ECOLE CENTRALE DE NANTES

MODELLING AND CONTROL OF MANIPULATORS

LAB REPORT – 2

Submitted by

REGULAN GOPI KRISHNAN

RAMACHANDRAN RAGESH

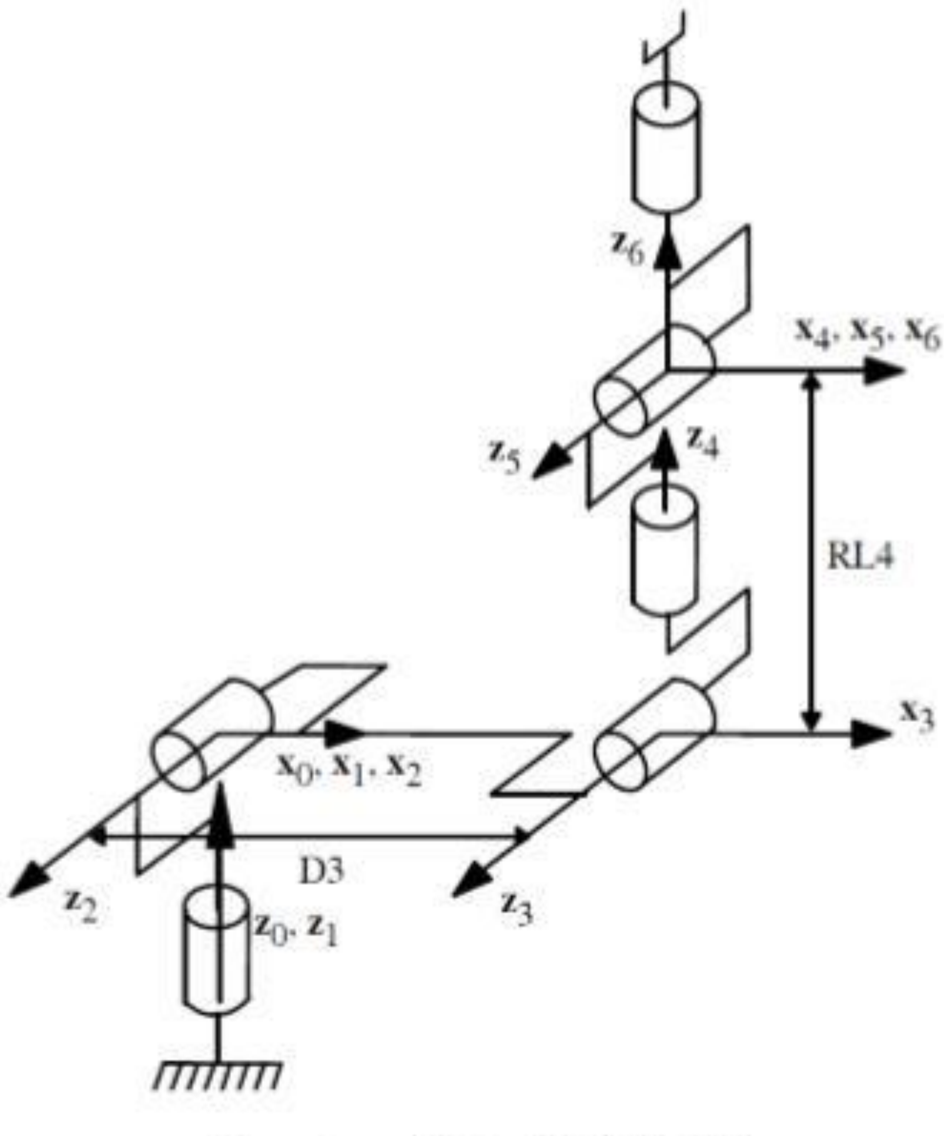


Fig. 1 – Stäubli RX90.

