**Project 1:**

Joint Variables:

• Joint Position A = [-106.12, -162.47, 135.49, 205.71, 15.19, 0.01]

• Joint Position B = [ -45.48, -183.35,120.62, 238.42, -69.27, 0.01]

**Part A:**

Do you encounter any problems? Do you have any initial hypotheses why some motions are

possible, and some are not? Write about what could have impacted what you observed.

1. **Singularity**: Singularities are configurations of the robot where the Jacobian loses rank and the robot loses one or more degrees of freedom. If jointPosA or jointPosB are singular configurations, the robot might not be able to reach them, or it might not be able to move smoothly between them.
2. **Joint limits**: Every robot has limits on the angles each joint can reach. If jointPosA or jointPosB are outside these limits, the robot won't be able to reach them.
3. **Obstacles**: If there are obstacles in the robot's workspace, it might not be able to move in a straight line between two points. This could be the case even if both points are within the robot's reach.
4. **Speed and acceleration limits**: Robots also have limits on the speed and acceleration of their joints. If the movement between two points requires exceeding these limits, the robot might not be able to perform the movement.

I encountered warning signs that say a singularity is about to occur due to the robots movement.

|  |
| --- |
| % % TCP Host and Port settings % host = '127.0.0.1'; % THIS IP ADDRESS MUST BE USED FOR THE VIRTUAL BOX VM % host = '192.168.230.128'; % THIS IP ADDRESS MUST BE USED FOR THE VMWARE clear all; host = '192.168.0.100'; % THIS IP ADDRESS MUST BE USED FOR THE REAL ROBOT port = 30003; % Calling the constructor of rtde to setup tcp connection rtde = rtde(host,port);  % Setting home home = [-588.53, -133.30, 371.91, 2.2214, -2.2214, 0.00];  jointPosA = deg2rad([106.12, -162.47, 135.49, 205.71, 15.19, 0.01]); jointPosB = deg2rad([45.48, -183.35,120.62, 238.42, -69.27, 0.01]);  disp("Part A: ") rtde.movel(home);pause(4) disp("Move second") % rtde.movel(jointPosA, "joint");pause(4); disp("Move third") rtde.movel(jointPosB, "joint");pause(4); rtde.drawPath(tcp\_pose);  rtde.close; |

A screenshot of a computer

Description automatically generated

Figure 1: Code for movement

Figure 2: Error from movement

**Part B:**

Part 1:

Joint Position A = [-106.12, -162.47, 135.49, 205.71, 15.19, 0.01]

We are interested in the all the joints, which has a position in degrees. We convert these joints to radians to get:

The DH parameters for the joints are:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| = 0 | = -0.425 | = -0.3922 | = 0 | = 0 | = 0 |
|  | = 0 | = 0 | = 0.1333 | = 0.0997 | = 0.0996 |
| = | = | = | = | = | = |

The general form of the transformation matrix using Denavit-Hartenberg parameters is:

Substituting the parameters for all the joints into this formula, we get:

After substitution and simplification of this, we get:

represents the transformation from the base frame to the end effector. The position of the end effector in the base frame can be found in the top-right column of this matrix. The orientation of the end effector can be found in the top-left 3x3 submatrix.

The position of the end effector in the base frame can be found in the top-right column of the transformation matrix and the orientation of the end effector can be found in the top-left 3x3 submatrix of the transformation matrix.

|  |
| --- |
| %% Part 2 =================================================================  % Import the Robotics Toolbox  clear all  startup\_rvc;  % Define the DH parameters  % [ theta d a alpha]  L(1) = Link([ 0 0.1625 0 pi/2], 'standard');  L(2) = Link([ 0 0 -0.425 0], 'standard');  L(3) = Link([ 0 0 -0.3922 0], 'standard');  L(4) = Link([ 0 0.1333 0 pi/2], 'standard');  L(5) = Link([ 0 0.0997 0 -pi/2], 'standard');  L(6) = Link([ 0 0.0996 0 0], 'standard');  % Create the robot  robot = SerialLink(L, 'name', 'UR5e');  % Display the robot model  robot.display();  Robot.plot([theta]) |

