Assignment II – CSC376 – Fall 2019

Fundamentals of Robot Design
Department of Mathematical and Computational Sciences
University of Toronto Mississauga

Abstract

This assignment covers various topics associated with forward and velocity kinematics (Module 4 & 5). It consists of three parts. In the first part, the forward kinematics of a serial manipulator have to be derived and implemented. In the second part, the velocity kinematics of the serial manipulator have to be determined. In the third part, different robot configurations are evaluated.

Remember to write your full names and student numbers prominently on your submission. To avoid suspicions of plagiarism: at the beginning of your submission, clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, that you consulted.

For Assignment II and Assignment III you are allowed to work in groups of 2 students or by yourself. Please note that the group cannot be changes between those two assignments. For assignment handling, group forming, and submission use MarkUs:

`https://mcsmark.utm.utoronto.ca/csc376f19`.

Please upload your submission as a single `.pdf` file alongside with an updated git-repository containing your code on MarkUs. By uploading, you confirm that this is your original work without using any references other than the listed ones.

Answer each question completely, always justifying your claims and reasoning and explaining your calculation/derivation. Furthermore, use meaningful and explanatory comments to go along your submitted code. Your solution will be graded not only on correctness, but also on clarity. Answers that are technically correct that are hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets].

Dean W[redacted]	Roman H[redacted]	11/18/2019
Name 1	Name 2 (if applicable)	Date

Due Date: Nov 8, 2019 (11:59pm)

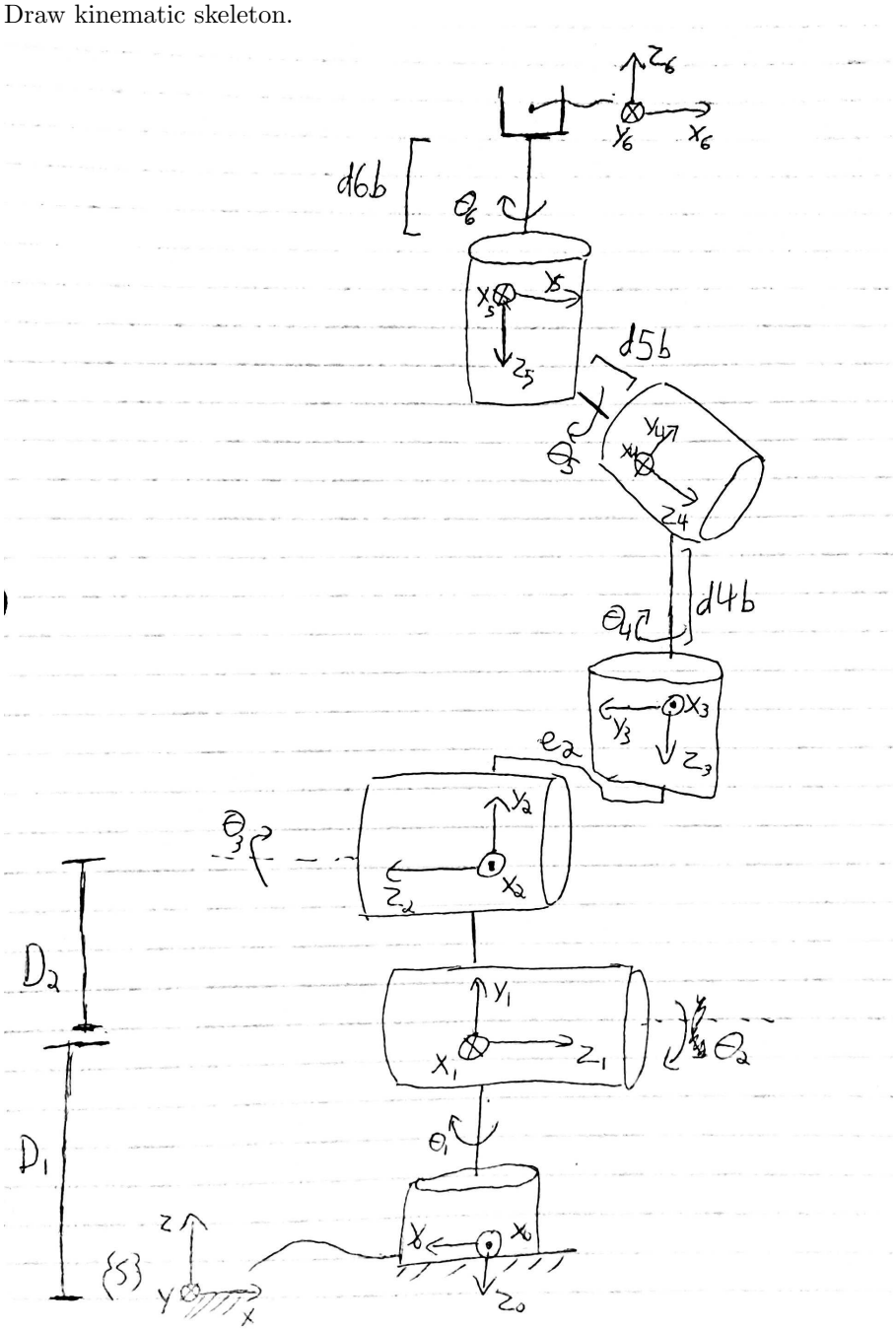
CSC376: Assignment 2: Due Nov 18, 2019 by 11:59pm

Roman H
Dean W

Nov 18, 2019

Part I

1. Draw kinematic skeleton.



2. Derive the forward kinematics using the Product of Exponential Formula in the space frame. Submit in writing. Implement your derived calculations in the function **forwardKinematicsPEF**.
- Fwd. Kin. takes the joint coordinates θ and gives position and orientation of end-effector frame.

$$M = \begin{bmatrix} 1 & 0 & 0 & e_2 - d5b \times s2a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_1 + D_2 + d4b + d5b \times c2a + d6b \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$q_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$v_1 = -\omega_1 \times q_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$S_1 = \begin{bmatrix} \omega_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$[S_1] = \begin{bmatrix} [\omega_1] & v_1 \\ 000 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\omega_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix}$$

$$v_2 = -\omega_2 \times q_2 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix} = \begin{bmatrix} 0 \\ D_1 \\ 0 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} \omega_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ D_1 \\ 0 \end{bmatrix}$$

$$[S_2] = \begin{bmatrix} [\omega_2] & v_2 \\ 000 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & D_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\omega_3 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

$$q_3 = \begin{bmatrix} 0 \\ 0 \\ D_1 + D_2 \end{bmatrix}$$

$$v_3 = -\omega_3 \times q_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ D_1 + D_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -(D_1 + D_2) \\ 0 \end{bmatrix}$$

$$S_3 = \begin{bmatrix} \omega_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ -(D_1 + D_2) \\ 0 \end{bmatrix}$$

