Fundamentels of Robotics

Assignment 1 - Kinematics. Innopolis University, Fall 2020

Name: Hany Hamed
Group Number: BS18-Robotics
GitHub Repository: here (It is public after the submission deadline)

Contents

FoR, Innopolis University, Fall 2020
Author: Hany Hamed

Section 1: Robot's Description

The robot is a manipulator from KUKA (KR 10 R1100-2). This robot has 6 joints and is 6 DoF manipulator, moreover, it has a spherical writs.

Data-sheet of the manipulator can be found here

Figure 1. Robot's image

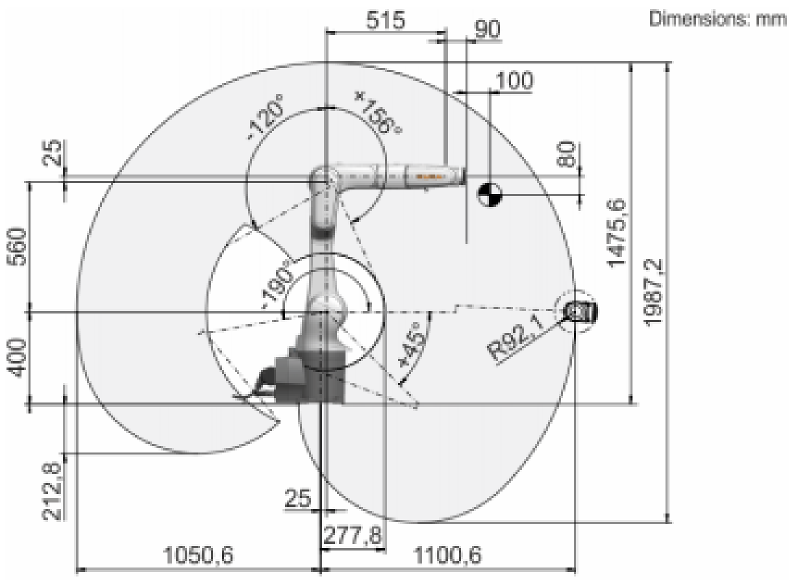

Figure 2. Robot's working space

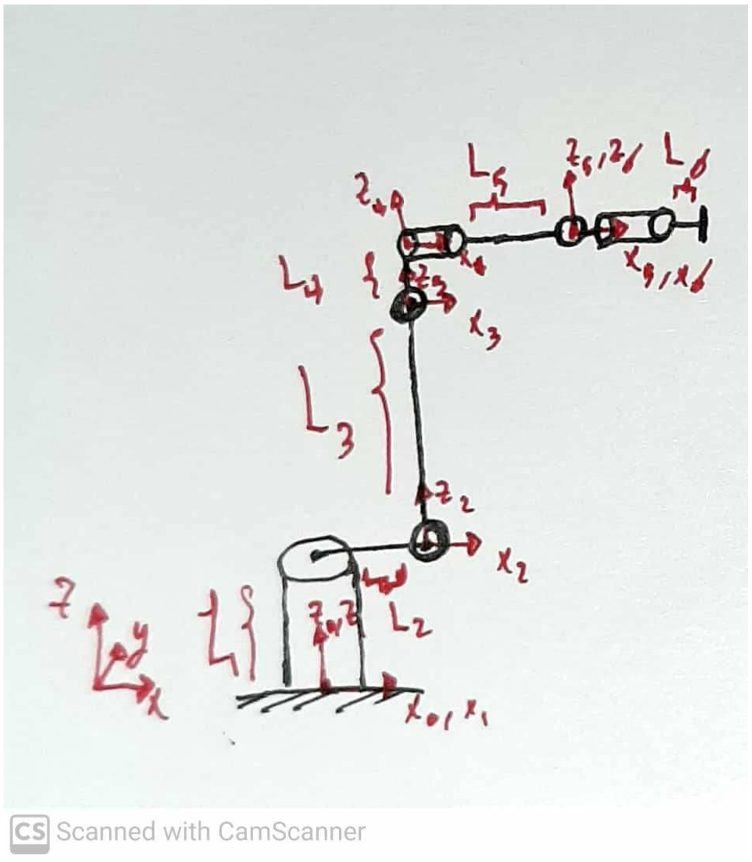

Figure 3. Robot's scheme in normal configuration

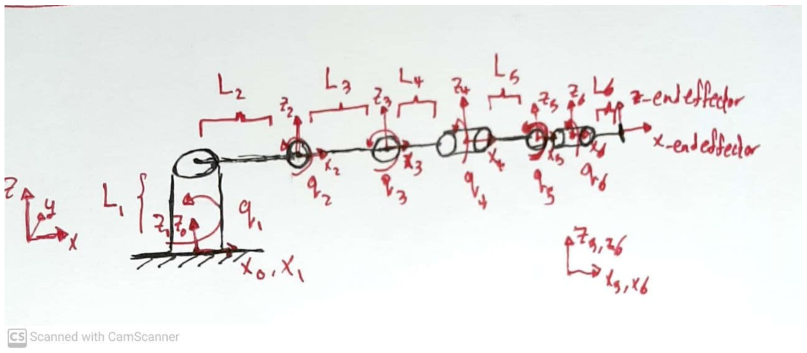

Figure 4. Robot's scheme in zero configuration

Note: I wanted to make it with this software but did not have time to do it

