SIX DEGREES OF FREEDOM ROBOTIC ARM

A Project Report

Submitted by

CHIRIKI PAVAN SAI (U42801124)

Department of Mechanical Engineering

University of South Florida

Tampa Florida 33612, USA.
November-2022

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to our Professor Redwan Alqasemi for his exemplary guidance, monitoring and constant encouragement throughout the course of Robotic Systems. The blessing, help and guidance given by him from time to time shall carry us a long way in the journey of life on which we are about to embark**.**

I also take this opportunity to express a deep sense of gratitude to my Teaching Assistant Urvish Trivedi for his cordial support, valuable information, and guidance, which helped me in learning new things and inspired us through various stages.

I am also grateful for having a chance to meet so many wonderful people who led me though this project period. I perceive this opportunity as a big milestone in my life.

Lastly, I thank The Almighty, our respective parents and friends for their constant encouragement without which this project would not have been possible.

CONTENTS

Introduction

Robots in the present have been a great asset because they can perform jobs that are too dangerous for humans. This has allowed workers to move on to more skilled jobs such as programming robots. Nowadays in this fast-growing industrial age, every company needs speed in manufacturing to cope with the customer's requirements. Robotics technology has matured to a point where there have been many research and industrial applications. Robots have been used to replace people in the production line and are ideally suited for repetitive work. Its use can be extended and exploited by a few modifications. The application of robotics technology is also used in medical, aerospace, and underwater.

The present model is a prototype replica of an Industrial Robotic Arm. After performing simulation tasks in the prototype Robot, we can understand the actual scenario such as Control of the Robotic Arm and Design Considerations in an Industrial Robot. This prototype model can be used to obtain the experimental results for any improvement or upgradations that can be implemented on an industrial robotic arm in a very economical and cost-effective way.

How robots will affect future generations may also be seen in the economy. High-quality products cause high sales, which keeps companies alive and running, and stimulates the economy in return. In the future service, robots are expected to surpass the industrial robot market. In our present report, we have emphasized the forward and inverse kinematic analysis of our robotic arm prototype. The forward kinematic analysis deals with the determination of the end effector position for the corresponding values of joint variables.

The present work describes a mechanical system design concept and a prototype implementation of a 6 DOF jointed-arm robot. The robot has 1 prismatic Joint and 5 revolute joints. The robotic arm was designed by the collaboration of the group and then hand sketched for approval. Upon approval, the robotic arm was given the name "Chitti 4.0" and then created using SolidWorks in which the frames were assigned to the joints using the right-handed rule. (For each frame). For each coordinate frame, the DH parameter was filled in according to each frame and then followed by creating the corresponding matrices. Once revision and corrections of the DH parameters and frames were made the physical robot was built and demonstrated the robot's ability to move in cartesian space as per the objective provided during the demonstration.

Problem Statement

The team was challenged to design a robot arm with the given constraint that "no two consecutive joints axes of rotation" were to be parallel. The robot had to be able to take input from a user in the form of a transformation matrix and use the matrix to execute motion and move the end effector to the desired location. The robot had to first determine whether the desired movement was valid or within the workspace. To achieve these requirements many concepts needed to be understood. These concepts include assigning optimal frames, formulating DH parameters, calculating Forward Kinematics, finding Jacobian, and using of resolved rate method.

Design of Robotic Arm

The robotic arm was initially hand sketched (Figure 1) with all frame assignments and sent for approval with the first joint being prismatic and the remaining five joints being revolute joints.

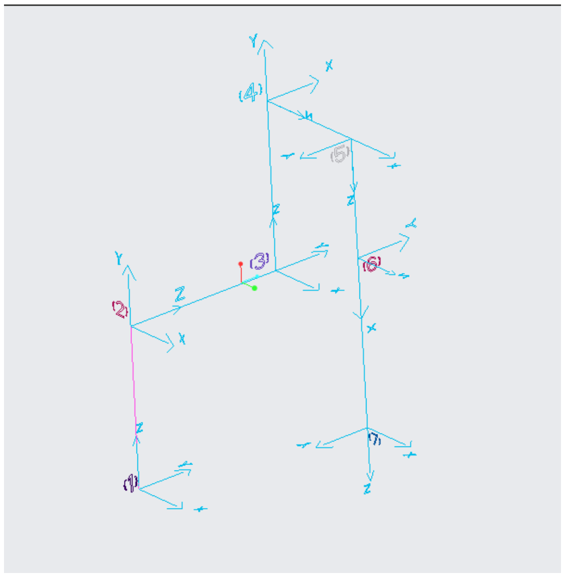


Fig 1: Initial hand sketch

After suggestions and improvements, the final design of the arm was achieved as per figure 2 with frames assigned and further DH parameters are calculated as per Table 1.

