Manipulater Robots

Integrated Master Degree in Mechanical Engineering

Scientific Area of Control, Automation, and Industrial Informatics

UR3 Kinematics

Author:
Luís Miranda, 81089
Diogo Catarino, 81772

Instructor:
Prof. Jorge Martins

April 24, 2018

Contents

1 Introduction

UR3 is a 6-degree-of-freedom (DoF) robotic manipulator manufactured by Universal Robots Company. The workspace of the UR3 robot extends 500 mm from the base joint The robot can be programmed to move a tool, and communicate with other machines using electrical signals.

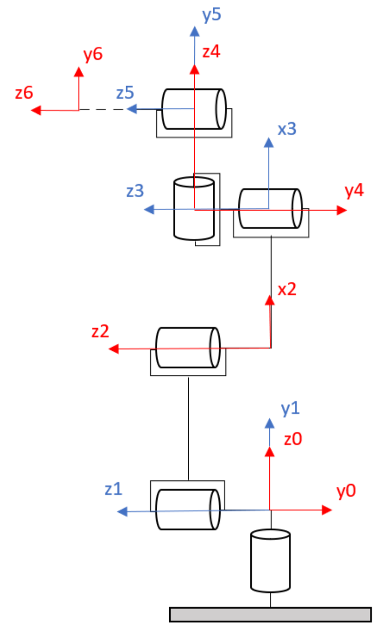
In this work the following tasks will be performed: the robot arm kinematic model will be built according to the Denavit-Hartenberg convention. A Simulink model for the direct kinematics and inverse kinematics of the robot will be created and validated. The geometric Jacobian of the robot will also be created to show the effect of the arm and wrist kinematic singularities on the rank of the Jacobian matrix. Finally, a Simulink model for the closed loop inverse kinematics will be implemented, using both the inverse and transpose geometric jacobian, and the results will be compared with the previous simulink model of the inverse kinematics.

2 Kinematics

2.1 Kinematics model

The official manual of this robotic manipulator and the figure 1 were used in order to build the kinematics model. The kinematics model was based on the Denavit-Hatenberg convention. The direct kinematic problem will be solved with the help of the table 1 that was build with previously placed joint frames.

(a)



(b)

Figure 1: Real Robotic manipulator and Kinematics model

$Link_i$	d_i	ϑ_i	a_i	α_i	Offset
1	0	ϑ_1	0	$\pi/2$	0
2	0	ϑ_2	a_2	0	$\pi/2$
3	0	ϑ_3	0	0	0
4	d_4	ϑ_4	0	$-\pi/2$	$-\pi/2$
5	d_5	ϑ_5	0	$\pi/2$	0
6	d_6	ϑ_6	0	0	0

Table 1: Denavit-Hatenberg parameters

2.2 Direct Kinematics

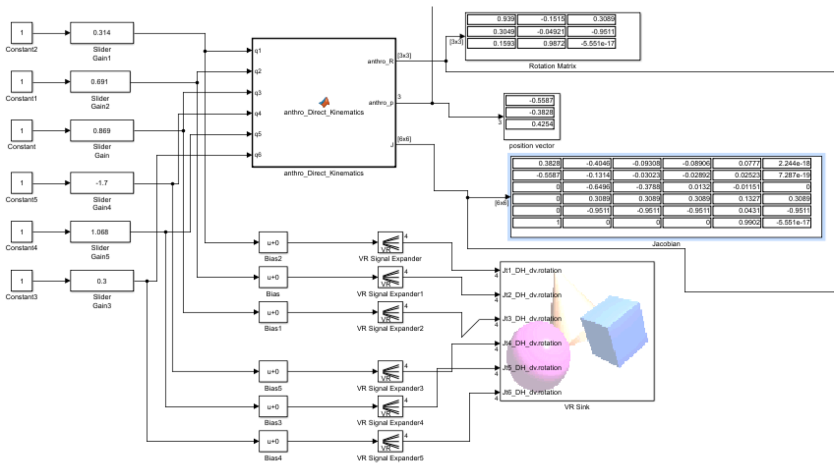
Solving the direct dynamics problem is useful for manipulator simulation. The aim of direct kinematics is to compute the pose of the end-effector as a function of the joint variables. The Denavit–Hartenberg convention allows the construction of the direct kinematics function by postmultiplication of the individual transformation matrices into one general homogeneous transformation matrix.

$$A_i^{i-1}(q_i) = \begin{bmatrix} C_{\vartheta_i} & -S_{\vartheta_i}C_{\alpha_i} & S_{\vartheta_i}S_{\alpha_i} & a_iC_{\vartheta_i} \\ S_{\vartheta_i} & C_{\vartheta_i}C_{\alpha_i} & -C_{\vartheta_i}S_{\alpha_i} & a_iS_{\vartheta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$T_6^0 = A_1^0 \cdot A_2^1 \cdot A_3^2 \cdot A_4^3 \cdot A_5^4 \cdot A_6^5 = \begin{bmatrix} n_{6\,x}^0 & s_{6\,x}^0 & a_{6\,x}^0 & p_{6\,x}^0 \\ n_{6\,y}^0 & s_{6\,y}^0 & a_{6\,y}^0 & p_{6\,y}^0 \\ n_{6\,z}^0 & s_{6\,z}^0 & a_{6\,z}^0 & p_{6\,z}^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

In the matrix 2, the end rotation matrix (R_{end}) is composed by the vectors n_6^0, s_6^0 and a_6^0 while the end position matrix (P_{end}) is composed of the vector p_6^0.

A Simulink model of the direct kinematics was created, as seen in figure 2 to observe the end position vector and Euler Angles' vector $\phi = \begin{bmatrix}\varphi & \vartheta & \psi\end{bmatrix}^T$ in 3 different *poses*. The vector ϕ is defined as a a sequence of Roll-Pitch-Yaw ZYX. The ordered sequence of rotations XYZ about axes of the fixed frame is equivalent to the sequence ZYX about axes of the current frame.



