Laboratory N.2

Michele Tartari, Stefano Castagnetta

October 23, 2017

Abstract

This paper represents the report for MoCom Laboratory N.2 held on the 16 October 2017 and during which the symbolic language of MATLAB was used to analyze the kinematic model of the RX90 robot. Firstly we computed the jacobian matrix for both the specific and the general case. We then validated the differential model by comparison with the DGM for given values. The second part of this paper analyzes the 3R planar robot, more specifically we defined the differential IGM and compared the end effector translation with the variation of the joint variables using two different techniques.

Contents

1. RX90 Robot

1.1. JACRX90.m

This function recives the joint variables as input and returns the the kinematic jacobian ${}^0J_6$ exclusively for the robot RX90.
First the matrix ${}^3J_6$ is computed from the MDH[1] table and given the know geometrical parameters R_{L_4} and D_3. Afterwards we made use of the relation:

$$ {}^0J_6 = \begin{bmatrix} {}^0R_3 & 0 \\ 0 & {}^0R_3 \end{bmatrix} \cdot {}^3J_6 $$

where ${}^0R_6$ in the rotation part of the homogeneous transformation matrix ${}^0T_6$ which has been computed using the MATLAB function DHSym that has been scripted in Lab N.1.

[1] Modified Denavit-Heaerg

1.2. genjac.m

The function genjac calculate the kinematic jacobian matrix ${}^0J_6$ from input MDH parameters (σ, α, d, θ, r). This function follow the steps descried below:

1. Compute ${}^0T_N$.
2. Extract p_N from ${}^0T_k = \begin{bmatrix} s_N & n_N & a_N & p_N \\ 0 & 0 & 0 & 1 \end{bmatrix}$.
3. Initialize ${}^0J_N$.
4. for every $k \in [0, N]$, $k \in$integer.
   (a) Compute ${}^0T_k$.
   (b) Extract a_k where ${}^0T_k = \begin{bmatrix} s_k & n_k & a_k & p_k \\ 0 & 0 & 0 & 1 \end{bmatrix}$.
   (c) Define ${}^0J_k = \begin{bmatrix} \sigma_k \cdot \alpha + \overline{\sigma} \cdot (a_k \times (P_N - P_k)) \\ \overline{\sigma} \cdot a_k \end{bmatrix}$
   (d) Insert ${}^0J_k$ as the k^{th} column of the kinematic jacobian matrix ${}^0J_N$.

To validate our code we compared it to the function JACRX90 by subtracting one function from the other and analyzing the difference. As visible in Figure 3 the error will always be of the order of 10^{-15}. This operation has been saved in the compare.m file.

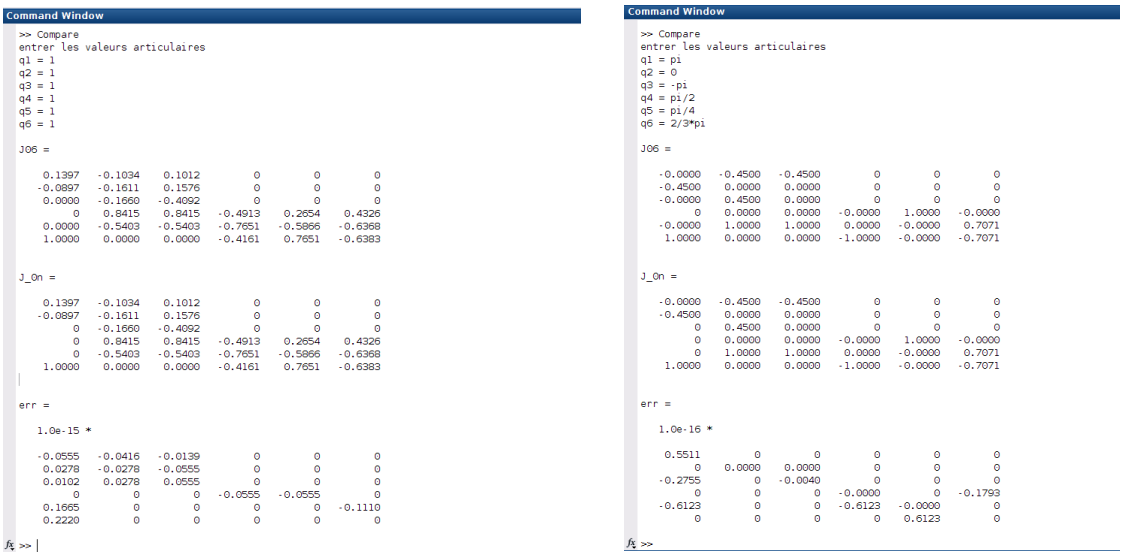

Figure 1: Error in the computation of ${}^0J_N$ between genjac and JACRX90.