The previous software can generate images like this via scripting language (but unfortunately, did not have time to modify it)

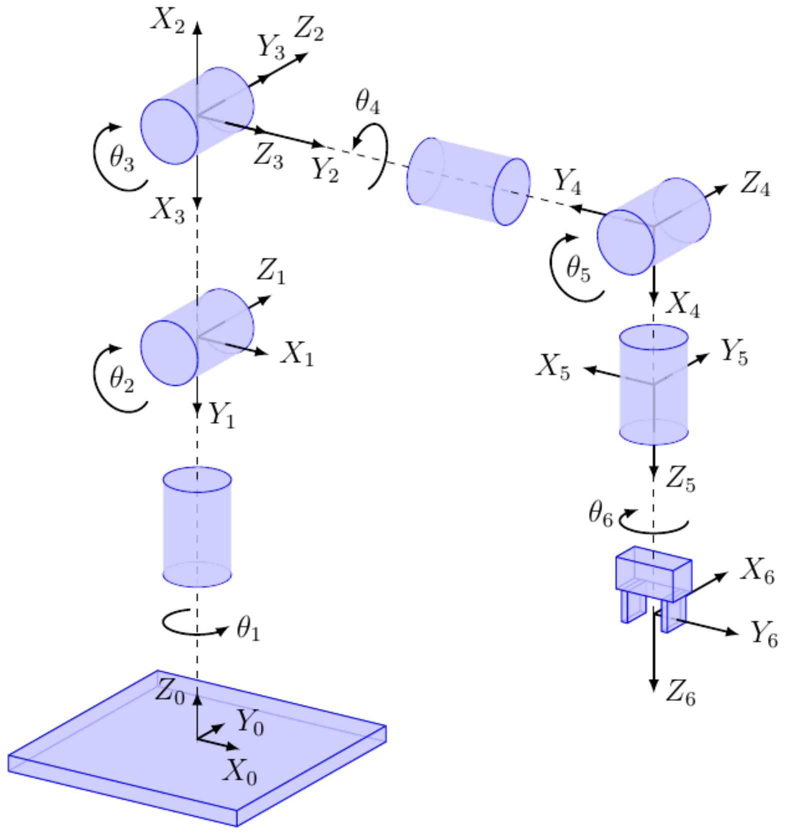

Figure 5. Example for the output of the software that can generate diagrams for manipulators (Example for 6DoF not the robot that is discussed here)

Robot's Links parameters

- $L_1 = 400mm$
- $L_2 = 25mm$
- $L_3 = 560mm$
- $L_4 = 25mm$
- $L_5 = 515mm$
- $L_6 = 90mm$

Robot's Joints limits

- $J_1 = \pm 170°$
- $J_2 = -190°/45°$
- $J_3 = -120°/156°$
- $J_4 = \pm 185°$
- $J_5 = \pm 120°$
- $J_6 = \pm 350°$

Section2: Forward Kinematics

Forward Kinematics can be solved in two ways: using DH parameters or using Translation and Rotation matrices directly from the scheme of the zero configuration.

Solution using DH parameters

Note: take into consideration that the previous scheme does not assume the same axis as it is used in DH as in DH z-axis is always the axis of rotation

Frame$_i d$	θ	d	a	α
1	0	0	0	0
2	-90	400	25	-90
3	0	0	560	0
4	0	0	25	-90
5	0	515	0	90
6	0	0	0	-90
7	0	90	0	0

Solution using lecture's method by translating and rotation

$T = T_{base}\, R_z(q_1)\, T_z(L_1)\, R_y(q_2)\, T_x(L_3)\, R_y(q_3)\, T_x(L_4)\, R_x(q_4)\, T_x(L_5)\, R_y(q_5)\, R_x(q_6)\, T_x(L_6)\, T_{tool}$

It can be changed a little bit with some swapping between the consecutive rotation and translation of the same axis, as following:

$T = T_{base}\, T_z(L_1)\, R_z(q_1)\, R_y(q_2)\, T_x(L_3)\, R_y(q_3)\, T_x(L_4)\, T_x(L_5)\, R_x(q_4)\, R_y(q_5)\, R_x(q_6)\, T_x(L_6)\, T_{tool}$

And here we can understand that we have a spherical wrist by observing the last three rotation matrices R_{xyx}

Section 3: Inverse Kinematics

Inverse Kinematics can be solved by multiple methods (numerical and analytical) according to the complexity of the problem. According to here, an explanation for analytical solutions like Pieper's solution and numerical solution like Newton's Method.

Solving Inverse Kinematics using Pieper's Solution

Pieper's solution is only used with manipulators with spherical wrist as the FK can be decomposed into two parts, one responsible for the first three joints which affect on only the position, and the other part responsible for the last three joints which affect on only the orientation of the end-effector.

First, it is needed to split the FK into multiple parts, as following:

$T = T_{base}\ T_z(L_1)\ T_{123}\ T_{456}\ T_x(L_6)\ T_{tool}$

Such that, T_{123} is for 1st,2nd and 3rd joints for position and T_{456} is for 4th, 5th and 6th joint for orientation.

In IK problems, T is given, then we can do as following:

$T_o = T_{123}\ T_{456} = T_z(L_1)^{-1}\ T_{base}^{-1}\ T\ T_{tool}^{-1}\ T_x(L_6)^{-1}$

and the previous matrix can be computed and get its values.

Then, we need to solve the part of the position first using T_{123}:

$T_{123} = T_o\ T_{456}^{-}1$

Computation of T_{123} with the expressions

$$T_{123} = \begin{bmatrix} -s_2s_3c_1 + c_1c_2c_3 & -s_1 & s_2c_1c_3 + s_3c_1c_2 & l_2c_1 + l_3c_1c_2 + l_4 - s_2s_3c_1 + c_1c_2c_3 + l_5 - s_2s_3c_1 + c_1c_2c_3 \\ -s_1s_2s_3 + s_1c_2c_3 & c_1 & s_1s_2c_3 + s_1s_3c_2 & l_2s_1 + l_3s_1c_2 + l_4 - s_1s_2s_3 + s_1c_2c_3 + l_5 - s_1s_2s_3 + s_1c_2c_3 \\ -s_2c_3 - s_3c_2 & 0 & -s_2s_3 + c_2c_3 & -l_3s_2 + l_4 - s_2c_3 - s_3c_2 + l_5 - s_2c_3 - s_3c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and here we only care about the position which means the last column in the homogeneous matrix, and this position is the position of the end-effector in T matrix (x, y, z). After calculating the previous expression we get:

Note: the following expressions have been computed with help of sympy: a library for symbolic derivations

Note: $l_1 = L_1$

$x = l_2 * cos(q_1) + l_3 * cos(q_1) * cos(q_2) + l_4 * (-sin(q_2) * sin(q_3) * cos(q_1) + cos(q_1) * cos(q_2) * cos(q_3)) + l_5 * (-sin(q_2) * sin(q_3) * cos(q_1) + cos(q_1) * cos(q_2) * cos(q_3))$
$= [l_2 + l_3 * cos(q_2) + l_4 * (-sin(q_2) * sin(q_3) + cos(q_2) * cos(q_3)) + l_5 * (-sin(q_2) * sin(q_3) + cos(q_2) * cos(q_3))] * cos(q_1)$
$= [l_2 + l_3 * cos(q_2) + l_4 * cos(q_2 + q_3) + l_5 * cos(q_2 + q_3)] * cos(q_1)$