Rotation Matrix

0.939	-0.1515	0.3089
0.3049	-0.04921	-0.9511
0.1593	0.9872	-5.551e-17

position vector

-0.5587
-0.3828
0.4254

Jacobian

0.3828	-0.4046	-0.09308	-0.08906	0.0777	2.244e-18
-0.5587	-0.1314	-0.03023	-0.02892	0.02523	7.287e-19
0	-0.6496	-0.3788	0.0132	-0.01151	0
0	0.3089	0.3089	0.3089	0.1327	0.3089
0	-0.9511	-0.9511	-0.9511	0.0431	-0.9511
1	0	0	0	0.9902	-5.551e-17

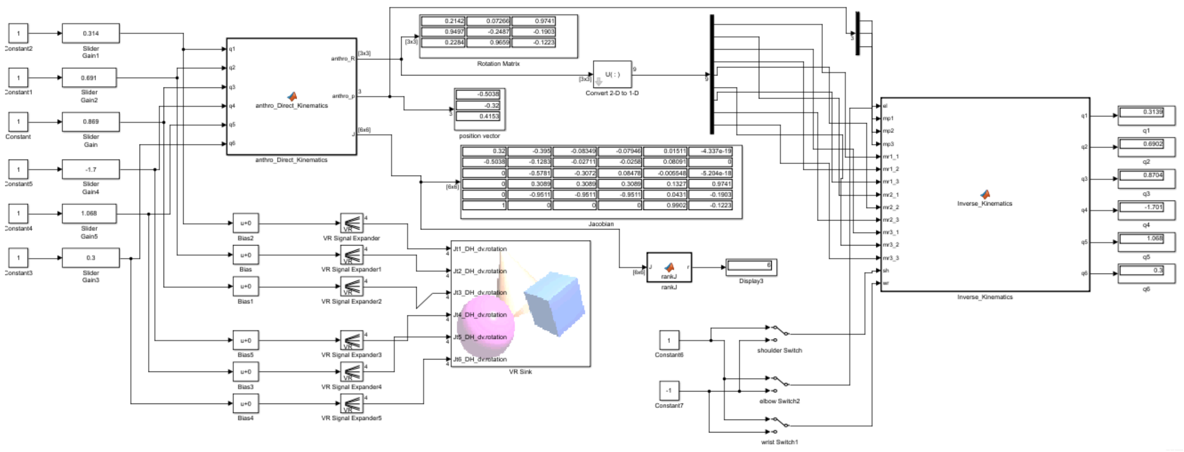
Figure 2: Simulink model for the direct kinematics

2.3 Inverse Kinematics

Solving the inverse kinematics problem is useful for manipulator trajectory planning and control algorithm implementation. The inverse kinematics problem consists on the determination of the joint variables corresponding to a given end-effector position and orientation. In order to solve this problem we started be decoupling the robot arm, computing first the q_1, then the q_5 and q_6, and finally from the q_2 to q_4.

The q_1 is an independent variable so we started by calculating this value from the reference frame T_5^1 in the position ${p_5^1}_z$. There are two possible solutions for this representing the shoulder up and down positions. With this value we calculate q_5 and after q_6 because it depends on the value of q_5. q_5 which also has two positions (wrist up and down) is obtained from the position ${a_6^1}_z$ of the reference frame T_6^1 and q_6 is obtained from the positions ${n_6^1}_z$ and ${s_6^1}_z$ of the same reference frame. Due to the fact q_2, q_3 and q_4 are planar, we used the 2D solution of the three-link planar arm learned in the class, in respect to the reference frame T_4^1. As learned in the class, there are two possible solutions (elbow up and down) for the q_3.

In order to validate this model, the calculations were implemented in simulink and validated against the direct kinematics model as seen in figure 3. By feeding the inverse kinematics calculations with the results of the direct kinematics, as well as the desired position for the shoulder, elbow and wrist, the values for the joint angles obtained were equal to the input, validating the model.



0.2142	0.07266	0.9741
0.9497	-0.2487	-0.1903
0.2284	0.9659	-0.1223

-0.5038
-0.32
0.4153

0.32	-0.395	-0.08349	-0.07946	0.01511	-4.337e-19
-0.5038	-0.1283	-0.02711	-0.0258	0.08091	0
0	-0.5781	-0.3072	0.08478	-0.005548	-5.204e-18
0	0.3089	0.3089	0.3089	0.1327	0.9741
0	-0.9511	-0.9511	-0.9511	0.0431	-0.1903
1	0	0	0	0.9902	-0.1223

Figure 3: Simulink model for the inverse kinematics

2.4 Diferencial Kinematics

Resorting to the direct kinematics of the robot we can calculate the Jacobian of the UR3. Given that the robot is composed by 6 revolute joints, the Jacobian will be a 6x6 matrix of the form:

$$J = \begin{bmatrix} z_0 \times p_6^0 & z_0 \times p_6^1 & z_0 \times p_6^2 & z_0 \times p_6^3 & z_0 \times p_6^4 & z_0 \times p_6^5 \\ z_0 & z_1 & z_2 & z_3 & z_4 & z_5 \end{bmatrix} \tag{3}$$

When the robot reaches a kinematic singularity it is expected that the rank of the Jacobian drops. In order to evaluate this phenomenon the MATLAB function rank was used and the manipulator was placed in the arm and wrist singularity positions.

For the wrist we have that for $\sin q_5 = 0$, which corresponds to the alignment of z_6 with the 3 link planar arm, the rank is 5. For the arm we have that for $\sin q_3 = 0$, which corresponds to the planar arm stretched, the rank is also 5. Other interesting values are that for the initial position (figure 1) the rank is 3 and for $q_2, q_3, q_4 = 0$ the rank is 4. This results can be seen in the appendix E.

2.5 Closed-Loop Inverse Kinematics

Problems like singularities and complexity of the manipulator structure, results in nonlinear relationships between the *workspace* and *jointspace* making it impossible to the use of inverse kinematics. The solution of this problem lies in the concept of implementing a closed-loop algorithm that solve both the relationship between end-effector speeds and jointspace speeds, as well as the Direct kinematics. This algorithm is called Closed-Loop Inverse Kinematics (CLIK).

