Final Design Project

Motion Planning for a 7-DOF Robotic Arm

Alexander Rufrano - 111270310

Faculty Advisor: Nilanjan Chakraborty

Department of Mechanical Engineering

May 17, 2020

I. Contents

II. Problem Statement 3
III. Description of the ScLERP Method and Algorithm used 4
IV. Simulation Results 11
A. Translation and Rotation 11
Figure. 1 Rotation of 45-degrees about y-axis and Translation 11
Figure. 2 Comparison between ROT+TRANS with and without Joint Limit 12
B. Difference Between Pure Translation vs Translational and Rotational 13
Figure. 3 Difference between Pure Translation and Trans/Rot mix. 13
C. Pure Translation 14
Figure. 4 Pure Translation and no change in end effector orientation. 14
Figure. 5 Comparing between two purely translational instances 15
Figure. 6 Comparing for translation with and without joint limits exceeding. 16
Figure. 7 Another view of showing joint limits exceeding. 16
D. Pure Rotation (Only about the y-axis) 16
Figure. 8 Pure Rotation about y-axis of 20 degrees 16
Figure. 9 Pure rotation of -45 and +20 degrees both from initial position 17
V. Animation Results 18
Figure. 10 Comparison between outputs of Baxter 18
Figure. 11 Code to plot simulation of Baxter 18
Figure. 12 Configuration of Baxter provided in Robotic System Toolbox 19
Figure. 13 My output of orientation and position of end effector in space. 19
Figure. 14 My output of Orientation and Position of end effector in space. 20
Figure. 15 My output of Orientation and Position of end effector in space 20
Figure. 16 Plotting Baxter's Movements based on my joint angles. 21
Figure. 17 Plotting Baxter's Movements based on my joint angles. 21
Figure. 18 Plotting Baxter's Movements based on my joint angles. 22
VI. Conclusion 22

II. Problem Statement

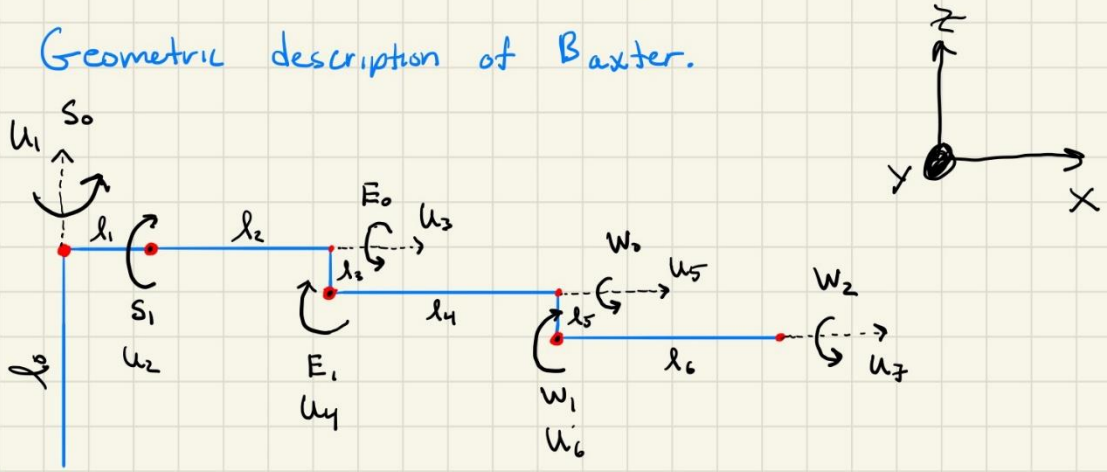
The purpose of this report is to demonstrate the utilization and design of a motion planning algorithm for a 7-DOF robot. The robot used for this report is the Baxter robot. To demonstrate the use of a motion planning algorithm, the robot must be given a specific task. This task can include anything as long as the task is within the robot's capabilities, such as moving the end effector from one point to another. The task that is chosen for this report is moving its end effector – or "hand"- from one random location to another for different instances. Each instance is unique. One instance is when the orientation is constant, where theta initial is equal to theta final, but the position of the end effector changes. Another instance is when the position of the end effector is constant, but the orientation changes – the end effector can only rotate about the rotational axis of the end effector. The last instance is when the position and orientation of the end effector change and reaching the desired goal.

