Indian Institute of Technology, Gandhinagar

ME 691

Advanced Robotics

Semester–I, Academic Year 2023-24

Project I

# *Analysis and Simulation of a Robotic Manipulator Kuka LBR iiwa 14 R820*

By-

Ashutosh Goyal 20110029

Rajesh Kumar 20110161

## Introduction:

The field of robotics has witnessed remarkable advancements in recent years, particularly in developing and applying robotic manipulators. With their high degree of flexibility and precision, these machines have found applications across various industries, from manufacturing to healthcare. In this technical report, we delve into Project 1, which focuses on analyzing and simulating a 7-degree-of-freedom robotic manipulator, specifically, the Kuka LBR iiwa 14 R820. Our project encompasses a comprehensive analysis of the manipulator's capabilities and a detailed simulation of its performance in a real-world manipulation task.

### Motivation and Significance:

The selection of the Kuka LBR iiwa 14R820 robotic manipulator for this project is grounded in its significance within the realm of collaborative robotics. Collaborative robots, or cobots, are increasingly being integrated into industrial settings thanks to their ability to work alongside humans safely and effectively. The Kuka LBR iiwa exemplifies this collaborative nature and its potential for use in various industries.

By conducting a literature survey, we have established the motivation and significance of our chosen manipulator. We have reviewed research papers, blogs, and work from the robotics community, finding that the Kuka LBR iiwa 14 R820 is employed in various applications, including assembly, quality control, and pick-and-place operations. Its collaborative capabilities, high precision, and advanced control systems make it valuable in modern automation processes.

### Analytical Analysis Procedure

Our project takes a step-by-step approach to analyze the Kuka LBR iiwa 14 R820 comprehensively. We have developed an analytical analysis procedure that includes the following components:

- **Task Representation:** We have defined the manipulator's workspace, which is essential for understanding its reach and capabilities.

- **Position Analysis:** We have conducted an in-depth analysis of the manipulator's kinematics to determine its position, orientation, and workspace limits.

- **Velocity and Statics Analysis:** We have explored the manipulator's velocity and statics properties, which are critical for controlling its movements and ensuring stability.

- **Redundancy Resolution:** Given the seven degrees of freedom of the manipulator, we have addressed redundancy resolution to optimize its movements and task execution.

- **Stiffness Analysis:** We have assessed the stiffness of the robotic system, providing insights into its compliance and ability to interact with objects delicately.

We have employed tools such as Matlab and CopeliaSim to perform these analyses, facilitating symbolic computation, numerical analysis, and graphical representation.

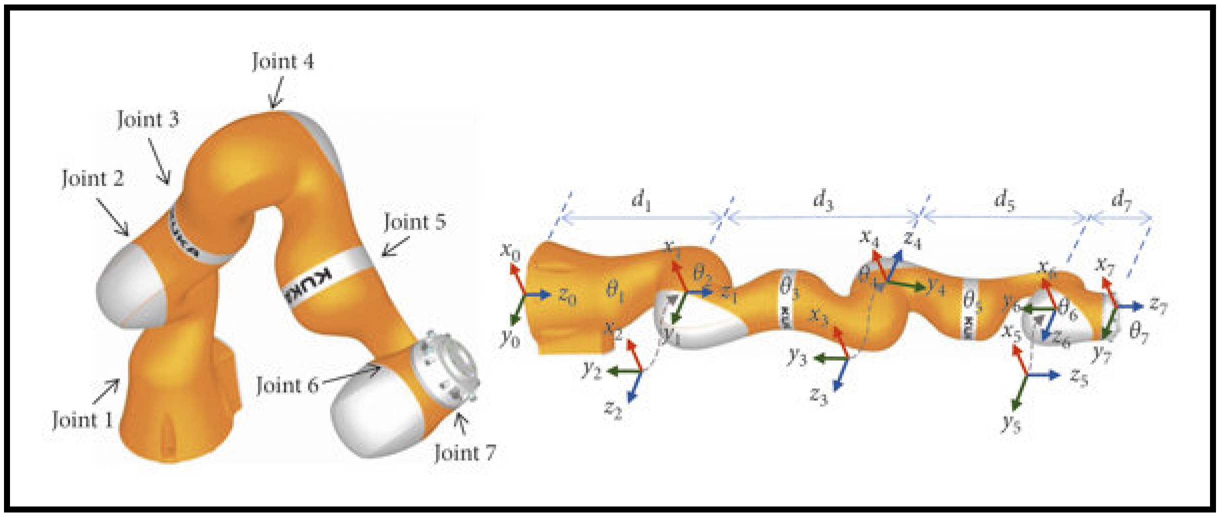
**Simulation and Implementation**

In addition to the analytical analysis, we have harnessed simulation software, specifically CoppeliaSim, to develop a virtual Kuka LBR iiwa 14 R820 virtual model. We have executed a 3-dimensional infinity trajectory within this simulated environment to showcase the manipulator's capabilities. This simulation, orchestrated through command inputs from Matlab, emulates real-world scenarios and demonstrates the versatility of the robotic manipulator.

Furthermore, we have taken a deep dive into manipulability, force, and stiffness analysis by constructing ellipsoids at various points along the trajectory. The force analysis also introduces the concept of the null space, which becomes crucial in optimizing the manipulation of a seven-degree-of-freedom manipulator.

Our project aims to comprehensively understand the Kuka LBR iiwa 14 R820, from theoretical analysis to practical simulation. Through this technical report, we aim to offer a detailed account of our findings and insights, which can contribute to the broader field of robotics and automation, particularly in the context of collaborative and flexible manipulation tasks.

## Configuration and D-H parameters of the robot:-



The configuration of the Kuka robot involves establishing a coordinate system for its various frames, with the first frame situated at the robot's base. These frames are arranged following the Denavit-Hartenberg (D-H) conventions, which are standard methods for defining the parameters of robotic manipulator joints. In our specific case, we have assigned values to these parameters as follows:

- **The first link's length, d1, is set to 0.36 units.**
- **The third link, d3, has a length of 0.42 units.**
- **The fifth link, d5, is measured at 0.4 units.**
- **Finally, the seventh link, d7, has a length of 0.126 units.**

This information is used to construct the D-H parameter table, which is crucial in defining the kinematic structure and relationships within the Kuka robot.

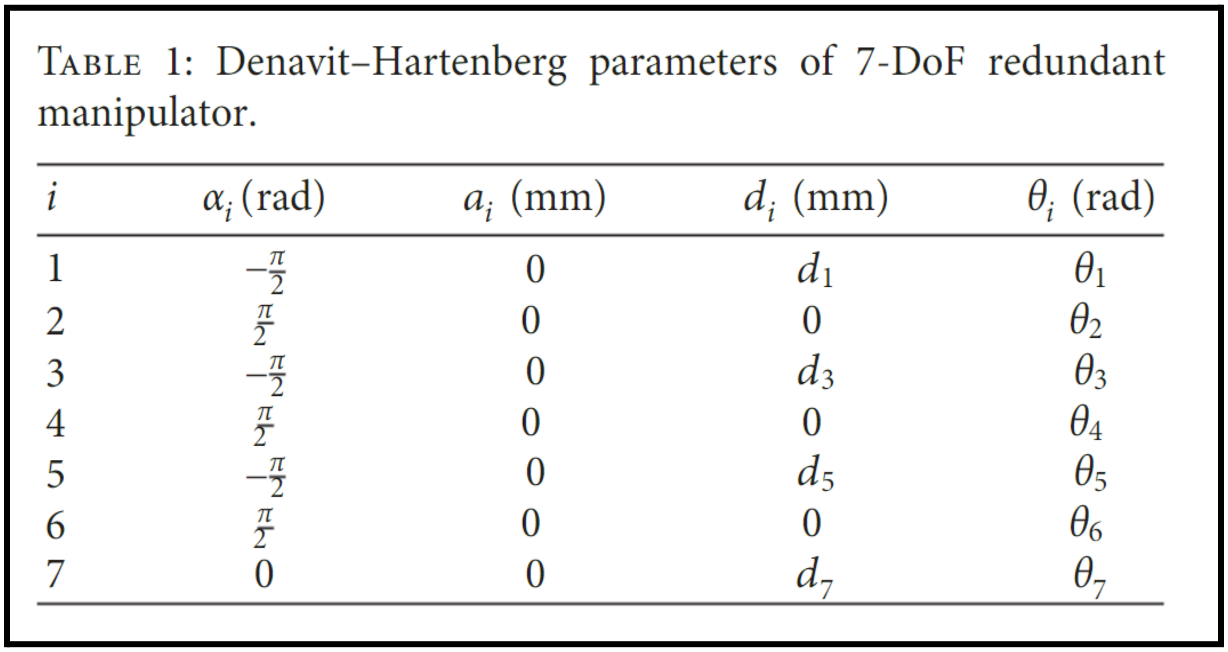


Table 1: Denavit–Hartenberg parameters of 7-DoF redundant manipulator.

| $i$ | $\alpha_i$ (rad) | $a_i$ (mm) | $d_i$ (mm) | $\theta_i$ (rad) |
|---|---|---|---|---|
| 1 | $-\frac{\pi}{2}$ | 0 | $d_1$ | $\theta_1$ |
| 2 | $\frac{\pi}{2}$ | 0 | 0 | $\theta_2$ |
| 3 | $-\frac{\pi}{2}$ | 0 | $d_3$ | $\theta_3$ |
| 4 | $\frac{\pi}{2}$ | 0 | 0 | $\theta_4$ |
| 5 | $-\frac{\pi}{2}$ | 0 | $d_5$ | $\theta_5$ |
| 6 | $\frac{\pi}{2}$ | 0 | 0 | $\theta_6$ |
| 7 | 0 | 0 | $d_7$ | $\theta_7$ |

Each $\theta_i$ represents the *ith* joint position variable. Column $d_i$ describes the kinematic link lengths. Once the DH parameters are determined, the Jacobian and the forward kinematics problem is easily solved. Four parameters are assigned to each joint, which converts to a transformation matrix that establishes the relation between one assigned reference frame *(i-1)* and the next *(i)*.

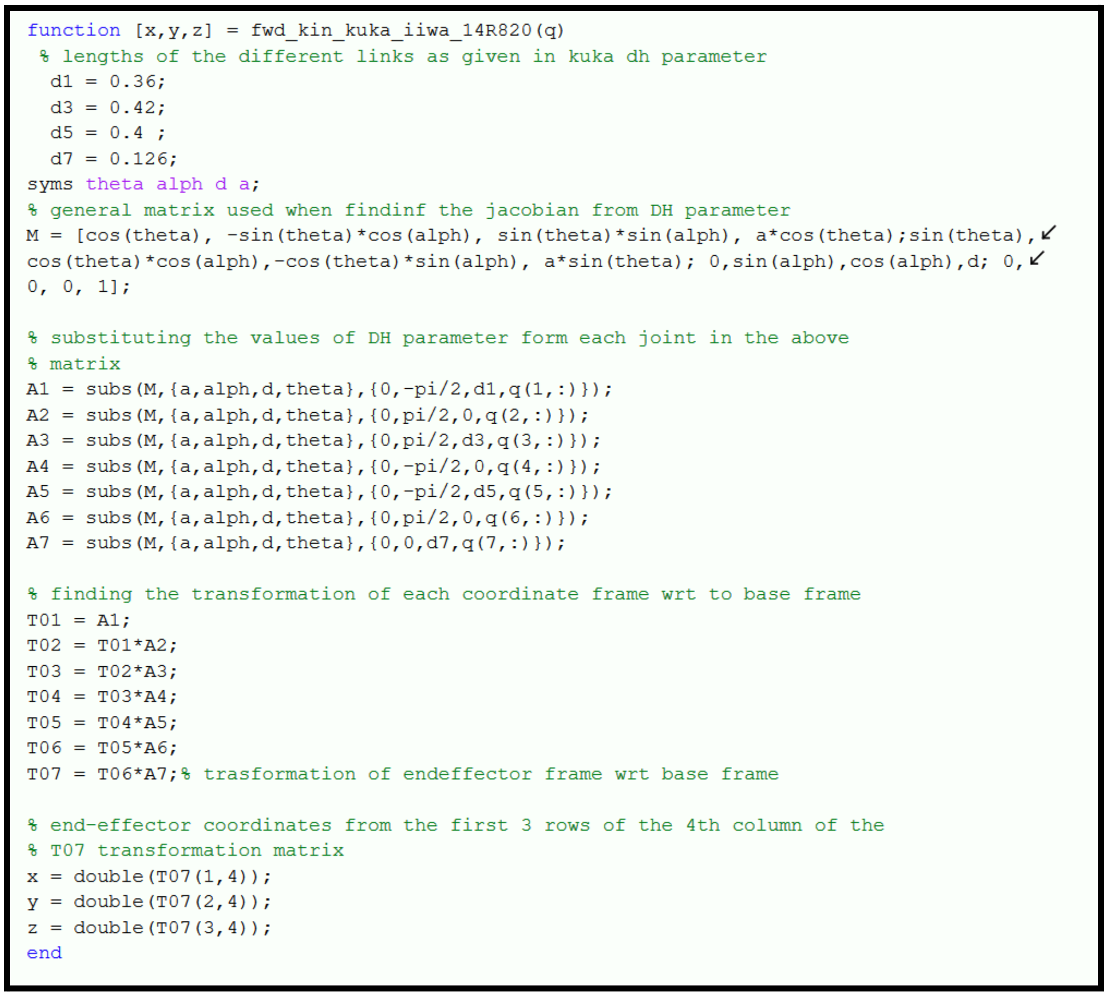
$$
{}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

The product of these matrices from the base to the flange, $^{0}\mathbf{T}_7 = {}^{0}\mathbf{T}_1{}^{1}\mathbf{T}_2{}^{2}\mathbf{T}_3{}^{3}\mathbf{T}_4{}^{4}\mathbf{T}_5{}^{5}\mathbf{T}_6{}^{6}\mathbf{T}_7$, returns the manipulator's pose in task space.

Once we get the $^{0}\mathbf{T}_7$ , we can easily define the end-effector position and orientation with respect to the base frame. Therefore, we can quickly solve for Jacobian and forward kinematics.

Here are the Matlab codes for forward kinematics and the Jacobian of the Kuka robot.

**Forward Kinematics:-**



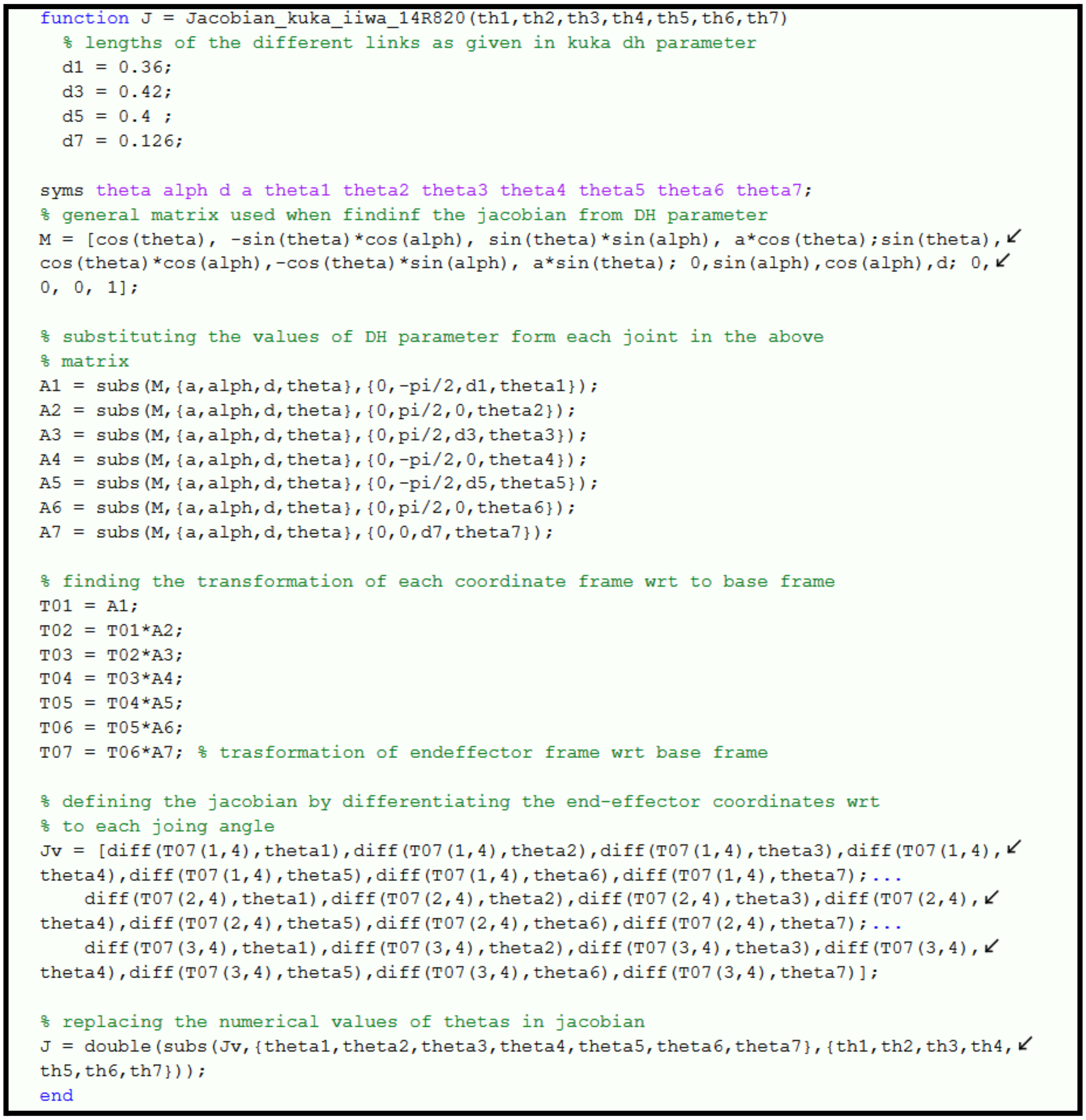
```
function [x,y,z] = fwd_kin_kuka_iiwa_14R820(q)
 % lengths of the different links as given in kuka dh parameter
  d1 = 0.36;
  d3 = 0.42;
  d5 = 0.4 ;
  d7 = 0.126;
syms theta alph d a;
% general matrix used when findinf the jacobian from DH parameter
M = [cos(theta), -sin(theta)*cos(alph), sin(theta)*sin(alph), a*cos(theta);sin(theta),↙
cos(theta)*cos(alph),-cos(theta)*sin(alph), a*sin(theta); 0,sin(alph),cos(alph),d; 0,↙
0, 0, 1];

% substituting the values of DH parameter form each joint in the above
% matrix
A1 = subs(M,{a,alph,d,theta},{0,-pi/2,d1,q(1,:)});
A2 = subs(M,{a,alph,d,theta},{0,pi/2,0,q(2,:)});
A3 = subs(M,{a,alph,d,theta},{0,pi/2,d3,q(3,:)});
A4 = subs(M,{a,alph,d,theta},{0,-pi/2,0,q(4,:)});
A5 = subs(M,{a,alph,d,theta},{0,-pi/2,d5,q(5,:)});
A6 = subs(M,{a,alph,d,theta},{0,pi/2,0,q(6,:)});
A7 = subs(M,{a,alph,d,theta},{0,0,d7,q(7,:)});

% finding the transformation of each coordinate frame wrt to base frame
T01 = A1;
T02 = T01*A2;
T03 = T02*A3;
T04 = T03*A4;
T05 = T04*A5;
T06 = T05*A6;
T07 = T06*A7;% trasformation of endeffector frame wrt base frame

% end-effector coordinates from the first 3 rows of the 4th column of the
% T07 transformation matrix
x = double(T07(1,4));
y = double(T07(2,4));
z = double(T07(3,4));
end
```

Here, we know the end effector homogeneous matrix with respect to the base frame, and we know that

$$H = \left[\begin{array}{ccc|c} & R & & P \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Hence, We can easily find the end-effector position and orientation.

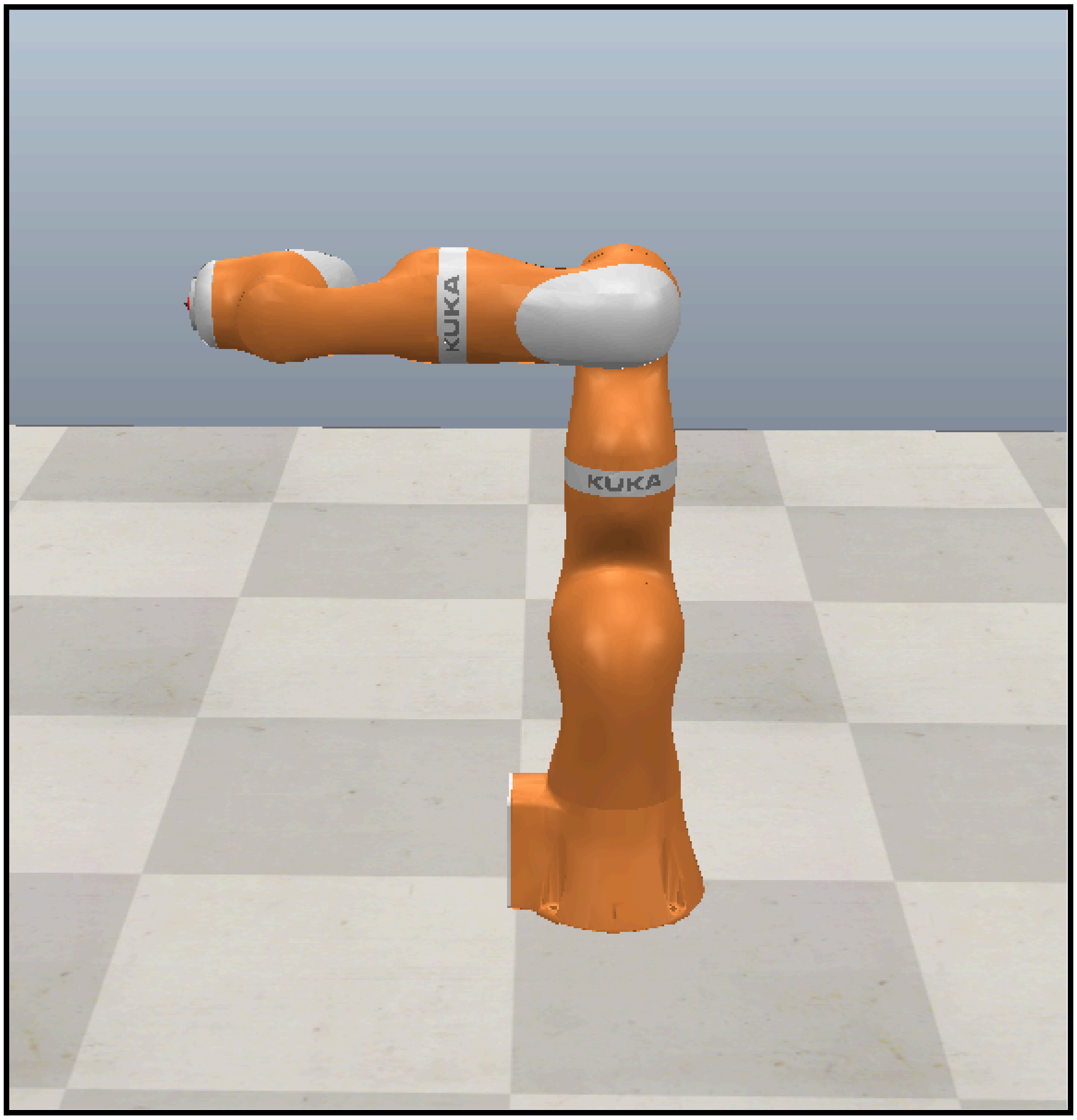
**Jacobian:-**



```
function J = Jacobian_kuka_iiwa_14R820(th1,th2,th3,th4,th5,th6,th7)
  % lengths of the different links as given in kuka dh parameter
  d1 = 0.36;
  d3 = 0.42;
  d5 = 0.4 ;
  d7 = 0.126;

syms theta alph d a theta1 theta2 theta3 theta4 theta5 theta6 theta7;
% general matrix used when findinf the jacobian from DH parameter
M = [cos(theta), -sin(theta)*cos(alph), sin(theta)*sin(alph), a*cos(theta);sin(theta),↙
cos(theta)*cos(alph),-cos(theta)*sin(alph), a*sin(theta); 0,sin(alph),cos(alph),d; 0,↙
0, 0, 1];

% substituting the values of DH parameter form each joint in the above
% matrix
A1 = subs(M,{a,alph,d,theta},{0,-pi/2,d1,theta1});
A2 = subs(M,{a,alph,d,theta},{0,pi/2,0,theta2});
A3 = subs(M,{a,alph,d,theta},{0,pi/2,d3,theta3});
A4 = subs(M,{a,alph,d,theta},{0,-pi/2,0,theta4});
A5 = subs(M,{a,alph,d,theta},{0,-pi/2,d5,theta5});
A6 = subs(M,{a,alph,d,theta},{0,pi/2,0,theta6});
A7 = subs(M,{a,alph,d,theta},{0,0,d7,theta7});

% finding the transformation of each coordinate frame wrt to base frame
T01 = A1;
T02 = T01*A2;
T03 = T02*A3;
T04 = T03*A4;
T05 = T04*A5;
T06 = T05*A6;
T07 = T06*A7; % trasformation of endeffector frame wrt base frame

% defining the jacobian by differentiating the end-effector coordinates wrt
% to each joing angle
Jv = [diff(T07(1,4),theta1),diff(T07(1,4),theta2),diff(T07(1,4),theta3),diff(T07(1,4),↙
theta4),diff(T07(1,4),theta5),diff(T07(1,4),theta6),diff(T07(1,4),theta7);...
    diff(T07(2,4),theta1),diff(T07(2,4),theta2),diff(T07(2,4),theta3),diff(T07(2,4),↙
theta4),diff(T07(2,4),theta5),diff(T07(2,4),theta6),diff(T07(2,4),theta7);...
    diff(T07(3,4),theta1),diff(T07(3,4),theta2),diff(T07(3,4),theta3),diff(T07(3,4),↙
theta4),diff(T07(3,4),theta5),diff(T07(3,4),theta6),diff(T07(3,4),theta7)];

% replacing the numerical values of thetas in jacobian
J = double(subs(Jv,{theta1,theta2,theta3,theta4,theta5,theta6,theta7},{th1,th2,th3,th4,↙
th5,th6,th7}));
end
```

Similarly, we can obtain the Jacobian matrix by using the D-H parameters.

Now, we have both the Jacobian and the forward kinematics equation. We can easily simulate the robot. We used the 3-D infinity trajectory as a desired trajectory to be followed by the robot, and the initial configuration set for the robot was q=[0,0,0,pi/2,0,0,0]. The 4th joint angle was 90 degrees, and the rest of the joint angles were 0.



Initial Configuration of Robot

# Task representation

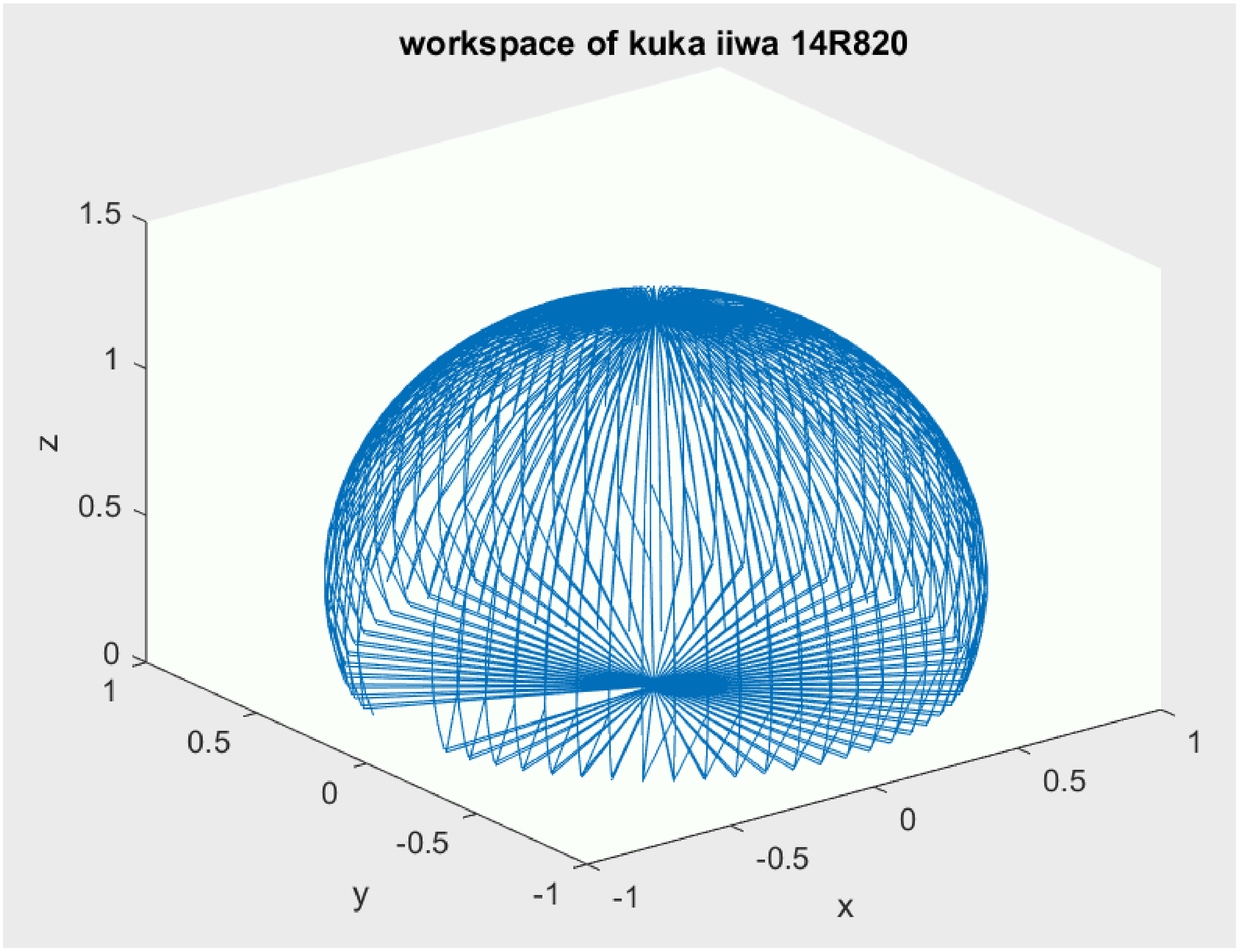
## C-space and Task space of the robot

- The C-space, also known as the joint space, represents all possible joint configurations of the Kuka robot. For a Kuka robot with seven degrees of freedom (DoF), its C-space consists of a 7-dimensional space. Each dimension corresponds to one of the seven joint angles or positions. The C-space defines the robot's mobility and the range of joint values it can take. It's often constrained by joint limits and collision considerations, making it essential for motion planning and control. The c-space of the Kuka iiwa LBR 14 R820 is $S^2 \times S^2 \times S^2 \times S^1$

- The task space, also known as the Cartesian space, represents the possible positions and orientations of the robot's end-effector in the world or workspace. It is typically a 6-dimensional space, encompassing three translational (x, y, z) and three rotational (roll, pitch, yaw) degrees of freedom. In the case of the Kuka robot, the task space is where the end-effector (the tool or gripper attached to the robot) can move and perform tasks. The task space is crucial for specifying target positions, orientation, and trajectories for the robot's end-effector, making it a key concept in path planning, control, and task execution. The task space for Kuka LBR iiwa 14 R820 can be considered as $R^3 \times S^2 \times S^1$

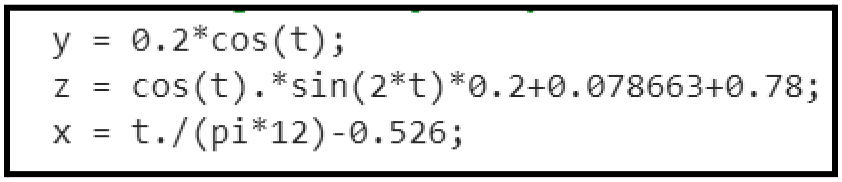
## Workspace of the robot:

- We have delineated the maximum achievable workspace for the Kuka iiwa LBR 14 R820. This workspace was constructed by systematically adjusting the first joint angles while keeping the others fixed. By doing so, we gain valuable insights into the robot's operational boundaries, enabling us to effectively define our trajectory within the confines of this workspace.



## Defined trajectory for the robot:

- We have followed an infinite trajectory in the workspace, enabling the Kuka robot to navigate complex 3-dimensional tasks. This approach allows us to conduct comprehensive analyses using null space projection techniques, including assessments for singularities and optimization.
- Equations for the defined trajectory:

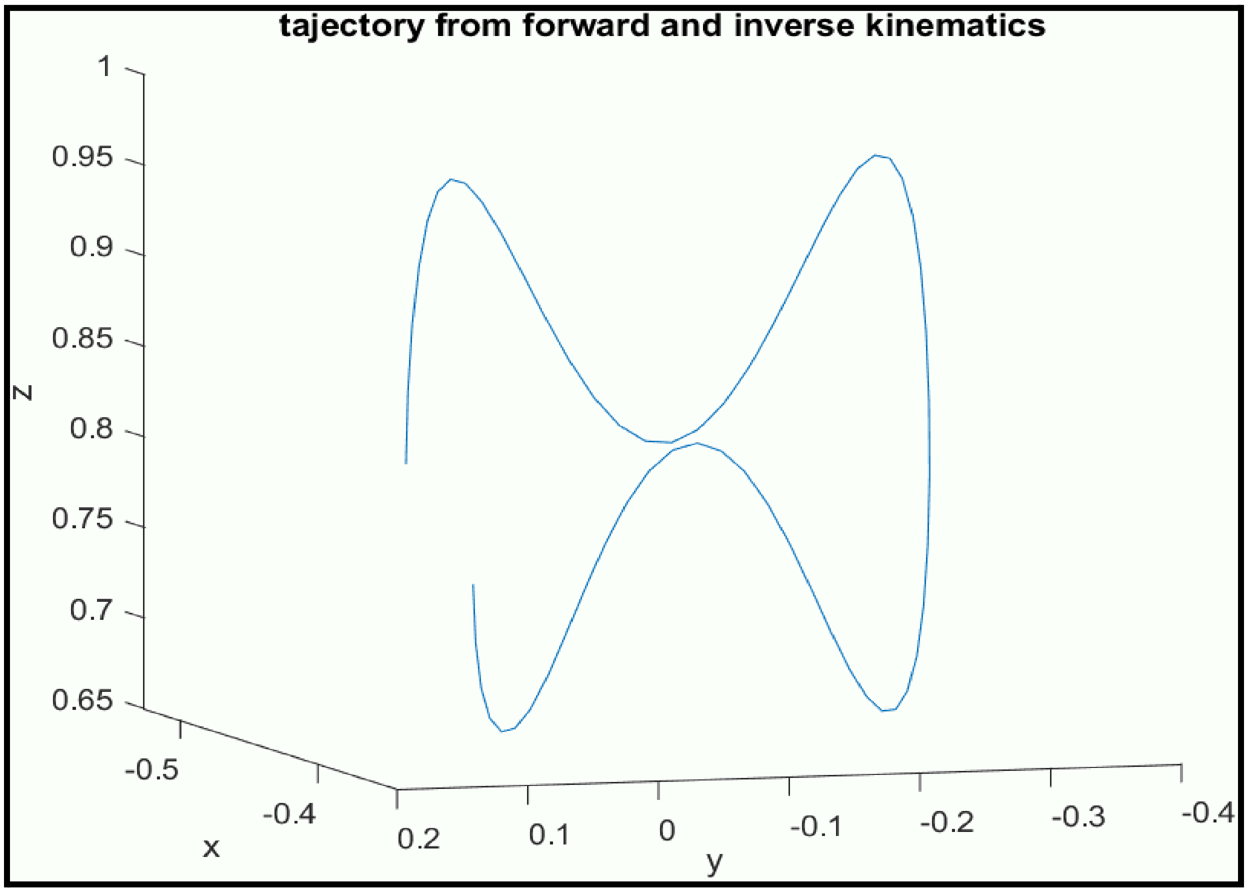


```
y = 0.2*cos(t);
z = cos(t).*sin(2*t)*0.2+0.078663+0.78;
x = t./(pi*12)-0.526;
```

In this context, we introduce a variable 't,' which spans the range from 0 to 4π, influencing the values of the x, y, and z coordinates. To ensure that the trajectory begins from the robot's end-effector position, we have set constant values of -0.526, 0.78, and 0.078663.

- The value 0.2 has been selected carefully, considering the robot's constrained workspace. This choice ensures that all points along the trajectory remain within the robot's operational boundaries.

- **Visualization of the trajectory**



*Note:- The starting point of the trajectory is x =-0.526, y=0.2, and z=0.8587 as you can see from the equations.*

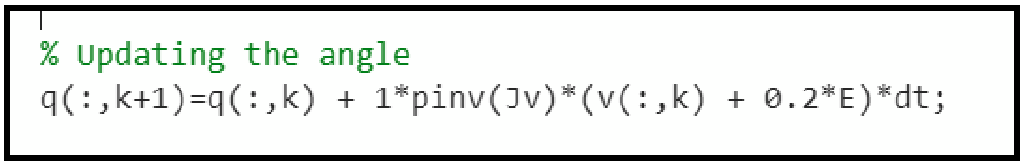
## Position analysis and Velocity analysis

In our position analysis, we have established a clear trajectory for the robot, which means we have precise knowledge of the end-effector's position at every point along the path. Initially, we initialized the robot with the joint angles set as q = [0;0;0;π/2;0;0;0], as previously defined.

To bring this trajectory to life, we employed inverse kinematics. This allowed us to calculate the joint angles required for each point on the trajectory, enabling us to simulate the robot's movement accurately. The simulation was conducted in CopeliaSim, with commands orchestrated from MATLAB to control the robot's actions.

Note: Given the redundancy of this robot, during the inverse kinematics calculation, we have deliberately chosen the minimum norm solution.

- **Numerical Formulation of the inverse kinematics:**



```
% Updating the angle
q(:,k+1)=q(:,k) + 1*pinv(Jv)*(v(:,k) + 0.2*E)*dt;
```

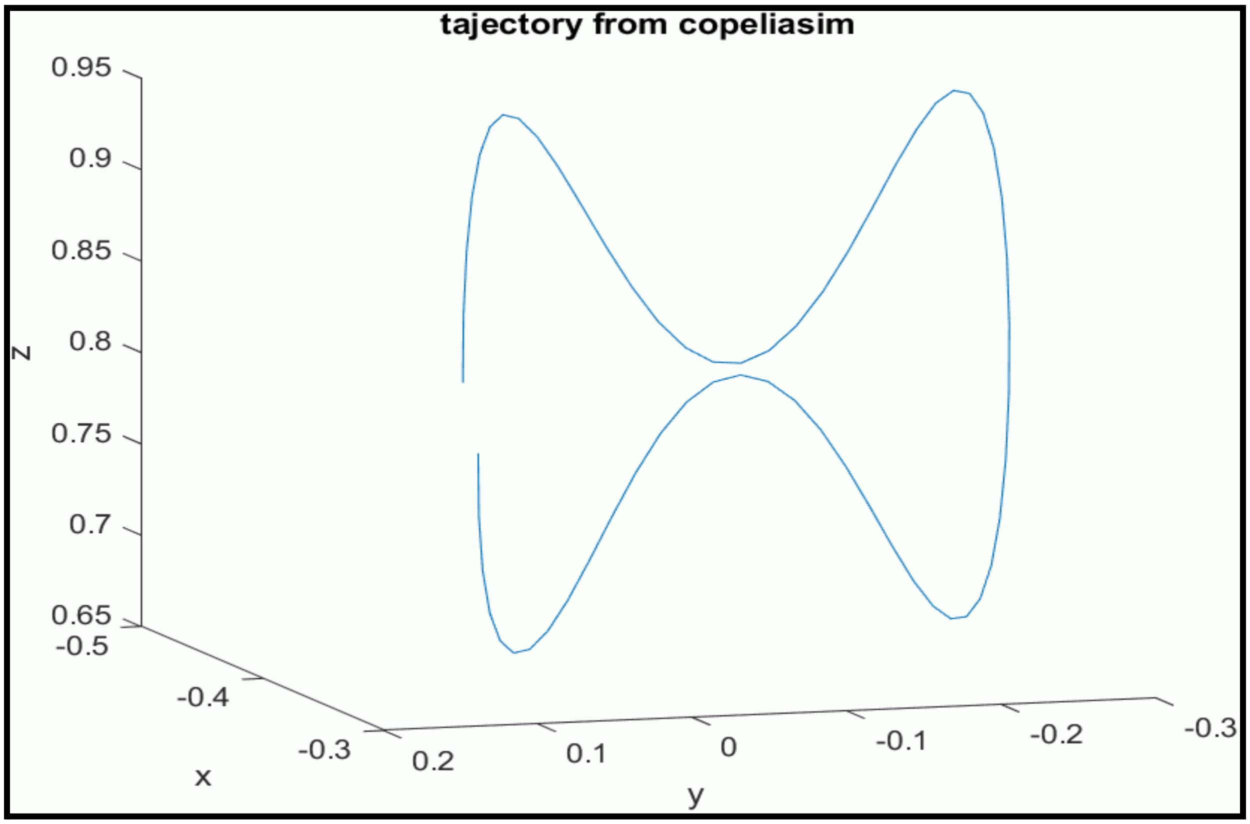
For the inverse kinematics problem, we've adopted a numerical solution approach. Given the initial configuration of the robot (q(:,k)), we can calculate the Jacobian for the robot. Additionally, as we've already defined the trajectory for the robot, we can differentiate it to obtain the end-effector velocity.

This velocity information allows us to compute the next angle of the robot, assuming a constant velocity for a very short duration (dt). This iterative process runs in a loop, continually evaluating the next joint angle vector.

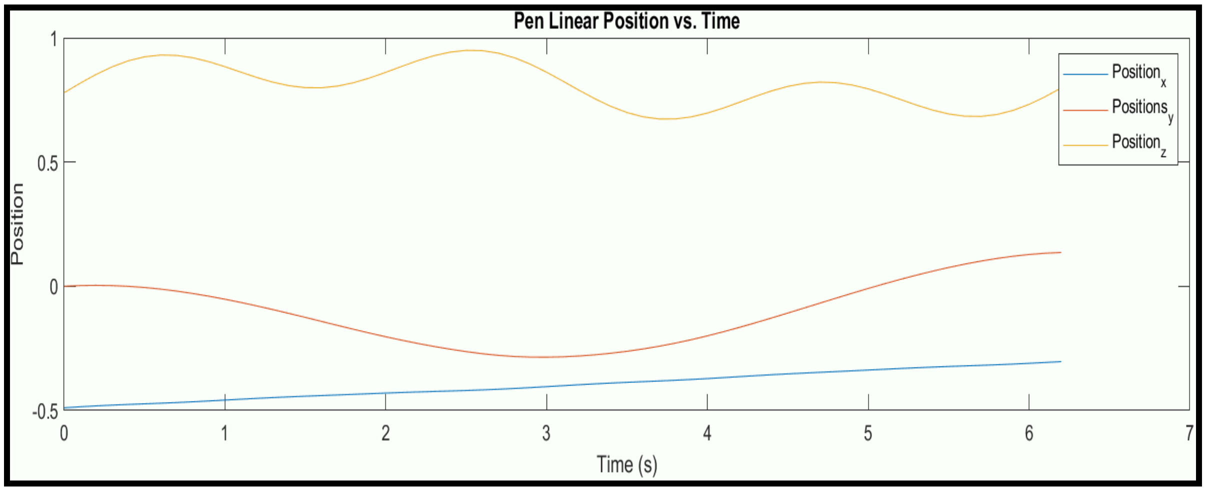
In our code, we've incorporated an error function that directly pertains to position control. We've set a gain of 0.2 for this position control. As a result, the trajectory smoothly follows the desired path, ensuring precise end-effector positioning throughout the motion.

- **Trajectory traced by the robot end effector:**

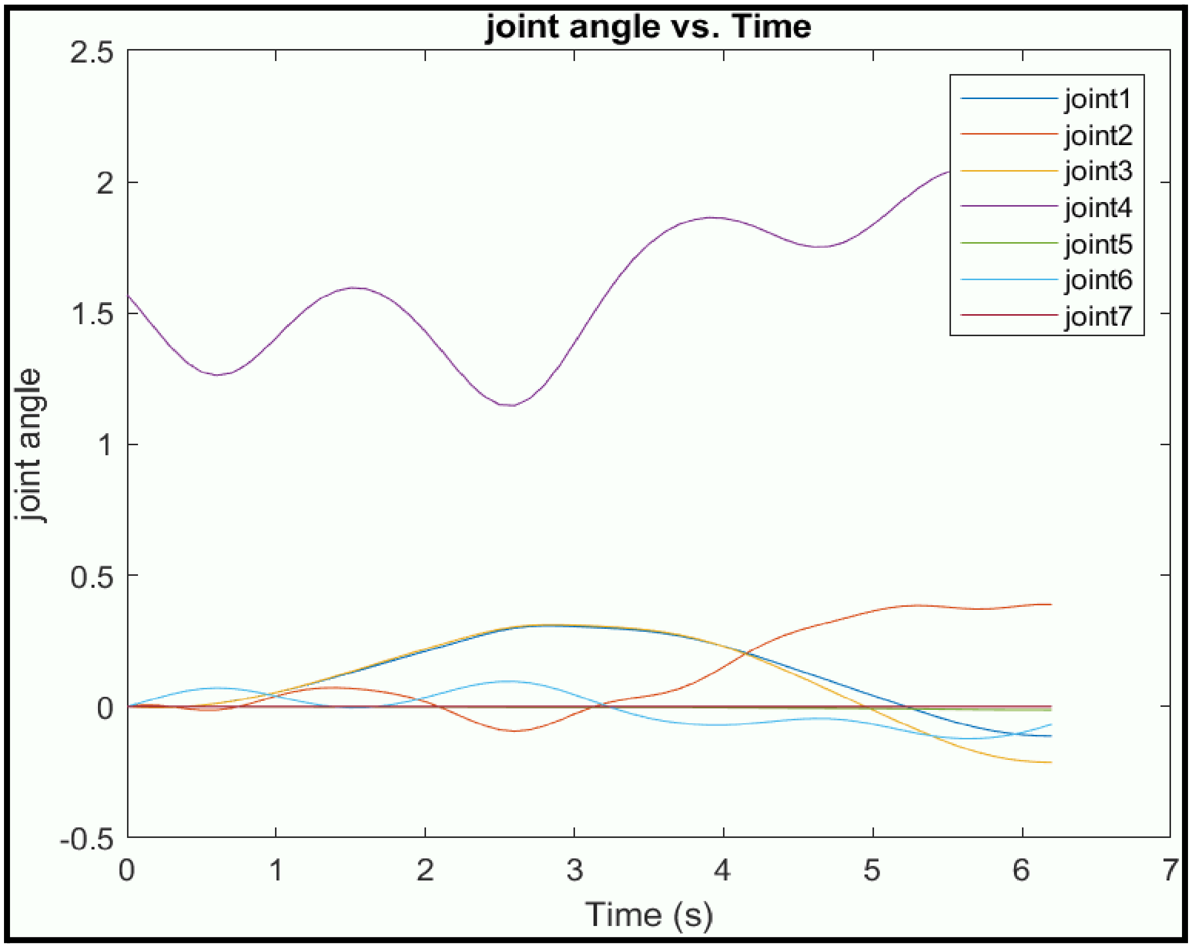
The trajectory we've obtained has been recorded directly from CopeliaSim, capturing the actual movement of the robot. Remarkably, it aligns seamlessly with the predefined trajectory we programmed for the robot to follow.



In tracing the trajectory, we've also recorded the end-effector's positions in the x-y-z coordinate frame. You can observe that the x-coordinate forms a linear path, while the y and z-coordinates exhibit sinusoidal behavior, as specified in the original trajectory. This correspondence between the expected and actual trajectory affirms the accuracy of our simulation and control.



Furthermore, we have also captured the joint angle with respect to time from CopeliaSim, and below is the representation of the joint angles.

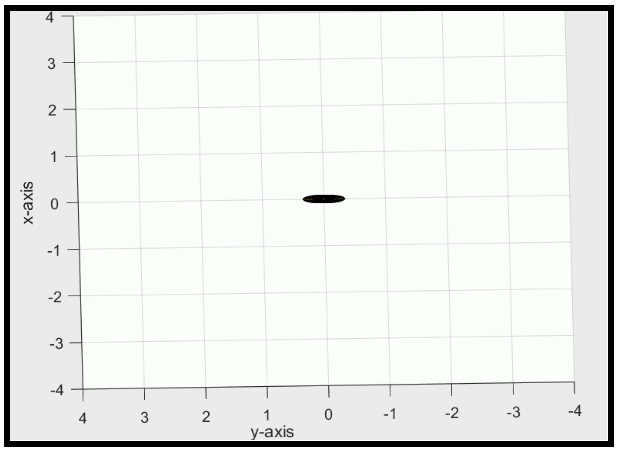


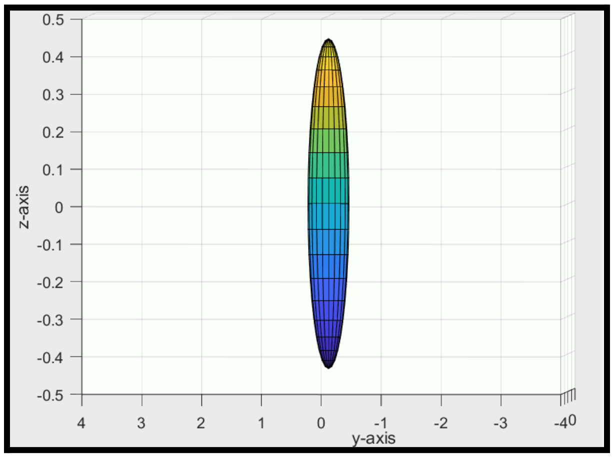
This concludes the discussion on Position and Velocity analysis.

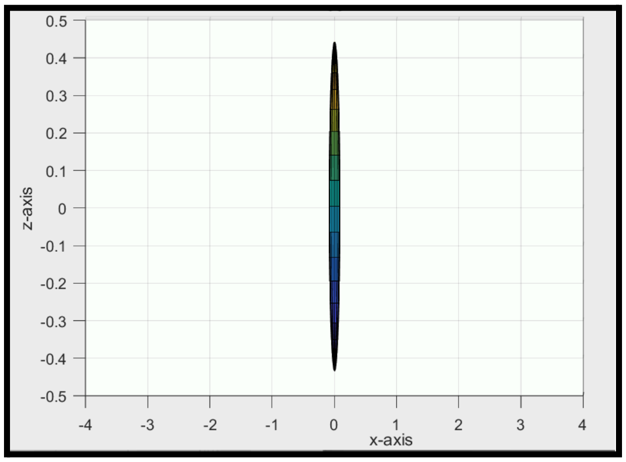
## Manipulability Ellipsoids at different points on the infinity trajectory

– *Note* all ellipsoids are drawn by taking (0,0,0) as a center, which means we have not specified the correct directions but we specify the behavior and magnitude of the ellipsoid at specific points.

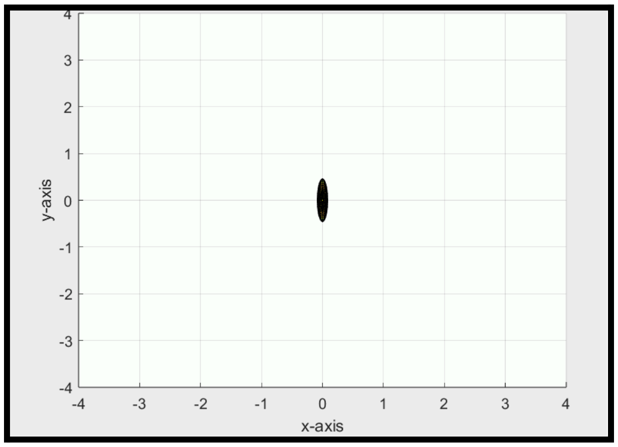
- For the lower value of manipulability - $\sqrt{det((JJ^T))}$

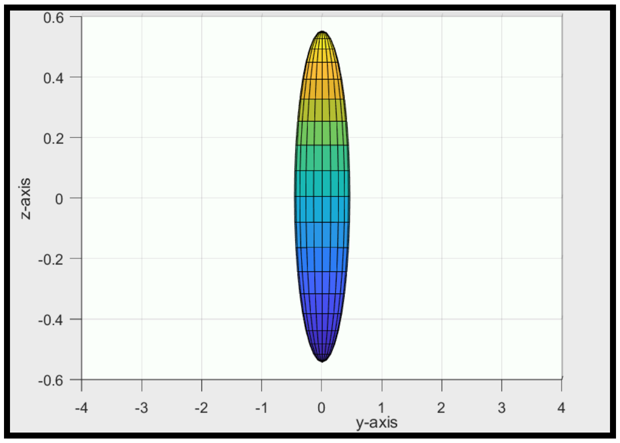


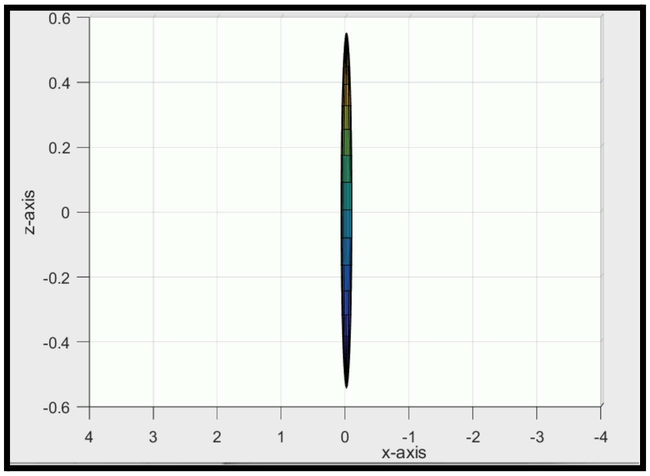




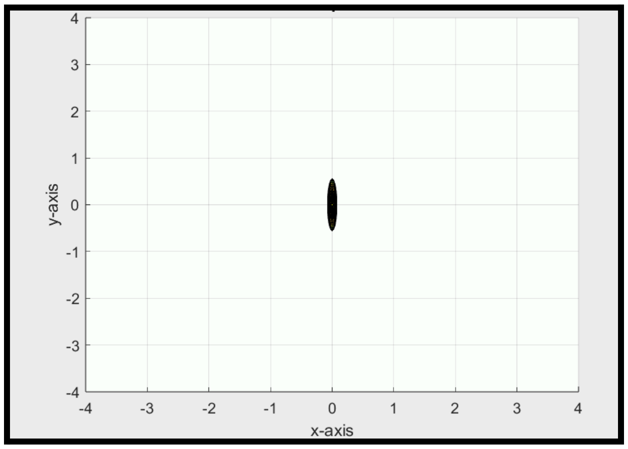
- For the middle value of manipulability - $\sqrt{det\left(\left(JJ^T\right)\right)}$

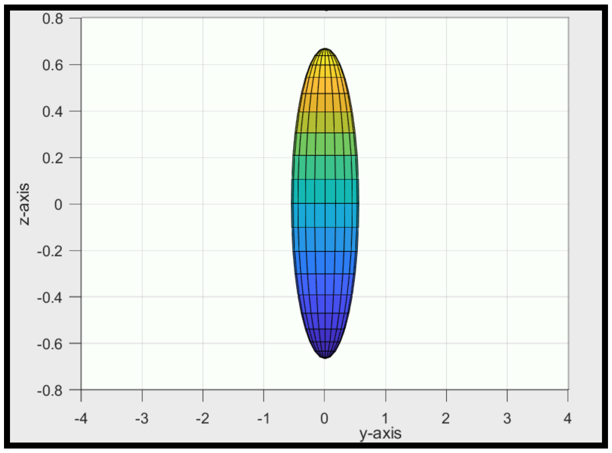


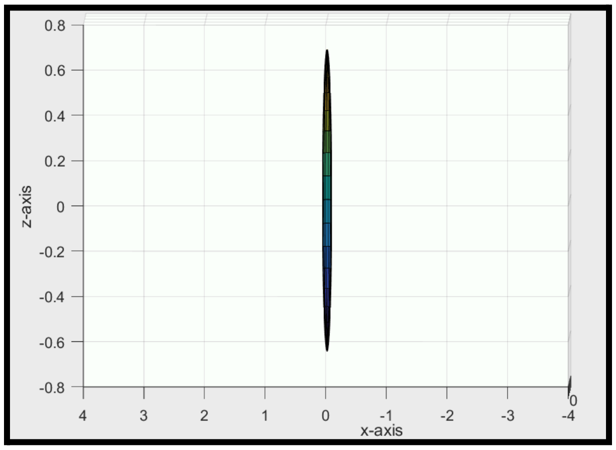




- For a high value of manipulability - $\sqrt{det\left(\left(JJ^T\right)\right)}$





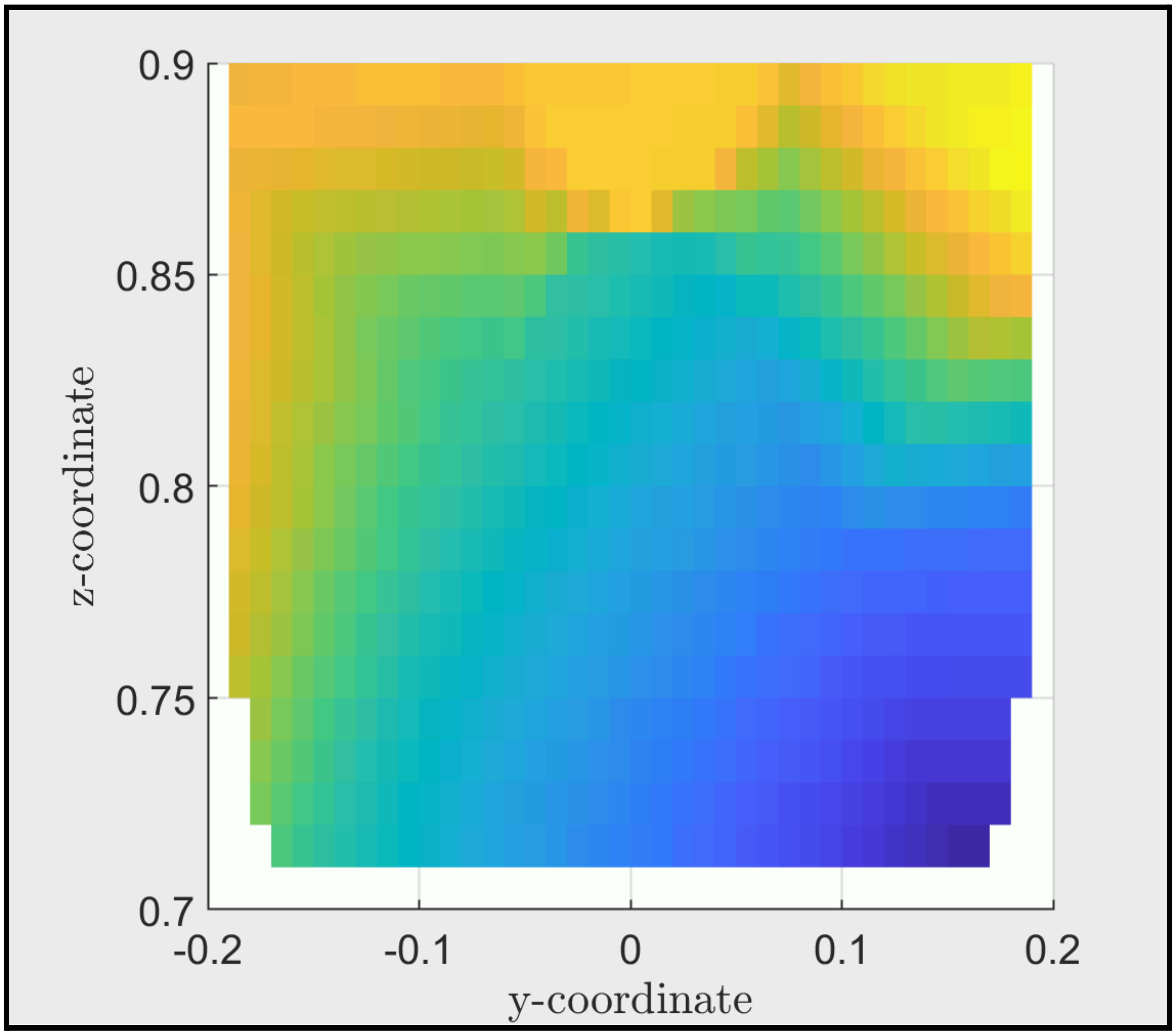


➢ The manipulability ellipsoids provide insights into the manipulability of a robotic manipulator. When the manipulability value is low, the ellipsoid representing the manipulability becomes stretched in one direction, which indicates that the manipulator has lost the freedom to move effectively in all other directions. In this situation, the robot's capability to perform tasks is constrained.

Conversely, the ellipsoid is less stretched when the manipulability value is high. This means that the manipulator is capable of movement in multiple directions. In our case, a high manipulability value suggests that the manipulator can move effectively in at least two basis vector directions, making it more versatile and capable of performing a more comprehensive range of tasks.

*– Also, note that we have given all the graph views, but for better understanding, please refer to the Matlab code.*

- Manipulability Index



*Heat map of the manipulability index*

Here, we have generated the heat map for the manipulability ellipsoid. This heat map is defined as the $\sqrt{det((JJ^T))}$ . Therefore, we have calculated this value for every point in the trajectory. This will physically provide the area of the velocity ellipsoid. The larger the area, the more the robot will be away from the singular configuration or singularity. Eigenvectors and eigenvalues of the ellipsoid give us the direction and magnitude of the maximum and minimum velocity possible. Here, we have provided this heat map in the x-y direction, but it was in 3-D. Code can be referred to properly understand the heat map. If the value of the manipulability index is 0, that particular configuration will be a singular configuration, and the robot will have limited capability in that particular configuration.

## Analytical Analysis of the inverse kinematics

The outlined inverse kinematics algorithm follows a conventional approach by splitting the manipulator into two segments: the upper arm, responsible for positioning, and the lower arm, responsible for orientation. In a similar manner to how forward kinematics calculates the final pose ($T_7^0$) and self-motion parameters (ψ, GC) based on joint positions, the inverse kinematics process reverses this, using these three variables to determine a set of joint positions.

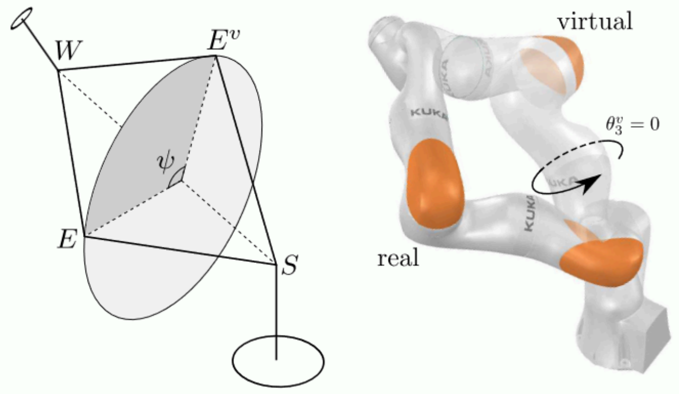
1. **Global Configuration (GC):** This parameter helps specify the branch of inverse kinematics solutions for the global configuration manifold. It involves three variables, GC2, GC4, and GC6, which express the sign of the joint angle coordinates at the shoulder, elbow, and wrist, respectively. These variables significantly influence the manipulator's arrangement in space.The GCk is given by,

$$\boxed{GC_k = \begin{cases} 1, & \text{if } \theta_k \geq 0 \\ -1, & \text{if } \theta_k < 0. \end{cases}, \forall k \in \{2,4,6\}.}$$

2. **Arm Angle (ψ):** The arm angle, introduced by Hollerbach, indicates the elbow position in the redundancy circle. It represents the angle formed by the shoulder-elbow-wrist plane (SEW) and the reference plane (SEvW). The center of

the ψ circle is located at half the distance along the line connecting the shoulder to the wrist, and it plays a crucial role in handling the manipulator's redundancy.

These parameters are determined in the forward kinematics problem from the joint angles and are subsequently used as input parameters for the inverse kinematics algorithm.



(a) Representation of the arm angle ($\psi$), as the angle between the real ($E$) and virtual elbow ($E^v$) in the redundancy circle.

(b) Configuration of the real and virtual manipulator at the same pose.

Figure 2: The "real manipulator" depicts a 7-DoF manipulator at an arbitrary pose. The "virtual manipulator" is the non-redundant (6-DoF) replica of the same manipulator reaching the same pose. The real manipulator is transformed into the virtual manipulator by blocking the 3$^{th}$ joint at zero ($\theta_3 = 0$).

For the sake of simplicity, we assumed joint 3 to be fixed at 0 degrees. As a result, our manipulator, which originally had 7 degrees of freedom (DOF), effectively operates as a 6-DOF manipulator in both joint space and task space. This simplification reduces redundancy and makes the problem more manageable. Although we're not considering the impact of joint 3 in this scenario, in future projects, we plan to address and incorporate its effects. This will allow us to gain a more comprehensive understanding of how to manage and utilize additional parameters that were not explored in this particular report.

We list the vectors from: base to shoulder $p_2^0$, shoulder to elbow $p_4^2$, elbow to wrist $p_6^4$ and wrist to flange $p_7^6$according to the DH parameters:

$$\boxed{\begin{aligned}{}^{0}\mathbf{p}_2 &= [0 \quad 0 \quad d_{bs}]^T \\ {}^{2}\mathbf{p}_4 &= [0 \quad d_{se} \quad 0]^T \\ {}^{4}\mathbf{p}_6 &= [0 \quad 0 \quad d_{ew}]^T \\ {}^{6}\mathbf{p}_7 &= [0 \quad 0 \quad d_{wf}]^T .\end{aligned}}$$

Where dbs = d1, dse = d2, dew = d3, dw f= d4

The initial step in the calculation involves determining the virtual elbow joint angle $\theta_4^v$, which relies solely on the manipulator's inherent structure and the shoulder-wrist vector. The shoulder-wrist vector is obtained through the following calculation:

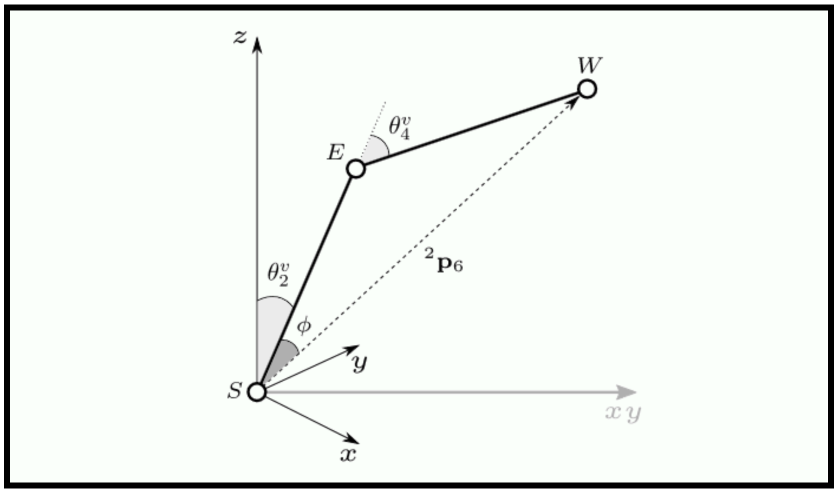
$$\boxed{{}^{2}\mathbf{p}_6 = {}^{0}\mathbf{p}_7 - {}^{0}\mathbf{p}_2 - ({}^{0}\mathbf{R}_7\, {}^{6}\mathbf{p}_7)}$$

and $\theta_4^v$ is computed using the law of cosines-

$$\boxed{\theta_4^v = GC_4 \arccos\left(\frac{\left\|{}^{2}\mathbf{p}_6\right\|^2 - (d_{se})^2 - (d_{ew})^2}{2\, d_{se}\, d_{ew}}\right). \qquad (4)}$$

Since $\theta_3^v$ is 0 , the shoulder-elbow vector ($p_4^2$), as well as the elbow-wrist vector ($p_6^4$) are aligned in the xy-plane. The joint $\theta_1^v$ is thus responsible for moving the virtual arm to the wrist x- and y- y-position coordinates. However, if the shoulder-wrist vector ($p_6^2$) is aligned to the z-axis of joint 1 ($R_{1,z}^0$ ), then joint 1 is no longer defined, and an algorithmic singularity occurs. Hence, the calculation of $\theta_1^v$ branches into,

$$\boxed{\theta_1^v = \begin{cases} \operatorname{atan2}\left({}^{2}\mathbf{p}_{6,y}, {}^{2}\mathbf{p}_{6,x}\right), & \text{if } \left\|{}^{2}\mathbf{p}_6 \times {}^{0}\mathbf{R}_{1,z}\right\| > 0 \\ 0, & \text{if } \left\|{}^{2}\mathbf{p}_6 \times {}^{0}\mathbf{R}_{1,z}\right\| = 0 \end{cases}. \qquad (5)}$$



The virtual shoulder joint $\theta_2^v$ is the last missing variable to find the virtual elbow position. As can be seen in the above figure , $\varphi$ can be calculated from the law of cosines:

$$\boxed{\phi = \arccos\left(\frac{(d_{se})^2 + \left\|{}^2\mathbf{p}_6\right\|^2 - (d_{ew})^2}{2\ d_{se}\ \ \left\|{}^2\mathbf{p}_6\right\|}\right),}$$

the value of $\theta_2^v$ which depends on the elbow configuration, is:

$$\boxed{\theta_2^v = \operatorname{atan2}\left(\sqrt{({}^2\mathbf{p}_{6,x})^2 + ({}^2\mathbf{p}_{6,y})^2},\ {}^2\mathbf{p}_{6,z}\right) + GC_4\ \phi.}$$

With $\theta_1^v, \theta_2^v, \theta_3^v$ *and* $\theta_4^v$ , we can calculate the pose of the virtual elbow from the base $T_4^0$ and $T_3^0$ using the forward kinematics. And from those we can find the corresponding rotation matrices $R_3^0$ and $R_4^0$ . Note since $\psi$ is 0 therefore

$\theta_1 = \theta_1^v, \theta_2 = \theta_2^v, \theta_3 = \theta_3^v$ *and* $\theta_4 = \theta_4^v$.

The elbow redundancy rotation matrix ($R_\psi^0$ ) codes the rotation of the angle $\psi$ around the shoulder-wrist vector $p_6^2$ Figure2a. It is calculated using Rodrigues's rotation formula in matrix notation

$$\boxed{{}^0\mathbf{R}_\psi = \mathbf{I}_3 + \sin(\psi)\left[\widehat{{}^2\mathbf{p}_6}\ \times\right] + (1-\cos(\psi))\left[\widehat{{}^2\mathbf{p}_6}\ \times\right]^2}$$

Where $[\hat{p}_6^2 \times]$ is the cross-product matrix for the unit vector $\hat{p}_6^2$.we have calculated it using the skew symmetric matrix of the unit vector $p_6^2$.

Also,

$$\boxed{\begin{aligned} {}^0\mathbf{R}_4 &= {}^0\mathbf{R}_\psi \; {}^0\mathbf{R}_4^v \\ {}^0\mathbf{R}_3 &= {}^0\mathbf{R}_\psi \; {}^0\mathbf{R}_3^v, \end{aligned}}$$

Since we have ψ = 0, therefore $R_\psi^0$= I3 and $R_3^0 = R_3^{0,v}$ and $R_4^0 = R_4^{0,v}$.

$R_3^0$ can be obtained in terms of three auxiliary matrices As, Bs and Cs from the above equations,

$$\boxed{{}^0\mathbf{R}_3 = \mathbf{A}_s \sin(\psi) + \mathbf{B}_s \cos(\psi) + \mathbf{C}_s}$$

where

$$\boxed{\begin{aligned} \mathbf{A}_s &= \left[\widehat{{}^2\mathbf{p}_6} \times\right] \; {}^0\mathbf{R}_3^v \\ \mathbf{B}_s &= -\left[\widehat{{}^2\mathbf{p}_6} \times\right]^2 \; {}^0\mathbf{R}_3^v \\ \mathbf{C}_s &= \left[\widehat{{}^2\mathbf{p}_6} \; \widehat{{}^2\mathbf{p}_6}^T\right] \; {}^0\mathbf{R}_3^v. \end{aligned}}$$

Once we know the rotation matrix relative to the shoulder joints $R_3^0$ , it is straightforward to compute the rotation matrix of the wrist joints $R_7^4$ .

$$\boxed{{}^4\mathbf{R}_7 = \mathbf{A}_w \sin(\psi) + \mathbf{B}_w \cos(\psi) + \mathbf{C}_w}$$

where,

$$\boxed{\begin{aligned} \mathbf{A}_w &= {}^3\mathbf{R}_4{}^T \; \mathbf{A}_s{}^T \; {}^0\mathbf{R}_7 \\ \mathbf{B}_w &= {}^3\mathbf{R}_4{}^T \; \mathbf{B}_s{}^T \; {}^0\mathbf{R}_7 \\ \mathbf{C}_w &= {}^3\mathbf{R}_4{}^T \; \mathbf{C}_s{}^T \; {}^0\mathbf{R}_7. \end{aligned}}$$

and now we can analytically extrapolate the joint positions of the wrist joints from the elements of the $R_7^4$(θ5,6,7) matrix,

$$
{}^4\mathbf{R}_7(\theta_{5,6,7}) = \begin{bmatrix} * & * & \cos\theta_5 \sin\theta_6 \\ * & * & \sin\theta_5 \sin\theta_6 \\ -\sin\theta_6 \cos\theta_7 & \sin\theta_6 \sin\theta_7 & \cos\theta_6 \end{bmatrix}.
$$

Respecting the global configuration parameter, we compute the remaining joint angles,

$$
\begin{aligned}
\theta_5 &= \operatorname{atan2}\left(GC_6[a_{w23}\sin\psi + b_{w23}\cos\psi + c_{w23}],\right. \\
&\qquad \left. GC_6\left[a_{w13}\sin\psi + b_{w13}\cos\psi + c_{w13}\right]\right) \\
\theta_6 &= GC_6 \arccos\left(a_{w33}\sin\psi + b_{w33}\cos\psi + c_{w33}\right) \\
\theta_7 &= \operatorname{atan2}\left(GC_6[a_{w32}\sin\psi + b_{w32}\cos\psi + c_{w32}],\right. \\
&\qquad \left. GC_6\left[-a_{w31}\sin\psi - b_{w31}\cos\psi - c_{w31}\right]\right).
\end{aligned}
$$

In conclusion, the joint angles are uniquely determined based on a target pose ( $T_7^0$), and two auxiliary parameters, the arm angle (ψ) and the joint configuration (GC).

*Code Explanation*

This code pertains to the analytical inverse kinematics for a 7-DOF Kuka LBR iiwa R820 robot with redundancy. The inverse kinematics problem involves providing the end-effector position and orientation obtained from a known homogeneous transformation matrix. This matrix encompasses both the rotational and translational components, enabling us to accurately extract the end-effector's position and orientation.

In this analytical approach, we assume one of the joint angles, specifically joint 3, to be fixed at 0 degrees. This simplification aligns with a kinematic decoupling approach, where we can independently solve for the first four joint angles to determine the translational kinematics and then focus on the remaining three joint angles to address the rotational aspects.

Following the methodology from a research paper, we acquire the first four joint angles. With these angles, we can calculate T04 and T47, which are crucial for determining R47. Subsequently, we proceed to find the last three joint angles of the robot.

To avoid certain angle configurations that could lead to instability or unexpected results, we've incorporated constraints in the calculation. For instance, when an angle should ideally be π/2 but the calculation results in -π/2, we handle this by considering the negative value of GC4, indicating that angle 4 was negative. This allows us to compute the other angles accordingly.

We've rigorously tested this analytical inverse kinematics approach by providing the robot with four configurations. The process involves calculating the homogeneous matrix using forward kinematics and then supplying this matrix to the analytical inverse kinematics. The fact that we consistently obtain the same angles through this function confirms the accuracy and reliability of our analytical inverse kinematics method.

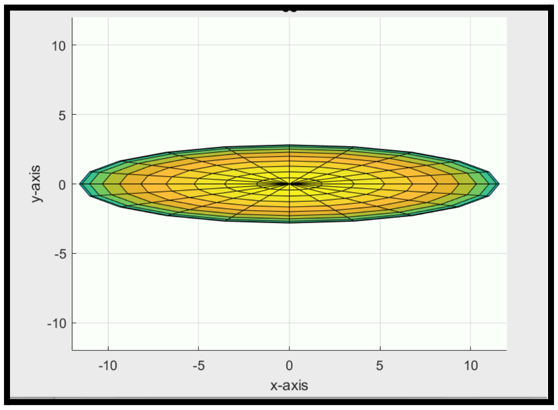
Note: The initial assumption of joint 3 being fixed at zero simplifies the problem, and the verification process has ensured the correctness of this analytical inverse kinematics solution.

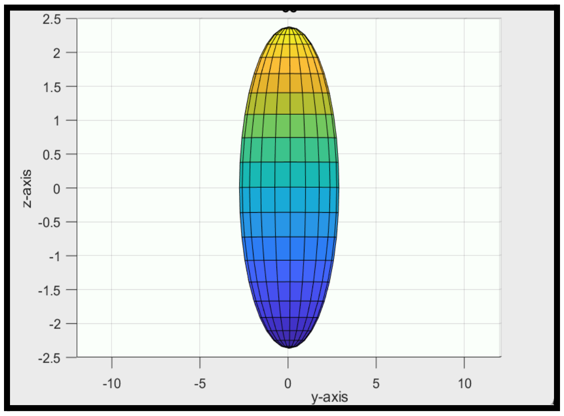
# Statics analysis and Redundancy Resolution

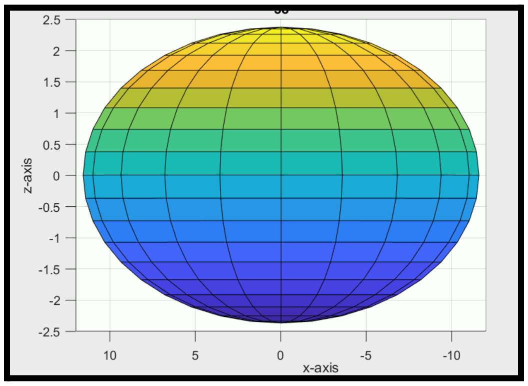
**Force Ellipsoid at different points on the infinity trajectory**

– *Note* all ellipsoids are drawn by taking (0,0,0) as a center, which means we have not specified the correct directions. Still, we specify the behavior and magnitude of the ellipsoid at specific points.

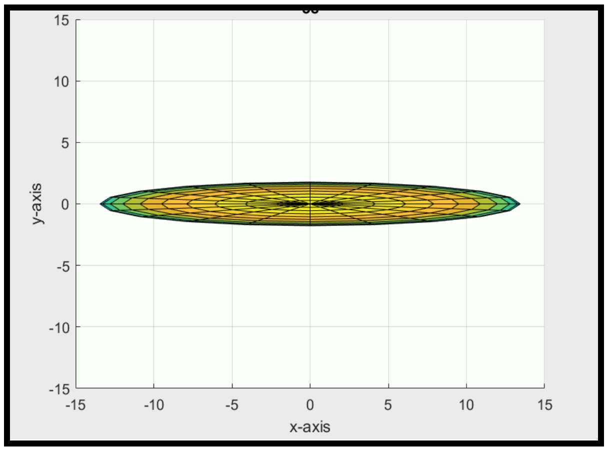
- For the high value of $\sqrt{det\left(\left(JJ^T\right)^{\dagger}\right)}$

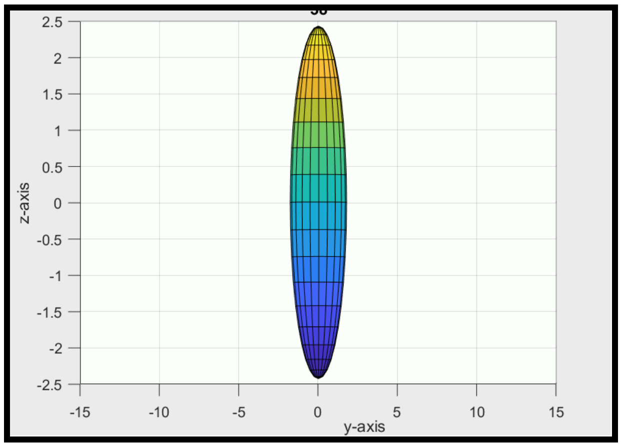


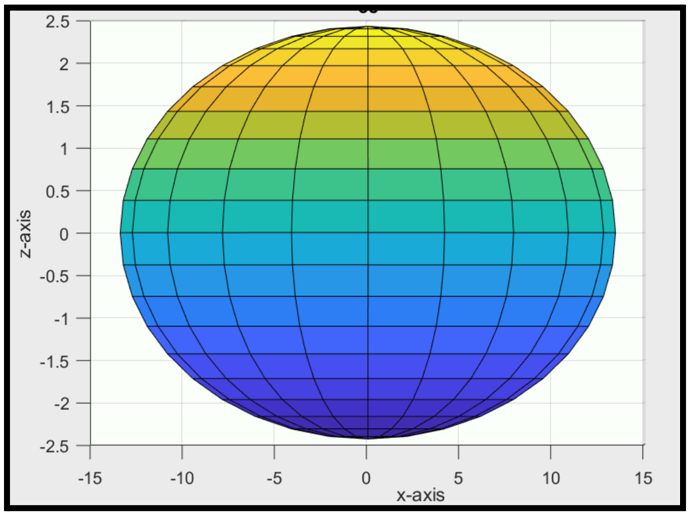




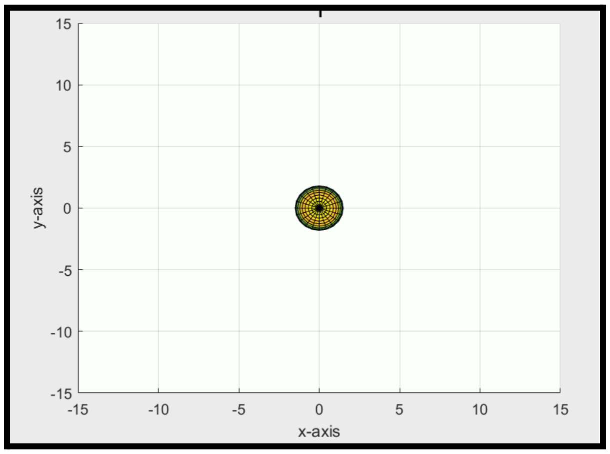
- For the mid value of $\sqrt{det\left(\left(JJ^T\right)^{\dagger}\right)}$

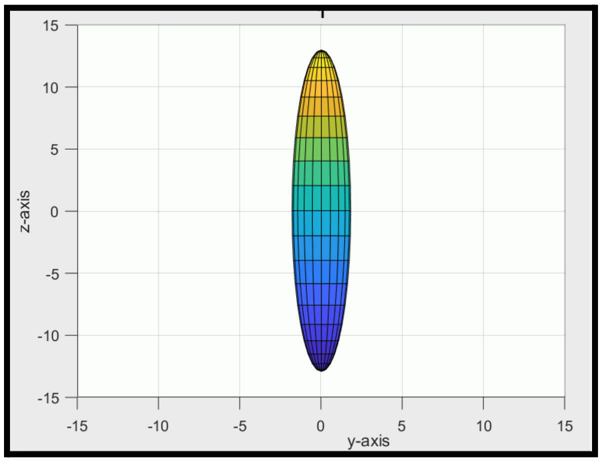


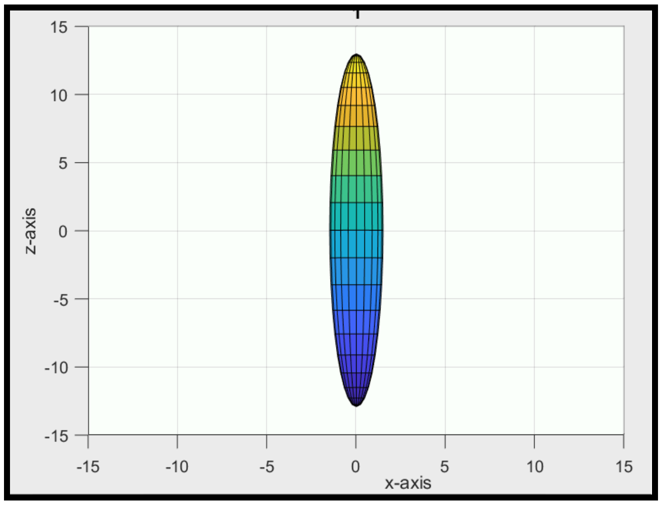




- For the low value of $\sqrt{det\left(\left(JJ^T\right)^{\dagger}\right)}$





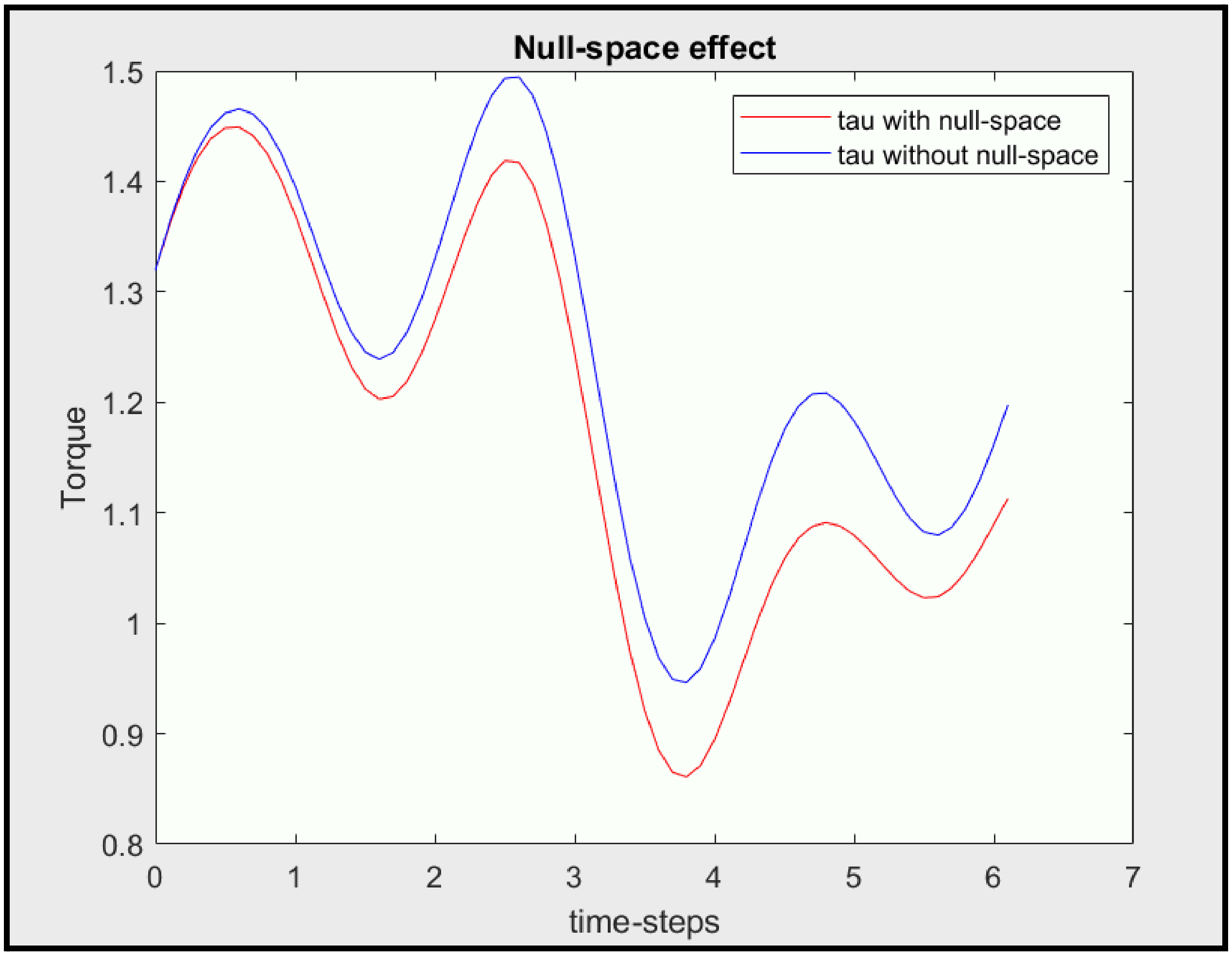


➢ The observations from the force ellipsoids at different locations along the trajectory reveal an intriguing contrast to the manipulability ellipsoids. Specifically, at certain points along the trajectory where the manipulator exhibits the capability to move its end-effector effectively in a particular direction, the force ellipsoid demonstrates an inability to apply force in that very direction.

This oppositional trend implies that there are critical points along the trajectory where the manipulator's dexterity to execute movements doesn't align with its capacity to exert force in those specific directions. In essence, the manipulator's ability to control its end-effector motion doesn't necessarily coincide with its capacity to apply force, highlighting the intricate dynamics involved in robotic manipulation.

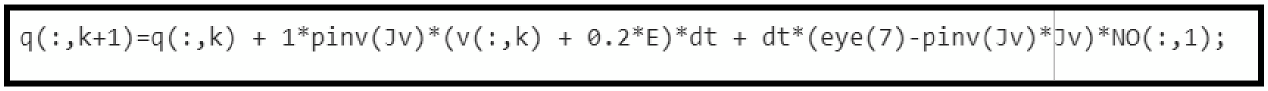
*– Also, note that we have given all the views of the graph but for better understanding please refer to the Matlab code.*

## Null space projection effect on redundant robots



In our project, we are working with a 7-degree-of-freedom (DOF) robot in a 6-degree-of-freedom (6-DOF) space, making our robot redundant. This redundancy can be advantageous, as it allows us to utilize null space projection to enhance the robot's performance. When solving for inverse kinematics, we often have multiple solutions, and if external forces are acting on the robot, they result in joint torques. To follow a trajectory while accommodating these forces, we can adjust the angular velocities of the robot's joints in a way that it can efficiently generate the required torques to counteract the external forces. We have demonstrated this process in Project 1, showcasing the benefits of utilizing the robot's redundancy to achieve desired tasks while maintaining stability in the presence of external forces.

Hence, we can update our inverse kinematic equation as



```
q(:,k+1)=q(:,k) + 1*pinv(Jv)*(v(:,k) + 0.2*E)*dt + dt*(eye(7)-pinv(Jv)*Jv)*NO(:,1);
```

Where  NO = [0.1;0.3;0.2;0.15;-0.2;0.1;-0.5]; represents the null-space projection vector. In our analysis, we have plotted the square root of the sum of the squares of all the joint torques. It is evident that through the use of null space projection, we have effectively minimized the net torque impact on the robot. This marks the conclusion of our discussion on redundancy resolution and the benefits of null space projection in enhancing the robot's performance and stability.

## Stiffness analysis

Stiffness analysis in robotics is a crucial step in understanding how compliant or rigid a robotic system is when interacting with its environment. In the case of the Kuka robot, performing stiffness analysis provides valuable insights into its ability to adapt to external forces and disturbances during tasks.

In robot manipulation, we can think of the stiffness matrix, $K_p$, as a measure of interaction between the robot end-effector tip and an object when an external force is applied to the end-effector. By using Taylor's expansion, we can write:

$$\boxed{df = \frac{\partial f}{\partial x}\, dx + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\, dx\right)\, dx}$$

which can be expressed as:

$$\boxed{K_p\, dx + \frac{1}{2}\left(\frac{\partial KE}{\partial x}\, dx\right)\, dx}$$

where $df$ and $dx$ are the force and displacement vectors in the Cartesian space relative to the robot's base frame.

If the higher-order small terms are neglected, the above equation is reduced to the familiar definition of the stiffness matrix:

$$\boxed{df = K_p\, dx}$$

The above equation represents the stiffness as an infinitesimal relationship. Additionally, the manipulation Jacobian matrix $J_\theta$ relates the differential joint displacements to the infinitesimal movement of the end-effector, i.e., $dx = J_\theta d\theta$

$$\boxed{d\theta = [d\theta_1, \ldots, d\theta_n]^T}$$

Where dθ is the displacement of the joints.

When we consider the forward kinematics of a serial robot manipulating an object under stiffness control, the stiffness matrix associated with the manipulator can be derived from the structural compliance and the joint stiffness matrix $K_\theta$

$$\boxed{\mathbf{K}_\theta = \mathbf{J}_\theta^T \mathbf{K}_p \mathbf{J}_\theta,}$$

**Derivation of the conservative congruence transformation:-**

From the principle of virtual work and the definition of the Jacobian, we obtain the following relationship between the joint torques and the force applied on the end-effector:

$$\boxed{\tau = \mathbf{J}_\theta^T \mathbf{f}.}$$

It is an important force/torque relation that enables us to determine the resulting torques, τ, reflected at the joints as a result of the force applied at the end-effector and vice versa. By the chain rule, we can differentiate the above equation.

$$\boxed{d\tau = (d\mathbf{J}_\theta^T)\,\mathbf{f} + \mathbf{J}_\theta^T\,(d\mathbf{f}).}$$

The above equation includes two separate terms. The first term is related to the changes in geometry, and the second term is related to differential external force. Using the definition of stiffness, we can formulate the above equation as

$$\boxed{\mathbf{J}_\theta^T \underbrace{\mathbf{K}_p d\mathbf{x}}_{d\mathbf{f}} = \underbrace{\mathbf{K}_\theta d\theta}_{d\tau} - \left(\frac{\partial \mathbf{J}_\theta^T}{\partial \theta} d\theta\right)\mathbf{f},}$$

Where the $\frac{\partial J_\theta^T}{\partial \theta}$ and $\frac{\partial J_\theta^T}{\partial \theta} d\theta$ are the third-order and second-order tensors or metrics and $dx = J_\theta d\theta$. Thus, we can simplify the first order tensor

$$\boxed{\begin{aligned}\left(\frac{\partial \mathbf{J}_\theta^T}{\partial \theta} d\theta\right)\mathbf{f} &= \left[\sum_{i=1}^{n} \left(\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_i} d\theta_i\right)\right]\mathbf{f} \\ &= \sum_{i=1}^{n} \left(\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_i}\mathbf{f}\right) d\theta_i \\ &= \left[ (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_1}\mathbf{f}) \;\; (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_2}\mathbf{f}) \;\; \cdots \;\; (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_n}\mathbf{f}) \right] d\theta.\end{aligned}}$$

Hence, we can rearrange the main equation.

$$\boxed{\mathbf{J}_\theta^T \mathbf{K}_p \mathbf{J}_\theta = \mathbf{K}_\theta - \left[ (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_1}\mathbf{f}) \;\; (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_2}\mathbf{f}) \;\; \cdots \;\; (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_n}\mathbf{f}) \right]}$$

Or

$$\boxed{\mathbf{J}_\theta^T \mathbf{K}_p \mathbf{J}_\theta = \mathbf{K}_\theta - \mathbf{K}_g,}$$

Where $\frac{\partial J_\theta^T}{\partial \theta_i}$ is a $n \times 1$ column vector with $i = 1. \ldots.. n$ and $n$ is the number of joints.

The $n \times n$ matrix $K_g$ defining the change in geometry via the differential jacobian is,

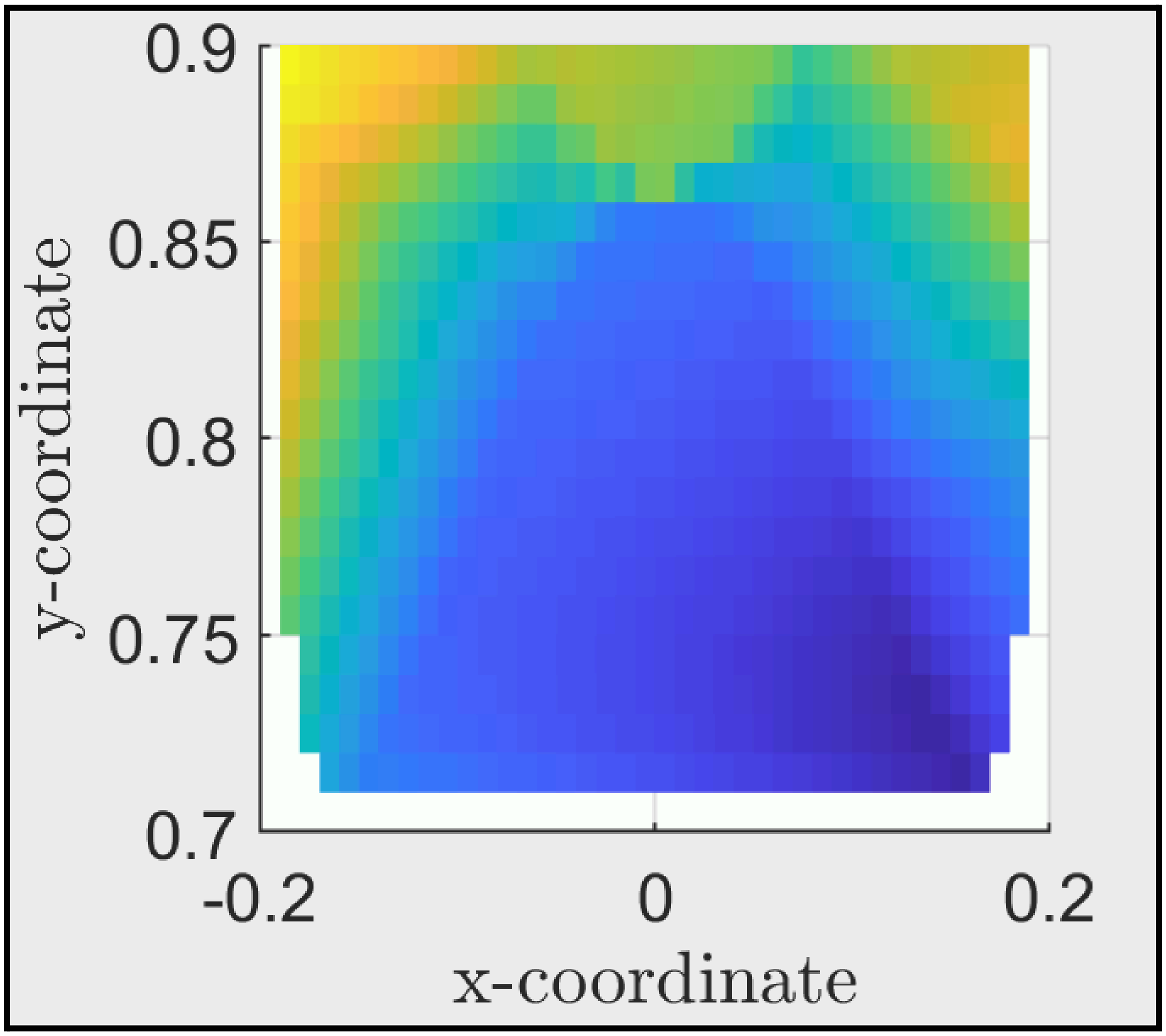
$$\boxed{\mathbf{K}_g = \underbrace{\begin{bmatrix} (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_1}\mathbf{f}) & (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_2}\mathbf{f}) & \cdots & (\frac{\partial \mathbf{J}_\theta^T}{\partial \theta_n}\mathbf{f}) \end{bmatrix}}_{n\times n} = \begin{bmatrix} \frac{\partial \mathbf{J}_\theta^T}{\partial \theta_n}\mathbf{f} \end{bmatrix},}$$

Finally, we can write the inverse of *conservative congruence transformation*

$$\boxed{\mathbf{K}_p = \mathbf{J}_\theta^{-T}(\mathbf{K}_\theta - \mathbf{K}_g)\mathbf{J}_\theta^{-1},}$$

In our simulation, we have assumed the $K_\theta$matrix and the force vector, and subsequently, we calculated the $K_P$ matrix. Knowing the matrix allows us to create a stiffness ellipsoid and evaluate the robot's stiffness in specific configurations. We visualized the stiffness index by plotting a heatmap, and based on the maximum and minimum values of the stiffness index, we generated the stiffness ellipsoid. This approach provides valuable insights into how the robot's stiffness varies across different configurations, helping us assess its performance and suitability for various tasks.

**Stiffness**



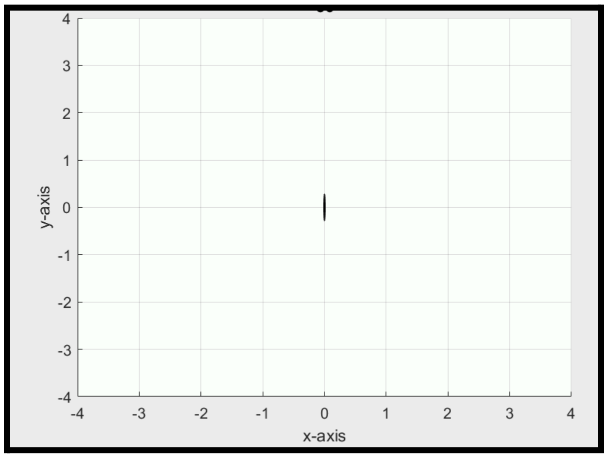
*(Heat Map for stiffness ellipsoid)*

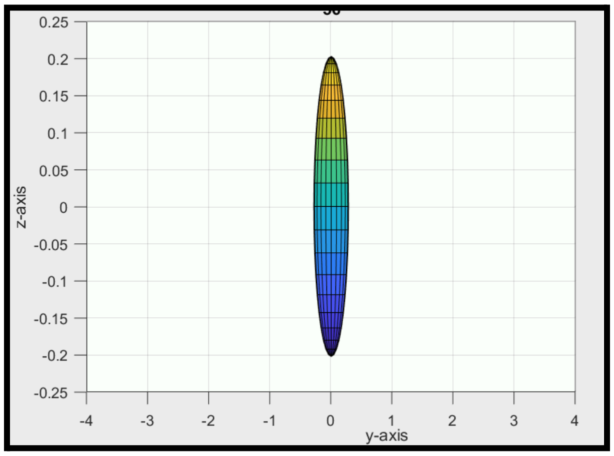
Here, we have generated the heat map for the stiffness ellipsoid. This heat map is defined as the $\sqrt{det\left(\left(K_x K_x^T\right)^{\dagger}\right)}$. Therefore, we have calculated this value for every point in the trajectory. This will physically provide the area of the stiffness ellipsoid. The larger the area, the more stiff the robot. Eigenvectors and eigenvalues of the ellipsoid give us the direction and magnitude of the maximum and minimum stiffness. Here, we have provided this heat map in the x-y direction but actually, it was in 3-D. Code can be referred to to get a proper understanding of the heat map.

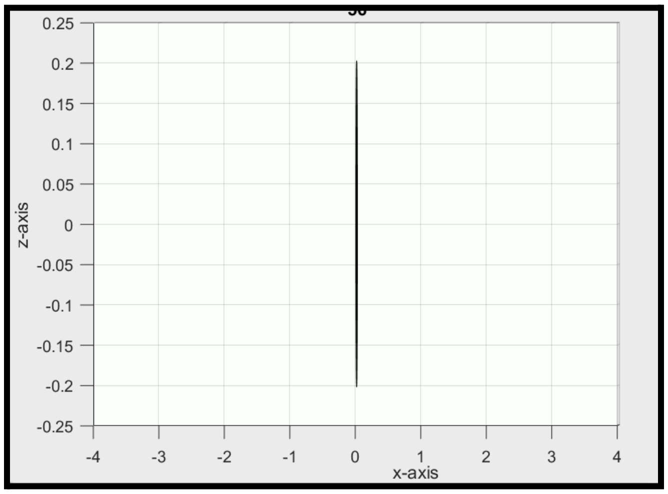
**Stiffness ellipsoid at different points on the infinity trajectory**

– *Note* all ellipsoids are drawn by taking (0,0,0) as a center which means we have not specified the correct directions but we specify the behavior and magnitude of the ellipsoid at certain points.

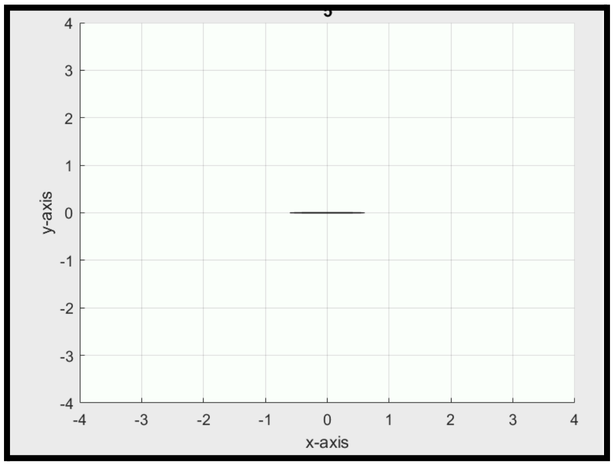
- For the low value of $\sqrt{det\left(\left(K_x K_x^T\right)^{\dagger}\right)}$

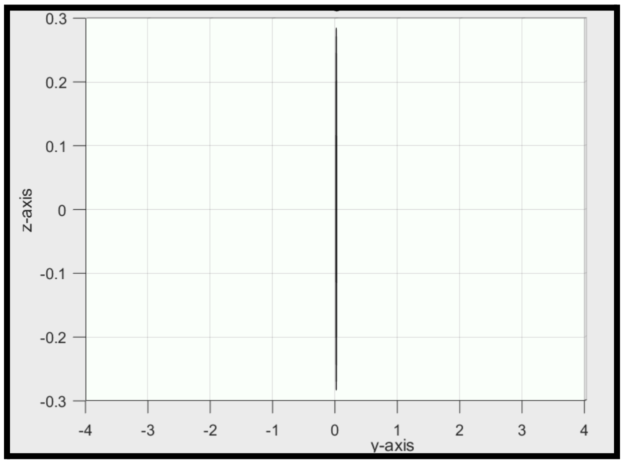


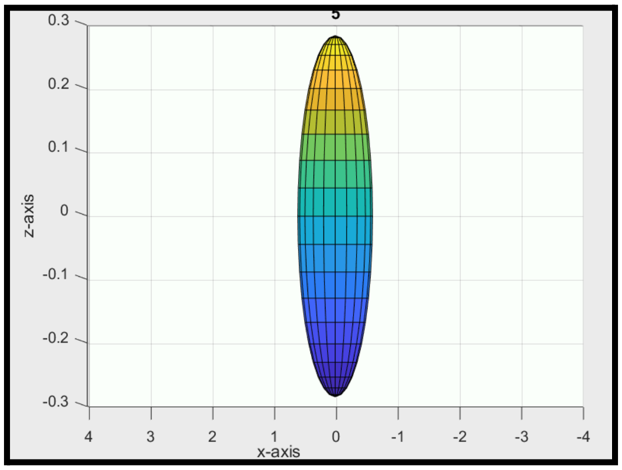




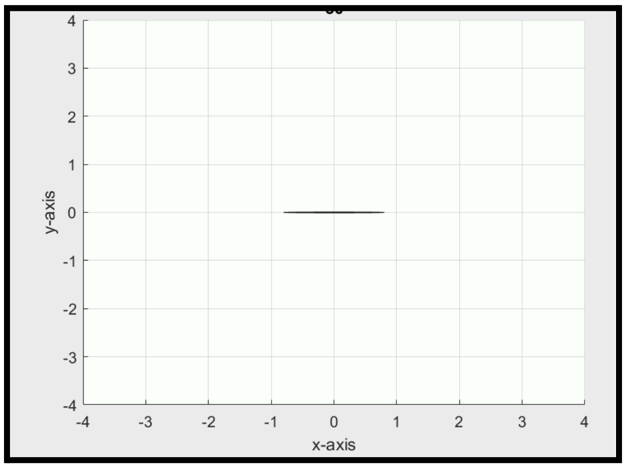
- For the mid value of $\sqrt{det\left(\left(K_x K_x^{T}\right)^{\dagger}\right)}$

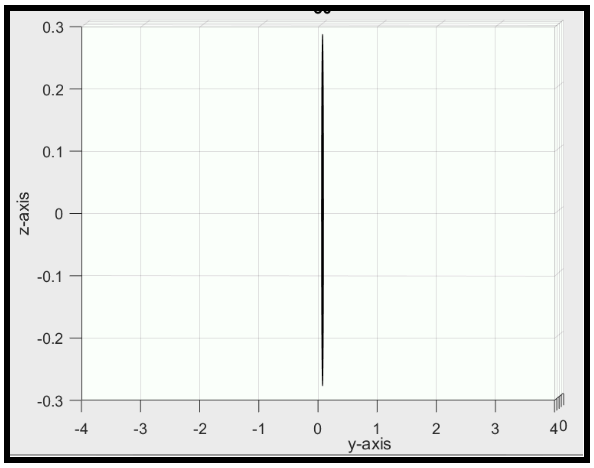


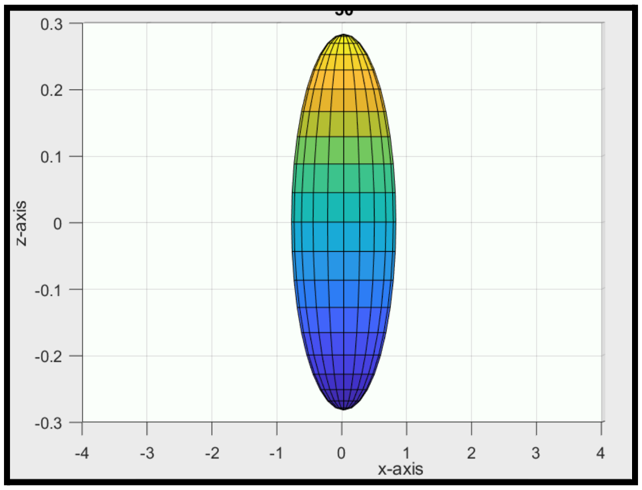




- For high-value $\sqrt{det\left(\left(K_x K_x^{T}\right)^{\dagger}\right)}$







Here, we can note that as the value of $\sqrt{det\left(\left(K_x K_x^{T}\right)^{\dagger}\right)}$ increases, the area of ellipsoid increases and the robot will become more stiff. Also, note that we have given all the graph views, but for better understanding, please refer to the Matlab code. From these views, we can conclude that the manipulator is highly resistant to the deformation (caused by the force applied on the end-effector) in some directions while low in other directions at the same point.

**References:-**

- https://www.sciencedirect.com/science/article/pii/S2405896317317147?ref=pdf_download&fr=RR-2&rr=817b2ee68efa8b0f
- https://www.sciencedirect.com/science/article/pii/S0094114X17306559?ref=pdf_download&fr=RR-2&rr=817b30833fd38afa
- https://www.kuka.com/en-in/products/robotics-systems/industrial-robots/lbr-iiwa
- https://www.researchgate.net/publication/220122616_Conservative_Congruence_Transformation_for_Joint_and_Cartesian_Stiffness_Matrices_of_Robotic_Hands_and_Fingers
- https://www.researchgate.net/publication/320895915_Position-based_kinematics_for_7-DoF_serial_manipulators_with_global_configuration_control_joint_limit_and_singularity_avoidance - for analytical inverse kinematics.

How to run the files

- First save the attached folder on your pc and open it on the matlab, and also open the coppeliaSim (*kuka_iiwa_14R820_trajectory.ttt*).
- For the kinematic analysis, open the *kuka_iiwa820_infinite_trajectory.m* in Matlab and first run the *kuka_iiwa_14R820_trajectory.ttt* file and then run the Matlab *file-kuka_iiwa820_infinite_trajectory.m*. It is also shown in the attached video.
- For drawing the manipulator's workspace, First make the initial joint angle4 to 0 value in *kuka_iiwa_14R820_trajectory.ttt*. Then, run *Workspace.m* after running the *kuka_iiwa_14R820_trajectory.ttt* file.
- For static and redundancy analysis, run *Static_analysis.m* file
- For Stiffness analysis, run *Stiffness_analysis.m* file
- For analytical analysis of inverse kinematics, run inv_kinematic_analytical_analysis.m
- Also, note there are other Matlab files on which the above files are dependent, for a better understanding of the codes go through them.