MOCOM/MOROB – LAB 2

PART I

I- Develop a Matlab function to calculate the kinematic Jacobian matrix ${}^0\mathbf{J}_6$ of the robot RX90 (see Figure 1). Make use of the symbolic expressions of the kinematic Jacobian matrix ${}^3\mathbf{J}_6$ and the symbolic expressions of the orientation matrix ${}^0\mathbf{R}_3$. This function will be called as follows: **J = JACRX90(q).**

$$
{}^3\mathbf{J}_6 = \begin{bmatrix}
0 & -\mathrm{RL4} + S3\mathrm{D3} & -\mathrm{RL4} & 0 & 0 & 0 \\
0 & C3\mathrm{D3} & 0 & 0 & 0 & 0 \\
S23\mathrm{RL4} - C2\mathrm{D3} & 0 & 0 & 0 & 0 & 0 \\
S23 & 0 & 0 & 0 & S4 & -S5C4 \\
C23 & 0 & 0 & 1 & 0 & C5 \\
0 & 1 & 1 & 0 & C4 & S5S4
\end{bmatrix}
$$

II- Develop a Matlab function that calculates the kinematic Jacobian matrix ${}^0\mathbf{J}_n$ 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 ${}^0\mathbf{a}_j$ and ${}^0\mathbf{P}_j$, for j = 1,…,n.
- For the comparison calculate:
  [JACRX90(q)- genjac (sigma,alpha,d,theta,r,q)] for a few random **q**.

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

$\mathbf{q} = [0.6 \quad 1.2\ 5 \quad -0.3 \quad 0.6 \quad 0.3 \quad 2.0]^T$

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

dq = $[0.08 \quad .012 \quad -0.02 \quad 0.006 \quad -0.03 \quad 0.03]^T$

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

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 $\mathbf{q1_{max}} = \mathbf{q2_{max}} = \mathbf{2.8\ rad}$ and $\mathbf{q1_{min}} = \mathbf{q2_{min}} = \mathbf{-2.6\ rad}$).

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.

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.

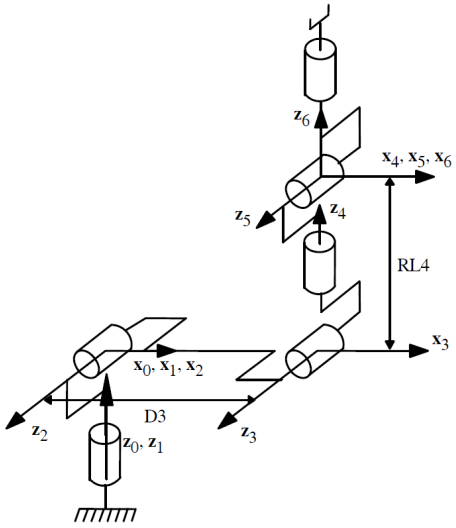

Fig. 1 – Stäubli RX90.

PART II

I- Develop for the 3R planar robot a numerical IGM using the differential model. Suppose that the geometric parameters and joint limits are given as (see Figure 2): **L1 = 0.6**, **L2 = 0.5**, **L3 = 0.3**, $\mathbf{q1_{max} = q2_{max} = 2.7\ rad}$ and $\mathbf{q1_{min} = q2_{min} = -2.5\ rad}$, $\mathbf{q3_{max} = 2.9\ rad}$ and $\mathbf{q3_{min} = -2.9\ rad}$):

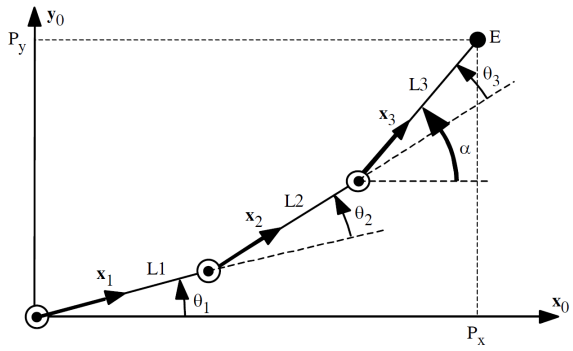

Fig. 2 – 3R planar robot.