$$[S_3] = \begin{bmatrix} [\omega_3] & v_3 \\ 000 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -(D_1 + D_2) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\omega_4 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$q_4 = \begin{bmatrix} e_2 \\ 0 \\ 0 \end{bmatrix}$$

$$v_4 = -\omega_4 \times q_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} e_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ e_2 \\ 0 \end{bmatrix}$$

$$S_4 = \begin{bmatrix} \omega_4 \\ v_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ e_2 \\ 0 \end{bmatrix}$$

$$[S_4] = \begin{bmatrix} [\omega_4] & v_4 \\ 000 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & e_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\omega_5 = \begin{bmatrix} ca \\ 0 \\ -sa \end{bmatrix}$$

$$q_5 = \begin{bmatrix} e_2 \\ 0 \\ D_1 + D_2 + d4b \end{bmatrix}$$

$$v_5 = -\omega_5 \times q_5 = \begin{bmatrix} -ca \\ 0 \\ sa \end{bmatrix} \times \begin{bmatrix} e_2 \\ 0 \\ D_1 + D_2 + d4b \end{bmatrix} = \begin{bmatrix} 0 \\ sa \cdot e_2 + ca \cdot (D_1 + D_2 + d4b) \\ 0 \end{bmatrix}$$

$$S_5 = \begin{bmatrix} \omega_5 \\ v_5 \end{bmatrix} = \begin{bmatrix} ca \\ 0 \\ -sa \\ 0 \\ sa \cdot e_2 + ca \cdot (D_1 + D_2 + d4b) \\ 0 \end{bmatrix}$$

$$[S_5] = \begin{bmatrix} [\omega_5] & v_5 \\ 000 & 0 \end{bmatrix} = \begin{bmatrix} 0 & sa & 0 & 0 \\ -sa & 0 & -ca & sa \cdot e_2 + ca \cdot (D_1 + D_2 + d4b) \\ 0 & ca & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\omega_6 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$q_6 = \begin{bmatrix} e_2 - d5b \times s2a \\ 0 \\ 0 \end{bmatrix}$$

$$v_6 = -\omega_6 \times q_6 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} e_2 - d5b \times s2a \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ e_2 - d5b \times s2a \\ 0 \end{bmatrix}$$

$$S_6 = \begin{bmatrix} \omega_6 \\ v_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ e_2 - d5b \times s2a \\ 0 \end{bmatrix}$$

$$[S_6] = \begin{bmatrix} [\omega_6] & v_6 \\ 000 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & e_2 - d5b \times s2a \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus, the forward kinematics using the Product of Exponentials Formula in the space frame is:

$$T_{06} = e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3} e^{[S_4]\theta_4} e^{[S_5]\theta_5} e^{[S_6]\theta_6} M$$

3. Derive the forward kinematics using the Denavit-Hartenberg method. Submit in writing. Implement your derived calculations in the function **forwardKinematicsDH**.

   DH Derivation:

Using our skeleton and a table for the DH parameters:

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	$\pi/2$	D1	0
2	D2	π	0	0
3	0	$\pi/2$	$-e_2$	0
4	0	$\pi/3$	$-d4B$	0
5	0	$\pi/3$	$-d5B$	0
6	0	π	$-d6B$	$-\pi/2$

We now derive a matrix using the form:

$$T_{i-1,i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0,1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1\cdot 0 & -1 & -D1\cdot 1 \\ 0 & 1\cdot 1 & 0 & D1\cdot 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{1,2} = \begin{bmatrix} 1 & 0 & 0 & D2 \\ 0 & 1\cdot -1 & 0 & 0 \\ 0 & 1\cdot 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{2,3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1\cdot 0 & -1 & e_2 \\ 0 & 1\cdot 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{3,4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1\cdot \frac{1}{2} & -\sqrt{3}/2 & d4b\cdot\sqrt{3}/2 \\ 0 & 1\cdot \sqrt{3}/2 & \frac{1}{2} & -d4b\cdot 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{4,5} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1\cdot 1/2 & -\sqrt{3}/2 & d5b\cdot\sqrt{3}/2 \\ 0 & 1\cdot\sqrt{3}/2 & \frac{1}{2} & -d5b\cdot 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{5,6} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1\cdot -1 & 0 & 0 & d6b\cdot 0 \\ -1\cdot 0 & 0 & -1 & -d6b\cdot -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We also need to rotate $\{5\}$ to frame $\{0\}$ first:

$$T(\theta) = Rot(z, -\tfrac{\pi}{2})\, Rot(y, \pi)\, T_{0,1}\, T_{1,2}\, T_{2,3}\, T_{3,4}\, T_{4,5}\, T_{5,6}$$

where $Rot(z, -\pi/2) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

and $Rot(y, \pi) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

4. Discuss the implementation differences for the PoE formula and D-H parameter function.

A key difference between the two is the fact that we must derive 4 parameters with DH and 6 with PoE. This means that a similar amount of information about the forward kinematics is stored between the 4 and 6 parameters, respectively. PoE uses 6 parameters to describe the screw axis, using two 3 dimensional unit vectors. PoE describes the transformations between the screw axis of each joint. Link reference frames do not need to be assigned because, from the links, we only require the joint axes to determine the forward kinematics. The link lengths and angles are still required for determining the M matrix, angular components, and linear components. The implementation for the computation of the PoE pose consisted of finding the screw axis of each joint. This was done by finding the angular component and linear component based on the kinematic skeleton, as done in the tutorials. The Cayle-Rodrigues formula was used to compute the transformation matrix of each exponential representing a screw axis. The M matrix was then pre-multiplied by all of the found transformation matrices to compute the end-effector pose.

DH parameters combine the same information about the transformations between frames into 4 parameters. This can be done because the frames are organized in such a way that we can calculate the parameters based on a highly specific arrangement of the link frames. The frames must be organized such that they follow rules about how the Z and X axis are determined. Only then can we derive the DH parameters, and create a set of transformations. Using the definitions of α_{i-1}, a_{i-1}, d_i and ϕ_i from tutorial, we fill out a joint-parameter table by carefully assigning frames to our robot such that DH rules are not broken. We then define a transformation matrix between each of the frames by using the definition of $T_{i-1,i}$. Since we must have a specific base frame that conforms to our DH rules, we first rotate the s frame to our 0 frame and then perform each transformation.

Normally a disadvantage of using DH parameters is that you have to differentiate between prismatic and revolute joint variables, but in this case we only had revolute joints.