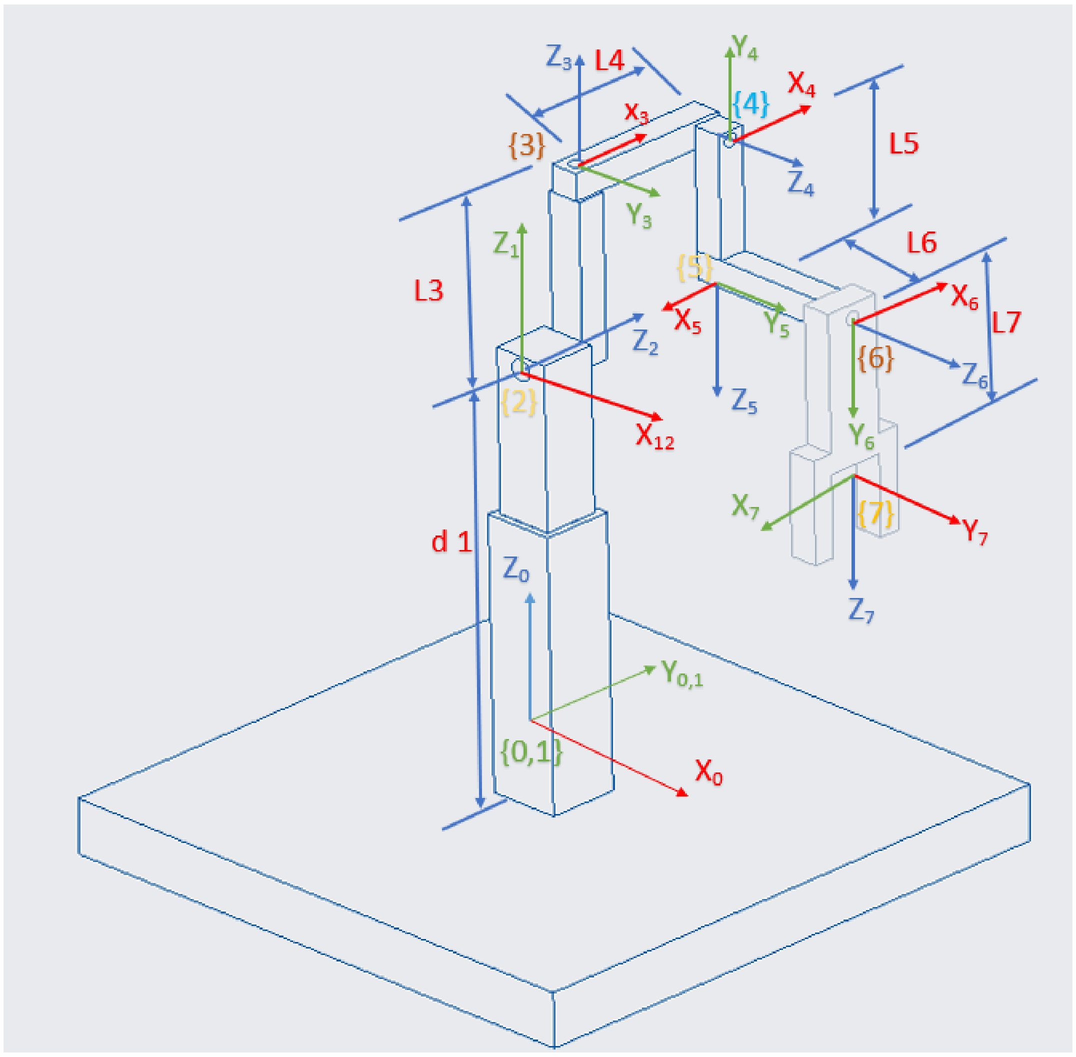


Fig 2: Final design of robotic arm with frames assigned

Table 1: DH Parameters of Robotic Arm

Joint i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	d1	0
2	0	-90°	0	Θ2
3	0	90°	L3	θ_3
4	L4	90°	0	θ_4
5	0	90°	L5	θ_5
6	0	90°	L6	θ_6
7	0	90°	L7	90°

Transformation Matrices of Robotic Arm:

$$ {}^{0}_{1}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} C\theta_2 & -sC\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S\theta_2 & -C\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & 0 \\ 0 & 0 & -1 & L3 \\ S\theta_3 & C\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} C\theta_4 & -S\theta_4 & 0 & L4 \\ 0 & 0 & -1 & 0 \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} C\theta_5 & -S\theta_5 & 0 & 0 \\ 0 & 0 & -1 & -L5 \\ S\theta_5 & C\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{6}T = \begin{bmatrix} C\theta_6 & -S\theta_6 & 0 & 0 \\ 0 & 0 & -1 & -L6 \\ S\theta_6 & C\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{6}_{7}T = \begin{bmatrix} 0 & 1 & 0 & l7 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The resultant matrix is as per below:

$$ {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T\,{}^6_7T = {}^0_ET = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} $$

$r_{11} = -\sin(\theta_3)\cos(\theta_5)+\sin(\theta_5)\cos(\theta_3)\cos(\theta_4)$

$r_{12} = \sin(\theta_3)\sin(\theta_5)\cos(\theta_4)+\cos(\theta_3)\cos(\theta_5)$

$r_{13} = \sin(\theta_4)\sin(\theta_5)$

$r_{14} = 0$

$r_{21} = -(\sin(\theta_3)\sin(\theta_5)+\cos(\theta_3)\cos(\theta_4)\cos(\theta_5))\cos(\theta_6)-\sin(\theta_4)\sin(\theta_6)\cos(\theta_3)$

$r_{22} = -(\sin(\theta_3)\cos(\theta_4)\cos(\theta_5)-\sin(\theta_5)\cos(\theta_3))\cos(\theta_6)-\sin(\theta_3)\sin(\theta_4)\sin(\theta_6)$

$r_{23} = -\sin(\theta_4)\cos(\theta_5)\cos(\theta_6)+\sin(\theta_6)\cos(\theta_4)$

$r_{24}=0$

$r_{31} = -(-\sin(\theta_3)\sin(\theta_5)-\cos(\theta_3)\cos(\theta_4)\cos(\theta_5))\sin(\theta_6)-\sin(\theta_4)\cos(\theta_3)\cos(\theta_6)$

$r_{32} = -(-\sin(\theta_3)\cos(\theta_4)\cos(\theta_5)+\sin(\theta_5)\cos(\theta_3))\sin(\theta_6)-\sin(\theta_3)\sin(\theta_4)\cos(\theta_6)$

$r_{33} = \sin(\theta_4)\sin(\theta_6)\cos(\theta_5)+\cos(\theta_4)\cos(\theta_6)$

$r_{34} = 0$

$r_{41}=$
$-L6(\sin(\theta_3)\cos(\theta_5)-\sin(\theta_5)\cos(\theta_3)\cos(\theta_4))-L7((-\sin(\theta_3))\sin(\theta_5)-\cos(\theta_3)\cos(\theta_4)\cos(\theta_5))\sin(\theta_6)+\sin(\theta_4)\cos(\theta_3)\cos(\theta_6))+(L4+L5\sin(\theta_4))\cos(\theta_3)$

$r_{42} =$

$d1+L4\sin(\theta_3)+L5\sin(\theta_3)\sin(\theta_4)+L6(\sin(\theta_3)\sin(\theta_5)\cos(\theta_4)+\cos(\theta_3)\cos(\theta_5))-L7((-\sin(\theta_3)\cos(\theta_4)\cos(\theta_5)+\sin(\theta_5)\cos(\theta_3))\sin(\theta_6)+\sin(\theta_3)\sin(\theta_4)\cos(\theta_6))$

$r_{43} =$

$L3-L5\cos(\theta_4)+L6\sin(\theta_4)\sin(\theta_5)-L7(-\sin(\theta_4)\sin(\theta_6)\cos(\theta_5)-\cos(\theta_4)\cos(\theta_6))$

$r_{44}=$

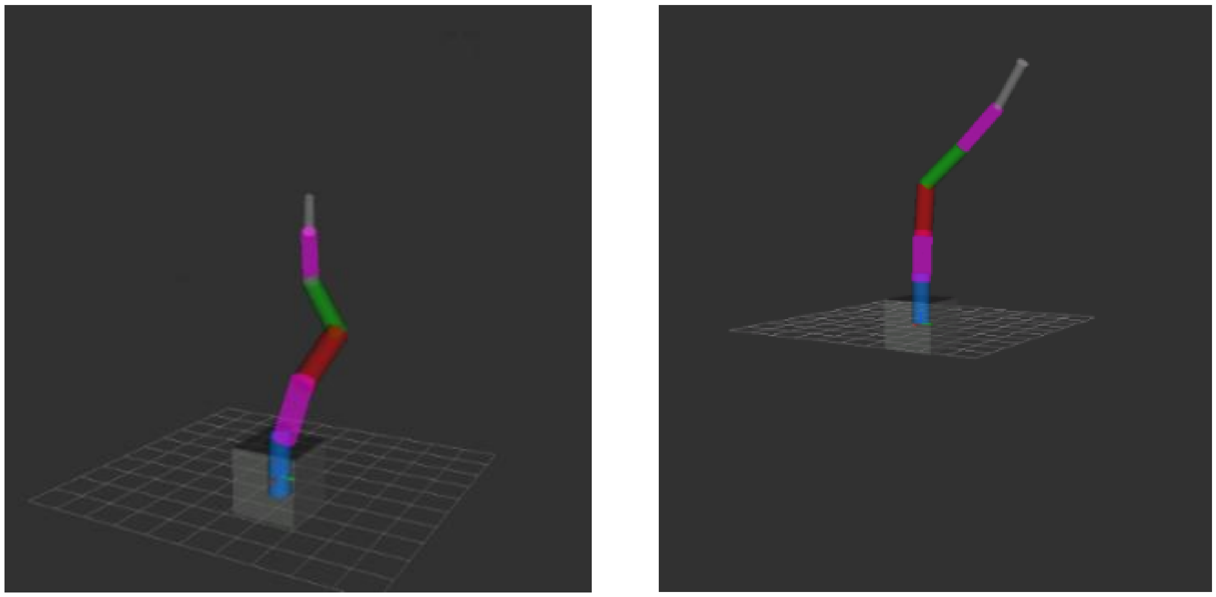
1

ROS Model and ROS Code

The program which was used to create a visual representation of the robotic arm design was ROS. It is an open-source robotic operating system that provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. It is based on a graph architecture where processing takes place in nodes that may receive, post, and multiplex sensor, control, state, planning, actuator, and other messages. ROS allows developers to easily simulate their robot in any environment, before deploying anything in the real world. Tools like Gazebo even allow you to create simulations with robots you don't possess. It's open source.

ROS has an online version that gives a GUI (Graphical User Interface) that is of higher quality than that of the downloadable version (VirtualBox). VirtualBox was used to visualize the robotic arm in this project in which the robotic arm design and intended movement were represented through the code. The VirtualBox version of ROS has a GUI known as RVIZ that allows you to visualize the arm once it has been coded (sdof.xacro code). Once the code is run in the terminal, it will open RVIZ and joint_state_publisher, giving the user the ability to move each link of the robotic arm.

Figures A: ROS Model for Six degree of freedom Robotic Arm (Chitti 4.0)



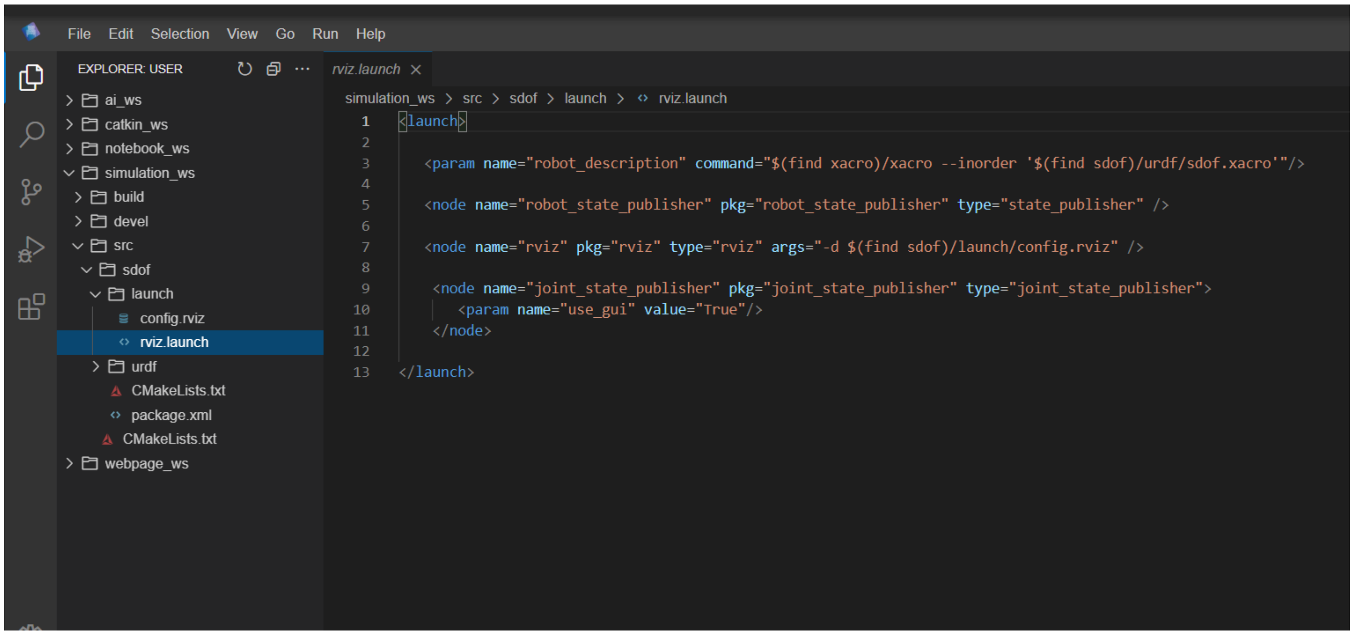
Note: The joint state angles were used to test the rotations of each joint to ensure that the robotic arm functioned properly.

The first step to creating the robotic arm was to create a ROS package named “sdof” within our “src” folder, which was in the “catkin_ws” folder. Within the “sdof” ROS package folder another folder was created called “urdf”, this folder would then include the “sdof.xacro”, a code represented in below page. The code begins with the name of the link you want to describe, within this code represents the base link. The next step was to use the origin and coordinates to describe the rotation and location of the link, in this case, it was zero since the base would be still without rotation. Geometry was then used to describe the shape and the size of the link which was represented as a box. We describe the joint name as well as the type of joint it is, in this example the joint name is “base_link_link_01” which represents the connection between the base link and link 1 and the joint type is a revolute joint. The axis is used to describe the axis on which the joint is on; the joint is represented as being on the z-axis. Origin is similar to that of describing the base link, in which you describe the rotation and location of the joint. The parent link was used to describe the previous link attached to the joint and the child link was used to describe the link after the joint, in this portion of the code our parent link would be the base link and the child link would be the link 1. These codes are repeated over and over until the end effector is met. You begin by describing the link before the joint, the joint itself, and then the link that follows the joint until the end effector is reached. The parent link would be link 1 when it comes to describing the prismatic joint and the child link would be link 2. Links 1 and 2 would be described similarly to the code for the base link with the axis, origin, and location, but instead have radius and length. All links after the first link would be described with a radius and length as the links are cylinders in the ROS representation (Figures A).

sdof.xacro code

```
<?xml version="1.0" ?>
<robot name="sdof" xmlns:xacro="http://www.ros.org/wiki/xacro">
    <link name="base_link">
        <visual>
            <material name="blue">
                <color rgba="0.5 0.5 0.5 1"/>
            </material>
            <origin rpy="0 0 0" xyz="0 0 0" />
            <geometry>
                <box size="2 2 2" />
            </geometry>
        </visual>
    </link>

    <joint name="base_link_link_01" type="prismatic">
        <axis xyz="0 0 1" />
        <limit effort="1000.0" lower="-0.69" upper="0.86" velocity="0.45"/>
        <origin rpy="0 0 0" xyz="0 0 0"/>
        <parent link="base_link"/>
        <child link="link_01"/>
    </joint>

    <link name="link_01">
        <visual>
            <material name="black">
                <color rgba="0 0.5 1 1"/>
            </material>

            <origin rpy="0 0 0" xyz="0 0 0.725" />
            <geometry>
                <cylinder radius="0.35" length="2" />
            </geometry>
        </visual>
    </link>

<joint name="link01_link02" type="revolute">
        <axis xyz="1 0 0" />
        <limit effort="1000." lower="-1.57" upper="0.80" velocity="0.45"/>
        <origin rpy="0 0 0" xyz="0 0 1"/>
        <parent link="link_01"/>
        <child link="link_02"/>
    </joint>

<link name="link_02">
        <visual>
            <material name="yellow">
                <color rgba="255 0 255 1"/>
            </material>

            <origin rpy="0 0 0" xyz="0 0 1.4" />
            <geometry>
                <cylinder radius="0.35" length="2" />
            </geometry>
        </visual>
    </link>
<joint name="link02_link03" type="revolute">
        <axis xyz="0 1 0" />
        <limit effort="1000." lower="-1.57" upper="1.57" velocity="0.45"/>
        <origin rpy="0 0 0" xyz="0 0 2.2"/>
        <parent link="link_02"/>
        <child link="link_03"/>
    </joint>

<link name="link_03">
        <visual>
            <origin rpy="0 0 0" xyz="0 0 1" />
            <geometry>
                <cylinder radius="0.35" length="2" />
            </geometry>
        </visual>
    </link>
<joint name="link03_link04" type="revolute">
        <axis xyz="1 0 0" />
        <limit effort="1000." lower="-1.57" upper="1.57" velocity="0.45"/>
        <origin rpy="0 0 0" xyz="0 0 2"/>
        <parent link="link_03"/>
        <child link="link_04"/>
    </joint>

<link name="link_04">
        <visual>
            <material name="green">
                <color rgba="0 1 0 1"/>
            </material>
            <origin rpy="0 0 0" xyz="0 0 1" />
            <geometry>
                <cylinder radius="0.3" length="2" />
            </geometry>
        </visual>
    </link>
<joint name="link04_link05" type="revolute">
        <axis xyz="1 0 0" />
        <limit effort="1000." lower="-1.57" upper="1.57" velocity="0.45"/>
        <origin rpy="0 0 0" xyz="0 0 2"/>
        <parent link="link_04"/>
        <child link="link_05"/>
    </joint>

<link name="link_05">
        <visual>
            <material name="yellow">
                <color rgba="128 128 0 1"/>
            </material>
            <origin rpy="0 0 0" xyz="0 0 1" />
            <geometry>
                <cylinder radius="0.25" length="2" />
            </geometry>
        </visual>
    </link>
    <joint name="link05_link06" type="revolute">
        <axis xyz="0 1 0" />
        <limit effort="1000." lower="-1.57" upper="1.57" velocity="0.45"/>
        <origin rpy="0 0 0" xyz="0 0 2"/>
        <parent link="link_05"/>
        <child link="link_06"/>
    </joint>
```

```
<link name="link_06">
        <visual>
            <material name="violet">
                <color rgba="1 1 1 1"/>
            </material>
            <origin rpy="0 0 0" xyz="0 0 1" />
            <geometry>
                <cylinder radius="0.2" length="2" />
            </geometry>
        </visual>
    </link>
</robot>
```



```
<launch>

    <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find sdof)/urdf/sdof.xacro'"/>

    <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />

    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find sdof)/launch/config.rviz" />

     <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
        <param name="use_gui" value="True"/>
     </node>

</launch>
```

Fig 3 rviz.launch code

Within the “sdof” folder another folder was created called “launch”, this folder would then include the “rviz.launch” file represented in Figure 3. This file enables the use of the RVIZ GUI, which gave us the visual representation needed. This file also allowed the joint_state_publisher to come up once the code was run in the terminal, the joint_state_publisher gave the user the ability to manipulate the links as seen in Figures A.

MATLAB Code and Simulation

It is evident that the accuracy of DH parameters was critical to calculating forward kinematics and performing Jacobian calculations. To calculate forward kinematics, the first transformation matrices were defined from one frame to the other in terms of joint variables. The textbook method shown in figure B shown below was utilized to calculate all the respective transformation matrices.

$$ {}^{i-1}_{i}T = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figure B

Having all the DH parameters and transformation matrices was a big step, but to begin any MATLAB programming, a pseudo code manuscript in the form of a flow chart was outlined (See Figure 5). The flow chart made it easy to visualize all the steps and corresponding scripts that would be required to be written to get a final working MATLAB program.

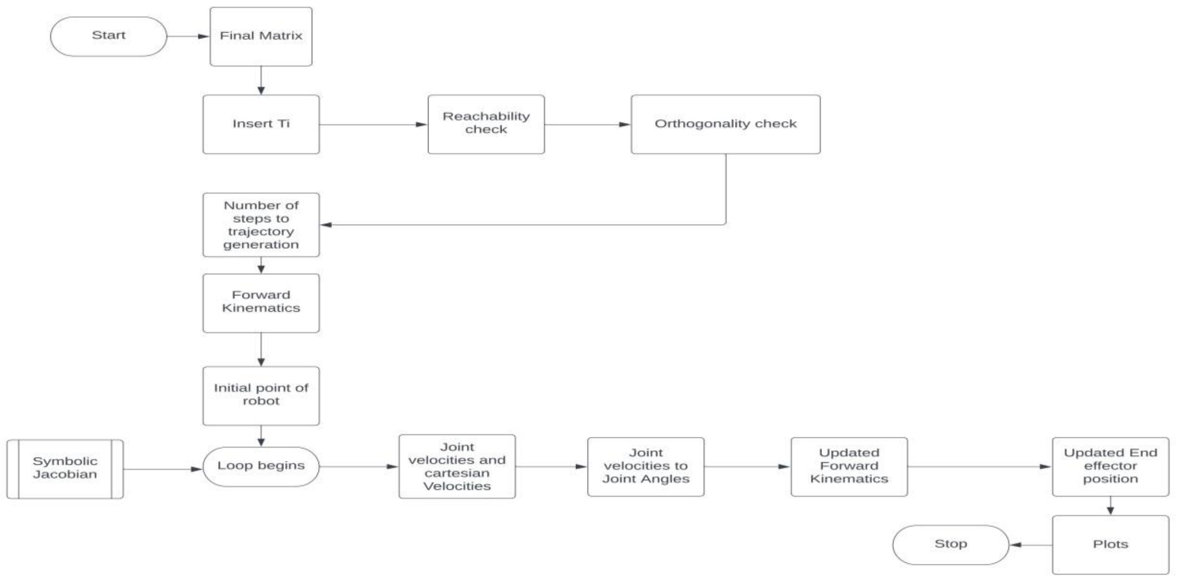
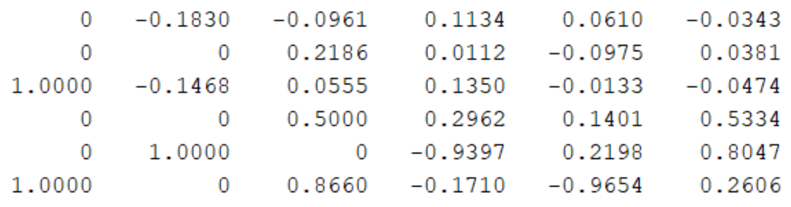

Start
Final Matrix
Insert Ti
Reachability check
Orthogonality check
Number of steps to trajectory generation
Forward Kinematics
Initial point of robot
Symbolic Jacobian
Loop begins
Joint velocities and cartesian Velocities
Joint velocities to Joint Angles
Updated Forward Kinematics
Updated End effector position
Plots
Stop

Figure 5: MATLAB program Simulation flowchart

The next step was to create a script to find the Jacobian. To keep the performance of the program fast, the Jacobian was found using a symbolic method in a stand-alone script and then the resultant Jacobian was transcribed to the program's main script (main.m). The Jacobian was found using MATLAB's built-in function "Jacob" and passing the derivative method argument to the function.

Based on the values provided for initial transformation matrix the resultant Jacobian is as below

J0 =



```
        0   -0.1830   -0.0961    0.1134    0.0610   -0.0343
        0         0    0.2186    0.0112   -0.0975    0.0381
   1.0000   -0.1468    0.0555    0.1350   -0.0133   -0.0474
        0         0    0.5000    0.2962    0.1401    0.5334
        0    1.0000         0   -0.9397    0.2198    0.8047
   1.0000         0    0.8660   -0.1710   -0.9654    0.2606
```

After the general Jacobian was defined, the resolved rate approach was used to simulate motion. The idea was that with an initial position and a desired position of the end effector, a time step could be calculated. The time step can be arbitrary, but the more time steps taken in 30 between the two locations, the better the motion can be simulated. With a time, step defined, the Jacobian and the inverse of the Jacobian are calculated at each step. These values are used to calculate a cartesian velocity, the position, and the angles of the robot joints at each time step. Furthermore, the new joint angles are used to recalculate the forward kinematics at each time step. The forward kinematics were then used to plot the links at each time step which yields the simulation of motion. In addition, graphs of the joint angles, joint velocities, end effector cartesian position, Euler Angles (Alpha, Beta, & Gamma), and Time vs Joint angles, Time vs Joint Velocity, Time vs Cartesian Position, Time vs Jacobian Determinant were plotted. These results are shown in the below images in Figures C, D, E, F.

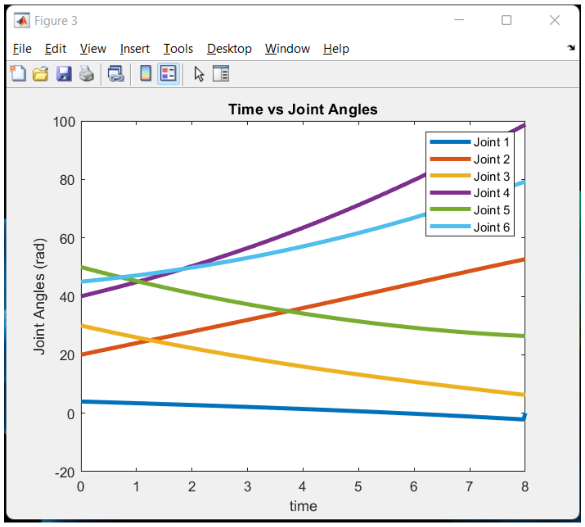


Fig: C

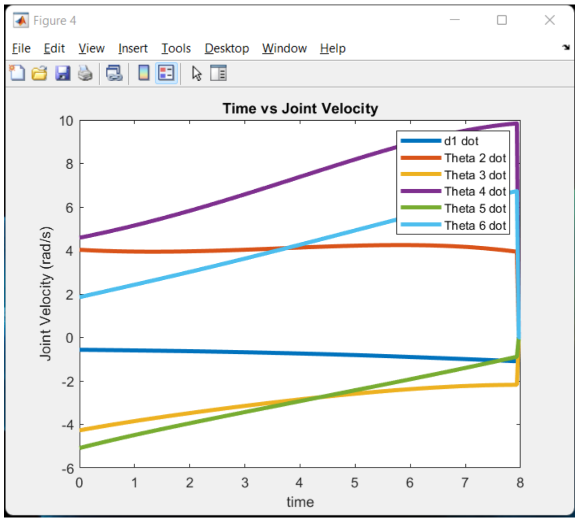


Fig: D

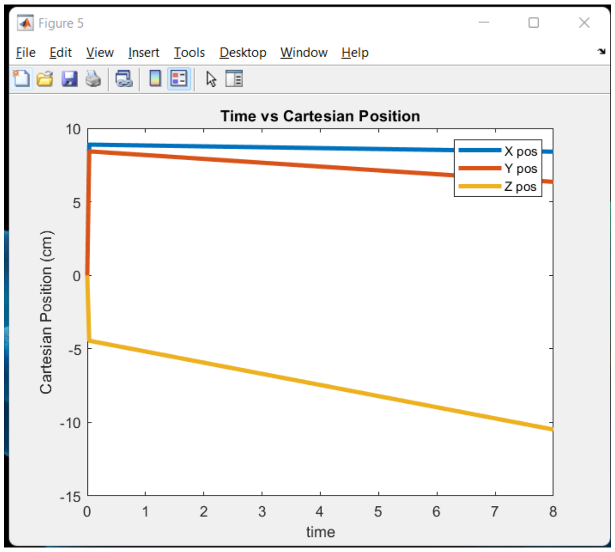


Fig: E

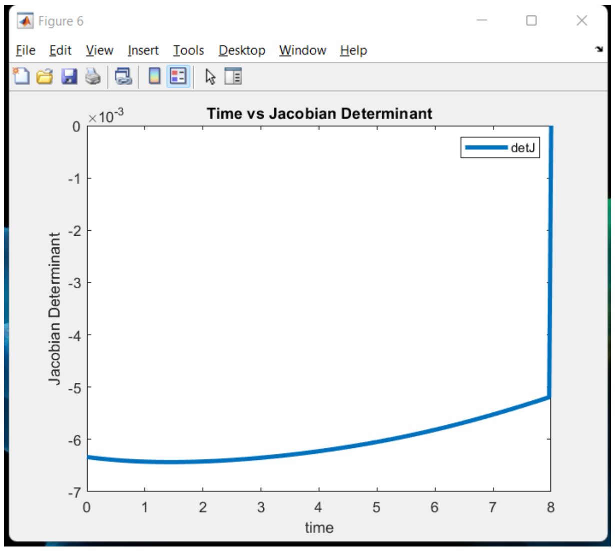


Fig: F

Conclusion/Future Scope

The concept of forward kinematics refers to the use of kinematic equations in a robot design to compute the position of the end-effector/gripper based on specified values for each of the joints and its corresponding coordinate frame. The objective of this project was to design, simulate, and test a 6-DoF robotic arm using forward kinematics. The program SolidWorks was used to design the arm, while it was digitally constructed using the ROS program, and simulated using MATLAB coding. The transformation matrices, the Euler angles, and the Jacobian matrix were found using forward kinematics in the MATLAB code. Once the script was run, the joint angles, cartesian position, Jacobian determinant, and joint velocity graphs were produced. Through these figures and code testing, it was concluded that the arm was functional at different positions. Possible future functions of the arm could include working on an assembly line or working in shipping. The Robotics arm revolute Joints can be designed in different way such that efficiency of the robotic arm movement can be improved.