III. Description of the ScLERP Method and Algorithm used

Motion Planning for a 7DOF Baxter arm

To begin the design of a Motion planning algorithm, there are certain quantities that must be given before we move forward. These quantities include the initial and final configuration, initial joint angles, joint angle limits, geometric description of the robot links, and if collision avoidance is required, then the geometric description of that obstacle.

Geometric description of Baxter.



There are a total of 7 revolute joints and $\emptyset$ prismatic. The axis of rotation and a point on that axis for each revolute joint must be known as it is required to know geometric description. These quantities are going to be used to find the intermediate position between the initial and final configuration using the initial configuration throughout the calculation because it only depends on its initial configuration for forward kinematics.

Let

$u_1 = [0, 0, 1]^T$ $q_1 = [0, 0, l_0]^T$ $q_6 = [0, l_1+l_2+l_4, l_0-l_3-l_5]^T$

$u_2 = [0, 1, 0]^T$ $q_2 = [0, l_1, l_0]^T$

$u_3 = [1, 0, 0]^T$ $q_3 = [0, l_1+l_2, l_0]^T$ $q_7 = [0, l_1+l_2+l_4+l_6, l_0-l_3-l_5]^T$

$u_4 = u_2$ $q_4 = [0, l_1+l_2, l_0-l_3]^T$

$u_7 = u_5 = u_3$ $q_5 = [0, l_1+l_2+l_4, l_0-l_3]^T$

$u_6 = u_2$

Since the initial and final configuration, along with geometric description of Baxter is known, we can know the position of Baxter for any change in its joint angles using direct foward kinematics of exponential configuration.
The iterative motion planning algorithm consist of Seven (7) steps.

Note: If the motion is purely translational, these steps differ slightly. Explanation is at the end of all steps.

Step 0

The zeroth step is knowing $g(t)$ and $\theta(t)$, when $t=0$, $t=$ final, where $\theta(t)$ is the joint angles of all the joints the start/End time. Use $g(0)$ and $\theta(0)$ as the starting point.

Step 1

Convert $g(\theta(t)) \xrightarrow[\text{reference}]{} Y(t)$ and $g(\theta(t)) \xrightarrow[\text{reference}]{} A(t)$

when $t=0$, $g_{ref} = g(0) = g_{initial}$

where $g(\theta(t))_{Reference} = \begin{bmatrix} R(t) R_{final} & p \\ 0 & 1 \end{bmatrix} = g_0^{-1} g_{final} \equiv D_0^* \otimes D_f$, $R_{ref} = R^T(\theta(t)) R_{final}$

where $A(t) = A_r(t) + \epsilon A_d(t)$

A_r is found from converting the rotation matrix R_{ref} of $g(\theta(t)) = \begin{bmatrix} R_{ref} & p \\ 0 & 1 \end{bmatrix}$ into a unit quarterion, $Q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$

where $A_r = Q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ and $A_d = \frac{1}{2} p_{\text{end effector}} \otimes A_r$

and $Y(t) = \begin{bmatrix} p \\ Q \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$

Step 2

Compute $A(t+1) = A(t) \otimes \underbrace{(A^*(t) \otimes A_f)}_{A_{reference}}{}^{\tau}$ where tau, $\tau = 0.1$

and $A^*(t) = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix} + \epsilon \begin{bmatrix} p_0 \\ -p_1 \\ -p_2 \\ -p_3 \end{bmatrix}$ and $A_f = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + \epsilon \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$

$A^*(t) \otimes A_f = A_{reference}$ which is a similar idea to $g_{reference}(t) = g_0^{-1} g_{final}$

To raise $A_{reference}$ to the τ power, there is a specific procedure that needs to be used as this is a dual quarterion.