$y = l_2 * sin(q_1) + l_3 * sin(q_1) * cos(q_2) + l_4 * (-sin(q_2) * sin(q_3) * sin(q_1) + sin(q_1) * cos(q_2) * cos(q_3)) + l_5 * (-sin(q_2) * sin(q_3) * sin(q_1) + sin(q_1) * cos(q_2) * cos(q_3))$
$= [l_2 + l_3 * cos(q_2) + l_4 * (-sin(q_2) * sin(q_3) + cos(q_2) * cos(q_3)) + l_5 * (-sin(q_2) * sin(q_3) + cos(q_2) * cos(q_3))] * sin(q_1)$
$= [l_2 + l_3 * cos(q_2) + l_4 * cos(q_2 + q_3) + l_5 * cos(q_2 + q_3)] * sin(q_1)$

$z = -l_3 * sin(q_2) + l_4 * (-sin(q_2) * cos(q_3) - sin(q_3) * cos(q_2)) + l_5 * (-sin(q_2) *$

$cos(q_3) - sin(q_3) * cos(q_2))$

Then we can get: $\begin{cases} if([l_2 + l_3 * cos(q_2) + l_4 * cos(q_2+q_3) + l_5 * cos(q_2+q_3)] \neq 0) & q_1 = atan2(y,x) \\ else & q_1\ any \end{cases}$

In the 2nd condition, q_1 is any as the point will be positioned on z-axis. So, rotation of the first joint will not matter.

Then, we can do as following:

$\pm\sqrt{x^2+y^2} - l_2 = l_3 * cos(q_2) + l_4 * cos(q_2+q_3) + l_5 * cos(q_2+q_3) = x`$

$-z = l_3 * cos(q_2) + (l_4+l_5) * sin(q_2+q_3) = y`$

$L`_1 = l_3,\ L`_2 = (l_4+l_5)$

and solve it as Planar 2-link manipulator with respect to $x`,\ y`,\ L`_1\ and\ L`_2$

$q_3 = q`_2 = arccos(\frac{x`^2 + y`^2 - L`^2_1 - L`^2_2}{2L`_1L`_2})$

$m = \begin{cases} -1 & q`_2 > 0 \\ +1 & q`_2 < 0 \end{cases}$

$q_2 = q`_1 = -m * atan(\frac{L`_2 * sin(q`_2)}{L`_1 + L`_2 * cos(q`_2)})$

Here, we got $q_1,\ q_2,\ q_3$, we can substitute them in T_{123} and get the expression of $T^{-1}_{123}\ T_o = T_{456}$ and we will observe that the last column is just zero which confirm that this matrix is only for the orientation of the end-effector and we are interested in the rotation matrix inside the homogeneous transformation (the first three rows x the first three columns).

$$T_{456} = \begin{bmatrix} c5 & s5s6 & s5c6 & 0 \\ s4s5 & -s4s6c5 + c4c6 & -s4c5c6 - s6c4 & 0 \\ -s5c4 & s4c6 + s6c4c5 & -s4s6 + c4c5c6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n``_x & s``_x & a``_x & 0 \\ n``_y & s``_y & a``_y & 0 \\ n``_z & s``_z & a``_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And we can solve it and get $q_4,\ q_5,\ q_6$ by getting the expressions from the previous matrix and substitute the values from computation of $T^{-1}_{123}\ T_o = T_{456}$

First, the single term determines the singularity

$$\begin{aligned} & if(|cos(q_4)| \neq 1): \\ & \qquad q_4 = atan2(n_y``, n_x``)\&q_6 = atan2(s``_x, a``_x) \\ & \qquad if(a``_x \neq 0): q_5 = atan2(a``_x/cos(q_6), n``_x) \\ & \qquad else: q_5 = atan2(s``_x/sin(q_6), n``_x) \end{aligned} \tag{1}$$

A singularity case: 4th and 6th joint on the same axis

$if(|cos(q_4)| = 1) : q_5 = arccos(n``_x)$ and the homogenous matrix will be as a matrix with zero translation and rotation around x-axis with angle $(q_4+q_6 = \alpha)$

$\alpha = atan2(s``_z, s``_y)$ and the summation of $q_4,\ q_6 = \alpha$. The choice of them, could be greedy by choosing the most of the angle from q_6 joint as it has bigger range of values and take the rest from the other joint, but it is not efficient as there will be a situation such that the 6th joint has to make big revolution and this revolution could be achieved by small revolution from the 4th joint.