Part II

1. Derive the velocity kinematics of the robot manipulator in the formulation of the 6 x 6 space Jacobian matrix.

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$q_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$v_1 = -\omega_1 \times q_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\omega_2 = \begin{bmatrix} c_1 \\ -s_1 \\ 0 \end{bmatrix}$$

$$q_1 = \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix}$$

$$v_2 = -\omega_2 \times q_2 = \begin{bmatrix} -c_1 \\ s_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix} = \begin{bmatrix} s_1 \cdot D_1 \\ c_1 \cdot D_1 \\ 0 \end{bmatrix}$$

$$\omega_3 = \begin{bmatrix} -c_1 \\ s_1 \\ 0 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix}$$

$$v_2 = -\omega_2 \times q_2 = \begin{bmatrix} -c_1 \\ s_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix} = \begin{bmatrix} s_1 \cdot D_1 \\ c_1 \cdot D_1 \\ 0 \end{bmatrix}$$

$$\omega_{s4} = Rot(z, -\theta_1)Rot(x, \theta_2)Rot(x, -\theta_3) \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$\omega_{s5} = Rot(z, -\theta_1)Rot(x, \theta_2)Rot(x, -\theta_3)Rot(z, -\theta_4) \begin{bmatrix} ca \\ 0 \\ -sa \end{bmatrix}$$

$$\omega_{s6} = Rot(z, -\theta_1)Rot(x, \theta_2)Rot(x, -\theta_3)Rot(z, -\theta_4)Rot(?, ?) \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

Thus, the velocity kinematics in the formulation of the 6×6 space Jacobian matrix $J_s(\theta)$ is:

$$J_s(\theta) = \begin{bmatrix} J_{s1} & J_{s2} & J_{s3} & J_{s4} & J_{s5} & J_{s6} \end{bmatrix}$$

$Ad_{T_{i-1}}(S_i)$ describes the i columns of $J_s(\theta)$ where:
$T_{i-1} = e^{[S_1]\theta_1} e^{[S_2]\theta_2} ... e^{[S_{i-1}]\theta_{i-1}}$
We get:
$J_{s1} = S_1$
$J_{s2} = Ad_{T_1}(S_2) = Ad_{e^{[S_1]\theta_1}}(S_2)$
$J_{s3} = Ad_{T_2}(S_3) = Ad_{e^{[S_1]\theta_1} e^{[S_2]\theta_2}}(S_3)$
$J_{s4} = Ad_{T_3}(S_4) = Ad_{e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3}}(S_4)$
$J_{s5} = Ad_{T_4}(S_5) = Ad_{e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_2]\theta_3} e^{[S_4]\theta_4}}(S_5)$
$J_{s6} = Ad_{T_5}(S_6) = Ad_{e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_2]\theta_3} e^{[S_4]\theta_4} e^{[S_5]\theta_5}}(S_6)$

Note: Our screw axis were previously calculated in part 1.

2. Implement **velocityKinematics**.

3. Implement **calcRCond**.
4. Implement **calcSingularValues**
5. Implement **drawManEllipsoid**

Config	Joint	1	2	3	4	5	6
1	θ in $^\circ$	0	0	0	0	0	0
2	θ in $^\circ$	0	90	90	0	90	0
3	θ in $^\circ$	45	-75	-90	90	45	90
4	θ in $^\circ$	-90	-30	90	90	0	45
5	θ in $^\circ$	-60	30	-45	35	-100	-120
6 (singularity)	θ in $^\circ$	0	90	0	0	0	0
7	θ in $^\circ$	90	0	60	110	30	50
8	θ in $^\circ$	90	100	-20	110	30	50
9	θ in $^\circ$	90	100	130	110	30	50
10	θ in $^\circ$	90	-60	40	110	150	50

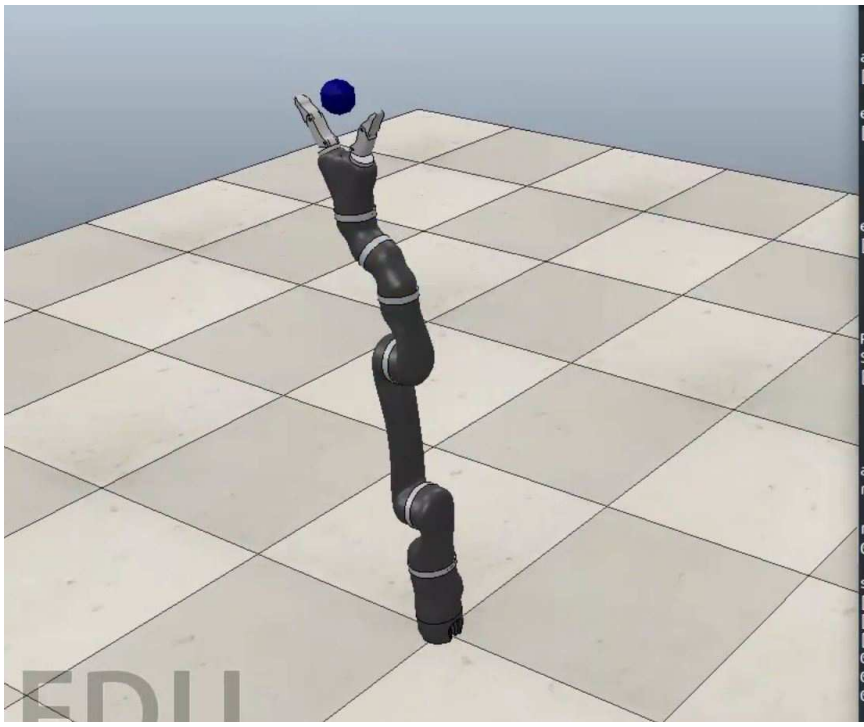
Part III

1. **Discussion:**
   Based on our implementations of PoE and DH, we found that our PoE formula correctly calculated the end-effector poses. The blue spheres in the screenshots shown in our results are calculated using our PoE formula. Unfortunately, our implementation based on the DH parameters did not result in correct end-effector poses.

   Our implementation of the velocity kinematics to calculate the 6×6 Jacobian matrix must have some errors since the rank was not full in non-singular poses. This lead to the values calculated for our reciprocal conditional number to be unexpected.

