**6-DOF Manipulator Differential Inverse Kinematics**

**1. Purpose**

In this assignment you will be writing a MATLAB script to calculate a numerical solution to the inverse kinematics problem for the DENSO 6-DOF manipulator. In lab 5, you will be writing a PAC script using this MATLAB code for the DENSO 6-DOF manipulator to pick up a Plexiglas cylinder at a given location and to place it at a second location.

**2. Objectives**

1. Calculate position and orientation geometric Jacobian of the 6DOF manipulator.
2. Implement an inverse differential kinematics algorithm that uses linear position error and quaternion based orientation error.

**3. Requirements**

Demonstrate to the laboratory faculty:

* A working MATLAB script that takes as input a vector of six joint angles and returns as output the geometric Jacobian of the 6DOF manipulator.
* A working MATLAB script that takes as input the end-effector position vector and the minimal representation of the end-effector orientation based on **absolute XYZ rotations**, and returns as output a vector of the corresponding joint angles. The script must also plot the magnitude of the position and orientation errors as function of the iteration number.

**4. Resources**

The kinematic characteristics of the 6DOF manipulator are:

* Joint limits for the *first revolute* joint : ± 160°.
* Joint limits for the *second revolute* joint : ± 120°.
* Joint limits for the *third revolute* joint : 19°+160°.
* Joint limits for the *fourth revolute* joint : ± 160°.
* Joint limits for the *fifth revolute* joint : ± 120°.
* Joint limits for the *sixth revolute* joint : ± 360°.

The dimensions of the links in are given in Figure 1 and Figure 2.

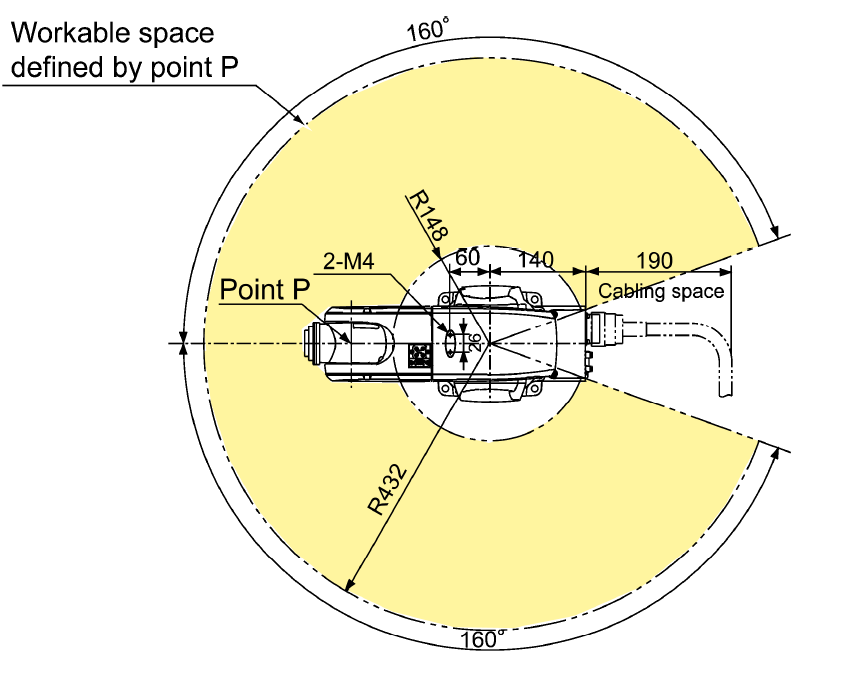


Figure 1: Top View ().