A CLIK algorithm can be built in order to overcome the numerical discrepancy/deviation concerned with discrete-time implementation. This algorithm uses the error characterizing the displacement between the desired and the current end-effector trajectory to calculate the revolute joint angles.

$$[\dot{q}]_{(6\times1)} = J^{-1}_{(6\times6)} \begin{bmatrix} \dot{p}_{e(3\times1)} \\ \omega_{e(3\times1)} \end{bmatrix} \tag{4}$$

With the equation 4 it is possible to relate the end-effector and the joints velocities. When the system is over a singularity, the Jacobian matrix rank is reduced and becomes non-invertible. In order to solve this problem we used two different functions when implementing the Simulink model, the pseudo-inverse and the transpose of the Jacobian matrix.

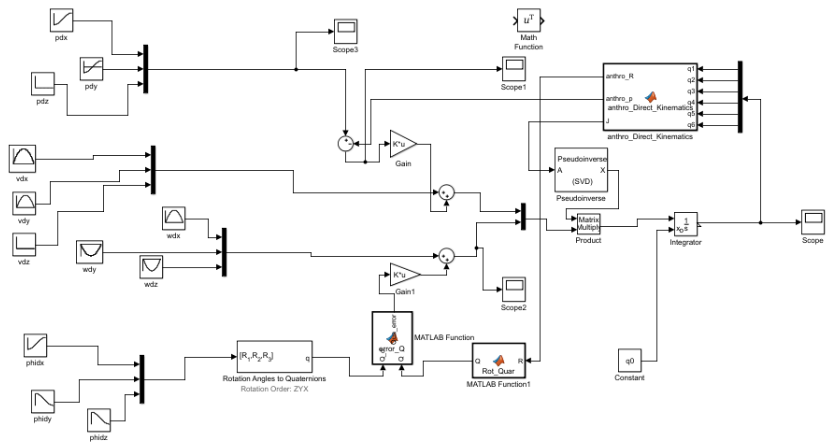


Figure 4: Simulink model for the closed-loop inverse kinematics

2.5.1 Trajectory Planning

In order to validate our results, two trajectories were tested, one rectilinear and one circular, using both geometric Jacobian functions, pseudo-inverse and transpose. The goal is to see the effect of the arm and wrist kinematic singularities and the difference in performance of the inverse and transpose Jacobian. The results will be compared with the previous Simulink model of the inverse kinematics.

Bellow we can see the calculated angles and errors in position when following a rectilinear trajectory for different gains.

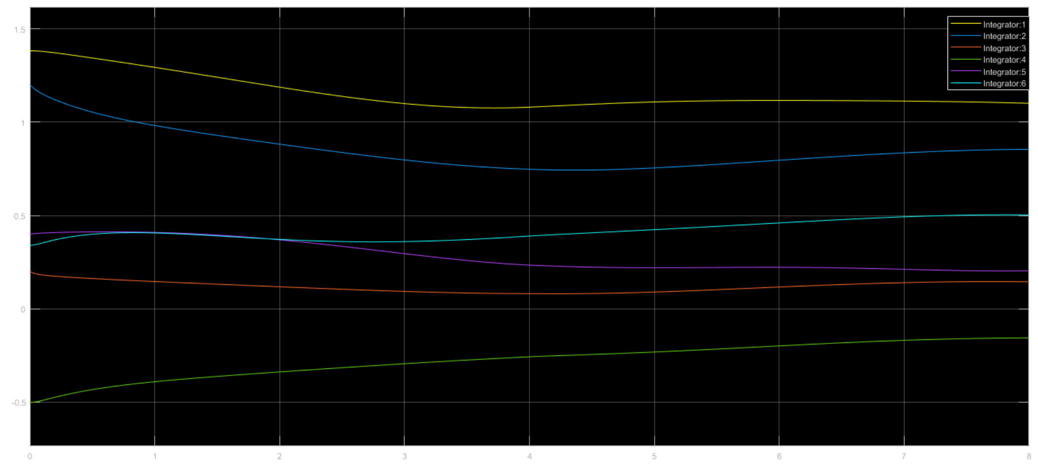


Figure 5: Click using transpose Jacobian response to rectilinear trajectory K=10

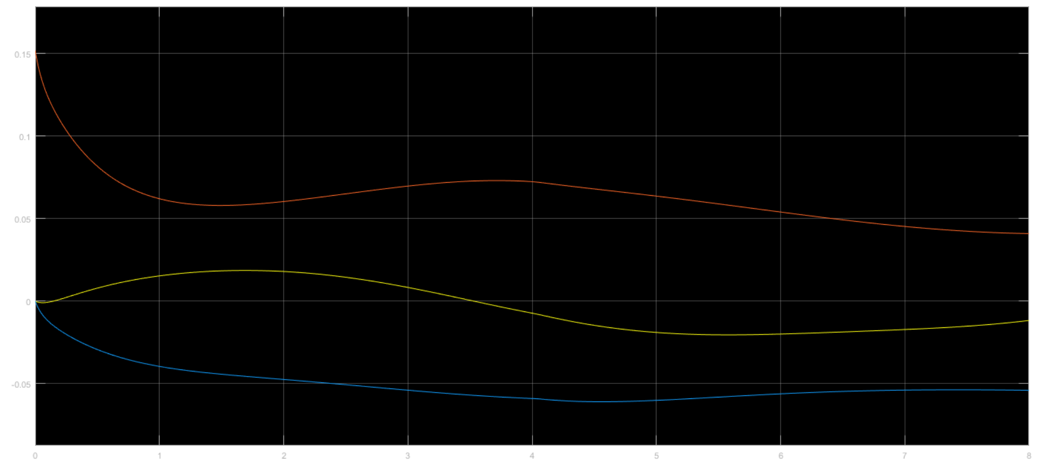


Figure 6: Click using transpose Jacobian response to rectilinear trajectory K=10 - error

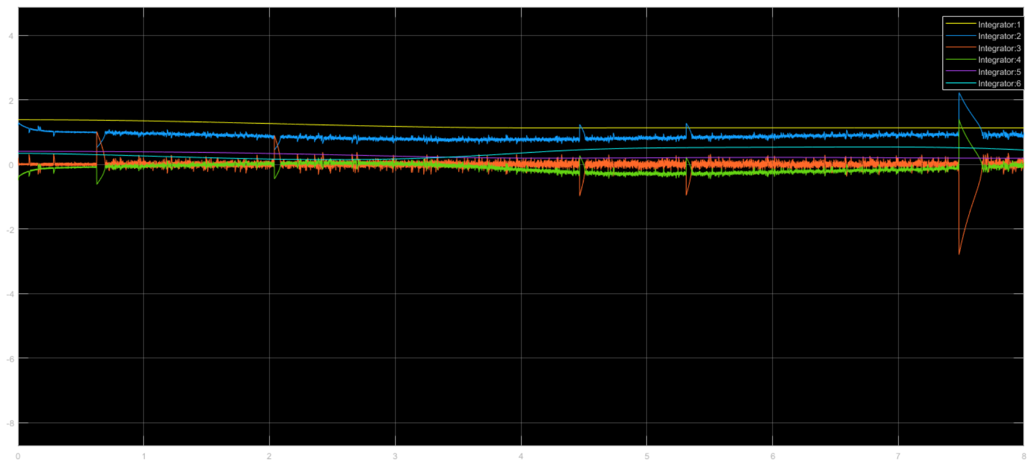


Figure 7: Click using inversed Jacobian response to rectilinear trajectory K=10

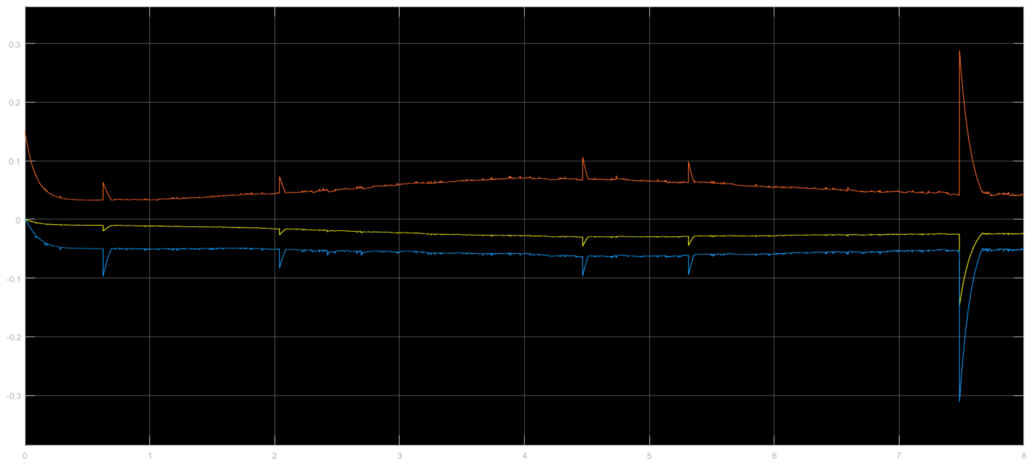


Figure 8: Click using inversed Jacobian response to rectilinear trajectory K=10 - error

As we can see from the graphics although the use of the inverse Jacobian provides less position errors, its computational time is very high and has problems with trajectories that pass through singularities (which is the case). The use of the transpose does not require too much computational time but it has more errors in terms of the desired position.

The trajectories used as well as more responses to circular and rectilinear trajectories can be seen in appendix D.

3 Conclusion

As this report demonstrates, the kinematic model was successfully built according to the Denavit-Hartenberg convention and the Simulinks for direct kinematics, inverse kinematics and closed-loop inverse kinematics were successfully implemented. The validations of the Simulink models were also achieved.

Regarding the Jacobian, the inverse of the Jacobian requires more processing computational power and cannot be used while on singularities, taking less time for the results and the error to converge. In the other hand, the transpose of Jacobian requires less processing computational power and can be used while on singularities, taking much more time for the results and the error to converge.

References

[1] $www.universal-robots.com$

[2] $www.universal{-}robots.com/media/207442/ur3usermanualenglobal.pdf$

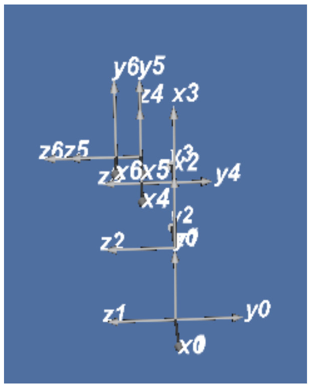
[3] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo (2009) *Robotics: Modelling, Planning and Control*. Springer-Verlag London, 1st edition.

[4] S. Chiaverini, B. Siciliano (1999) *The Unit Quaternion: A Useful Tool for Inverse Kinematics of Robot Manipulators.*

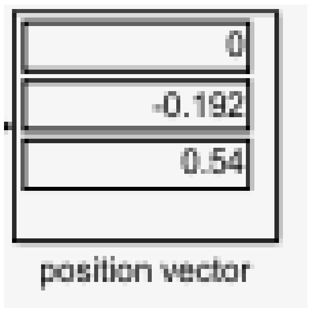
[5] K. P. Hawkins (2013) *Analytic Inverse Kinematics for the Universal Robots*

A Direct Kinematics Validation

To validate the direct kinematics model we placed the manipulator in three distinct positions which have known coordinates for the end factor.
The first position corresponds to the robotic arm completely stretched and is given by equaling all joint coordinates to 0.