PART I:

I- Develop a Matlab function to calculate the kinematic Jacobian matrix 0J6 of the robot RX90 (see Figure 1). Make use of the symbolic expressions of the kinematic Jacobian matrix 3J6 and the symbolic expressions of the orientation matrix 0R3. This function will be called as follows: J = JACRX90(q).

$$
{}^{3}\mathbf{J}_{6} = \begin{bmatrix} 0 & -RL4+S3D3 & -RL4 & 0 & 0 & 0 \\ 0 & C3D3 & 0 & 0 & 0 & 0 \\ S23RL4-C2D3 & 0 & 0 & 0 & 0 & 0 \\ S23 & 0 & 0 & 0 & S4 & -S5C4 \\ C23 & 0 & 0 & 1 & 0 & C5 \\ 0 & 1 & 1 & 0 & C4 & S5S4 \end{bmatrix}
$$

SOLUTION:

In general, we calculate v_n and ω_n in frame R_n or frame R_0. The corresponding Jacobian matrix is denoted by ${}^{n}\mathbf{J}_n$ or ${}^{0}\mathbf{J}_n$ respectively. These matrices can also be computed using any matrix ${}^{i}\mathbf{J}_n$, for i = 0, ..., n, thanks to the following expression:

$$
{}^{s}\mathbf{J}_{n} = \begin{bmatrix} {}^{s}\mathbf{R}_{i} & \mathbf{0}_{3} \\ \mathbf{0}_{3} & {}^{s}\mathbf{R}_{i} \end{bmatrix} {}^{i}\mathbf{J}_{n}
$$

where ${}^{s}\mathbf{R}_i$ is the (3x3) orientation matrix of frame R_i relative to frame R_s.

MATLAB CODE:

```
function J06 = JACRX90(q)
 D3=0.45;
 RL4=0.45;

q1=q(1);
q2=q(2);
q3=q(3);
q4=q(4);
q5=q(5);
q6=q(6);

J36=[0, -RL4+sin(q3)*D3, -RL4, 0, 0, 0;
    0, cos(q3)*D3, 0, 0, 0, 0;
    (sin(q2+q3)*RL4)-(cos(q2)*D3), 0, 0, 0, 0, 0;
    sin(q2+q3), 0, 0, 0, sin(q4), -sin(q5)*cos(q4);
    cos(q2+q3), 0, 0, 1, 0, cos(q5);
    0, 1, 1, 0, cos(q4), sin(q5)*sin(q4)];
```

```
R03=[cos(q1)*cos(q2+q3),-cos(q1)*sin(q2+q3),sin(q1);
     sin(q1)*cos(q2+q3),-sin(q1)*sin(q2+q3),-cos(q1);
     sin(q2+q3),cos(q2+q3),0];

J03=[R03, zeros(3);
    zeros(3), R03];


J06=J03*J36;

end
```

II- Develop a Matlab function that calculates the kinematic Jacobian matrix 0Jn of a general serial robot. This function will be called by: J0n = genjac (sigma, alpha, d, theta, r, q), where (sigma, alpha, d, theta, r) are the geometric parameters of the robot, and q is the vector of joint variables. Compare the results of this function with the function JACRX90 developed in I using three random values for the joint variables. Note: - Determine at first the vectors 0aj and 0Pj, for j = 1,…,n. - For the comparison calculate: [JACRX90(q)- genjac (sigma,alpha,d,theta,r,q)] for a few random q.

SOLUTION:

Since the vector product, $a_k \times L_{k,n}$ can be computed by, $\hat{a}_k . L_{k,n}$, the k^{th} column of ${}^iJ_n$, denoted as ${}^iJ_n(:,k)$,

$$ {}^i\mathbf{J}_n(:,k) = \begin{bmatrix} \sigma_k {}^i\mathbf{a}_k + \bar{\sigma}_k {}^i\mathbf{R}_k {}^k\hat{\mathbf{a}}_k {}^k\mathbf{L}_{k,n} \\ \bar{\sigma}_k {}^i\mathbf{a}_k \end{bmatrix} $$

Since ${}^ka_k = [0 \quad 0 \quad 1]^T$ and ${}^kL_{k,n} = {}^kP_n$, we obtain:

$$ {}^i\mathbf{J}_n(:,k) = \begin{bmatrix} \sigma_k {}^i\mathbf{a}_k + \bar{\sigma}_k (-{}^kP_{n_y} {}^i\mathbf{s}_k + {}^kP_{n_x} {}^i\mathbf{n}_k) \\ \bar{\sigma}_k {}^i\mathbf{a}_k \end{bmatrix} $$

where ${}^kP_{nx}$ and ${}^kP_{ny}$ denote the x and y components of the vector ${}^kP_n$ respectively.