With $A_{reference} = A_{r,reference} + \epsilon A_{d,reference}$

$A_{r,reference} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = Q_{reference}$, where $Q_{reference} = Q_{reference}(\theta_{reference}, u_{reference} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix})$

where $\theta_{reference} = 2\cos^{-1}(q_0)$

$u_1 = q_1 / \sin(\frac{\theta_{ref}}{2})$

$u_2 = q_2 / \sin(\frac{\theta_{ref}}{2})$

$u_3 = q_3 / \sin(\frac{\theta_{ref}}{2})$

$A_{d,reference} = \begin{bmatrix} P_{reference} \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$

Since $\theta_{reference}, u_{reference}, P_{reference}$, and τ_{dual} are known, $(A^*(t) \otimes A_f)^{\mp} = (A_{reference})^{\mp}$ can be found.

Also, $(A_{reference})^{\mp} = \cos(\frac{\mp\bar{\theta}}{2}) + \bar{u}\sin(\frac{\mp\bar{\theta}}{2})$

where $\sin(\frac{\mp\bar{\theta}}{2}) = \sin(\frac{\mp\theta}{2} + \epsilon\frac{\mp d}{2}) = \sin(\frac{\mp\theta}{2}) + \epsilon\frac{\mp d}{2}\cos(\frac{\mp\theta}{2})$

$\cos(\frac{\mp\bar{\theta}}{2}) = \cos(\frac{\mp\theta}{2} + \epsilon\frac{\mp d}{2}) = \cos(\frac{\mp\theta}{2}) - \epsilon\frac{\mp d}{2}\sin(\frac{\mp\theta}{2})$

$\bar{u} = u + \epsilon m$, $d = p \cdot u = 2A_d \otimes A_r^* \cdot u$

$m = \frac{1}{2}(p \times u + (p - du)\cot(\frac{\theta}{2}))$

* All values used $\theta_{reference}$, $P_{reference}$, and $u_{reference}$ are from $A_{reference}$

At this point, $(A^*(t) \otimes A_{final})^{\mp} = (A_{reference})^{\mp}$ can be computed.

This means $A(t+1) = A(t) \otimes A_{reference}$ can be computed as well.

where $A(t+1) = A_r(t+1) + \epsilon A_d(t+1)$

Note:

Dual quarteron Multiplication

$A = P + \epsilon Q$, $B = u + \epsilon V$

$A \otimes B = P \otimes u + \epsilon(Q \otimes u + P \otimes V)$

($A \otimes B$: dual quarteron; $P \otimes u$, $Q \otimes u$, $P \otimes V$: quarterions)

quarteron multiplication

$A = P \otimes q = \begin{bmatrix} q_0 \cdot p_0 - dot(q,p) \\ q_0 p + p_0 q + q \times p \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$

where, $P = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$, $q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ are quarterions.

Step 3 Convert $A(t+1) \rightarrow y(t+1)$

Similarly in step ①, $A(t+1) = A_r + \epsilon A_d = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + \epsilon \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$, $p_0 = 0$

$y(t+1) = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$

Step 4 Compute $\Theta(t+1) = \Theta(t) + \beta B(\Theta(t))(\gamma(t+1) - \gamma(t))$

Where β changes due to joint constraint, but for this Algorithm, I did something unique where if $\Theta(t+1)$ goes beyond joint limits, it automatically sets it to a value within the joint limits but also makes it within 10% of being near that joint limit and not exactly at the limit. The value of β was kept at a constant value of $\beta = 0.1$.

The only variable that is missing is $B(\Theta(t))$

where $B(\Theta(t)) = (J^s)^T (J^s (J^s)^T)^{-1} J_2$

$$J_2 = \begin{bmatrix} I & 2\hat{P}J_1 \\ 0 & 2J_1 \end{bmatrix}, \quad J_1 = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}$$

where $A_r = Q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ is from Step one based on $R_{reference}$ and is used to find J_1.

J_2 uses J_1 and P based on Step one from the q_{ref} configuration. NOTE for J_2 $P \neq \hat{P}$ = Skewsymmetric matrix.

J^s is the Spatial Jacobian, where $J^s = J^s(u, q, \text{Joint type}, \Theta(t+1))$

The q for the Spatial Jacobian is the points on the axis of rotation, u, that both were found when determining the Robot configuration on the first page.