The function will be called as **[Q, err, k, m] = IGMDIFF3R (X)**, With **X = (Px, Py, α)** are the coordinates of the third link. **Px**, **Py** are the coordinates of the terminal point of link 3, in meters, and **α** is the angle between **x0** and the **x3** axis, in rad. This function will give one solution (**Q**) of the inverse geometric model for the desired **X**. If there is no solution err =1.

k and **m** are counters that are defined in the following algorithm. Test this algorithm using the Cartesian values corresponding to the following joint configurations:

q1 = $[0.6\ 0.4\ 1.3]^T$, **q2 =** $[-0.5\ 0.9\ 0.7]^T$, **q3 =** $[0.7\ 0.0\ 0.8]^T$, **q4 =** $[2.9\ -0.4\ 0.8]^T$.

The algorithm:

i– Initialize the first counter **k = 0** (this counter will give the number of trials for the initial joint configuration).

ii- **k = k + 1**, if **k = 500**, put **err = 1** and stop the programme.

iii- pick up a random value for $\mathbf{q^c}$ using the Matlab function rand[1]. Put **m = 0**, (this counter gives the number of iterations which have been intialized by this joint configuration).

iv– Calculate the Cartesian coordinates $\mathbf{X^c}$ using the DGM of the 3R robot;

v- Calculate the difference **dX** between the desired $\mathbf{X^d}$ and the current $\mathbf{X^c}$

$$\mathbf{dX = X^d - X^c}$$

If **max(abs(dX)) < ε** , put $\mathbf{q^d = q^c}$, if the joint angles are within the joint domains. Put **err = 0** and stop the programme.

vi- **m = m + 1**, if **m = 1000** goto ii,

viii- Update the value of $\mathbf{q^c}$ as follows:

- Calculate the Jacobian matrix **J = J3R(q^c)**;

- calculate **dq** corresponding to **dX** using the pseudo-inverse;

- **$q^c = q^c + dq$**

ix – goto iv.

Notes:

- take **$\varepsilon = 10^{-10}$**

- If **dX** is too big we have to reduce it as follows in order that the differential model will be valid

- if **norm(dX)>S**, then **dX = [dX/norm(dX)]·S**, with **S = 0.2**

- **J3R** is the Jacobian matrix of 3R robots, where the task coordinates are as defined previously.

- if any angle exceeds 2π use the matlab function rem[1]

II- Plot the evolution of the joint variables and the Cartesian position variables (**Px**, **Py**) of the 3R robot while executing a straight line motion between two points. The linear trajectory generator is composed of 50 steps with equal length. The transformation from Cartesian coordinates into joint coordinates will be carried out using the redundant Jacobian matrix (RJ3R) (where the Cartesian coordinates are the coordinates **x** and **y** of the terminal point). The straight line has as terminal configurations:

q^i = $[\pi/6, -\pi/6, \pi/6]^T$ (initial configuration), **q^f =** $[\ 3\pi/4, \pi/2, \pi/3]^T$ (final conf.)

Note. The Algorithm can be as follows:

Calculate **Xi** and **Xf** using DGM (the direct geometric model of the 3R robot where the task vector is composed of the terminal point coordinates of the third link only, thus the robot is redundant wrt the task).

Nstep = 50

qc = qi

Xd = Xi

Xc = Xi

---

[1]After each updating of **q** find the remainder wrt 2*pi, using **q**= rem(**q**, 2*pi), such that the angle will be < 2*pi.

stepX = (Xf-Xi)/Nstep

For k = 1 : Nstep

Xd = Xd + stepX

dX = Xd – Xc; **Xc** gives the current **Px** and **Py**

calculate **dq** corresponding to **dX** using pseudo inverse to minimize the norm of **dq**,

qc = qc + dq

calculate **Xc** using the RDGM

end

III- Repeat the previous problem but use the pseudo inverse with optimization component avoiding the joint limits. Calculate the coefficient **α** such that the two parts of the optimisation will have equal norm for the corresponding **dq** terms, i.e **norm $(J^{+}dX) = norm(\alpha(I - J^{+}J)Z)$**, thus **$\alpha = norm(J^{+}dX)/norm((I - J^{+}J)Z)$,** with **Z=gradient(Φ).**