Dynamics of Nonlinear Robotic Systems Assignment 2

Ali Jnadi

September 12, 2021

Abstract

This report is part of Dynamics of Nonlinear Robotic Systems course for 1st year master students at Innopolis University.

In this report I will calculate the Jacobian matrix for Niryo one manipulator that I have already made the DKM and IKM for it in the last assignment. I will also analyze the kinematics' singularities and how to use the Jacobian matrix to detect them. The implementation of all topics will be in Matlab and and the code will be uploaded to a GitHub repo.

1. Jacobian Matrix

Jacobian Matrix describe the relationship between each and every joint with the end effector, in other words it shows what is the effect of change in one joint to the end effector.
Figure 1. is describe the kinematic scheme for Niryo one manipulator

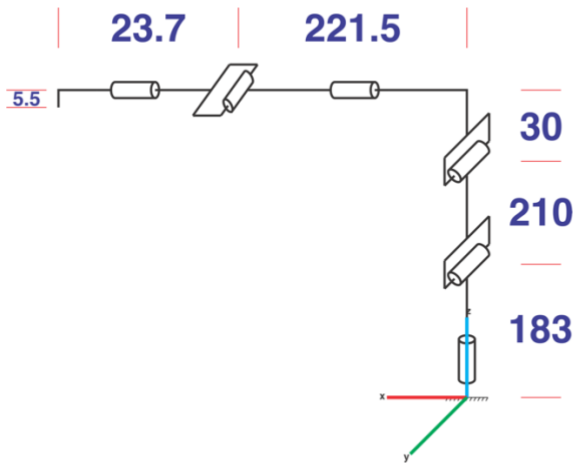


Figure 1 Kinematic scheme for Niryo one manipulator

To calculate the Jacobian matrix, we firstly need to calculate the forward kinematic:

$$T = T_z(l1)R_Z(\theta_1)R_y(\theta_2)T_z(l2)R_y(\theta_3)T_z(d)T_x(l3)R_x(\theta_4)R_y(\theta_5)R_x(\theta_6)T_x(l4)Tz(-l5) \quad (1)$$

We can use DKM function written in the previous assignment:

$$T = DKM(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, 0{,}0) \quad (2)$$

From this matrix we will obtain R and assign it to a zero position to get R$_{ee}$:

$$R_{ee} = \begin{pmatrix} R & (0{,}0{,}0)' \\ 0 & 1 \end{pmatrix} \quad (3)$$

Now for each joint we compute the following:

$$\dot{T_i} = T_{left} * \dot{H_i} * T_{right} * R_{ee}^{-1} \quad (4)$$

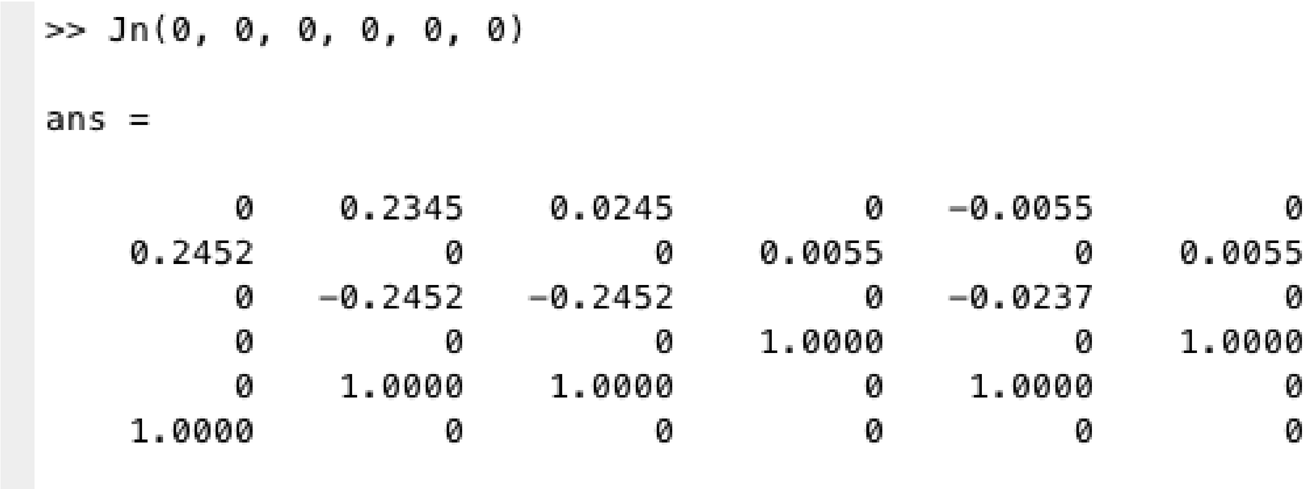
Where $\dot{H_i}$ is the derivative of the transformation that depends on the ith joint. From (4) we can calculate Jacobian as following:

$$J_i = \begin{bmatrix} \dot{H_i}(1{,}4) \\ \dot{H_i}(2{,}4) \\ \dot{H_i}(3{,}4) \\ \dot{H_i}(3{,}2) \\ \dot{H_i}(1{,}3) \\ \dot{H_i}(2{,}1) \end{bmatrix} \quad (5)$$

The following table shows: $\dot{H}$ for Rx, Ry, Rz:

DR$_x$	DR$_y$	DR$_z$
$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -S & -C & 0 \\ 0 & C & -S & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -S & 0 & C & 0 \\ 0 & 0 & 0 & 0 \\ -C & 0 & -S & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -S & -C & 0 & 0 \\ C & -S & 0 & 0 \\ 0 & C & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

Figure 2. illustrate implementation on Matlab for zero configuration.



```
>> Jn(0, 0, 0, 0, 0, 0)

ans =

         0    0.2345    0.0245         0   -0.0055         0
    0.2452         0         0    0.0055         0    0.0055
         0   -0.2452   -0.2452         0   -0.0237         0
         0         0         0    1.0000         0    1.0000
         0    1.0000    1.0000         0    1.0000         0
    1.0000         0         0         0         0         0
```

Figure 2 Jacobian for zero configuration.

There is another approach for calculating Jacobian matrix called numerical method, it depends on the definition of differentiation. The following two equations illustrate the implementation:

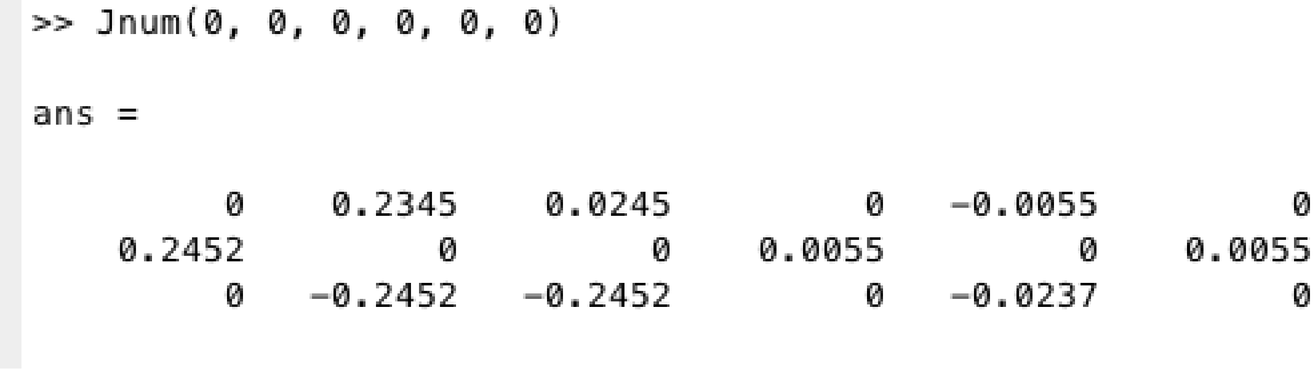
$$J_{temp} = \frac{DKM(\theta_1 + eps, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) - DKM(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, 0,0)}{eps} : eps < 10^{-3} \qquad (6)$$

$$J_1 = J_{temp}(1{:}3,4) \qquad (7)$$

After applying the previous for every joint, we can have the full Jacobian matrix for (x, y, z):

$$J = [J_1, J_2, J_3, J_4, J_5, J_6] \qquad (8)$$

I implement the previous in Matlab for the zero configuration.



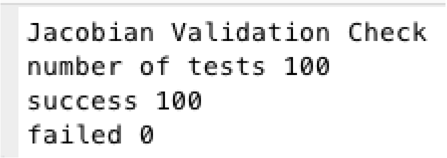
```
>> Jnum(0, 0, 0, 0, 0, 0)

ans =

         0    0.2345    0.0245         0   -0.0055         0
    0.2452         0         0    0.0055         0    0.0055
         0   -0.2452   -0.2452         0   -0.0237         0
```

Figure 3. Jacobian numerical method.

In the code I check the validation of the numerical method and I got 100%.



```
Jacobian Validation Check
number of tests 100
success 100
failed 0
```

Figure 4. Numerical approach validation.