From this expression, we obtain the k^{th} column of ${}^nJ_n$ as:

$$ {}^n\mathbf{J}_n(:,k) = \begin{bmatrix} \sigma_k {}^n\mathbf{a}_k + \bar{\sigma}_k (-{}^kP_{n_y} {}^n\mathbf{s}_k + {}^kP_{n_x} {}^n\mathbf{n}_k) \\ \bar{\sigma}_k {}^n\mathbf{a}_k \end{bmatrix} $$

The column ${}^nJ_n(:,k)$ is computed from the elements of the matrix ${}^kT_n$ resulting from the DGM.

In a similar way since $L_{k,n} = L_{k,i} + L_{i,n} = L_{i,n} - L_{i,k}$ thus the k^{th} column of ${}^iJ_n$ can be written as:

$${}^iJ_n(:,k) = \begin{bmatrix} \sigma_k {}^i\mathbf{a}_k + \bar{\sigma}_k {}^i\hat{\mathbf{a}}_k ({}^i\mathbf{P}_n - {}^i\mathbf{P}_k) \\ \bar{\sigma}_k {}^i\mathbf{a}_k \end{bmatrix}$$

which gives for $i = 0$:

$${}^0J_n(:,k) = \begin{bmatrix} \sigma_k {}^0\mathbf{a}_k + \bar{\sigma}_k {}^0\hat{\mathbf{a}}_k ({}^0\mathbf{P}_n - {}^0\mathbf{P}_k) \\ \bar{\sigma}_k {}^0\mathbf{a}_k \end{bmatrix}$$

MATLAB CODE:

```
function J = genjac (sigma, alpha, d, theta, r,q)
n=length(sigma);
P=zeros(3,n);
a=zeros(3,n);
T=eye(4);
J = zeros(6,n);
for k=1:n
     T = T*TRANSMAT(alpha(k),d(k),theta(k),r(k));
     P(1:3,k)=T(1:3,4);
     a(1:3,k)=T(1:3,3);
end


for k=1:n
     J(:,k) = [sigma(k).*a(:,k) + ~sigma(k).*cross(a(:,k),(P(:,6)-(P(:,k))));
         ~sigma(k).*a(:,k)];

end

end
```

Matlab function that calculates a kinematic Jacobian matrix 0Jn of a general serial robot:

```
sigma=[0 0 0 0 0 0];
alpha=[0 pi/2 0 -pi/2 pi/2 -pi/2];
d=[0 0 .45 0 0 0 ];
r=[0 0 0 .45 0 0];
q=[0.6  1.25  -0.3  0.6  0.3  2.0];
theta=q;

J0N = genjac (sigma, alpha, d, theta, r, q)
```

Comparing the results of this function with the function **JACRX90** developed in I using three random values for the joint variables.

```
for i = 1:3

    q = rand(1:6);
    sigma = [0,0,0,0,0,0];
    alpha = [0 pi/2 0 -pi/2 pi/2 -pi/2];
    theta = [q(1) q(2) q(3) q(4) q(5) q(6)];
    d = [0 0 0.45 0 0 0];
    r = [0 0 0 0.45 0 0];
    Q1=JACRX90(theta)
    Q2= genjac(sigma,alpha,d,theta,r,q)
    Q3=Q2-Q1

end
```