1.3. Differential Translation and Rotation Vectors

We calculated the differential translational vector and the differential rotation vector corresponding to the differential joint vector defined as follows:

$$\begin{aligned} q &= \begin{bmatrix} 0.6 & 1.25 & -0.3 & 0.6 & 0.3 & 2.0 \end{bmatrix}^T \\ dq &= \begin{bmatrix} 0.08 & .012 & -0.02 & 0.006 & -0.03 & 0.03 \end{bmatrix}^T \end{aligned}$$

We used the JACRX90 to obtain the kinematic jacobian and compute $dx_{jac} = j \cdot dq$. By exploiting the relation:

$$\begin{bmatrix} dP_N \\ \delta \end{bmatrix} = J_N \cdot dq$$

and by scripting the custom function BryantRotPos we then computed the direct geometrical model for q and $q + dq$, which will give us x and $(x + dx)_{dmg}$. To validate our model we compare $x + dx_{dmg}$ and $x + dx_{Jac}$ by subtraction.

```
%Hom trans matrix for q and q+dq
Tx = DGMRX (q);
Txdx= DGMRX (q+dq);

% Pose with Bryant angles convention for x and x+dx
X = BryantRotPos (Tx);
XdX = BryantRotPos (Txdx);

% Error with differential model
err = XdX - (X+dx.')
```

Through this procedure we validated our model getting the following absolute error:

$$err = \begin{bmatrix} 0.0007 & 0.0002 & -0.0000 & 0.0346 & 0.0704 & -0.0473 \end{bmatrix}$$

1.4. 2R robot Plot

In this section we plotted the joint space and the working space of the 2R robot ($L1 = 0.5m$, $L2 = 0.4m$ and whose angles limits are $q_{1,max} = q_{2,max} = 2.8rad$ and $q_{1,min} = q_{2,min} = -2.6rad$). The joint space representation is a rectangle delimited by $[q_{1,min}, q_{1,max}]$ along the q_1 axes and by $[q_{2,min}, q_{2,max}]$ along the q_2 axes as seen in figure 3.
We then obtained the working space by iteratively computing the $x, y = DGM2R(L_1, L_2, q_1, q_2)$ for $q_1 \in [q_{1,min}, q_{1,max}]$ and $q_2 \in [q_{2,min}, q_{2,max}]$, where $DGM2R$ is a function which returns the solution of the 2R planar robot's DGM for given L_1,L_2,q_1,q_2.

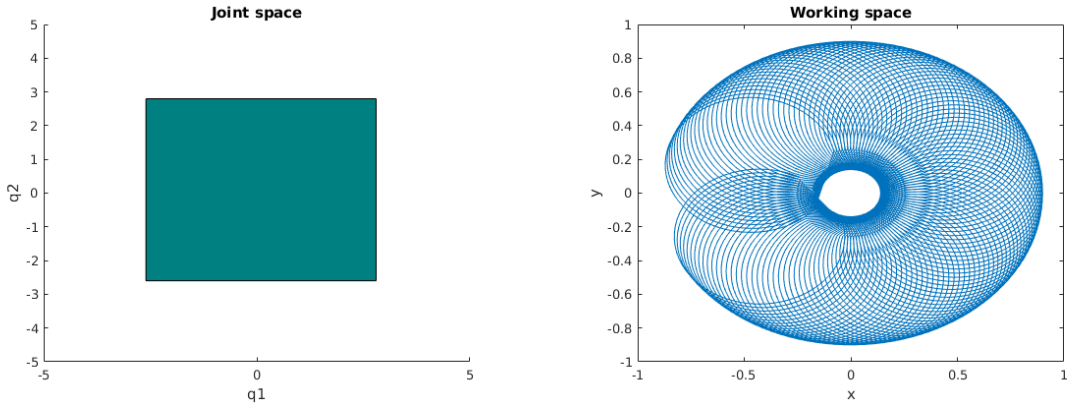

Figure 2: Joint space (left) and working space (right) of the 2R robot.