(a)



0

-0.192

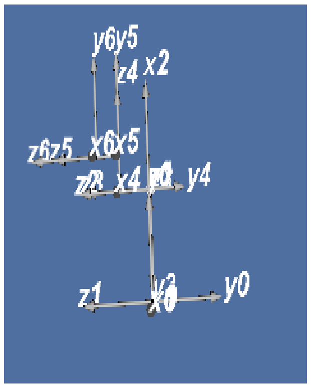
0.54

position vector

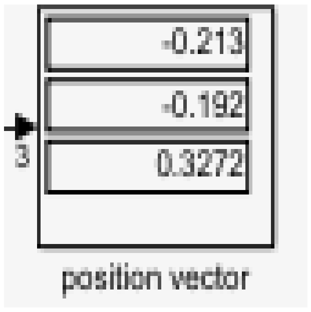
(b)

Figure 9: Stretched Position

The second position was the rotating the third joint and the fourth joint by 90^o in opposite directions.



(a)



-0.213

-0.192

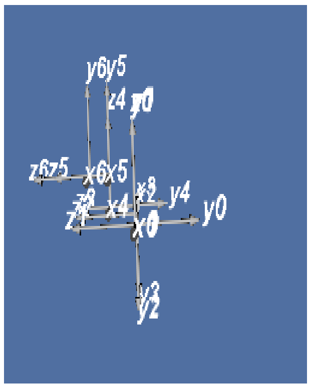
0.3272

position vector

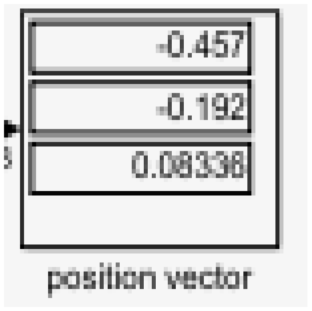
(b)

Figure 10: Bench position

For the last one we gave the joints 2 and 4 also 90^o rotation in opposite directions.



(a)



(b)

Figure 11: Lying position

All three positions are accurate according to the dimensions off the robot seen bellow. For better comprehension we suggest to run simulink UR3_Kinematics.slx and input the angles described.

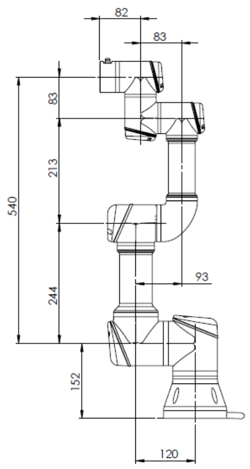


Figure 12: Dimensions of the UR3

B Inverse Kinematics Validation

To validate the inverse kinematics model we calculated a position (figure 14) from the direct kinematics model for a set of initial angles (figure 13). We also calculated from the inverse kinematics model a second set of angles (figure 15) (for elbow up and down) from the position calculated previously. The second set of angles were used for the calculation of a position (figure 16) with the direct kinematics model. Comparing the first position calculated with the last one we achieved the same result. This way we guarantee the validation of inverse kinematics model.

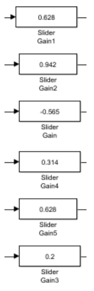
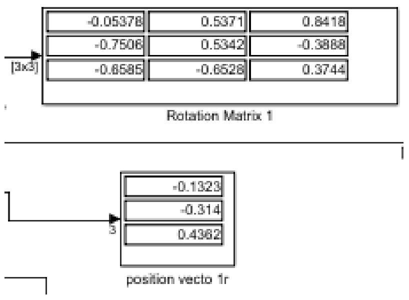


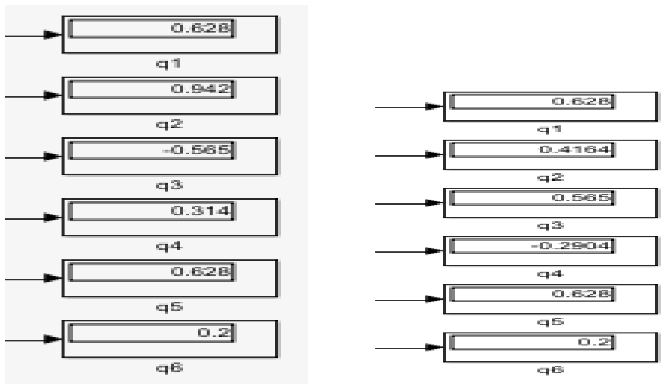
Figure 13: Initial set of angles



```
-0.05378   0.5371   0.8418
-0.7506    0.5342  -0.3868
-0.6585   -0.6528   0.3744
```

```
-0.1323
-0.314
0.4362
```

Figure 14: Inicial position from the direct kinematics



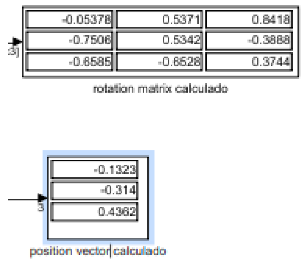
```
0.628
q1
0.942
q2
-0.565
q3
0.314
q4
0.628
q5
0.2
q6
```

(a)

```
0.628
q1
0.4164
q2
0.565
q3
-0.2904
q4
0.628
q5
0.2
q6
```

(b)

Figure 15: Angles from the inverse kinematics model for elbow up and down



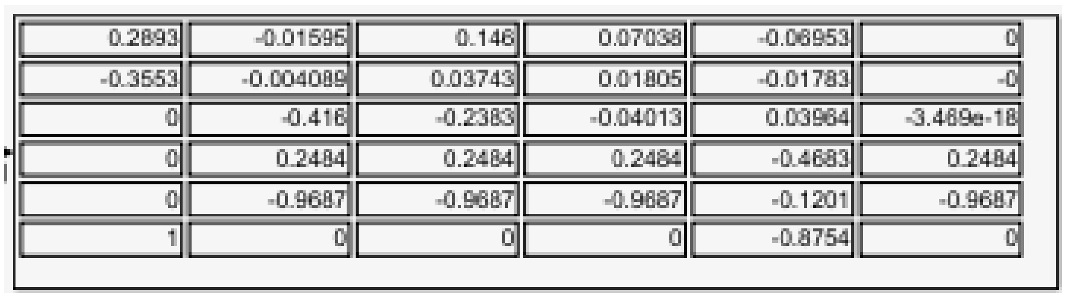
```
-0.05378    0.5371    0.8418
 -0.7506    0.5342   -0.3888
 -0.6585   -0.6528    0.3744
rotation matrix calculado
```

```
-0.1323
 -0.314
 0.4362
position vector calculado
```