Solutions for the three random variables using the above code:

```
Q1 =

   -0.0353   -0.3381   -0.3253         0         0         0
    0.1449   -0.0824   -0.0793         0         0         0
         0    0.1492   -0.3006         0         0         0
         0    0.2369    0.2369   -0.6490    0.2424   -0.9453
         0   -0.9715   -0.9715   -0.1583   -0.9702   -0.2350
    1.0000         0         0    0.7441    0.0051    0.2263


Q2 =

   -0.0353   -0.3381   -0.3253         0         0         0
    0.1449   -0.0824   -0.0793         0         0         0
         0    0.1492   -0.3006         0         0         0
         0    0.2369    0.2369   -0.6490    0.2424   -0.9453
         0   -0.9715   -0.9715   -0.1583   -0.9702   -0.2350
    1.0000    0.0000    0.0000    0.7441    0.0051    0.2263


Q3 =

   1.0e-15 *

    0.0139    0.0555    0.0555         0         0         0
   -0.0278    0.0139   -0.0139         0         0         0
         0   -0.0278         0         0         0         0
         0         0         0         0         0    0.1110
         0         0         0   -0.0278    0.1110         0
```

```
   -0.0570   -0.4671   -0.4224         0         0         0
    0.3083   -0.0863   -0.0780         0         0         0
         0    0.3135   -0.1342         0         0         0
         0    0.1817    0.1817   -0.2932    0.3061   -0.5662
         0   -0.9834   -0.9834   -0.0542   -0.9511   -0.1481
    1.0000         0         0    0.9545    0.0401    0.8109


Q2 =

   -0.0570   -0.4671   -0.4224         0         0         0
    0.3083   -0.0863   -0.0780         0         0         0
         0    0.3135   -0.1342         0         0         0
         0    0.1817    0.1817   -0.2932    0.3061   -0.5662
         0   -0.9834   -0.9834   -0.0542   -0.9511   -0.1481
    1.0000    0.0000    0.0000    0.9545    0.0401    0.8109


Q3 =

   1.0e-15 *

    0.0069         0    0.0555         0         0         0
         0    0.0139   -0.0139         0         0         0
         0    0.1110   -0.0278         0         0         0
         0         0         0         0         0   -0.1110
         0         0         0         0         0   -0.0278
fx       0    0.0612    0.0612         0    0.0555         0
```

```
Q1 =

    0.0282   -0.5432   -0.2256         0         0         0
   -0.0784   -0.1958   -0.0813         0         0         0
         0   -0.0833   -0.3808         0         0         0
    0.0000    0.3390    0.3390   -0.7961    0.4136   -0.9104
         0   -0.9408   -0.9408   -0.2869   -0.9007   -0.4095
    1.0000         0         0    0.5328    0.1330    0.0584


Q2 =

    0.0282   -0.5432   -0.2256         0         0         0
   -0.0784   -0.1958   -0.0813         0         0         0
         0   -0.0833   -0.3808         0         0         0
         0    0.3390    0.3390   -0.7961    0.4136   -0.9104
         0   -0.9408   -0.9408   -0.2869   -0.9007   -0.4095
    1.0000    0.0000    0.0000    0.5328    0.1330    0.0584


Q3 =

   1.0e-15 *

   -0.0173         0   -0.0278         0         0         0
    0.0833    0.0278   -0.0416         0         0         0
         0    0.0971         0         0         0         0
   -0.0555         0         0    0.1110         0    0.1110
         0         0         0         0         0         0
```

III- Supposing that the RX90 robot is at the configuration defined by:

q = [0.6   1.2 5 -0.3   0.6   0.3   2.0]T

a-Calculate the differential translational vector and the differential rotation vector corresponding to the differential joint vector defined as follows:

dq = [0.08 .012   -0.02   0.006   -0.03   0.03]T

b- Verify the result of a) using the direct geometric model at q and at q+dq.

SOLUTION:

The differential transformation of the position and orientation – or location – of a frame R_i attached to any body may be expressed by a differential translation vector dP_i expressing the translation of the origin of frame R_i, and of a differential rotation vector $\boldsymbol{\delta}_i$, equal to $\mathbf{u}_i\, d\theta$, representing the rotation of an angle $d\theta$ about an axis, with unit vector $\mathbf{u}_i$, passing through the origin O_i.