1.5. RX90 singularities

We used the MATLAB function ezplot to plot the symbolic expression of singularities ($R_{L_4} \cdot S_{23} - D_3 \cdot C_2 = 0$) and ($C_3 = 0$) of the RX90 for case $D_3 = R_{L_4} = 4.5$ (visible in fig 4 on the left) and for the case $D_3 = 0.55$, $R_{L_4} = 4.5$ (on the right).

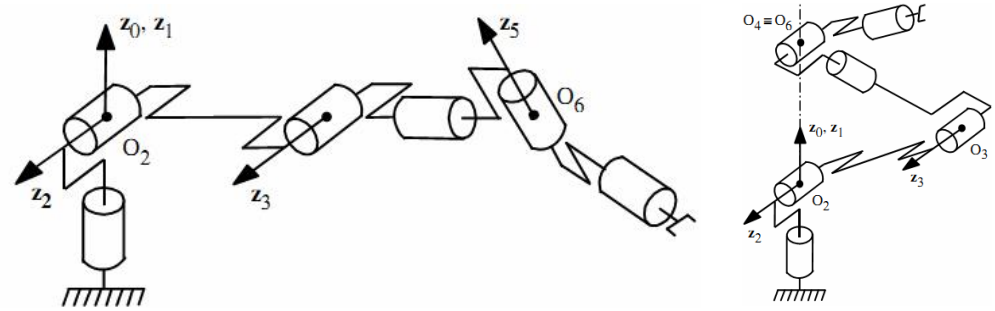

Figure 3: Elbow singularity(left) and Shoulder singularity (right)

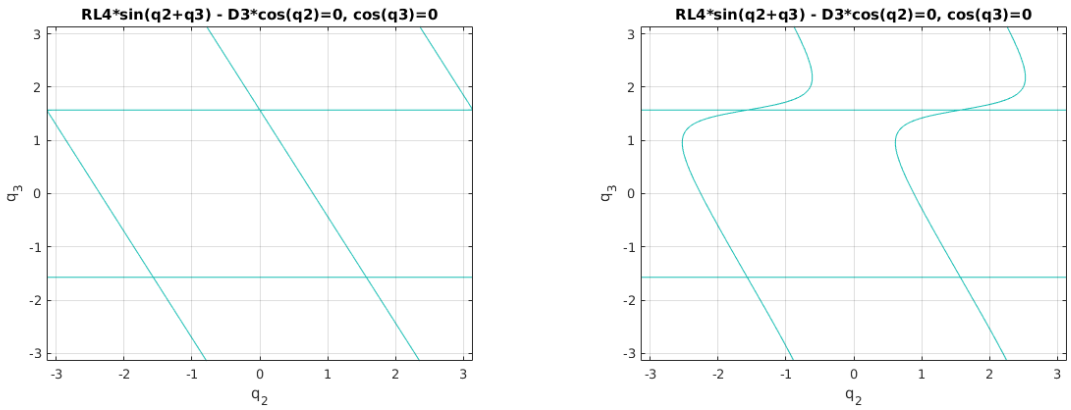

Figure 4: Symbolic expression of singularities for different R_{L_4} and D_3.

1.6. RX90 subspace configuration

Finally we computed the dimensions and orthogonal basis of the following subspaces for the RX90 robot at the given configuration $q_1 = \begin{bmatrix} 0.2 & -0.3 & -\pi/2 & 0.6 & 0 & 0.7 \end{bmatrix}^T$ using the Singular value decomposition theory and thanks to the matlab function svd that returns S which is a diagonal matrix, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that X = U*S*V'.

- 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.

The results obtained are the following:

$$S = \begin{bmatrix} 1.8760 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5290 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.2439 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5693 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1521 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0000 \end{bmatrix}$$

Null space base

$$N_j = \begin{bmatrix} 0 \\ -0.0000 \\ 0.0000 \\ 0.7071 \\ -0.0000 \\ -0.7071 \end{bmatrix}$$

Dimension of null space = 1.

Subspace generating Cartesian velocities base

$$B_v = \begin{bmatrix} 0.1423 & -0.6158 & 0.6948 & 0.3241 & -0.1127 \\ -0.6796 & -0.1425 & 0.1746 & -0.5118 & -0.4748 \\ -0.5686 & -0.1085 & 0.1140 & 0.0786 & 0.8036 \\ -0.0277 & 0.5385 & 0.4535 & 0.0571 & -0.0168 \\ -0.4394 & 0.0934 & -0.2496 & 0.7876 & -0.3400 \\ -0.0277 & 0.5385 & 0.4535 & 0.0571 & -0.0168 \end{bmatrix}$$

Dimension of the subspace generating Cartesian velocities = 5.