The only component that changes for the Spatial Jacobian is $\Theta(t)$ and $u, q,$ type stay constants.

Now that $\Theta(t)$, J^s, J_1, and J_2 are known, $B(\Theta(t))$ can be computed.

Since $B(\theta(t))$, $\gamma(t)$, $\gamma(t+1)$, β and $\theta(t)$ are known, $\theta(t+1)$ can be found using the formula above.

$$\theta(t+1) = \theta(t) + \beta B(\theta(t))(\gamma(t+1) - \gamma(t))$$

Ensure joint angles do not go above or below joint limits

$$\theta_{k,min} \leq \theta(t+1) \leq \theta_{k,max}$$

$$\theta_{k,min} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix} \leq \theta(t+1) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix} \leq \theta_{k,max} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix}$$

When $\theta(t+1) > \theta_{k,max}$

$$\theta_{k,max} - \theta(t+1) = \theta_{high}$$

Set all positive values in θ_{high} to $\emptyset$

Multiply θ_{high} by 1.1. $\theta_{high} = 1.1\theta_{high}$

$\theta(t+1) = \theta(t+1) - \theta_{high}$ (Now all values beyond limit are within the limit)

When $\theta(t+1) < \theta_{k,min}$

$$\theta_{k,min} - \theta(t+1) = \theta_{Low}$$

Set all negative values in θ_{Low} to $\emptyset$

$\theta_{Low} = 1.1\theta_{Low}$

$\theta(t+1) = \theta(t+1) + \theta_{Low}$

(All values below limit are within the limit + 10%.)

Step 5 Compute $g(t+1) = FK(\theta(t+1))$

Since $\theta(t+1)$, u, q, and the type of joints are known, $g(t+1)$ can be found by finding each exponential twist for each joint using $\theta(t+1)$, u, and q.

$$g_{st}[\theta(t+1)] = e^{\xi_1\theta_1} e^{\xi_2\theta_2} e^{\xi_3\theta_3} e^{\xi_4\theta_4} e^{\xi_5\theta_5} e^{\xi_6\theta_6} e^{\xi_7\theta_7} g_{st}(0)$$

Since $\xi = \begin{bmatrix} -u \times q \\ u \end{bmatrix}$ and $e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & p \\ 0 & 1 \end{bmatrix}$, $e^{\hat{\omega}\theta} = I + \hat{\omega}\sin(\theta) + \hat{\omega}^2(1 - \cos(\theta))$

Step 6

If $|g_{final} - g(t+1)| < 0.1\%$

then Stop iteration

else, go to Step 7

STep 7

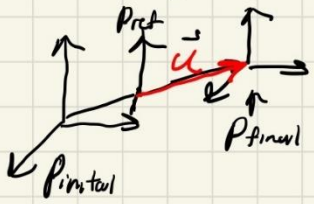
Store $t \leftarrow t+1$, $g(t) \leftarrow g(t+1)$, $\theta(t) = \theta(t+1)$

and go back to Step 1.

When $\theta(t) = \theta(0) = \theta(t+1) = [0,0,0]^T$

When the end effector does not change its orientation but it changes its position, there are a couple of modifications that take place.

For Step 2, the u will be found from finding the direction of $P = \underbrace{P_{final}}_{\text{Desired position}} - \underbrace{\text{Preference}}_{\text{position you're currently in.}}$



To find the unit vector

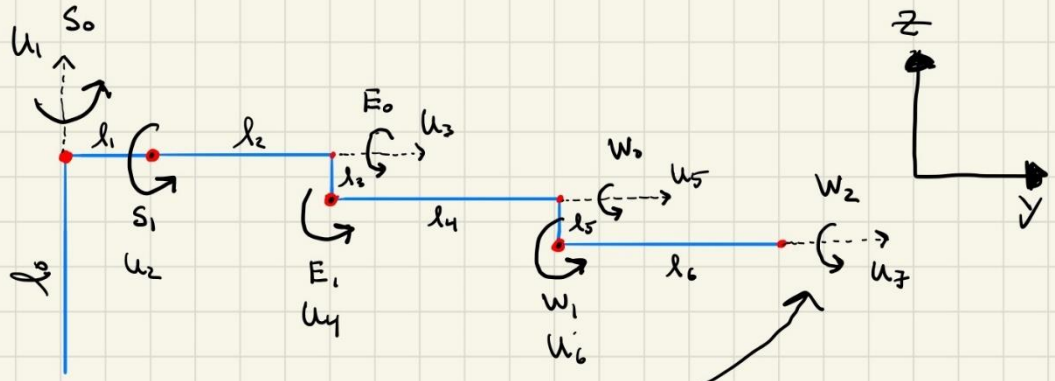
$$u = \frac{P}{\|P\|} \quad \left[\text{Direction in current State to final state}\right]$$