Then the differential transformation matrix delta is defined as

$$\boldsymbol{\Delta} = [\mathbf{Trans}(dx, dy, dz)\ \mathbf{Rot}(\mathbf{u}, d\theta) - \mathbf{I}_4]$$

such that:

$$d\,{}^i\mathbf{T}_j = {}^i\boldsymbol{\Delta}\ {}^i\mathbf{T}_j$$

or:

$$d\,{}^i\mathbf{T}_j = {}^i\mathbf{T}_j\ {}^j\boldsymbol{\Delta}$$

$${}^j\boldsymbol{\Delta} = \begin{bmatrix} {}^j\hat{\boldsymbol{\delta}}_j & & & {}^j\mathbf{dP}_j \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} {}^j\hat{\mathbf{u}}_j\, d\theta & & & {}^j\mathbf{dP}_j \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\hat{\mathbf{u}}$ and $\hat{\boldsymbol{\delta}}$ represent the skew-symmetric matrices defined by the vectors $\mathbf{u}$ and $\boldsymbol{\delta}$ respectively.

Note that the transformation matrix between screws can also be used to transform the differential translation and rotation vectors between frames:

$$\begin{bmatrix} {}^j\mathbf{dP}_j \\ {}^j\boldsymbol{\delta}_j \end{bmatrix} = {}^j\mathbf{S}_i \begin{bmatrix} {}^i\mathbf{dP}_i \\ {}^i\boldsymbol{\delta}_i \end{bmatrix}$$

Using the genjac( ) function we can find the Jacobian matrix and using the DGMRX code, we can find the dq and q + dq matrices.

MATLAB CODE:

```
q = [0.6 1.25 -0.3 0.6 0.3 2.0]';
dq= [0.08 .012 -0.02 0.006 -0.03 0.03]';
sigma = [0,0,0,0,0,0]';
alpha = [0 pi/2 0 -pi/2 pi/2 -pi/2]';
theta = [q(1) q(2) q(3) q(4) q(5) q(6)]';
d = [0 0 0.45 0 0 0]';
r = [0 0 0 0.45 0 0]';
J=genjac(sigma,alpha,d,theta,r,q)
dX=J*dq
dX2=DGMRX(q+dq)-DGMRX(q)
```

OUTPUT OF THE MATLAB CODE:

```
J =

    0.1266   -0.5685   -0.2160         0         0         0
   -0.1850   -0.3889   -0.1478         0         0         0
         0   -0.2241   -0.3660         0         0         0
         0    0.5646    0.5646   -0.6713    0.7371   -0.6642
         0   -0.8253   -0.8253   -0.4593   -0.4957   -0.6566
    1.0000    0.0000    0.0000    0.5817    0.4593    0.3573


dX =

    0.0076
   -0.0165
    0.0046
   -0.0506
   -0.0010
    0.0804


dX2 =

   -0.0116    0.0597    0.0534    0.0084
   -0.0891    0.0096   -0.0327   -0.0163
   -0.0104    0.0376    0.0323    0.0046
         0         0         0         0
```

IV- Plot the joint space and the working space of the 2R robot (L1 = 0.5 m, L2 = 0.4 m and whose angles limits are q1max = q2max = 2.8 rad and q1min= q2min = -2.6 rad).

MATLAB CODE:

```
L1 = 0.5;
L2 = 0.4;
q1min=-2.6;
q1max=2.8;
q2min=-2.6;
q2max=2.8;
for q1=q1min:0.05:q1max;
 for q2=q2min:0.05:q2max;
   q=[q1,q2];
   X=DGM2R(L1,L2,q);
   x1=[x1,X(1)];
   x2=[x2,X(2)];
 end
end


%JOINT SPACE
t=q1min:0.05:q1max;
figure(1);
jointspace = plot(t,q1min*ones(size(t)),
t,q1max*ones(size(t)),q2min*ones(size(t)),t,q2max*ones(size(t)),t);
xlabel('Joint Q1');ylabel('Joint Q2')

%WORK SPACE
figure(2);
plot(x1,x2);
xlabel('X1');ylabel('X2')
```

V- Plot the surfaces of singularity of the robot RX90 in the plane (q2-q3). Consider the case where D3 = RL4 = 0.45 and the case where D3 = 0.55 and RL4 = 0.45. Make use of the symbolic expressions of the singularities C3 = 0, and RL4 S23 – D3 C2 = 0, and the function “ezplot” of Matlab.