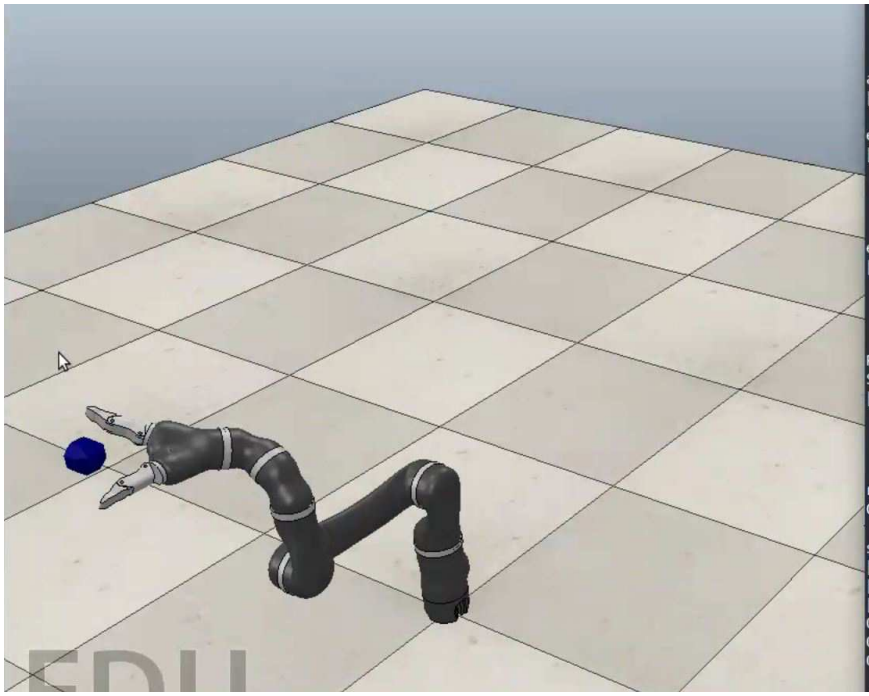
   We have added 5 more configurations to the table above. Configuration 6 was our chosen singularity sample, and we produced it by setting the angle of joint 2 to 90°. This results in the robot extending the rest of the body frames towards our calculated end-effector pose (using PoE). We know this is a singularity because the robot cannot exceed beyond this calculated point. If we had implemented our maniplulability ellipsoid, then we would observe in this state that the ellipsoid resembles a line, since it would need infinite velocity to move in any outwards direction.
   Screenshots and Info:

Config 1
Image and info below:



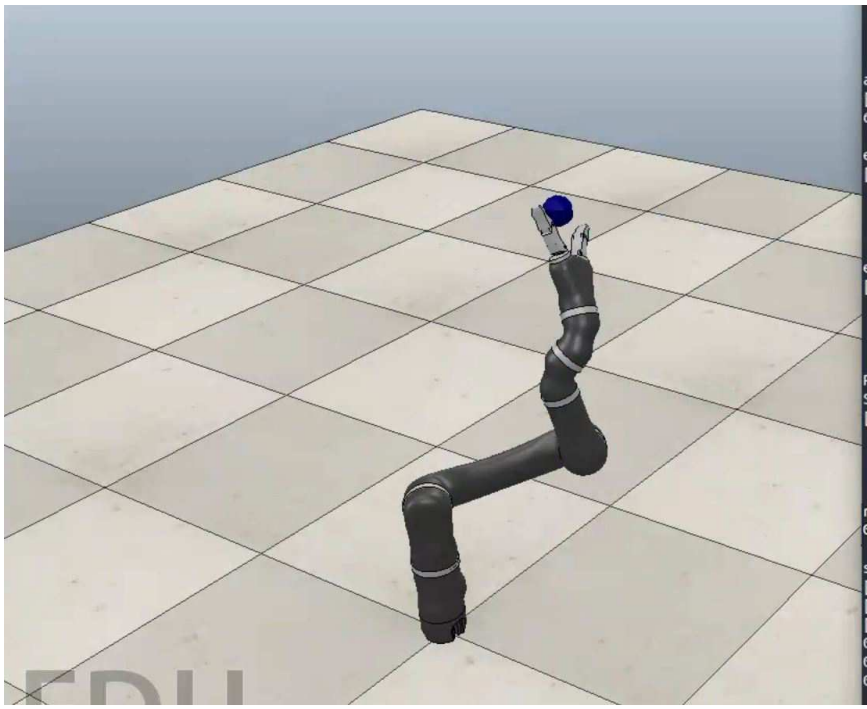
```
 Connected!

angles:
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

eePose forwardKinematicsPEF:
[[ 1.          0.          0.         -0.0671    ]
 [ 0.          1.          0.          0.        ]
 [ 0.          0.          1.          1.00846331]
 [ 0.          0.          0.          1.        ]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[ 1.00000000e+00  0.00000000e+00  0.00000000e+00 -6.69904575e-02]
 [ 0.00000000e+00  1.00000000e+00  0.00000000e+00  1.52330991e-04]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00  9.77099419e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

assignmentII.py:462: RuntimeWarning: divide by zero encountered in double_scala
rs
  rcond = 1 / (np.max(w) / np.min(w))
reciprocal conditional number:
0.0

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

Config 2
Image and info below:



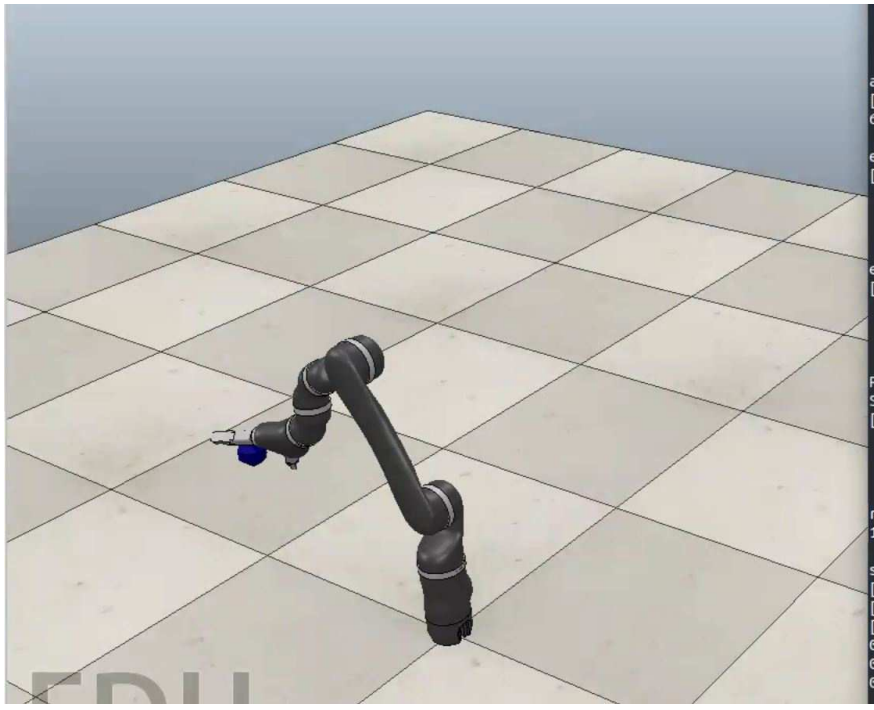
```
 Connected!