where $u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$

Also, Set $m = \frac{1}{2}(p \times u)$ where, $\bar{u} = u + \epsilon m = u + \epsilon \frac{1}{2} p \times u$ which eliminates, $(p - du)\cot\frac{\theta}{2}$.

When $P_{final} = P_{initial}$, but $\theta(0) \neq \theta(t+1) \neq \theta(t)$

Based on the configuration of the baxter robot arm, the only thing that changes is the orientation of the frame. This limits baxter to only be able rotate purely only in the y-axis because the arm is not capable of purely rotating in any other direction.



As you can see, at u_7, the arm is only able to purely rotate about the y-axis and nothing else, in terms of rotating the end effector and not changing the position of the end effector.

IV. Simulation Results

All simulations below were successful. The end effector successfully reached the desired goal with the proper orientation and was within 0.1% of absolute error.

A. Translation and Rotation

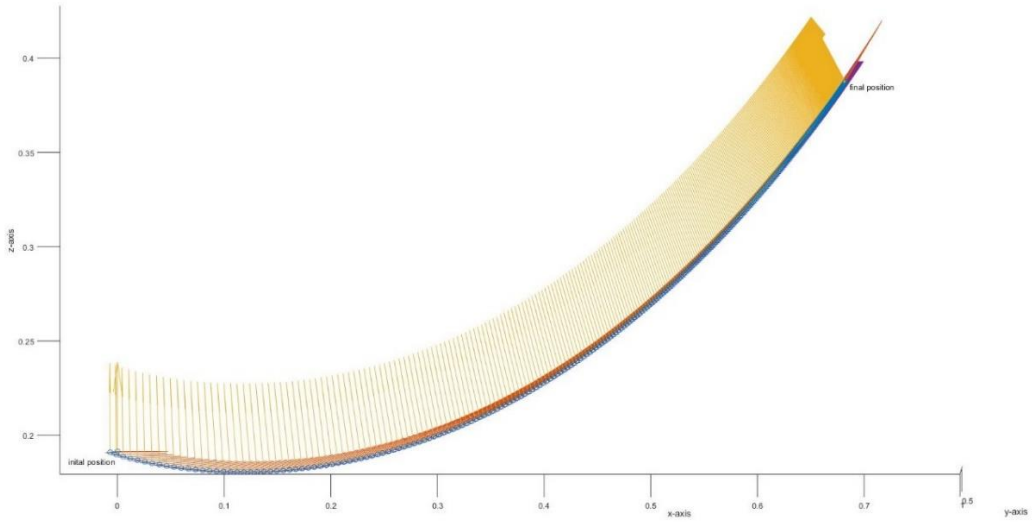


Figure. 1 Rotation of 45-degrees about y-axis and Translation

This is showing the rotation about the y-axis which is pointing in the page and the plane of the z- and x- axis. There were no joint angle limitations that were observed. **NOTE**: The orientation of the end effector is given by arrows pointing in a the x, y, and z direction, as shown in all figures. This shows the instantaneous change of the orientation of the end effector.