Figure 3: Code for robot model

A screenshot of a computer

Description automatically generated

Figure 4: FKINE results for position A

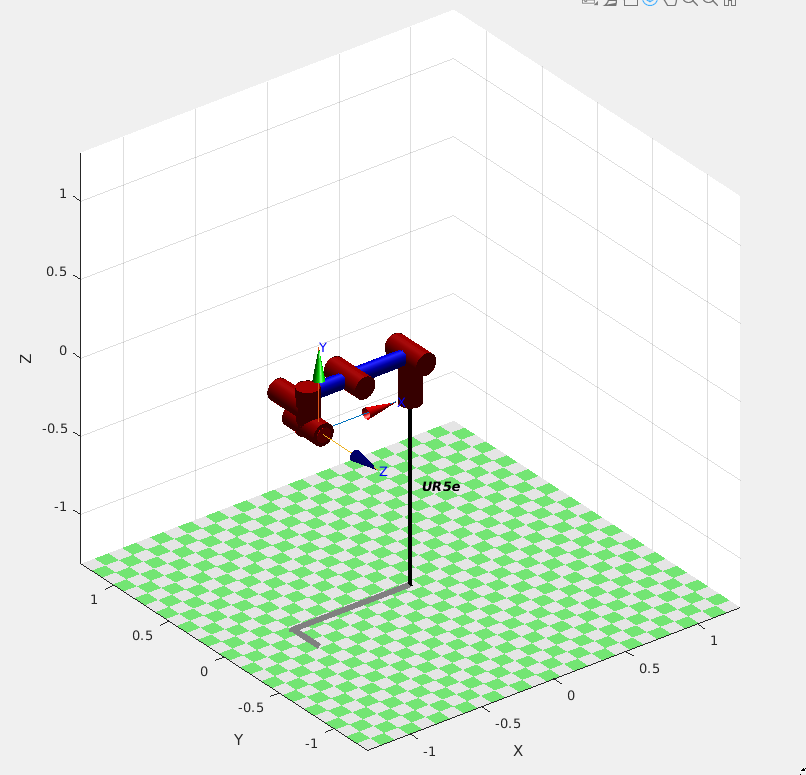


Figure 5: FKINE results for position A

|  |
| --- |
| Part 3 =================================================================  % Define the joint angles for Joint Variable A  % Convert the joint angles from degrees to radians  jointAnglesA = deg2rad([-106.12, -162.47, 135.49, 205.71, 15.19, 0.01]);  % Compute the forward kinematics  F = robot.fkine(jointAnglesA);  % Compare the matrices  isEqual = isequal(T, F);  % Display the result  disp(isEqual); |

Figure 6: Mathematically checking movement of A

**A computer screen shot of a machine

Description automatically generated**

Figure 7: Position A movement

**Part C:**

|  |
| --- |
| %% Fixed Code with a Via point.  clear all;  host = '127.0.0.1'; % THIS IP ADDRESS MUST BE USED FOR THE VIRTUAL BOX VM  % host = '192.168.230.128'; % THIS IP ADDRESS MUST BE USED FOR THE VMWARE  % host = '192.168.0.100'; % THIS IP ADDRESS MUST BE USED FOR THE REAL ROBOT  port = 30003;  % Calling the constructor of rtde to setup tcp connection  rtde = rtde(host,port);  % Setting home  home = [-588.53, -133.30, 371.91, 2.2214, -2.2214, 0.00];  jointPosA = deg2rad([106.12, -162.47, 135.49, 205.71, 15.19, 0.01]);  jointPosB = deg2rad([45.48, -183.35,120.62, 238.42, -69.27, 0.01]);  % Define via point as halfway between jointPosA and jointPosB  viaPoint = deg2rad([-85,-155,82,150,15,0.01]);  % Move to home position  [pose2,a2,b2,c2,d2] = rtde.movel(jointPosA, "joint");  [pose3,a3,b3,c3,d4] = rtde.movel(viaPoint, "joint");  [pose4,a4,b4,c4,d4] = rtde.movel(jointPosB, "joint");  [pose5,a5,b5,c5,d5] = rtde.movel(viaPoint, "joint");  [pose6,a6,b6,c6,d6] = rtde.movel(jointPosA, "joint");  % pose7 = rtde.movej(home);  tcp\_poses = [pose2;pose3;pose4;pose5;pose6];  rtde.drawPath(tcp\_poses);  rtde.close; |