angles:
[0.0, 1.5707963267948966, 1.5707963267948966, 0.0, 1.5707963267948966, 0.0]

eePose forwardKinematicsPEF:
[[ 7.50000000e-01  5.00000000e-01 -4.33012702e-01 -1.68468274e-01]
 [-5.00000000e-01  1.11022302e-16 -8.66025404e-01 -4.92736547e-01]
 [-4.33012702e-01  8.66025404e-01  2.50000000e-01  5.42888310e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[ 7.50007348e-01  4.99991852e-01 -4.33009384e-01 -1.54781759e-01]
 [-4.99993677e-01  4.48470328e-07 -8.66029054e-01 -4.65622276e-01]
 [-4.33007276e-01  8.66030108e-01  2.49993101e-01  5.35038114e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

reciprocal conditional number:
0.002372149395781009

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

Config 3
Image and info below:



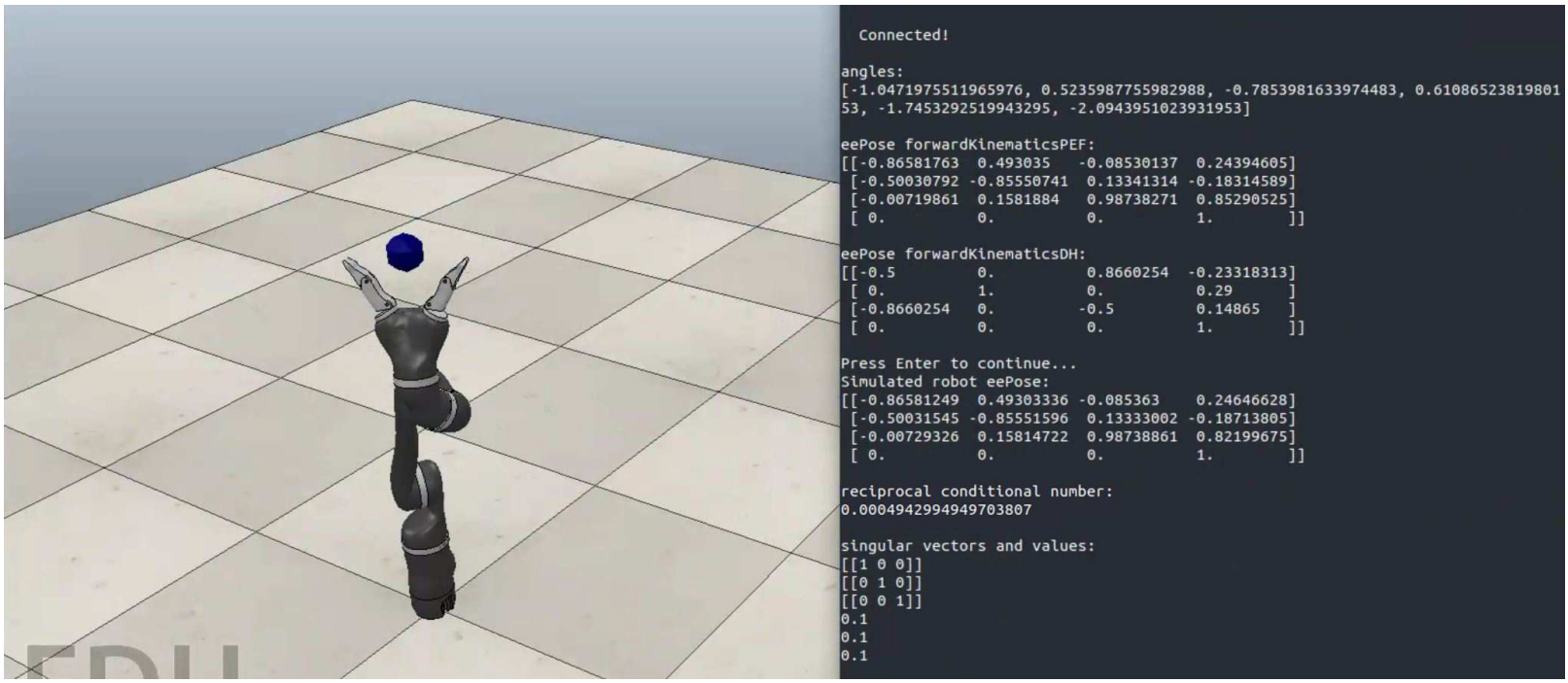
```
  Connected!

angles:
[0.7853981633974483, -1.3089969389957472, -1.5707963267948966, 1.57079632679489
66, 0.7853981633974483, 1.5707963267948966]

eePose forwardKinematicsPEF:
[[-0.14644661 -0.8597894  -0.48919892  0.10088857]
 [ 0.85355339 -0.3597894   0.37682648  0.29372562]
 [-0.5        -0.36237244  0.78656609  0.7556176 ]
 [ 0.          0.          0.          1.        ]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[-0.14641904 -0.85982011 -0.48915318  0.1161315 ]
 [ 0.85352669 -0.35976969  0.37690577  0.28172779]
 [-0.50005365 -0.36231912  0.78655655  0.73093593]
 [ 0.          0.          0.          1.        ]]

reciprocal conditional number:
0.0015423088737050678

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

Config 4
Image and info below:



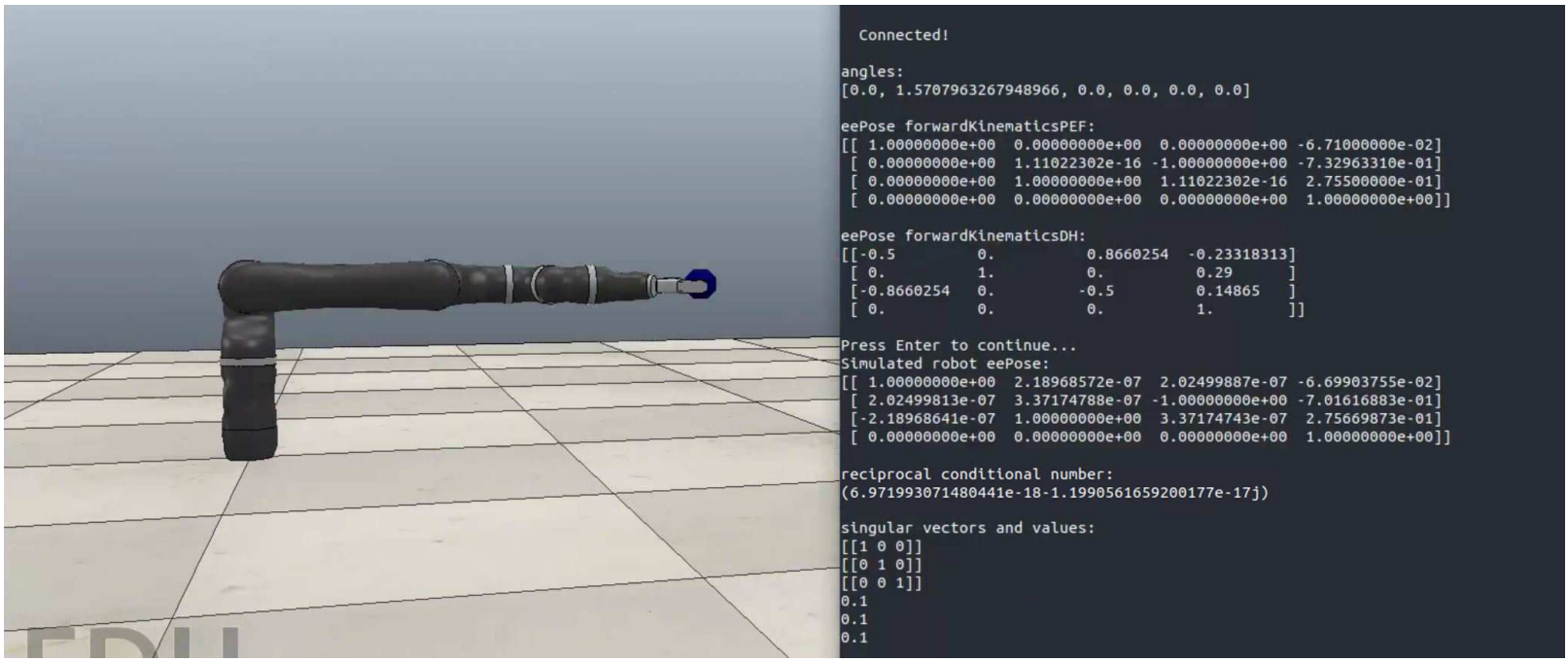
```
  Connected!

