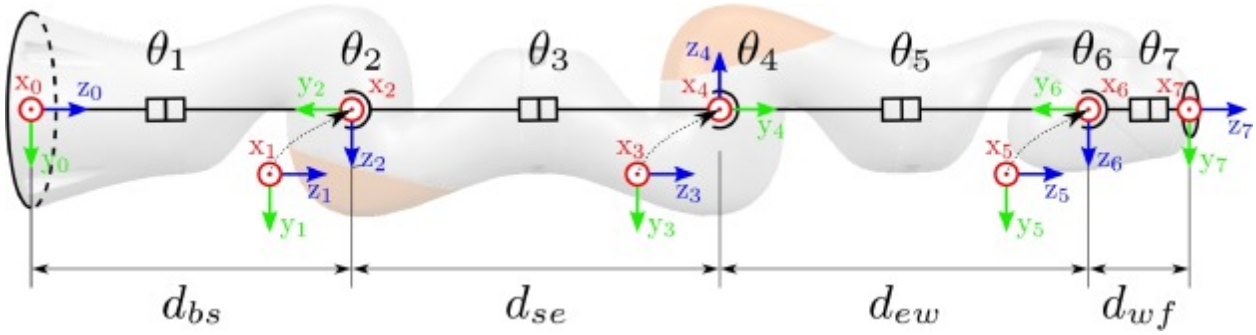
Kuka LBR iiwa 7 R800 robot

1. Configuration and D-H parameters of the robot:-



This is the configuration of the Kuka robot and the assigned D-H conventions. The first origin is set up at the base frame, and then the succeeding origins are set up according to the D-H conventions. In our case, we used
dbs = 0.36;
dse = 0.42;
dew= 0.4;
dwf= 0.126, which specified the length of the different links.
Hence, according to the assigned conventions, the D-H parameter table will be:-

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	d_{bs}	θ_1
2	0	$\pi/2$	0	θ_2
3	0	$\pi/2$	d_{se}	θ_3
4	0	$-\pi/2$	0	θ_4
5	0	$-\pi/2$	d_{ew}	θ_5
6	0	$\pi/2$	0	θ_6
7	0	0	d_{wf}	θ_7

Each θ_i represents the *ith* joint position variable. Column d_i describes the kinematic link lengths. The jacobian and the forward kinematics problem is easily solved once the DH parameters are determined. 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 ${}^{0}_{T_7}$, 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:-

```
function [Data] = forward_kinematics_kuka(q)
syms theta alph d a;
q=q';
a1 = 0.36;
a2 = 0.42;
a3 = 0.4;
a4 = 0.126;
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,a1,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,a2,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,a3,q(5,:)});
A6 = subs(M,{a,alph,d,theta},{0,pi/2,0,q(6,:)});
A7 = subs(M,{a,alph,d,theta},{0,0,a4,q(7,:)});
T01 = A1; T02 = T01*A2; T03 = T02*A3; T04 = T03*A4; T05 = T04*A5; T06 = T05*A6; T07 = T06*A7;
x = T07(1,4);
y = T07(2,4);
z = T07(3,4);
angle_y = pi/2;
angle_x = 0;
angle_z = 0;
Data = [x,y,z,angle_x,angle_y,angle_z];
end
```

Jacobian:-

```
function J = Jacobian_kuka_iiwa(q)
syms theta alph d a theta1 theta2 theta3 theta4 theta5 theta6 theta7;
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});
A5 = subs(M,{a,alph,d,theta},{0,-pi/2,0.4,theta5});
A6 = subs(M,{a,alph,d,theta},{0,pi/2,0,theta6});
A7 = subs(M,{a,alph,d,theta},{0,0,0.126,theta7});
T01 = A1; T02 = T01*A2; T03 = T02*A3; T04 = T03*A4; T05 = T04*A5; T06 = T05*A6; T07 = T06*A7;
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)];
z4 = [T04(1,3);T04(2,3);T04(3,3)];
z5 = [T05(1,3);T05(2,3);T05(3,3)];
z6 = [T06(1,3);T06(2,3);T06(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(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)];
Jw = [z0 z1 z2 z3 z4 z5 z6];
J = [Jv;Jw];
J = subs(J,{theta1,theta2,theta3,theta4,theta5,theta6,theta7},{q(1),q(2),q(3),q(4),q(5),q(6),q(7)});
end
```

Now, as we have both the Jacobian and the forward kinematics equation. We can easily simulate the robot. We used the circular 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.