Now, we got all the information from the Ik (the six joints' angles: $q_1, q_2, q_3, q_4, q_5, q_6$).

Section 4: Code Architechture

Code consists of several parts:

assets directory: Datasheet of the robot

imgs directory: images for the manipulator for the scheme

report directory: the report source (LaTex and pdf)

src directory: (source codes)

- main.py: has the examples and how to use the code
- robot.py: has the classes of the robots and its configurations (links' dimensions, joints' limits)
- FK.py: Forward Kinematics implementation (As it was stated in the assignment description to be in a separate file)
- IK.py: Inverse Kinematics implementation (As it was stated in the assignment description to be in a separate file)
- utils.py: (utilities) functions for rotations and translations, getting positions and orientation from homogeneous matrix and calculating the error between two homogeneous matrices (To be used to select $\vec{q}$ feed it in FK implementation get T (transformation matrix), feed it into IK get $\vec{q}_{calc}$, then feed it again into FK and get T_{calc} and calculate the error between T & T_{calc}, if it is small, then the implementations works fine.
- exp_helper.py: (expression helper) it is used to generate the symbolic expressions for matrices and all the expressions instead of doing it by hand.
- visualization.py: implementation of a visualizer based on vpython (implemented) and mplot3d(did not have time to implement this).
- gen_test.py: (generate test) generate some test sets for the FK and IK and save them in files with .npy extension (3 methods: Random: random generation with constraints on the joints limits, Likelihood: closer to a selected configuration, Hard-coded configurations)

Examples for the output of the visualization

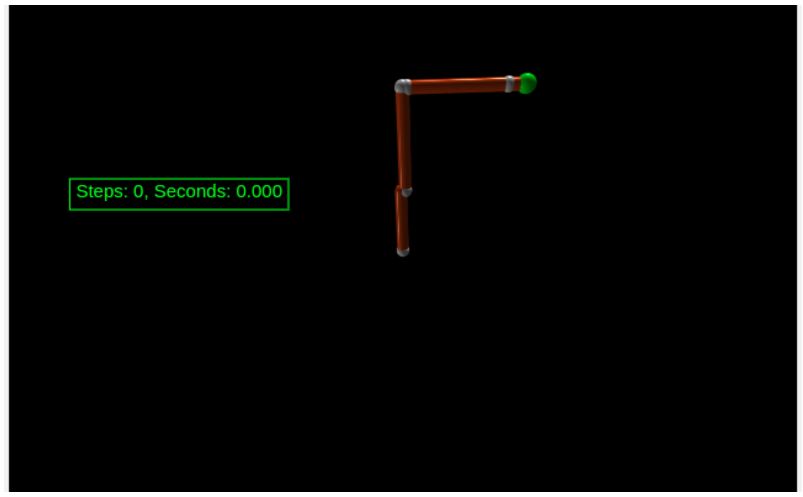

Figure 6. Example1

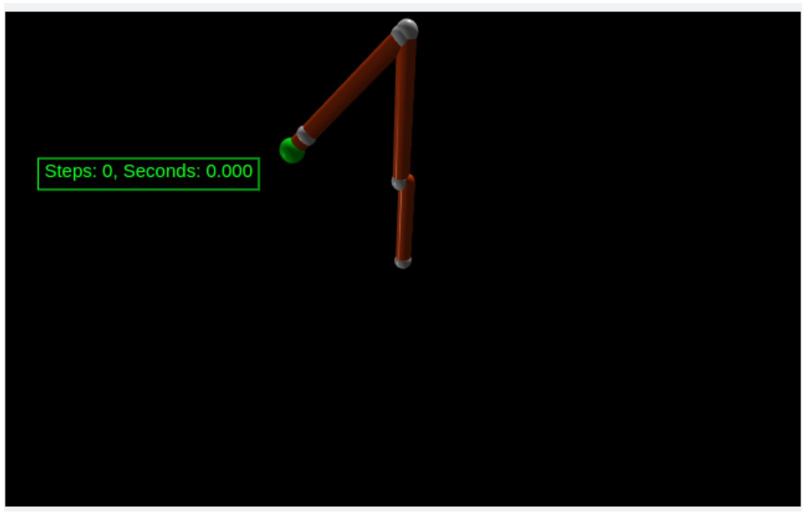

Figure 7. Example2

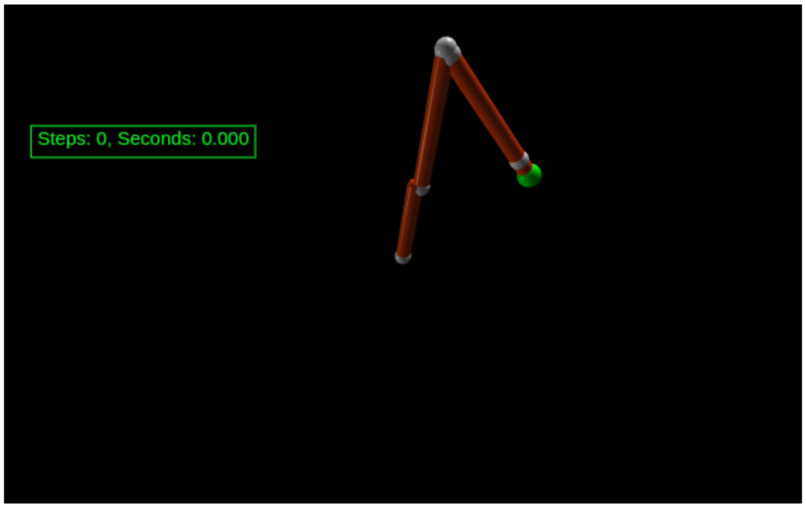

Figure 8. Example3

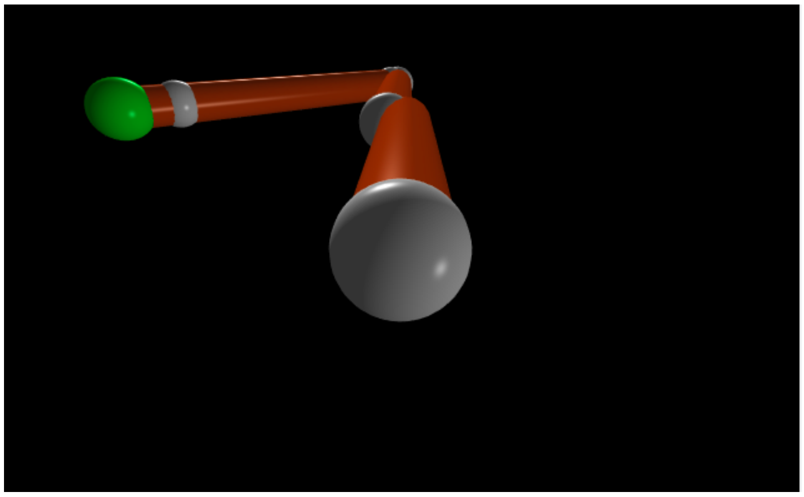