angles:
[-1.5707963267948966, -0.5235987755982988, 1.5707963267948966, 1.57079632679489
66, 0.0, 0.7853981633974483]

eePose forwardKinematicsPEF:
[[-3.53553391e-01 -3.53553391e-01 -8.66025404e-01 -4.91567479e-01]
 [-7.07106781e-01  7.07106781e-01  9.61481343e-17  7.00000000e-03]
 [ 6.12372436e-01  6.12372436e-01 -5.00000000e-01  2.40993230e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[-3.53538245e-01 -3.53593988e-01 -8.66015012e-01 -4.64509189e-01]
 [-7.07106923e-01  7.07106638e-01 -4.53986905e-05  7.15133082e-03]
 [ 6.12381016e-01  6.12349160e-01 -5.00017997e-01  2.56797791e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

reciprocal conditional number:
1.0815716297514675e-07

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

Config 5
Image and info below:



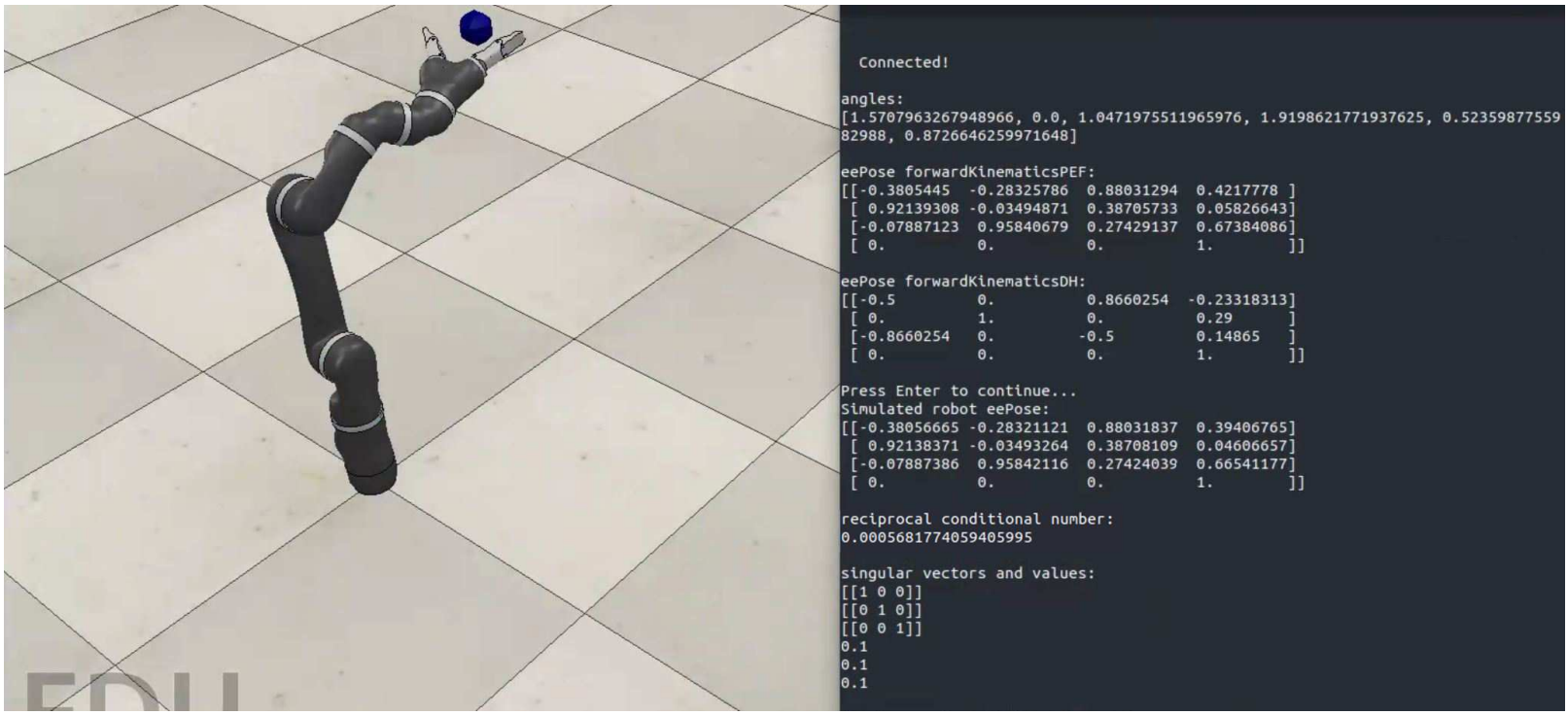
```
  Connected!

angles:
[-1.0471975511965976, 0.5235987755982988, -0.7853981633974483, 0.61086523819801
53, -1.7453292519943295, -2.0943951023931953]

eePose forwardKinematicsPEF:
[[-0.86581763  0.493035   -0.08530137  0.24394605]
 [-0.50030792 -0.85550741  0.13341314 -0.18314589]
 [-0.00719861  0.1581884   0.98738271  0.85290525]
 [ 0.          0.          0.          1.        ]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[-0.86581249  0.49303336 -0.085363    0.24646628]
 [-0.50031545 -0.85551596  0.13333002 -0.18713805]
 [-0.00729326  0.15814722  0.98738861  0.82199675]
 [ 0.          0.          0.          1.        ]]

reciprocal conditional number:
0.0004942994949703807

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

Config 6 (Singularity)
Image and info below:

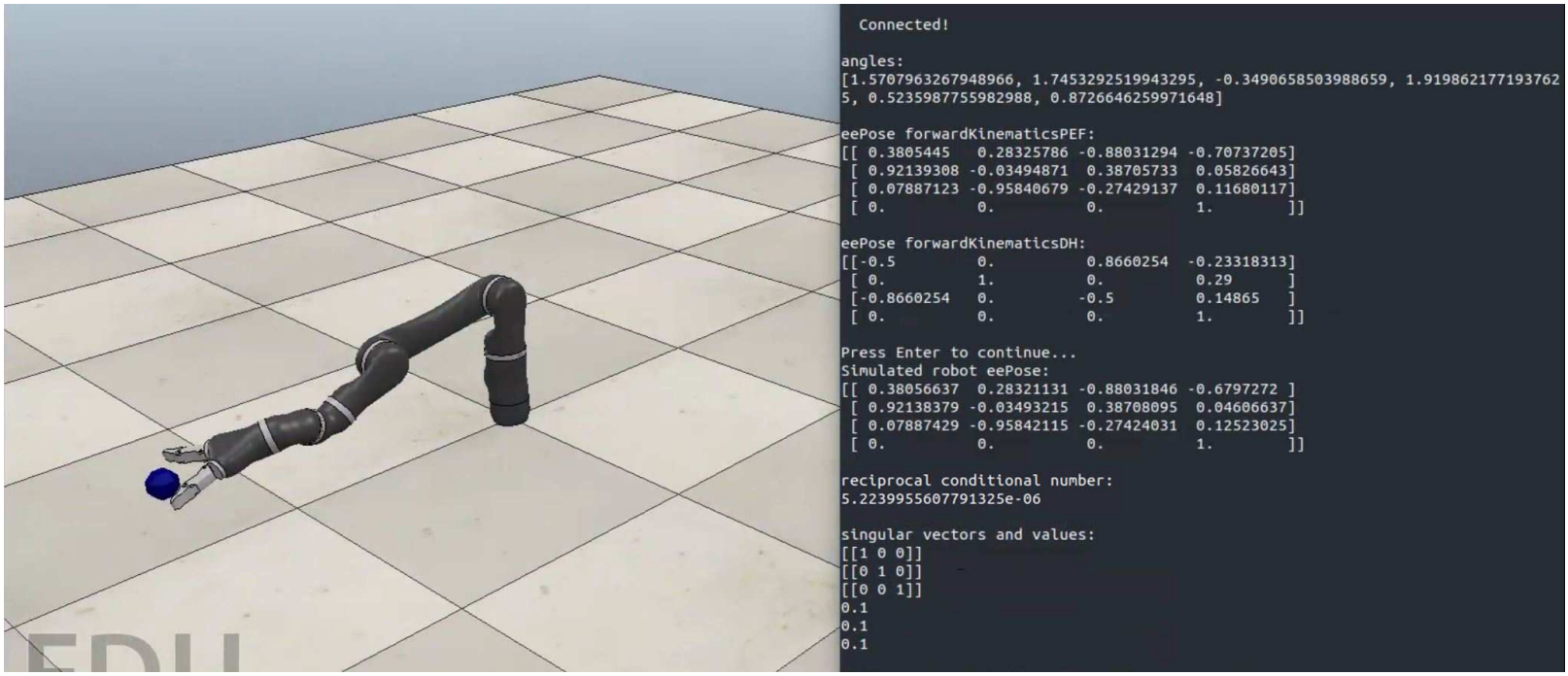


```
  Connected!

angles:
[0.0, 1.5707963267948966, 0.0, 0.0, 0.0, 0.0]

eePose forwardKinematicsPEF:
[[ 1.00000000e+00  0.00000000e+00  0.00000000e+00 -6.71000000e-02]
 [ 0.00000000e+00  1.11022302e-16 -1.00000000e+00 -7.32963310e-01]
 [ 0.00000000e+00  1.00000000e+00  1.11022302e-16  2.75500000e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[ 1.00000000e+00  2.18968572e-07  2.02499887e-07 -6.69903755e-02]
 [ 2.02499813e-07  3.37174788e-07 -1.00000000e+00 -7.01616883e-01]
 [-2.18968641e-07  1.00000000e+00  3.37174743e-07  2.75669873e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

reciprocal conditional number:
(6.971993071480441e-18-1.1990561659200177e-17j)

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

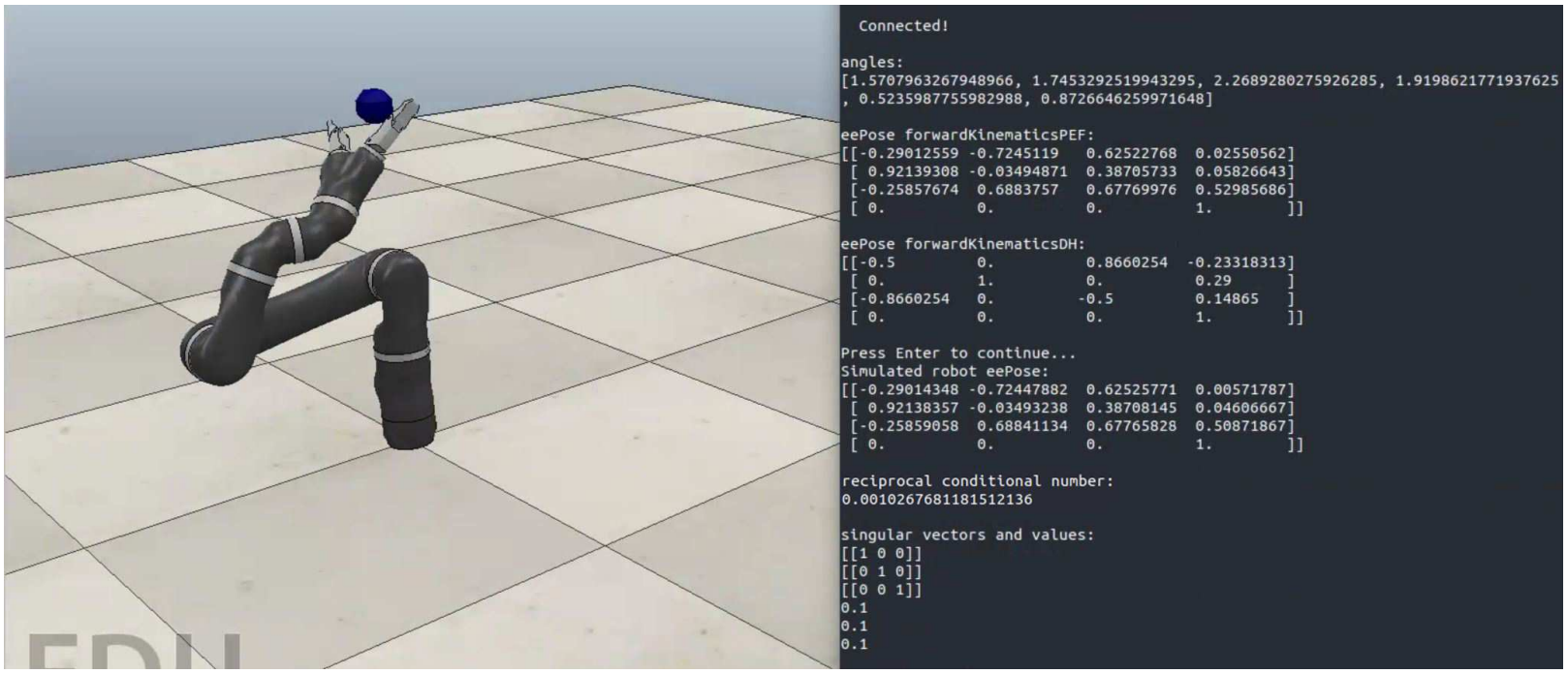
Config 7
Image and info below:



Config 8
Image and info below:

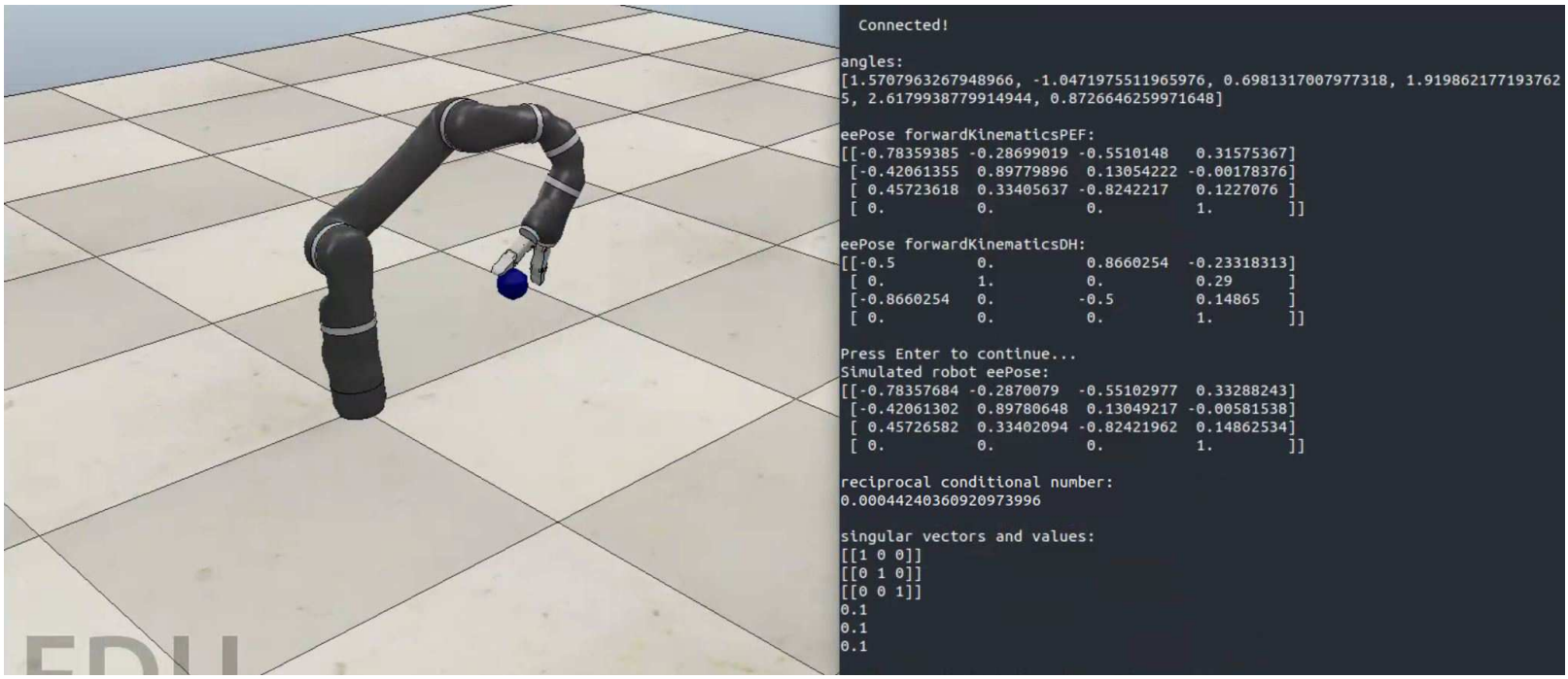


Config 9
Image and info below:



```
  Connected!

angles:
[1.5707963267948966, 1.7453292519943295, 2.2689280275926285, 1.9198621771937625
, 0.5235987755982988, 0.8726646259971648]

eePose forwardKinematicsPEF:
[[-0.29012559 -0.7245119   0.62522768  0.02550562]
 [ 0.92139308 -0.03494871  0.38705733  0.05826643]
 [-0.25857674  0.6883757   0.67769976  0.52985686]
 [ 0.          0.          0.          1.        ]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[-0.29014348 -0.72447882  0.62525771  0.00571787]
 [ 0.92138357 -0.03493238  0.38708145  0.04606667]
 [-0.25859058  0.68841134  0.67765828  0.50871867]
 [ 0.          0.          0.          1.        ]]

reciprocal conditional number:
0.0010267681181512136

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```

Config 10
Image and info below:



```
  Connected!

angles:
[1.5707963267948966, -1.0471975511965976, 0.6981317007977318, 1.919862177193762
5, 2.6179938779914944, 0.8726646259971648]

eePose forwardKinematicsPEF:
[[-0.78359385 -0.28699019 -0.5510148   0.31575367]
 [-0.42061355  0.89779896  0.13054222 -0.00178376]
 [ 0.45723618  0.33405637 -0.8242217   0.1227076 ]
 [ 0.          0.          0.          1.        ]]

eePose forwardKinematicsDH:
[[-0.5         0.          0.8660254  -0.23318313]
 [ 0.          1.          0.          0.29      ]
 [-0.8660254   0.         -0.5         0.14865   ]
 [ 0.          0.          0.          1.        ]]

Press Enter to continue...
Simulated robot eePose:
[[-0.78357684 -0.2870079  -0.55102977  0.33288243]
 [-0.42061302  0.89780648  0.13049217 -0.00581538]
 [ 0.45726582  0.33402094 -0.82421962  0.14862534]
 [ 0.          0.          0.          1.        ]]

reciprocal conditional number:
0.00044240360920973996

singular vectors and values:
[[1 0 0]]
[[0 1 0]]
[[0 0 1]]
0.1
0.1
0.1
```