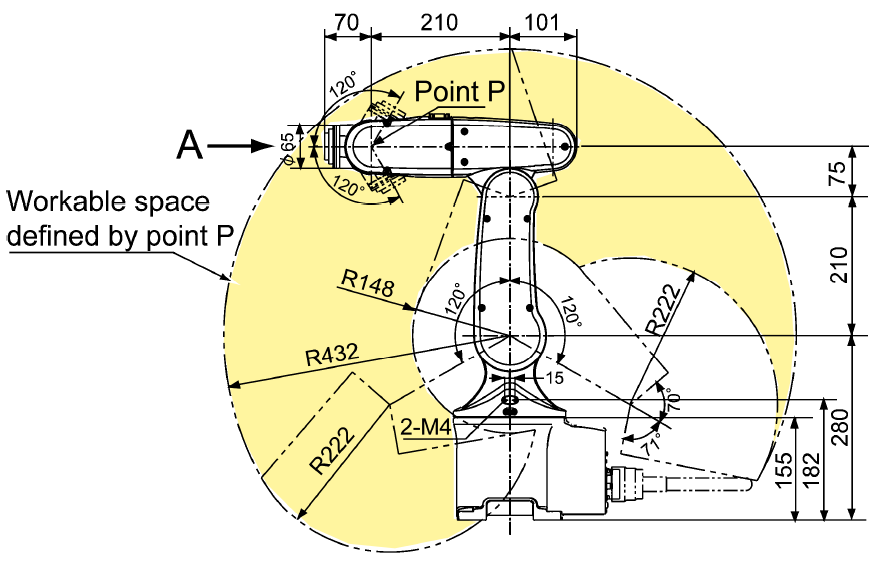


Figure 2: Lateral View ().

**5. Procedure**

After deriving the Denavit-Hartenberg table for the given manipulator, complete the following three steps.

**Part 1**: Write a MATLAB function J = sixDofJ(q) that takes as input the column vector containing the values of the six joint angles and returns the (6x6) geometric Jacobian matrix of the 6DOF manipulator.

To calculate the geometric Jacobian use the procedure covered in ME302 where you define each column of using the rule for revolute joints. Note that in this case your forward kinematics needs to be derived using the Denavit-Hartenberg notation.

If the input vector is used, then the geometric Jacobian is (**note that all the angles must be in radians for this assignment**!):

|  |  |
| --- | --- |
|  | (5) |

**Part 2:** Write a MATLAB function [o, phi] = sixDofFK(q)that takes the joint values (q) and returns the end-effector position (o) and orientation in absolute X, Y, and Z angles (phi). You can test your code using the following data:

,

**Part 3**: Write a MATLAB function [q, posErr, oriErr] = sixDofIK(o, phi, q0) that takes as input the desired end-effector position vector , the minimal representation of the end-effector orientation based on **absolute XYZ rotations**, and an initial guess for the inverse kinematic solution, and returns as output:

* The vector q containing the six joint variables.
* The vector posErr containing the magnitude of the position error as function of the iteration number.
* The vector oriErr containing the magnitude of the quaternion based orientation error as function of the iteration number.

Start testing your implementation with a value , and observe the plots of and vs. iteration to verify if the algorithm is converging or not. You might need to adjust the values and for fast convergence. A diagonal 3x3 matrix with diagonal entries between 1 to 10 can be a good start. As test inputs use the values of and calculated for thetest vector given in Part 1. Notice that the inverse kinematics of a 6DOF manipulator is not unique, and you can have up to 8 different solutions (figures) depending on your choice of .

The Denavit-Hartenberg table of the 6DOF manipulator is reported in Table 1. Transformations from the base frame to frame 0, and from frame 6 to the end-effector frame are both defined by identity matrices. The forward kinematics can be thus determined using the Denavit-Hartenberg notation.

Table 1: DH table of 6DOF manipulator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DH** |  |  |  |  |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |

The pseudo code of the inverse differential kinematics algorithm is provided in Figure 3.

differential\_inverse\_kinematics

Input: , desired position

, desired orientation based on absolute XYZ rotations

, maximum number of attempts for the solver

Ouput: , set of desired configurations

1. convert into
2. **For**  **to**  **do**
3. calculate and
4. calculate


8. **If**  **and**  **then**
10. **return**
11. **end if**
12. **end** **for**

Figure 3: Differential inverse kinematics pseudo code.

**6. Deliverables:**

In a zipped folder named with your first and last name and the lab number, include the following items and submit on canvas under assignments/lab report 4.

1. Lab report in pdf.
2. Working MATLAB codes that performs the tasks illustrated in parts 1 and 2 of the Procedure Section.