GlobalSearch(ms,'XTolerance',1e-3,'StartPointsToRun','
   bounds');
parpool('local');
parfor i=1:n
    fun=@(x) double(subs(-abs(obj(i,:)),Q,x'));
    p=createOptimProblem('fmincon','x0',lowbound,'objective',
       fun,'lb',lowbound,'ub',upbound,'options',options);
    [~,tmp]=run(gs,p);
    max(i)=-tmp;
end
delete(gcp);
end
```

Now for the efficiency of MATLAB function, I only use the i-th row of $\mathbf{J}_v^{-1}$ for finding the maximum velocity of joint i actuator. I substitute the known parameters in the velocity Jacobians of the robot. To find the maximum velocity of each joint actuator, I find the minimum of the negative sign of the absolute of actuator velocity. With this manner, I find the actuator's maximum velocity considering both positive and negative actuation, for example clockwise and counter-clockwise rotation for the revolute joint. I use the built-in optimization function `fmincon` by defining the upper and lower bounds for the exploration space. Using the code above, I found the following maximum velocities for the joint actuator velocities:

$$\vec{\Theta}_{max} = \begin{bmatrix} \theta_{1max} \\ \theta_{2max} \\ d_{max} \end{bmatrix} = \begin{bmatrix} 0.6374\ rad/s \\ 0.3188\ rad/s \\ 139.6165\ mm/s \end{bmatrix}$$