Subspace of the achievable Cartesian velocity base

$$R_j = \begin{bmatrix} -0.1469 & 0.0353 & -0.0469 & -0.3136 & 0.0014 \\ 0.0368 & -0.3462 & 0.4805 & 0.4358 & -0.6495 \\ -0.4418 & -0.1106 & 0.1601 & -0.7136 & -0.4128 \\ -0.1599 & 0.6269 & 0.7288 & 0.0374 & 0.2211 \\ 0.8437 & 0.2431 & 0.0800 & -0.3811 & -0.2783 \\ 0.2109 & -0.6439 & 0.4513 & -0.2363 & 0.5304 \end{bmatrix}$$

Dimension of the subspace of the achievable Cartesian velocity = 5.

Subspace of the Cartesian velocities that cannot be generated base

$$M_j = \begin{bmatrix} 0.9363 \\ 0.1898 \\ -0.2955 \\ -0.0000 \\ -0.0000 \\ 0.0000 \end{bmatrix}$$

Dimension of the subspace of the Cartesian velocities that cannot be generated = 1

2. 3R plan robot

In thi section we will analyze a 3R planar robot with the following geometric parameters and joint limits:
$L_1 = 0.6m$, $L_2 = 0.5m$, $L_3 = 0.3m$,
$q_{1,max} = q_{2,max} = 2.7rad$, $q_{1,min} = q_{2,min} = -2.5rad$, $q_{3,max} = 2.9rad$, and $q_{3,min} = -2.9rad$

2.1. Differential IGM

We developed a Matlab file IGMDIFF3R.m which is a function that takes as input the Cartesian coordinates of a 3Rplan robot, that can be expressed as $X = (P_x, P_y, \alpha)$ and it calculates the joints values through the inverse differential model. It is does several iteration in order to update joints values n_g with different initial values and it is returned the number of times qi is initialized with random value k. The number of iteration with specific initial value qi that are needed to reach the required minim error is also given in the value of counter m. In the case that no solution could be found it returns $err = 1$.
To make use of such model we implemented the following algorithm:

1. Initialize the first counter $k = 0$ (this counter will give the number of trials for the initial joint configuration).
2. $k = k + 1$, if $k = 500$, put $err = 1$ and stop the program.
3. pick up a random value for qc using the Matlab function rand1. Put $m = 0$, (this counter gives the number of iterations which have been intialized by this joint configuration).
4. Calculate the Cartesian coordinates X^c using the DGM of the 3R robot;
5. Calculate the difference dX between the desired X^d and the current X^c

   $$dX = X^{d} \breve{} X^c$$

   If $max(abs(dX)) < \epsilon$ (for $\epsilon < 10^{-10}$), put $q^d = q^c$, if the joint angles are within the joint domains. Put $err = 0$ and stop the program.
6. $m = m + 1$, if $m = 1000$ go back to the second step.
7. Update the value of q_c as follows:

- Calculate the Jacobian matrix $J = J3R(q^c)$, where $J3R$ is a function that compute the Jacobian matrix for the 3R planar robot.
- Calculate dq corresponding to dX using the pseudo-inverse;
- Set $q^c = q^c + dq$.

8. go back to the fourth step.

As visible in fig the algorithm has been tested for the given configuration:

$$\begin{aligned} q^1 &= \begin{bmatrix} 0.6 & 0.4 & 1.3 \end{bmatrix}^T \\ q^2 &= \begin{bmatrix} -0.5 & 0.9 & 0.7 \end{bmatrix}^T \\ q^3 &= \begin{bmatrix} 0.7 & 0.0 & 0.8 \end{bmatrix}^T \\ q^4 &= \begin{bmatrix} 2.9 & -0.4 & 0.8 \end{bmatrix}^T \end{aligned}$$