Figure 16: Last position after the inverse and direct kinematics

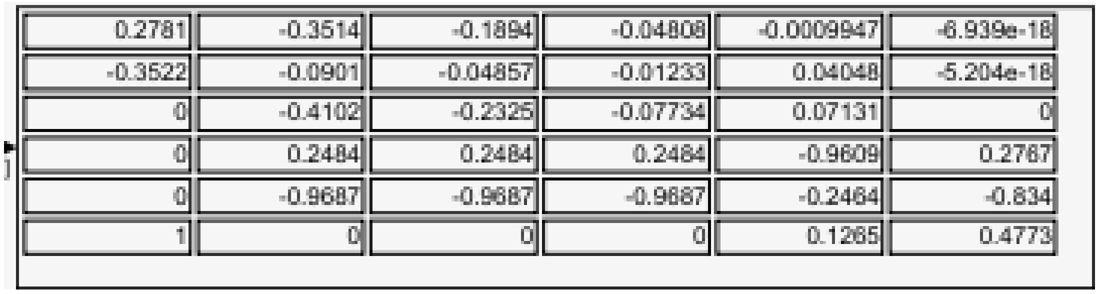
C Singularities and its effects on the Jacobian rank

Here are presented the Jacobian matrices for the cases, described in the Section 2.4, when the rank is decreased.



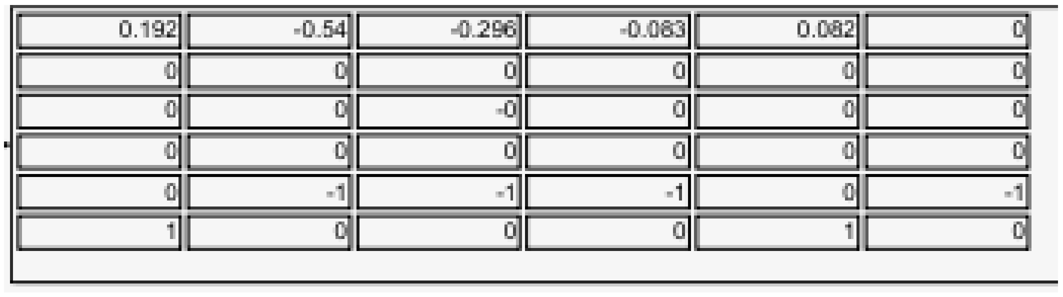
0.2893	-0.01595	0.146	0.07038	-0.06953	0
-0.3553	-0.004089	0.03743	0.01805	-0.01783	-0
0	-0.416	-0.2383	-0.04013	0.03964	-3.469e-18
0	0.2484	0.2484	0.2484	-0.4683	0.2484
0	-0.9687	-0.9687	-0.9687	-0.1201	-0.9687
1	0	0	0	-0.8754	0

Figure 17: Jacobian matrix when the q5=0



0.2781	-0.3514	-0.1894	-0.04808	-0.0009947	-6.939e-18
-0.3522	-0.0901	-0.04857	-0.01233	0.04048	-5.204e-18
0	-0.4102	-0.2325	-0.07734	0.07131	0
0	0.2484	0.2484	0.2484	-0.9609	0.2767
0	-0.9687	-0.9687	-0.9687	-0.2464	-0.834
1	0	0	0	0.1265	0.4773

Figure 18: Jacobian matrix when the q3=0



0.192	-0.54	-0.296	-0.083	0.082	0
0	0	0	0	0	0
0	0	-0	0	0	0
0	0	0	0	0	0
0	-1	-1	-1	0	-1
1	0	0	0	1	0

Figure 19: Jacobian matrix when the all the q's=0

D Click Responses

To compute the trajectories run MATLAB scripts circular_trajectoryP_dyn and linear_trajectoryP.

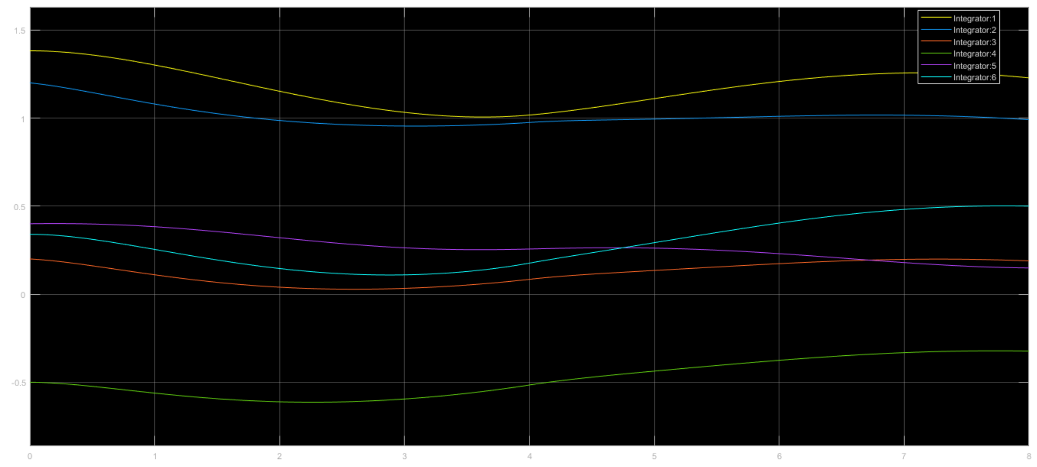


Figure 20: Click for transpose Jacobian response to rectilinear trajectory K=1

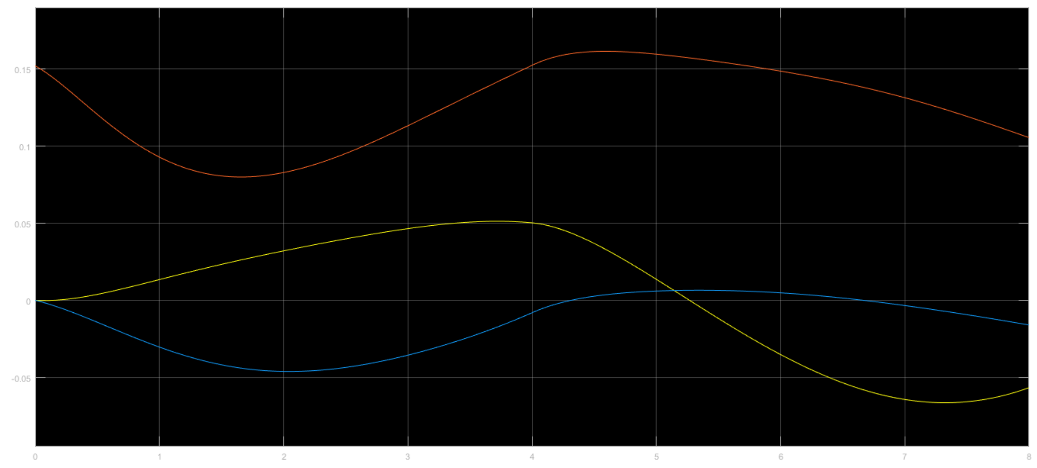


Figure 21: Click for transpose Jacobian response to rectilinear trajectory K=1 - erro

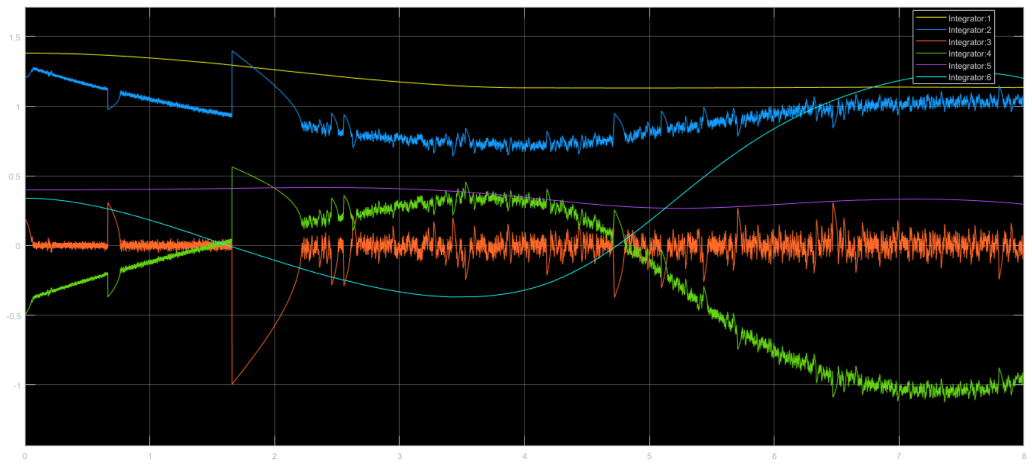


Figure 22: Click for inverse Jacobian response to rectilinear trajectory K=1

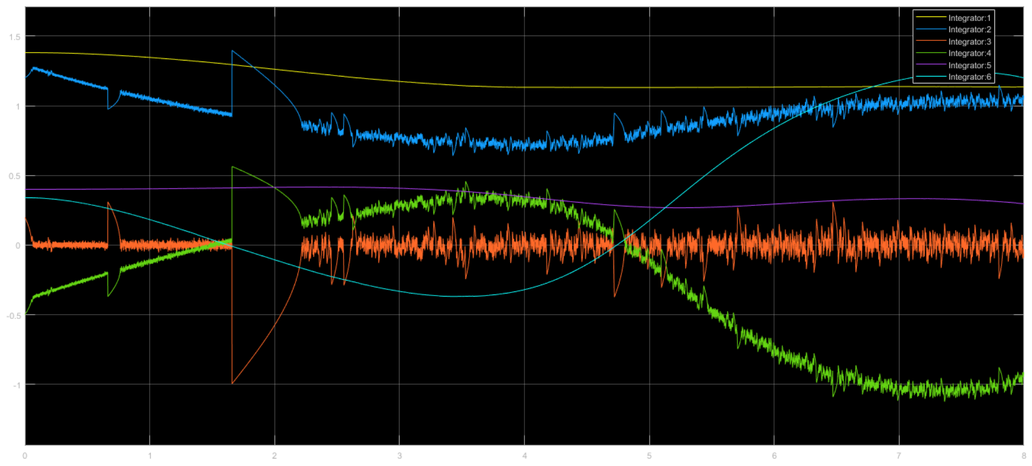


Figure 23: Click for inverse Jacobian response to rectilinear trajectory K=1 - error

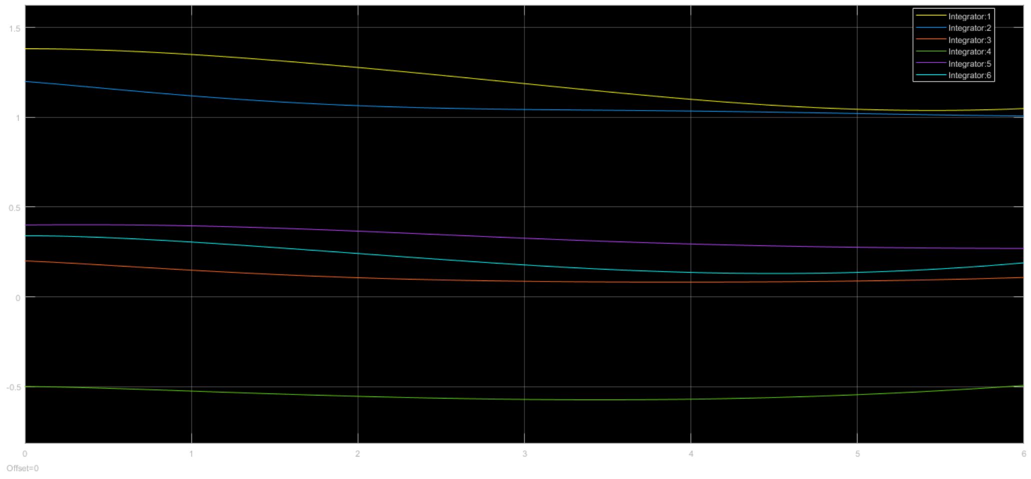


Figure 24: Click for inverse Jacobian response to circular trajectory K=1

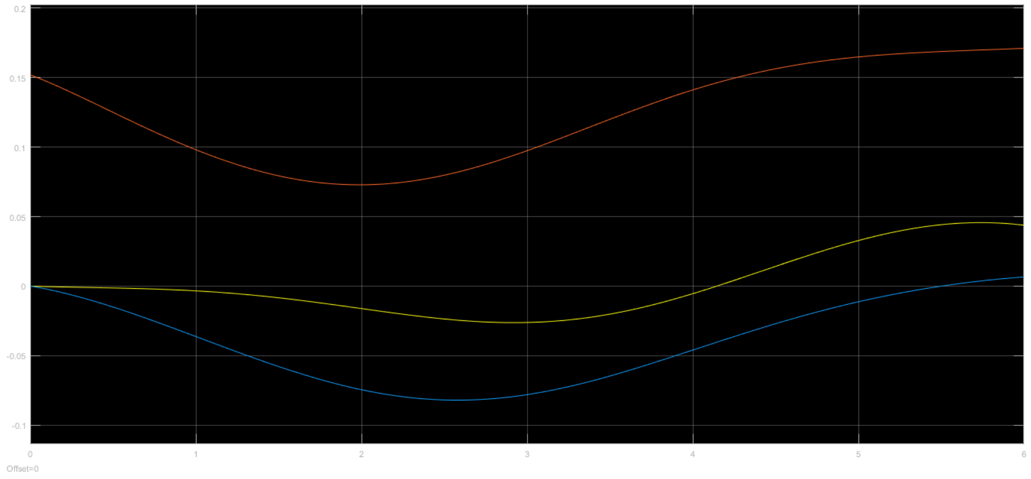


Figure 25: Click for transpose Jacobian response to circular trajectory K=1 - error

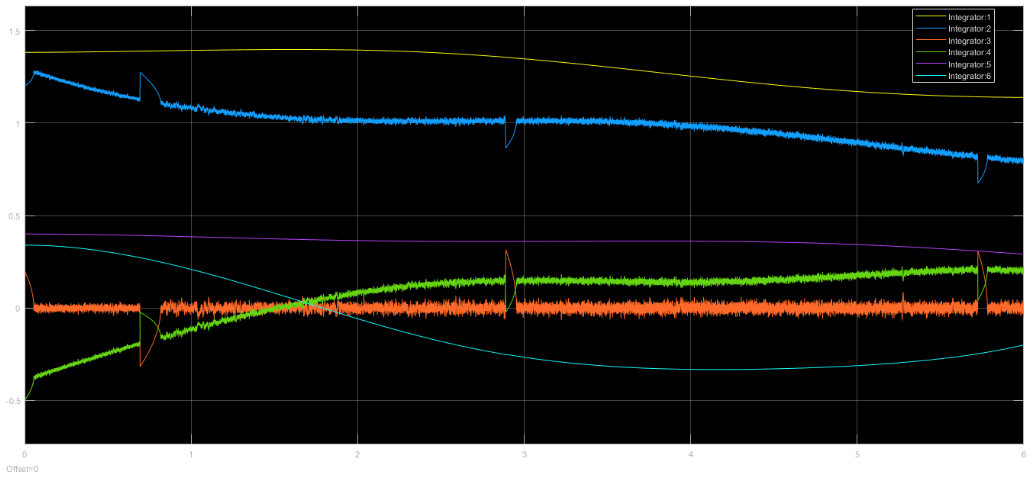


Figure 26: Click for inverse Jacobian response to circular trajectory K=1

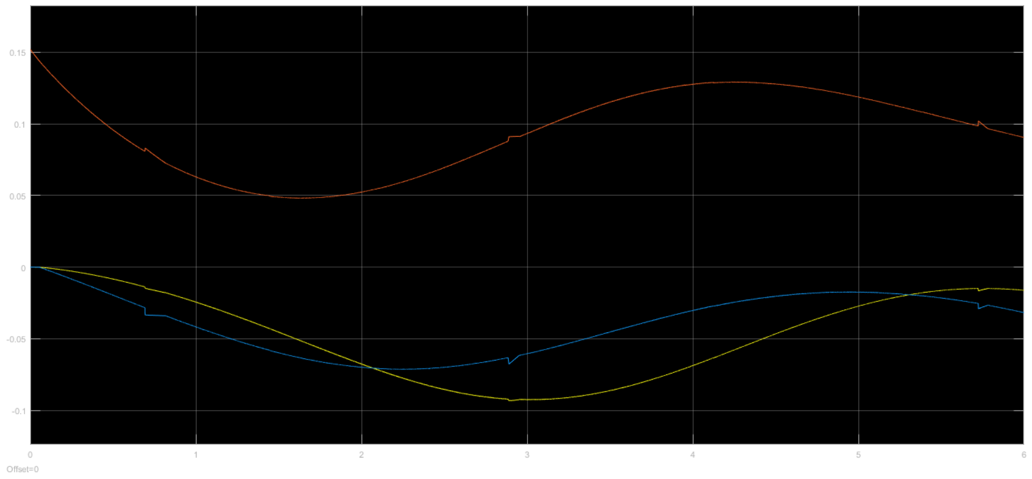


Figure 27: Click for inverse Jacobian response to circular trajectory K=1 - error