Effort reduction and optimization:-

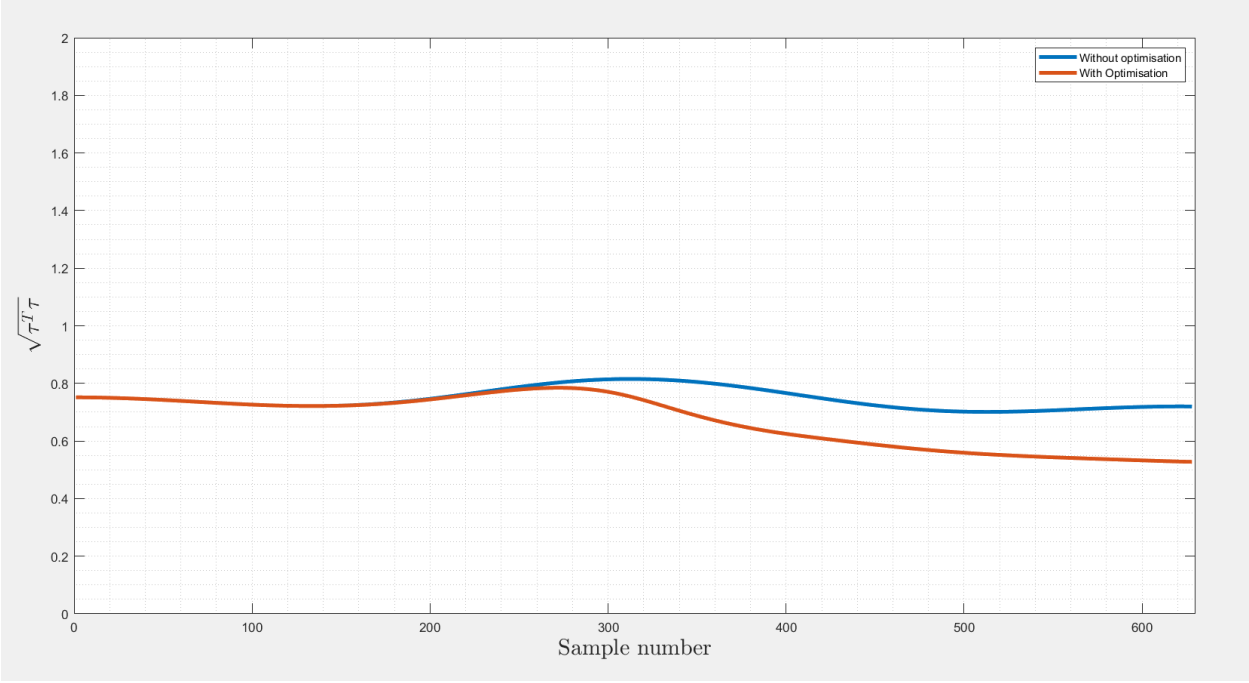
Now, we know that the KUKA is a 7-DOF robot, and we are tracing a trajectory in a single plane. Therefore, there will be redundancy in the traced path. But we can use these redundancies to reduce the effort applied by the motor at the joint. In simple words, whenever there is a redundancy in the robot, it means that there are various configurations possible for that particular end-effector location. Therefore, in the optimization/effort reduction method, we are using that configuration where the motor has to provide the lowest joint torque.

We are using null space projection for the optimization method. Null space projection is defined as the `NC=(eye(7) - pinv(J)*J)*no ;`
Where no is the null space contribution matrix and is given by

```
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]); % These are the forces and moment in x,y and z direction
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')
```

Now, by applying the optimization, we can clearly see the difference in the torque at the joints.



We also plotted the force ellipsoid at the end-effector when there was the force or moment acting at the tip of the robot. We can clearly see that in the optimization method, the force ellipsoid always tries to align with the direction of the force, which concludes that the robot is trying to align itself in such a way it can bear the maximum force or torque.

Here is the Google Drive link to access the videos of the force ellipsoid and all the codes.