Figure 9. Example4

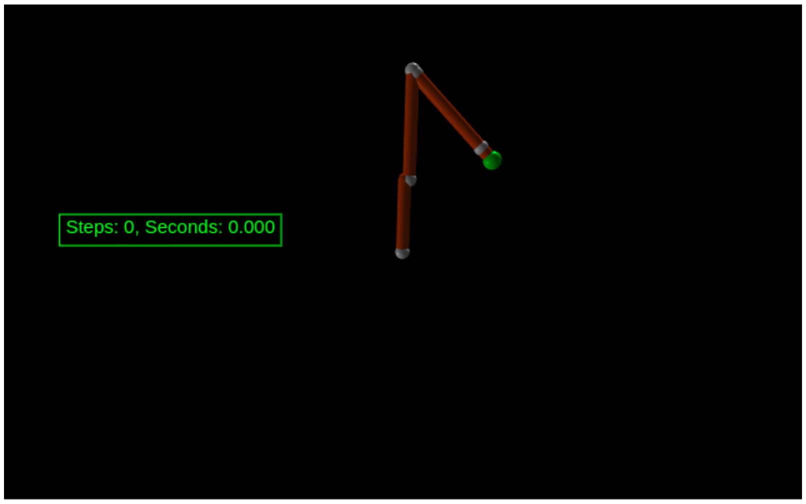

Figure 10. Example5

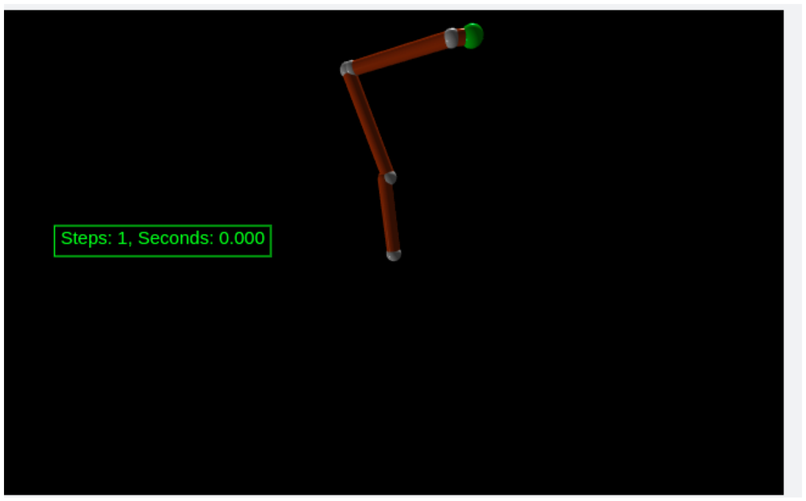

Figure 11. Example6

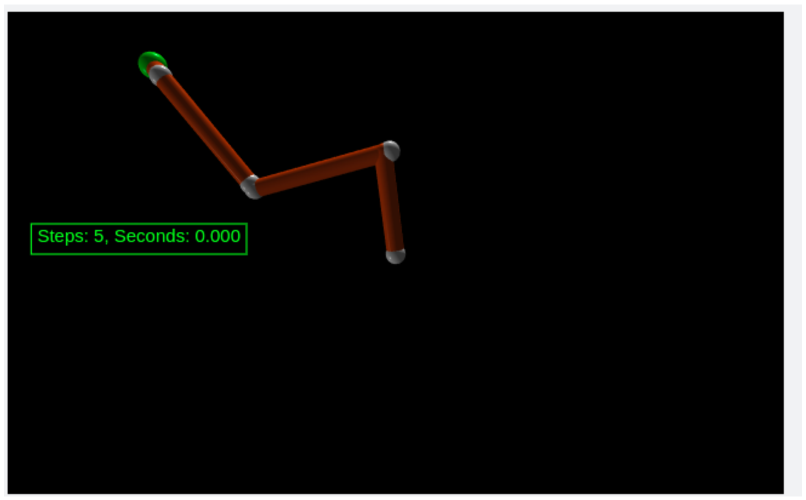

Figure 12. Example7

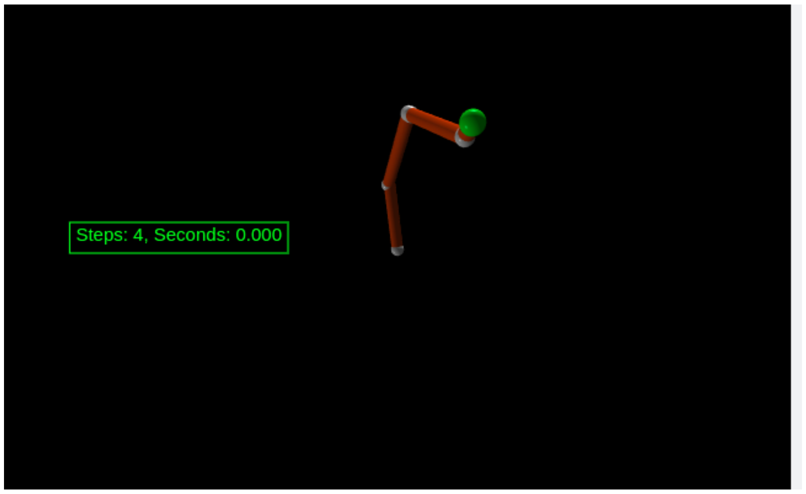

Figure 13. Example8