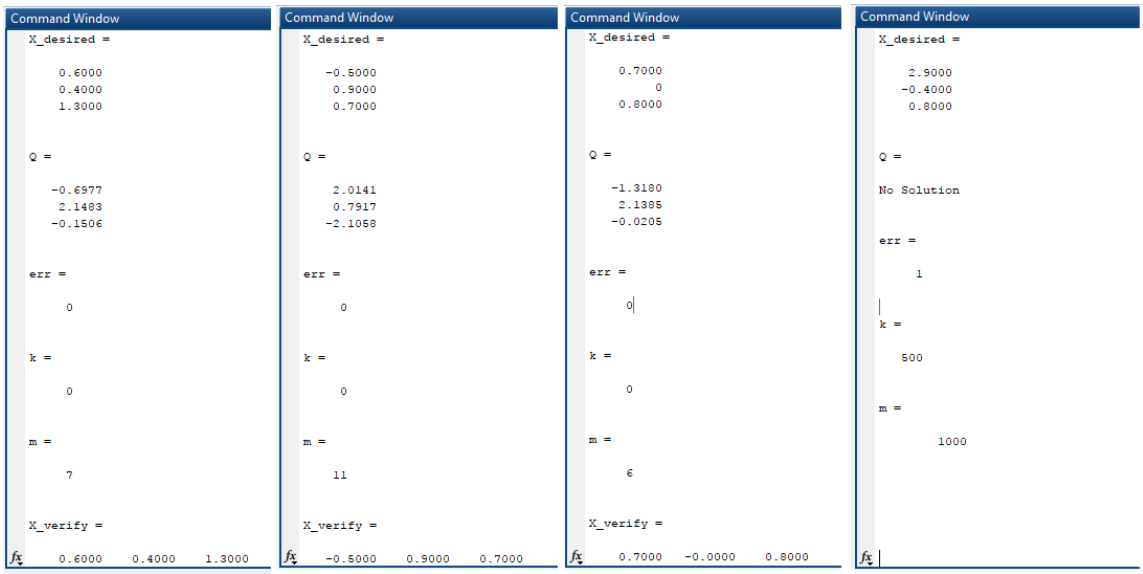

Figure 5: results of the script for different q.

2.2. Joint variable evolution plot

The following step required to plot the evolution of the joint variables and the Cartesian position variables (P_x, P_y) of the 3R robot while executing a straight line motion between two points (q^i and q^f).

$$\begin{aligned} q^i &= \begin{bmatrix} \pi/3 & -\pi/3 & \pi/3 \end{bmatrix}^T \\ q^f &= \begin{bmatrix} 3\pi/4 & \pi/2 & \pi/3 \end{bmatrix}^T \end{aligned}$$

Firstly we computed the Px, Py coordinates at the initial and the final value Xi and Xf then we divided the motion in 50 equally spaced intermediate points and we calculated the configuration of

q^i for each point. This calculation is done using the inverse differential model and by taking into account that the transformation from cartesian coordinates to joint coordinates will be done through a redundant Jacobian matrix (which is computed by the function $RJ3R$), which is a consequence of the fact that cartesian coordinates of the end effector can only be described as P_x, P_y (without α). Figure 6 shows the trajectory of the end effector in the (x, y) plane (on the left) and the respective variation in the joint variables with respect to every step (on the right).

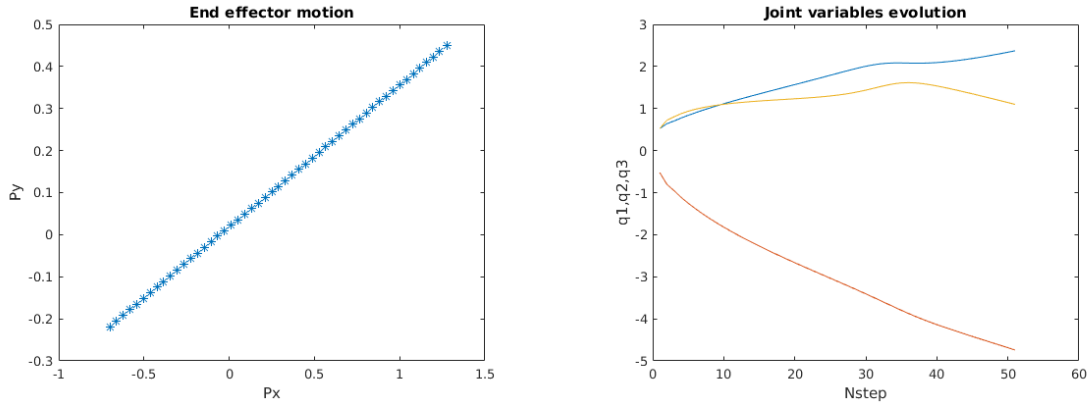

Figure 6: End effector position vs. Joint variables value for linear trajectory.