SOLUTION:

The ezplot function is used for the plotting of the surface of the singularity robot.

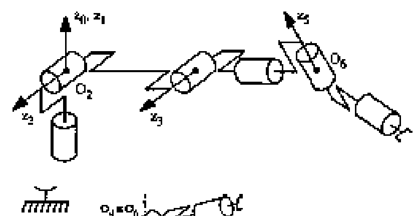
Case 1: D3 = RL4 = 0.45.

Case 2: D3 = 0.55 and RL4 = 0.45.

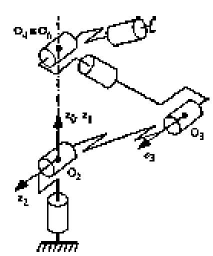
The symbolic expressions of the singularities C3 = 0, and RL4 S23 – D3 C2 = 0 are used.

Singularities explained in the following figures:

– when C3 = 0 (elbow singularity), the robot is fully extended or fully folded. In this case, the origin O_6 is located on the boundary of its workspace.



– the singularity S23 RL_4 – C2 D_3 = 0 (shoulder singularity), corresponds to a configuration in which O_6 is located on the z_0 axis. In this configuration, where Px = Py = 0, the third row of ${}^3J_6$ is zero.



MATLAB CODE:

```
syms q2 q3
%CASE 1
D3 = 0.45;
RL4 = 0.45;
figure(1);
ezplot((RL4*sin(q2+q3) - D3*cos(q2)),[-pi pi -pi pi]);
hold on;
ezplot(cos(q3)*(RL4*sin(q2+q3) - D3*cos(q2)),[-pi pi -pi pi]);

%CASE 2
D3 = 0.55;
RL4 = 0.45;
figure(2);
ezplot((RL4*sin(q2+q3) - D3*cos(q2)),[-pi pi -pi pi]);
hold on;
ezplot(cos(q3)*(RL4*sin(q2+q3) - D3*cos(q2)),[-pi pi -pi pi]);
```

OUTPUT OF THE MATLAB CODE:

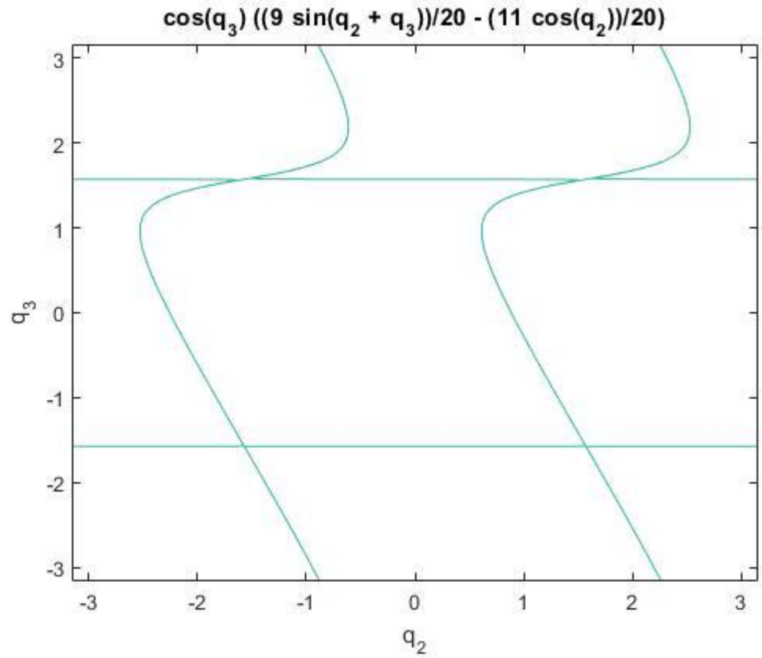


Fig. 2 Case 1: D3 = RL4 = 0.45. Plot of singularities

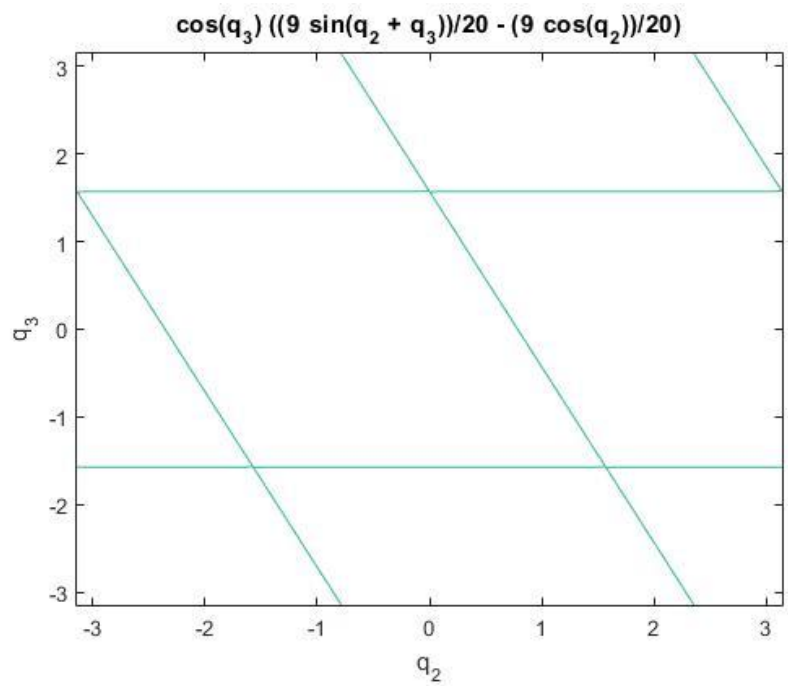


Fig. 2 Case 2: D3 = 0.55 and RL4 = 0.45. Plot of singularities

VI- Determine the dimension and orthogonal basis of the following subspaces for the RX90 robot at the configuration q1=[0.2 -0.3 -pi/2 0.6 0.0 0.7]T - the null space of J; - the subspace generating Cartesian velocities; - the subspace of the achievable Cartesian velocity; - the subspace of the Cartesian velocities that cannot be generated.

SOLUTION:

The dimension and orthogonal basis of the subspaces is calculated for the given robot using the (SVD) theory.

Singular value decomposition theory states that for any (mxn) matrix J of rank r there exist orthogonal matrices U and V of dimensions (mxm) and (nxn) :

Singular value decomposition $\mathbf{J} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^T$

U (m×m) orthogonal matrix columns of U are the eigenvectors of $J.J^T$

V (n×n) orthogonal matrix: columns of V are the eigenvectors of $J^T.J$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{S}_{rxr} & \mathbf{0}_{rx(n-r)} \\ \mathbf{0}_{(m-r)xr} & \mathbf{0}_{(m-r)x(n-r)} \end{bmatrix}$$ Using svd of J r = rank of J

S is an (r×r) diagonal matrix, formed by the non-zero singular values of J, which are arranged in decreasing order such that $s_1 \geq s_2 \geq \ldots \geq s_r$. The singular values of J are the square roots of the eigenvalues of the matrix J^TJ if $n \geq m$ (or $J.J^T$ if $n \leq m$).

$$\dot{\mathbf{X}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\dot{\mathbf{q}} \qquad \dot{\mathbf{X}} = \sum_{i=1}^{r} s_i \mathbf{U}_i \mathbf{V}_i^T \dot{\mathbf{q}} \qquad \text{since } s_i = 0 \text{ for } i > r$$

Singular value decomposition

From the last relation, we deduce that:

– the vectors $V_1, \ldots, V_r$ form an orthonormal basis for the subspace of $\mathbf{q}$ generating an end-effector velocity;

– the vectors $V_{r+1}, \ldots, V_n$ form an orthonormal basis for the subspace of $\mathbf{q}$ giving $\mathbf{X}$=0. They define the null space of J, denoted by $\mathcal{N}(J)$;