2. Singularity Analysis

Singularity cases can be obtained from Jacobian matrix using three methods:

1. Checking the rank of the Jacobian matrix.
2. The diagonal matrix s from SVD has extremely small values.
3. The Jacobian matrix determinant is equal to zero.

Before implementation, it worth to illustrate that we have three types of singularity:

2.1.Shoulder Singularity.

The end-effector coincides with the rotation axis of the base joint, that mean any move of the first joint won’t affect (x, y) position of the end effector. The following example illustrate this situation:

$$\theta = \begin{bmatrix} 3.1416 \\ -1.2269 \\ 1.05 \\ 3.1416 \\ 1.8575 \\ 0 \end{bmatrix}$$

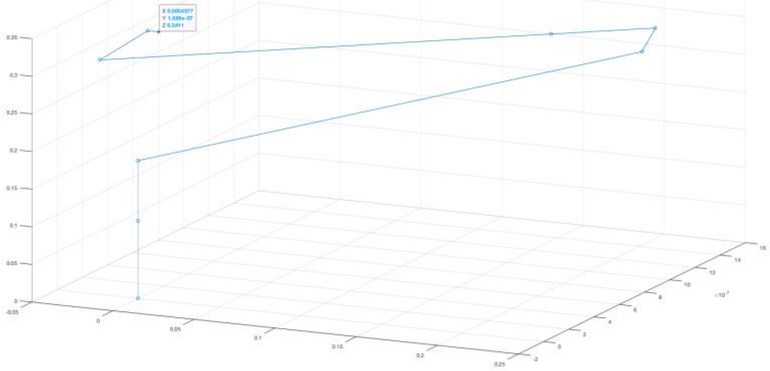


Figure 5. Shoulder Singularity.

2.2.Elbow Singularity.

The manipulator is fully extended, that lead to lost control on several axis and directions. For example, the robot can move on the Z axis neither the X nor the Y axis:

$$\theta = \begin{bmatrix} 0 \\ \frac{\pi}{2} \\ -\frac{\pi}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

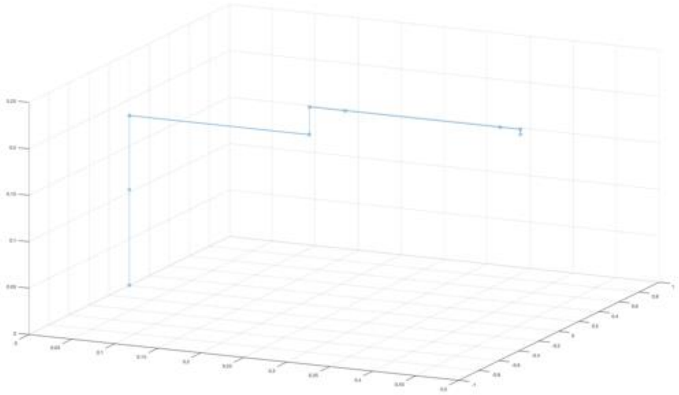


Figure 6. Elbow Singularity.

2.3.Wrist Singularity.

The axis of joint 4 and joint 6 are colinear, the robot will lose 1DOF. In the IKM we see that there are infinite solutions for the wrist, beside that if j4 and j6 move in opposite directions and same value the end effector won't move, that's why it is singularity:

$$\theta = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

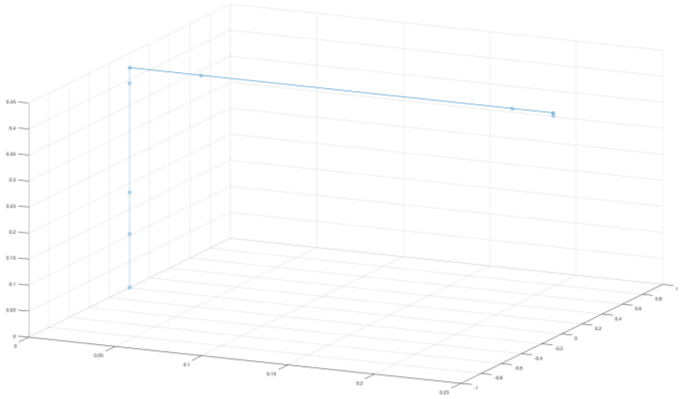


Figure 7. Wrist Singularity.

In Matlab code I have checked the singularity with rank function and SVD function in the second case I added small value to joint 5 to avoid wrist singularity results.

3. GitHub

The project can be found here