2.3. Optimized joint variable evolution plot

The same procedure has been then repeated using the pseudo inverse with optimization component to maintain the joint variables away from their limits q_{max} and q_{min}:

$$\begin{aligned} \dot{q} &= J^+ \cdot \dot{X} + (I_N - J^+ J) \cdot Z \\ \dot{q} &= J^+ \cdot \dot{X} + \alpha\,(I_N - J^+ J) \cdot \nabla\phi \quad \text{and,} \quad \phi(q) = \sum_{i=1}^{N} \left[\frac{q_i - q_{i,moy}}{\triangle\, q_i}\right]^2 \\ \nabla\phi &= \left[\frac{\partial\phi}{\partial q_1} \cdots \frac{\partial\phi}{\partial q_n}\right]^T \end{aligned}$$

where

$$\begin{aligned} q_{i,moy} &= \frac{1}{2}\,(q_{max} + q_{min}) \\ \triangle\, q_i &= q_{max} + q_{min} \end{aligned}$$

For each joint we calculate

$$Z_i = \frac{\alpha\partial\phi(q)}{\partial q} = \frac{2\alpha\partial(q_i - q_{i,moy})}{\partial q_i^2}$$

Finally

$$Z = (2 \cdot \alpha) \cdot \begin{bmatrix} \dfrac{q_1^c - q_{1,moy}}{q_{1,max} - q_{1,min}^2} \\ \dfrac{q_2^c - q_{2,moy}}{q_{2,max} - q_{2,min}^2} \\ \dfrac{q_3^c - q_{3,moy}}{q_{3,max} - q_{3,min}^2} \end{bmatrix}$$

By analyzing α we can say that:

- $\alpha < 0$ will minimize the function $\phi(q)$
- $\alpha > 0$ will maximize the function $\phi(q)$

For $\alpha = 0.3$, as visible in fig. 7 the algorithm tries to keep the variables away from their limit values.

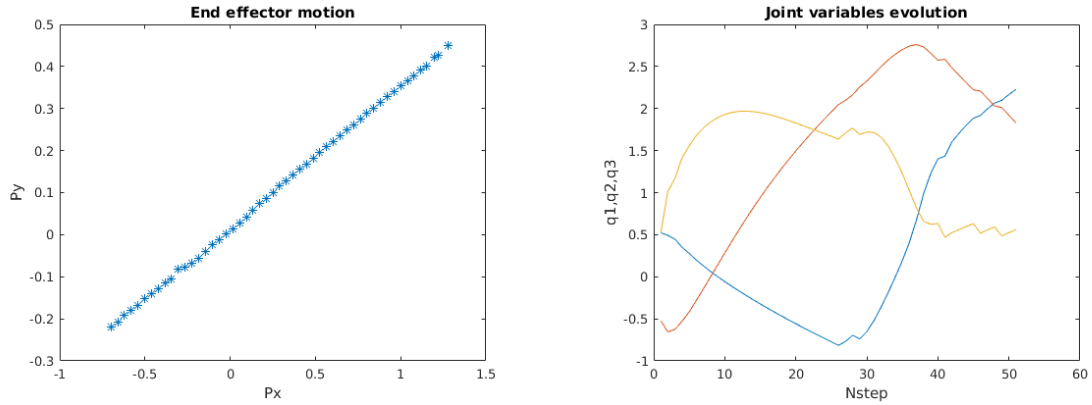

Figure 7: End effector position vs. Joint variables value for linear trajectory.

Conclusions

For this paper we scripted the Jacobian matrix for the specific case of the RX90 robot and for the general case as described in sections 1.1 and 1.2. We then defined the differential model for such robot and validated it with the use of the DGM with satisfactory results visible in section 1.3. Later, in section 1.5 and 1.6 we analyzed the RX90 singularities subspaces for a given configuration. In section 2.1 we modeled the differential IGM and finally, in section 2.2 and 2.3 we compared the motion of the end effector and the variation of the joint variables value using the differential model. From a comparison of the result of the two sections one can see the potentials of using a redundant robot (in this case it became redundant because we were not controlling the orientation anymore) in order to be able to make use of the pseudo inverse with optimization component to maintain the joint variables away from their limits.