**Part D:**

The root cause preventing the robot from completing a specified motion in this case is likely a singularity. A singularity is a configuration of the robot where the joints align in such a way that the robot loses one degree of freedom and cannot move in certain directions. This can happen, for example, when the robot's arm is fully extended or when two joint axes align.

In the context of robotic arms, singularities can occur in several situations:

* Wrist singularity: This occurs when the wrist center aligns with the desired end-effector position. In this case, the robot cannot maintain the end-effector orientation while moving the end-effector position.
* Shoulder singularity: This occurs when the shoulder joint is fully extended or fully retracted. In this case, the robot loses a degree of freedom in moving the end-effector position.
* Elbow singularity: This occurs when the elbow joint is fully extended or fully retracted. Similar to the shoulder singularity, the robot loses a degree of freedom in moving the end-effector position.

Mitigation methods for singularities include:

* Redesigning the task: If possible, the task can be redesigned so that the robot does not need to move into a singular configuration.
* Using redundancy: If the robot has more degrees of freedom than necessary for the task (i.e., it is redundant), these extra degrees of freedom can be used to avoid singularities.
* Using software solutions: Some robot control software can detect when the robot is approaching a singularity and automatically adjust the robot's motion to avoid it.
* Using via points: As you did in this task, you can define via points that the robot should move through to avoid singularities. The via points should be chosen so that they are not in singular configurations.

Remember that singularities are not inherently bad and can sometimes be used to the robot's advantage. However, they do need to be managed carefully to prevent unexpected robot behaviour.

**Part E:**

Obtain the DH Matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kinematics | Theta [rad] | A[m] | D[m] | Alpha [rad] |
| Joint 1 |  |  |  | 0 |
| Joint 2 |  |  | 0 | 0 |
| Joint 3 | 0 | 0 |  | 0 |

Calculate the Jacobian:

|  |
| --- |
| % Define the symbolic variables  clear all  syms theta1 theta2 d3 L1 L2 d1 d4 real  % DH parameters  theta = [theta1, theta2, 0]; % theta values in radians  a = [L1, L2, 0]; % a values in meters  d = [d1, 0, (d3-d4)]; % d values in meters  alpha = [0, 0, 0]; % alpha values in radians  % Initialize transformation matrix  T = eye(4);  for i = 1:3  % Calculate the individual transformation matrix  A = [cos(theta(i)), -sin(theta(i))\*cos(alpha(i)), sin(theta(i))\*sin(alpha(i)), a(i)\*cos(theta(i));  sin(theta(i)), cos(theta(i))\*cos(alpha(i)), -cos(theta(i))\*sin(alpha(i)), a(i)\*sin(theta(i));  0, sin(alpha(i)), cos(alpha(i)), d(i);  0, 0, 0, 1];  disp("A" + i)  disp(A)  % Multiply the overall transformation matrix by the individual transformation matrix  T = T \* A;  end  disp("Transformation: ")  T = simplify(T);  disp(T)  % Assuming you have the transformation matrix T  % Extract the position vector from the transformation matrix  p = T(1:3, 4);  % Calculate the Jacobian matrix  J = simplify([diff(p, theta1), diff(p, theta2), diff(p, d3)]) |

Forward Kinematics between joints:

A math equations on a white background

Description automatically generated

From the abovce matrices, we can get the first 3 values of the 3rd column to form and the values of the 4th column for .

A white background with black text

Description automatically generated

Determine singularities:

To find the singularities, I need to find the conditions under which the determinant of the Jacobian matrix becomes zero. When this is set to 0, it shows that the robot will have a singularity whenever θ2 is π or 0.

A math equations on a white background

Description automatically generatedInverse kinematics: