Indian Institute of Technology, Gandhinagar

ME 691
Advanced Robotics
Semester–I, Academic Year 2023-24

Project II

Torque Optimization by using Null Space Projection 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 the development and application of robotic manipulators. These machines, with their high degree of flexibility and precision, have found applications across various industries, from manufacturing to healthcare. This technical report delves into Project 2, where the focus is on the motion control and torque optimization for the joints of a 7-degree-of-freedom Kuka LBR iiwa 14 R820 robot. The project specifically addresses the challenge of applying force and moment to the end-effector, simulating a practical scenario such as the robot carrying a glass of water. In this scenario, a constant force is exerted on the end-effector in the -z direction due to the weight of the glass. The objective is to optimize the joint space of the robot, allowing it to follow a predefined trajectory with a constant force/torque at the tip while minimizing the net joint torque. The project involves a comprehensive analysis of the manipulator's capabilities and includes a detailed simulation to evaluate its performance in a real-world manipulation task.

Configuration and D-H parameters of the robot:-

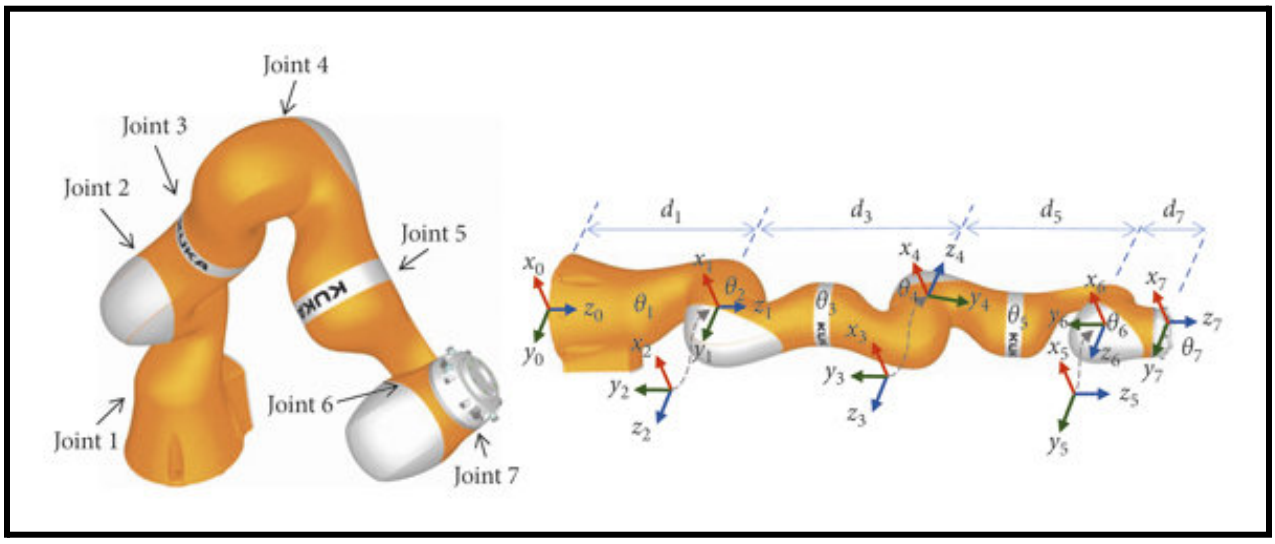

Fig 1: Manipulator configuration and joint frame [Source]

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 plays a crucial role in defining the kinematic structure and relationships within the Kuka robot.

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

i	α_i (rad)	a_i (mm)	d_i (mm)	θ_i (rad)
1	$-\frac{\pi}{2}$	0	d_1	θ_1
2	$\frac{\pi}{2}$	0	0	θ_2
3	$-\frac{\pi}{2}$	0	d_3	θ_3
4	$\frac{\pi}{2}$	0	0	θ_4
5	$-\frac{\pi}{2}$	0	d_5	θ_5
6	$\frac{\pi}{2}$	0	0	θ_6
7	0	0	d_7	θ_7

Fig 2: DH Parameters for Kuka iiwa 14R 820

Each θ_i represents the i-th 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 easily solve for Jacobian and forward kinematics.

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

Forward Kinematics formulation:-

```
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
```

The above code is for the jacobian of the end-effector. Hence, we can easily formulate the kinematics of the robot. But, in order to develop the dynamics equation, we will need the velocity of the center of mass of each link, which we can only find if we know the Jacobians of the C.O.M of each link. Therefore, to find the Jacobian for the C.O.M of ith-link, we simply consider it as an end-effector and calculate the Jacobian with the above approach.

One sample function for finding Jacobian for 4th link is given below:

```
function [Jv,Jw,R,Y] = Jacobian_kuka_4(theta1,theta2,theta3,theta4)
syms theta alph d a;
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];
A1 = subs(M,{a,alph,d,theta},{0,-pi/2,0.36,theta1});
A2 = subs(M,{a,alph,d,theta},{0,pi/2,0,theta2});
A3 = subs(M,{a,alph,d,theta},{0,pi/2,0.42,theta3});
A4 = subs(M,{a,alph,d,theta},{0,-pi/2,0,theta4});
T01 = A1;
T02 = T01*A2;
T03 = T02*A3;
T04 = T03*A4;
T05 = T04;
T06 = T05;
T07 = T06;
z0 = [0; 0; 1];
z1 = [T01(1,3);T01(2,3);T01(3,3)];
z2 = [T02(1,3);T02(2,3);T02(3,3)];
z3 = [T03(1,3);T03(2,3);T03(3,3)];
Jv = [diff(T07(1,4),theta1),diff(T07(1,4),theta2),diff(T07(1,4),theta3),diff(T07(1,4),theta4);
    diff(T07(2,4),theta1),diff(T07(2,4),theta2),diff(T07(2,4),theta3),diff(T07(2,4),theta4);
    diff(T07(3,4),theta1),diff(T07(3,4),theta2),diff(T07(3,4),theta3),diff(T07(3,4),theta4)];
Jw = [z0 z1 z2 z3];
R = [T07(1,1),T07(1,2),T07(1,3);
    T07(2,1),T07(2,2),T07(2,3);
    T07(3,1),T07(3,2),T07(3,3)];
Y = T07(3,4);
end
```

Now, we can easily calculate the Jacobian$\left(J_i\right)$, rotation matrix $\left(R_i\right)$ (representing the rotation of i-th joint frame to the base frame) and the height$\left(y_i\right)$ of the C.O.M of each link. Hence, we are now ready to formulate the dynamics of the Kuka manipulator.

1 KUKA iiwa 14R 820 Dynamics Formulation

1.1 Equations of Motion

The general equation of Kinetic Energy for n-th link manipulator is given by:

$KE = \frac{1}{2}\dot{q}^T \sum_i^n [(J_{wi}^T R_i I_i R_i^T J_{wi}) + m_i (J_{vi}^T J_{vi})]\dot{q}$

For a 7-DOF,with inertia $I_1, I_2, I_3, I_4, I_5, I_6, I_7$ and masses $m_1, m_2, m_3, m_3, m_4, m_5, m_6, m_7$.

$m_1 = 3.94781; m_2 = 4.50275; m_3 = 2.45520; m_4 = 2.61155; m_5 = 3.41000; m_6 = 3.38795; m_7 = 0.35432$

$$I_1 = \begin{bmatrix} 0.00455 & 0 & 0 \\ 0 & 0.00454 & 0.00001 \\ 0 & 0.00001 & 0.00029 \end{bmatrix}$$

$$I_2 = \begin{bmatrix} 0.00032 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.00042 \end{bmatrix}$$

$$I_3 = \begin{bmatrix} 0.00223 & -0.00005 & 0.00007 \\ -0.00005 & 0.00219 & 0.00007 \\ 0.00007 & 0.00007 & 0.00073 \end{bmatrix}$$

$$I_4 = \begin{bmatrix} 0.03844 & 0.00088 & -0.00112 \\ 0.00088 & 0.01144 & -0.00111 \\ -0.00112 & -0.00111 & 0.04988 \end{bmatrix}$$

$$I_5 = \begin{bmatrix} 0.00277 & -0.00001 & 0.00001 \\ -0.00001 & 0.00284 & 0 \\ 0.00001 & 0 & 0.00012 \end{bmatrix}$$

$$I_6 = \begin{bmatrix} 0.0005 & -0.00005 & -0.00003 \\ -0.00005 & 0.00281 & -0.00004 \\ -0.00003 & 0.00004 & 0.00232 \end{bmatrix}$$

$$I_7 = \begin{bmatrix} 0.00795 & 0.00022 & -0.00029 \\ 0.00022 & 0.01089 & -0.00029 \\ -0.00029 & -0.00029 & 0.00294 \end{bmatrix}$$

$KE_1 = \frac{1}{2}\dot{\theta_1}((J_{w1}^T R_1 I_1 R_1^T J_{w1}) + m_1 (J_{v1}^T J_{v1}))\dot{\theta_1}$

$KE_2 = \frac{1}{2}[\dot{\theta_1}, \dot{\theta_2}]((J_{w2}^T R_2 I_2 R_2^T J_{w2}) + m_2 (J_{v2}^T J_{v2}))[\dot{\theta_1}, \dot{\theta_2}]^T$

$KE_3 = \frac{1}{2}[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}]((J_{w3}^T R_3 I_3 R_3^T J_{w3}) + m_3 (J_{v3}^T J_{v3}))[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}]^T$

$KE_4 = \frac{1}{2}[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}]((J_{w4}^T R_4 I_4 R_4^T J_{w4}) + m_4 (J_{v4}^T J_{v4}))[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}]^T$

$KE_5 = \frac{1}{2}[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}]((J_{w5}^T R_5 I_5 R_5^T J_{w5}) + m_5 (J_{v5}^T J_{v5}))[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}]^T$

$KE_6 = \frac{1}{2}[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}, \dot{\theta_6}]((J_{w6}^T R_6 I_6 R_6^T J_{w6}) + m_6 (J_{v6}^T J_{v6}))[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}, \dot{\theta_6}]^T$

$KE_7 = \frac{1}{2}[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}, \dot{\theta_6}, \dot{\theta_7}]((J_{w7}^T R_7 I_7 R_7^T J_{w7}) + m_7 (J_{v7}^T J_{v7}))[\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}, \dot{\theta_6}, \dot{\theta_7}]^T$

$KE = KE_1 + KE_2 + KE_3 + KE_4 + KE_5 + KE_6 + KE_7$

$PE = m_1 g y_1 + m_2 g y_2 + m_3 g y_3 + m_4 g y_4 + m_5 g y_5 + m_6 g y_6 + m_7 g y_7$

where,

- $y_1, y_2, y_3, y_4, y_5, y_6, y_7$ are the vertical distance of the center of mass of each link.

- $\theta_1(t), \theta_2(t), \theta_3(t), \theta_4(t), \theta_5(t), \theta_6(t), \theta_7(t)$ are the joint angle w.r.t x-axis.

- J_{wi} and J_{vi} are the angular and linear velocity Jacobian of the center of mass of the i^{th} link respectively.

- R_i is the rotation matrix which is used to transfer inertia from i^{th} joint frame to the base frame.

1.2 Dynamics

Lagrangian is given as -
$\mathcal{L} = KE - PE$

The equations of motion can now be expressed in terms of the Lagrangian as follows -
$\tau_i = \frac{d}{dt}(\frac{\partial \mathcal{L}}{\partial \dot{q_i}}) - \frac{\partial \mathcal{L}}{\partial q_i}$

The equations of motion are given as:

$$\tau_1 = M_{11}\ddot{\theta}_1 + M_{12}\ddot{\theta}_2 + M_{13}\ddot{\theta}_3 + M_{14}\ddot{\theta}_4 + M_{15}\ddot{\theta}_5 + M_{16}\ddot{\theta}_6 + M_{17}\ddot{\theta}_7 + H_1 + G_1$$
$$\tau_2 = M_{21}\ddot{\theta}_1 + M_{22}\ddot{\theta}_2 + M_{23}\ddot{\theta}_3 + M_{24}\ddot{\theta}_4 + M_{25}\ddot{\theta}_5 + M_{26}\ddot{\theta}_6 + M_{27}\ddot{\theta}_7 + H_2 + G_2$$
$$\tau_3 = M_{31}\ddot{\theta}_1 + M_{32}\ddot{\theta}_2 + M_{33}\ddot{\theta}_3 + M_{34}\ddot{\theta}_4 + M_{35}\ddot{\theta}_5 + M_{36}\ddot{\theta}_6 + M_{37}\ddot{\theta}_7 + H_3 + G_3$$
$$\tau_4 = M_{41}\ddot{\theta}_1 + M_{42}\ddot{\theta}_2 + M_{43}\ddot{\theta}_3 + M_{44}\ddot{\theta}_4 + M_{45}\ddot{\theta}_5 + M_{46}\ddot{\theta}_6 + M_{47}\ddot{\theta}_7 + H_4 + G_4$$
$$\tau_5 = M_{51}\ddot{\theta}_1 + M_{52}\ddot{\theta}_2 + M_{53}\ddot{\theta}_3 + M_{54}\ddot{\theta}_4 + M_{55}\ddot{\theta}_5 + M_{56}\ddot{\theta}_6 + M_{57}\ddot{\theta}_7 + H_5 + G_5$$
$$\tau_6 = M_{61}\ddot{\theta}_1 + M_{62}\ddot{\theta}_2 + M_{63}\ddot{\theta}_3 + M_{64}\ddot{\theta}_4 + M_{65}\ddot{\theta}_5 + M_{66}\ddot{\theta}_6 + M_{67}\ddot{\theta}_7 + H_6 + G_6$$
$$\tau_7 = M_{71}\ddot{\theta}_1 + M_{72}\ddot{\theta}_2 + M_{73}\ddot{\theta}_3 + M_{74}\ddot{\theta}_4 + M_{75}\ddot{\theta}_5 + M_{76}\ddot{\theta}_6 + M_{77}\ddot{\theta}_7 + H_7 + G_7$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \\ \tau_7 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} & M_{17} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} & M_{26} & M_{27} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} & M_{36} & M_{37} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} \\ M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} & M_{57} \\ M_{61} & M_{62} & M_{63} & M_{64} & M_{65} & M_{66} & M_{67} \\ M_{71} & M_{72} & M_{73} & M_{74} & M_{75} & M_{76} & M_{77} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\theta}_4 \\ \ddot{\theta}_5 \\ \ddot{\theta}_6 \\ \ddot{\theta}_7 \end{bmatrix} + \begin{bmatrix} H_1 + G_1 \\ H_2 + G_2 \\ H_3 + G_3 \\ H_4 + G_4 \\ H_5 + G_5 \\ H_6 + G_6 \\ H_7 + G_7 \end{bmatrix}$$

Challenges Faced and Execution of the Task

Problem with coppeliaSim

Due to the lack of a proper inertia matrix frame of reference for each joint in CoppeliaSim and incorrect DH parameters (only 6 provided for a 7-DOF manipulator), dynamic simulation of our robot in this environment has been challenging. Therefore, DH parameters from a paper were adopted; however, the joint frames defined in CoppeliaSim differ from those in the paper. Consequently, directly using the inertia matrix from CoppeliaSim is not feasible for dynamic simulation. Hence, we cannot use coppeliaSim.

Problem with the Dynamics of the Kuka Robot

We've derived the equations of motion for the dynamics, but they're too complex for MATLAB to simplify them adequately(more time-consuming). As a result, we cannot obtain the complete equations, preventing us from isolating the mass matrix (M) and the Coriolis and gravity terms (H+G) separately. Consequently, calculating angular acceleration, angular velocity, and joint angles from provided torques is currently unfeasible. However, we have managed to generate a .mat file from MATLAB containing all variables with their full expressions. By loading this file, we can input values for angles, angular velocity, and acceleration to determine the corresponding torques.

Problem with motion control incorporating dynamics

Since we are unable to incorporate the forward dynamic equations and we don't know the technique to find the value of controller parameters for our system, that's why we cannot perform motion control by incorporating the dynamics of kuka.

Execution of the task in MATLAB

Therefore, we have simulated our manipulator in Matlab using kinematics and optimized the torques$\left(J^T F\right)$ by following the circular trajectory while continuously applying the constant force and moment on the end-effector of the Kuka robot.

In our exploration of the Kuka robot's capabilities, we conducted a demonstration using a circular trajectory in a plane to illustrate the manipulator's ability to securely hold a glass of water. When the robot holds an object, such as a glass of water, a constant force is exerted at the tip of the robot. To extend this functionality, we tasked the robot with transporting the glass from one point to another, essentially requiring the end-effector to follow a specific trajectory while maintaining a constant force at its tip.

Executing this task necessitates supplying joint torques, a critical factor in trajectory design. Higher torques imply increased energy consumption and potential difficulty in execution. To address this, we leveraged the Null space contribution of the robot to minimize the net joint torque. The formulation of the Null-space projection vector was strategically designed to account for both joint configuration and the force/moment at the end-effector tip, ensuring consistent optimization of joint torques throughout the circular trajectory.

Additionally, we delved into manipulability force analysis, constructing ellipsoids at various points along the trajectory. This simulation, orchestrated through command inputs from Matlab, replicates real-world scenarios and highlights the versatility of the robotic manipulator in handling dynamic tasks with a constant force application at the end-effector.

Null Space Contribution:

As we know that the Kuka is a 7-DOF robot, meaning it has redundancy. Therefore, we can use its extra degree of freedom to optimize any task. We have done this by using its null space contribution. We know that for a redundant robot, the numerical inverse kinematic is defined as:

$$q(:,k+1) = q(:,k) + pinv(J) \times (v(:,k)) \times dt + NC \times dt$$

where $pinv(J)$ is the pseudo inverse of the jacobian and NC is the null space contribution which can be defined for 7-DOF kuka as:

$$NC = (eye(7) - pinv(J) \times J) \times no$$

Here, *no* is the Null space projection vector.

The physical significance of the null space projection is that, we can take any value of the *no* vector, so that the joint velocity of each joint will change but the end-effector velocity will remain the same. Hence, our approach involves employing the null space vector to generate joint velocities in a manner that minimizes the effort or torque required by the joints to achieve the desired velocity.

Null space projection vector:

```
clear
close all
clc
%%
syms th1 th2 th3 th4 th5 th6 th7 fx fy fz mx my mz
K=diag([fx,fy,fz,mx,my,mz]);
th=[th1 th2 th3 th4 th5 th6 th7];
J = Jacobian_kuka_iiwa(th);
A=J*J.';
B=K;
A=A/trace(A);
B=B/trace(B);
W=sqrt(trace((A-B)*(A-B).'));
qo=[diff(W,th1);diff(W,th2);diff(W,th3);diff(W,th4);diff(W,th5);...
    diff(W,th6);diff(W,th7)];
qof=symfun(qo,[th1 th2 th3 th4 th5 th6 th7 fx fy fz mx my mz]);
matlabFunction(qof,'File','nullspce_cotribution_qo')
```

Here, the construction of the Null space projection vector depends on the joint configuration and the forces/moment at the joint tip. The joint configuration enables the determination of the Jacobian of the robot and, consequently, the matrix $\left(JJ^T\right)$. The trace of the matrix is used to just normalize the matrix. Subsequently, the W matrix is constructed based on the JJ^T and the normalized force/moment diagonal matrix. This establishes our objective function. To obtain the optimized values at each configuration, we utilize the Gradient Descent method. Therefore, W is differentiated, and we get the null space projection vector as

$$No = -\left[\frac{\partial W}{\partial q_i}\right]$$

Simulation results and verification:

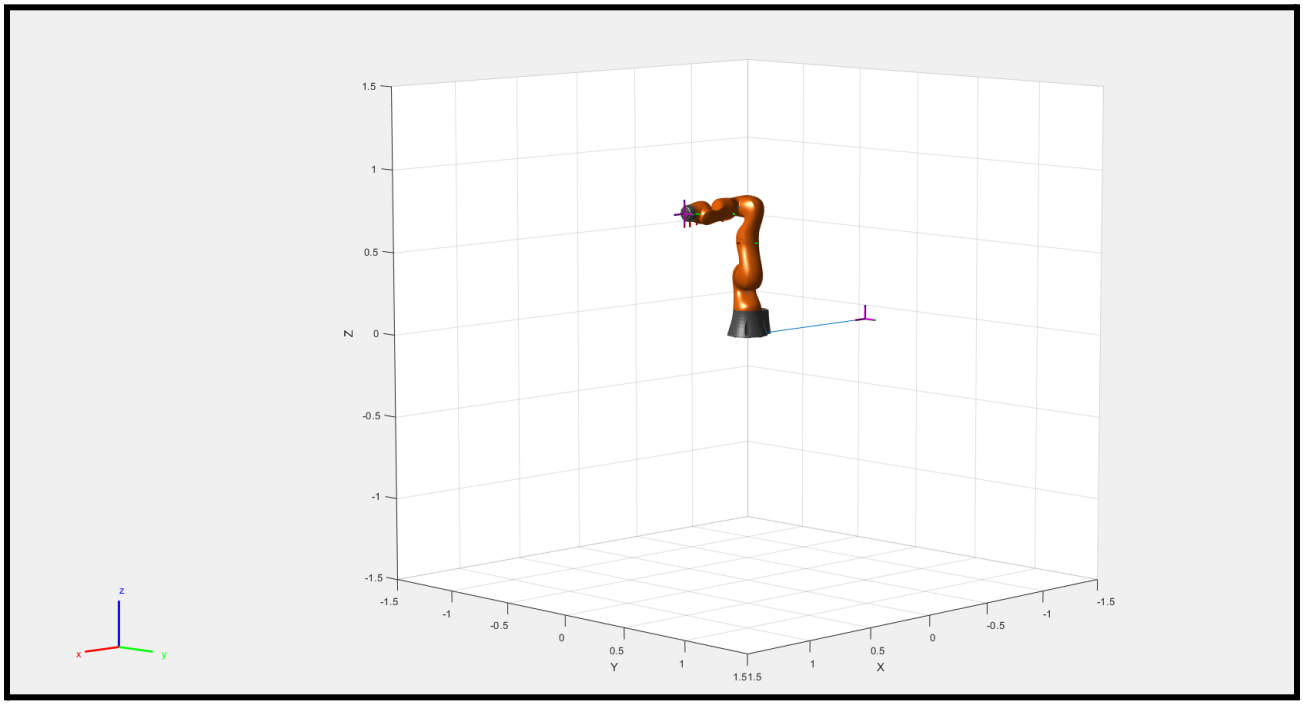

Fig 3: Coordinates representation and the initial configuration of the manipulator

Case 1:- Force at the tip is -1N in z-direction

Now, we have performed the simulation to verify our optimization technique. Here, we have plotted the norm of the torque vector at each point of the trajectory for both optimized versions as well as without optimization (where we just use minimum norm solution). For the most realistic task, force at the tip acts in the -ve z direction due to gravity. We can see the differences between the norm of the torque vector from the graph described below

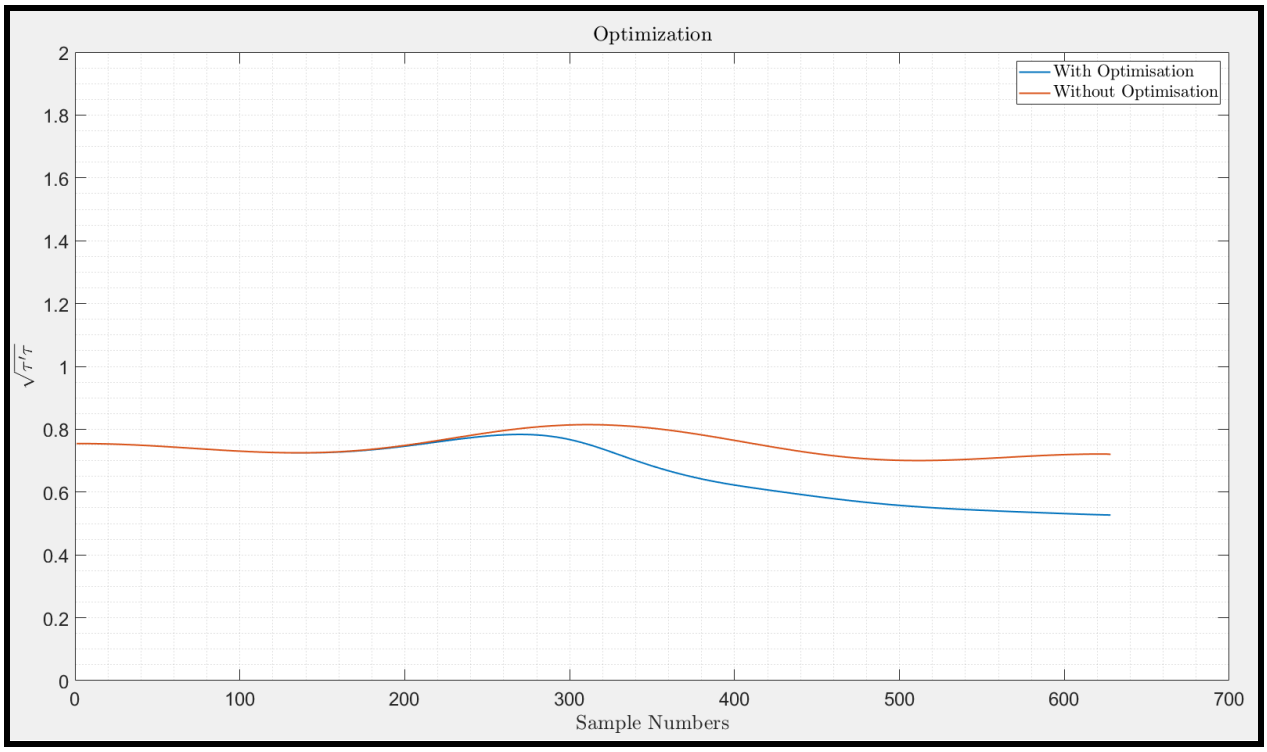

Fig 4: Plots between norm of joint torque and points on circular trajectory

In this scenario, we applied a force in the negative z-direction with a magnitude of 1N at the tip, while the robot end-effector traced a circular trajectory in a specific plane. We monitored the norm of the torque vector. The graphical representation indicates that although the manipulator initiates from identical configurations in both cases, the optimization process consistently leads to a reduction in the net torque. This outcome demonstrates the manipulator's ability to optimize the net joint torque for a given task.

We have also plotted the same norm of the torque for different conditions of the end-effector tip forces/moment and from all the graphs we can clearly see the effect of the optimization process.

Case 2:- Force at the tip is 1N in x-direction

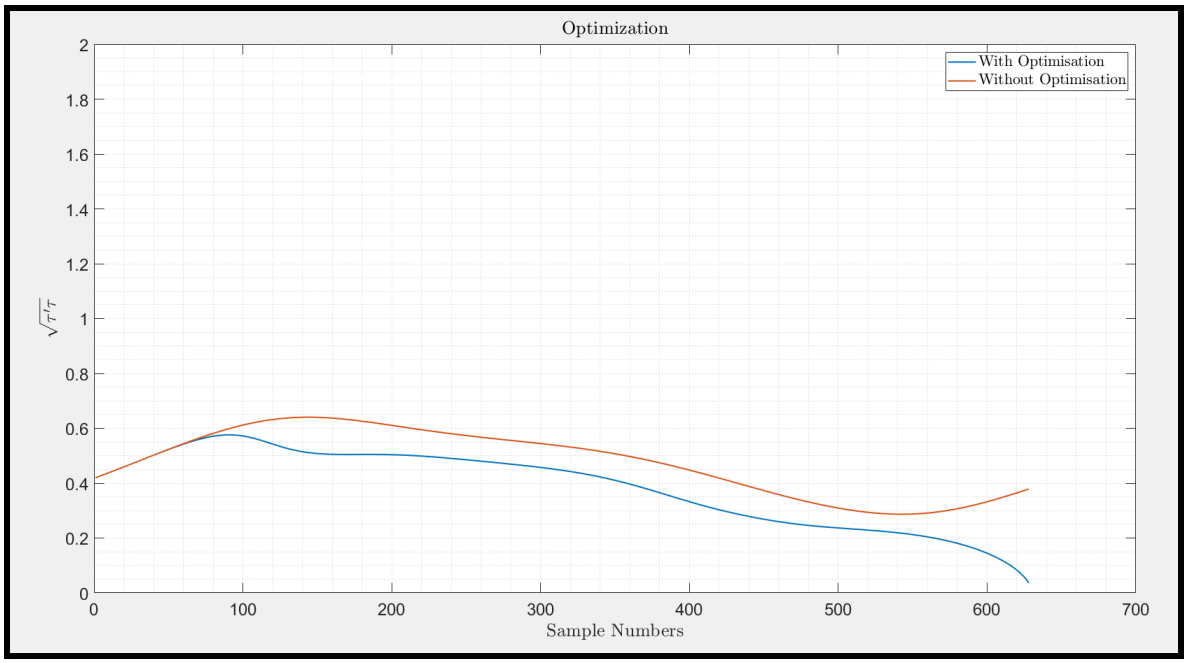

Fig 5: Plots between norm of joint torque and points on circular trajectory

Case 3: Force at the tip is -2N in y-direction

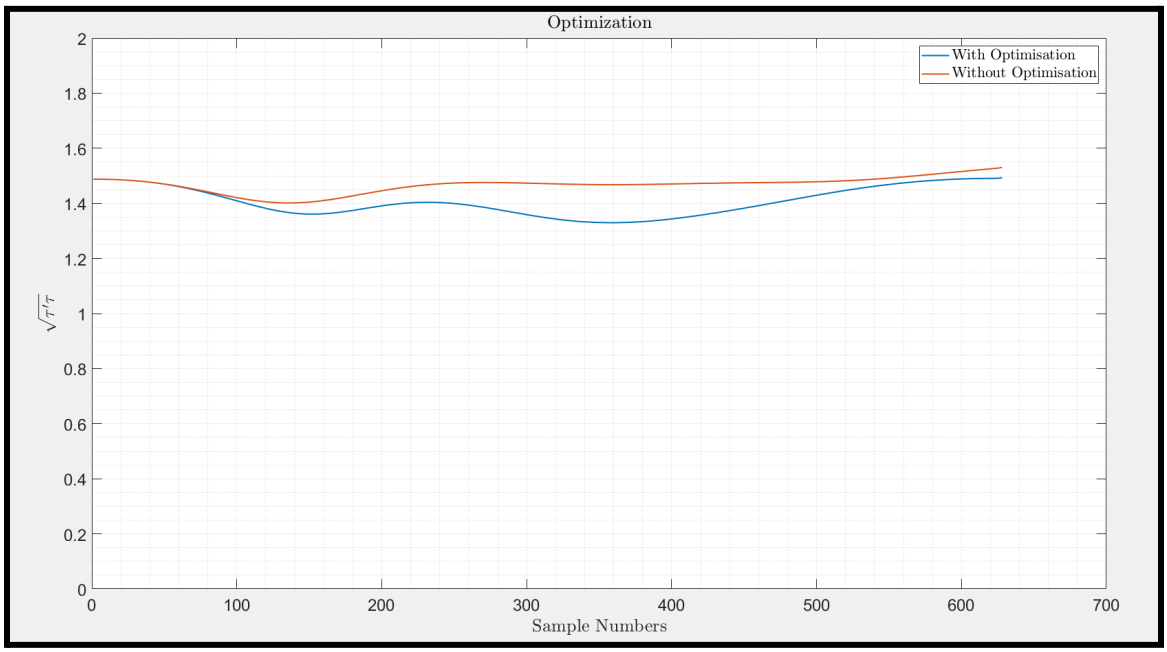

Fig 6: Plots between norm of joint torque and points on circular trajectory

Additionally, we assessed the optimization process for the moment applied at the end-effector and confirmed that this approach is effective in optimizing moments as well.

Case 4: Moment at the tip -1N-m in the x-direction

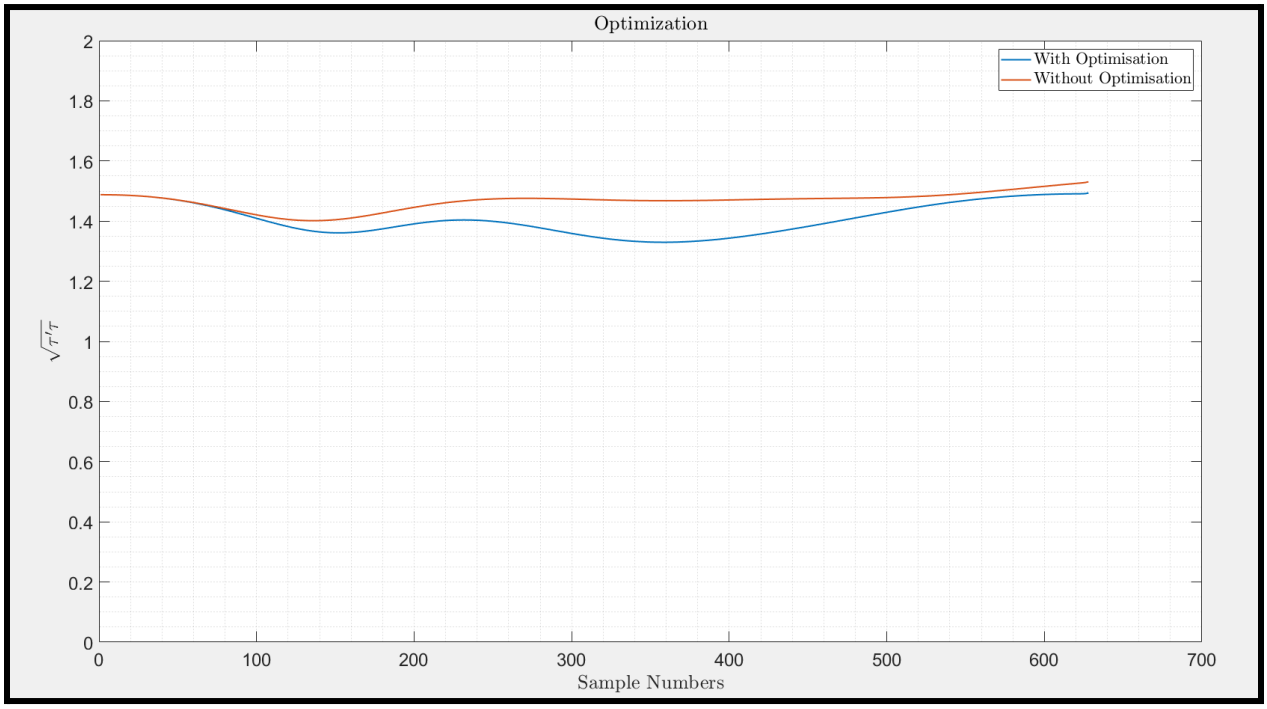

Fig 7: Plots between norm of joint torque and points on circular trajectory

Hence, we are able to demonstrate and verify the optimization technique.

Discussion:

- Apart from the graph, we have also captured the motion of the kuka robot with a real-time force ellipsoid set on the tip of the end effector. We can also visualize from the video that the force ellipsoid defined at the tip of the robot always tries to align in the direction of the applied force so that the same process can be executed with a minimal amount of effort.
- Although we had the aim of doing the same by incorporating the feedback and feedforward model through dynamic equations, due to very high symbolic calculation in the case of the 7-DOF manipulator, it is very difficult to construct Mass and Inertia matrices separately.

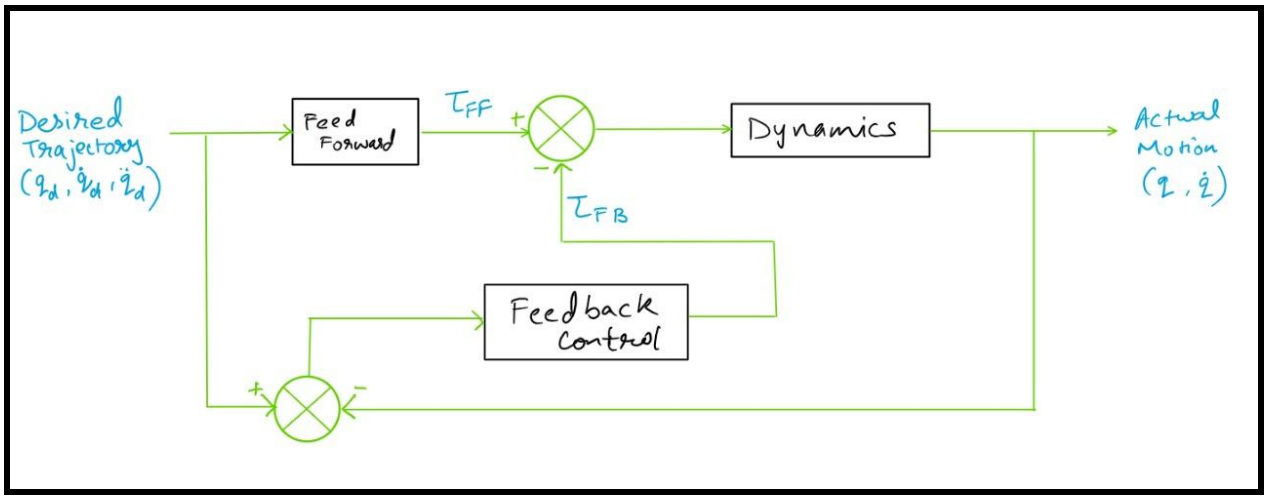

Fig 8: Motion Control

- We have derived the torque equation, and our approach for deriving the torque is true (validated for the conservation of the 3-R manipulator in homework 5) but somehow, we are not able to separate the gravity and inertia matrix from the torque equation, which leads us to do the optimization for the kinematics purpose.

Videos:

- This is the video link for the kuka manipulator following circular trajectory with a real-time force ellipsoid set on the tip of the end-effector - https://drive.google.com/drive/folders/1ts-MRJDd1BtLfHt4XZIMSj12O6B_8nK8?usp=sharing

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://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7139300
- 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
- https://www.sciencedirect.com/science/article/pii/S2212827118310710
- https://mediatum.ub.tum.de/doc/1585197/file.pdf

How to run optimization files:

- First, save the attached folder on your PC and open it on MATLAB.
- Open the kuka_ellipse_optimize.m file.
- There is an NR_x.mat file which contains all the information about the KUKA iiwa 14R 820 robots, like D-H parameters, mass, Inertia matrix, and all the frames.
- There is an Av matrix (6x1) at the start of the code. It contains the value of 3-forces and 3-moments at the robot's tip. Initially, all the elements are set to zero. We can change the value of any element to visualize the optimization effect of that trajectory.
- Now, we can make the Null space contribution to be zero by multiplying it by zero, and consistently, we can also visualize the normal trajectory.
- Finally, we will run the plot_graph.m file to understand the difference between optimized torque and torque without optimization.

To run Dynamics files:

- First run the *kuka_dynamics_generation.m*, which has all the formulation of the dynamics of the kuka manipulator. If you only want to see the torque equations, then you can just load the *Torque_dynamics_Kuka_820.mat* into your workspace, and you will get the torque expressions from the variable (T1, T2, T3, T4, T5, T6, T7).
- Second, in order to execute the feedforward dynamics, run the *FeedForwardDynamics.m* file, which takes the desired angle, angular velocity, and acceleration as input and outputs the joint torques at those input values.