– the vectors $U_1, \ldots, U_r$ form an orthonormal basis for the set of the achievable end-effector velocities X. Hence, they define the range (image) space of J, denoted by $\Re(J)$;

– the vectors $U_{r+1}, \ldots, U_m$ form an orthonormal basis for the subspace composed of the set of X that cannot be generated by the robot. In other words, they define the complement of the range space, denoted by $\Re(J)^{\perp}$;

– the singular values represent the velocity transmission ratio from the joint space to the task space. In fact, multiplying the previous equation by U_i^T yields:

$$U_i^T\,\dot{\mathbf{X}} = s_i\,V_i^T\,\dot{\mathbf{q}} \qquad \text{for } i = 1, \ldots, r$$

MATLAB CODE:

```
q = [0.2, -0.3, -pi/2, 0.6, 0, 0.7];
J = JACRX90(q);
[m n] = size(J);
[U,S,V] = svd(J);
    display(U);
    display(S);
    display(V);

 r = rank(S);
 display('the null space of J');
 nullspace = V(:,r+1:n)
 display('The subspace generating Cartesian velocities');
 cartesianvel = V(:,1:r)
 display('The subspace of the achievable Cartesian velocity');
 AV = U(:,1:r)
 display('The subspace of the Cartesian velocities that cannot be
generated');
 UAV = U(:,r+1:m)
```

OUTPUT OF THE MATLAB CODE:

```
U =

   -0.1469    0.0353   -0.0469   -0.3136    0.0014    0.9363
    0.0368   -0.3462    0.4805    0.4358   -0.6495    0.1898
   -0.4418   -0.1106    0.1601   -0.7136   -0.4128   -0.2955
   -0.1599    0.6269    0.7288    0.0374    0.2211   -0.0000
    0.8437    0.2431    0.0800   -0.3811   -0.2783    0.0000
    0.2109   -0.6439    0.4513   -0.2363    0.5304   -0.0000


S =

    1.8760         0         0         0         0         0
         0    1.5290         0         0         0         0
         0         0    1.2439         0         0         0
         0         0         0    0.5693         0         0
         0         0         0         0    0.1521         0
         0         0         0         0         0    0.0000


V =

    0.1423   -0.6158    0.6948    0.3241   -0.1127         0
   -0.6796   -0.1425    0.1746   -0.5118   -0.4748    0.0000
   -0.5686   -0.1085    0.1140    0.0786    0.8036   -0.0000
   -0.0277    0.5385    0.4535    0.0571   -0.0168   -0.7071
   -0.4394    0.0934   -0.2496    0.7876   -0.3400   -0.0000
   -0.0277    0.5385    0.4535    0.0571   -0.0168    0.7071
```

```
the null space of J

nullspace =

         0
    0.0000
   -0.0000
   -0.7071
   -0.0000
    0.7071

The subspace generating Cartesian velocities

cartesianvel =

    0.1423   -0.6158    0.6948    0.3241   -0.1127
   -0.6796   -0.1425    0.1746   -0.5118   -0.4748
   -0.5686   -0.1085    0.1140    0.0786    0.8036
   -0.0277    0.5385    0.4535    0.0571   -0.0168
   -0.4394    0.0934   -0.2496    0.7876   -0.3400
   -0.0277    0.5385    0.4535    0.0571   -0.0168
```

```
The subspace of the achievable Cartesian velocity

AV =

   -0.1469    0.0353   -0.0469   -0.3136    0.0014
    0.0368   -0.3462    0.4805    0.4358   -0.6495
   -0.4418   -0.1106    0.1601   -0.7136   -0.4128
   -0.1599    0.6269    0.7288    0.0374    0.2211
    0.8437    0.2431    0.0800   -0.3811   -0.2783
    0.2109   -0.6439    0.4513   -0.2363    0.5304

The subspace of the Cartesian velocities that cannot be generated

UAV =

    0.9363
    0.1898
   -0.2955
   -0.0000
    0.0000
   -0.0000
```