Results when the end effector orientation and position change with joint limitation consideration:

```
K>> gst_step(1:4,1:4,1)

ans =

    1.0000         0         0         0
         0    1.0000         0    1.0372
         0         0    1.0000    0.1914
         0         0         0    1.0000

g_final =

    0.7071    0.7071         0    0.5000
   -0.7071    0.7071         0    0.3000
         0         0    1.0000    0.5000
         0         0         0    1.0000

ans =

    0.7074    0.7069   -0.0010    0.4999
   -0.7069    0.7074    0.0001    0.3001
    0.0008    0.0006    1.0000    0.5002
         0         0         0    1.0000
```

Based on the results above, any simulation that is within the limits of the robot and the orientation and position of the end effector frame are not constant, then it will be successful.

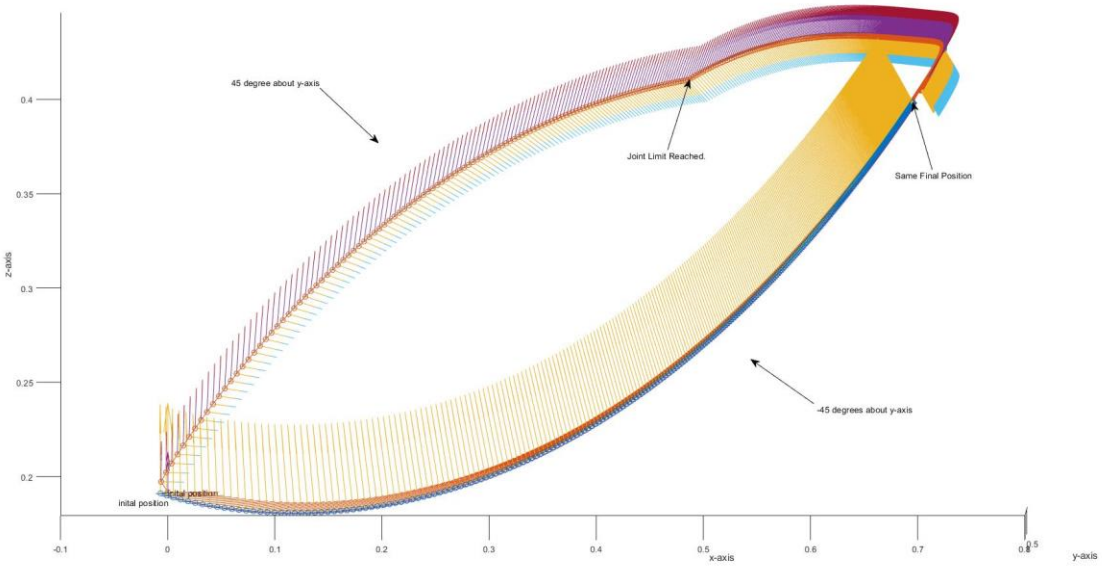


Figure. 2 Comparison between ROT+TRANS with and without Joint Limit

Both of these paths start from the same initial position and end in the same final position. The purple and yellow path is when the orientation is rotated negative 45 degrees and the yellow and orange path, which is also in the figure above has its orientation rotated (+ve) 45 degrees. Comparing to the previous figure, the purple and yellow arrows are illustrating that there is a joint constraint in the current path – shown by the arrow in the figure pointing to when it is constrained. The program accounts for joint constraints by forcing it not to exceed the limitations that it is provided by going to a new path where the limits are not exceeded. It was expected that when the orientation is rotated about a (-ve) 45 degrees that it will endure some type of constraint. **NOTE**: The final position that is chosen is not directly on the X/Z plane.

Results when the end effector orientation and position change with joint limitation consideration:

```
K>> gst_step(1:4,1:4,1)

ans =

    1.0000         0         0         0
         0    1.0000         0    1.0372
         0         0    1.0000    0.1914
         0         0         0    1.0000

g_final =

    0.7071   -0.7071         0    0.5000
    0.7071    0.7071         0    0.3000
         0         0    1.0000    0.5000
         0         0         0    1.0000

ans =

    0.7079    0.7063   -0.0030    0.4998
   -0.7063    0.7079    0.0004    0.3002
    0.0024    0.0019    1.0000    0.5007
         0         0         0    1.0000
```

Based on the results above, any simulation that is within the limits of the robot and the orientation and position of the end effector frame are not constant, then it will be successful even when the joint limits were accounted for.

B. Difference Between Pure Translation vs Translational and Rotational

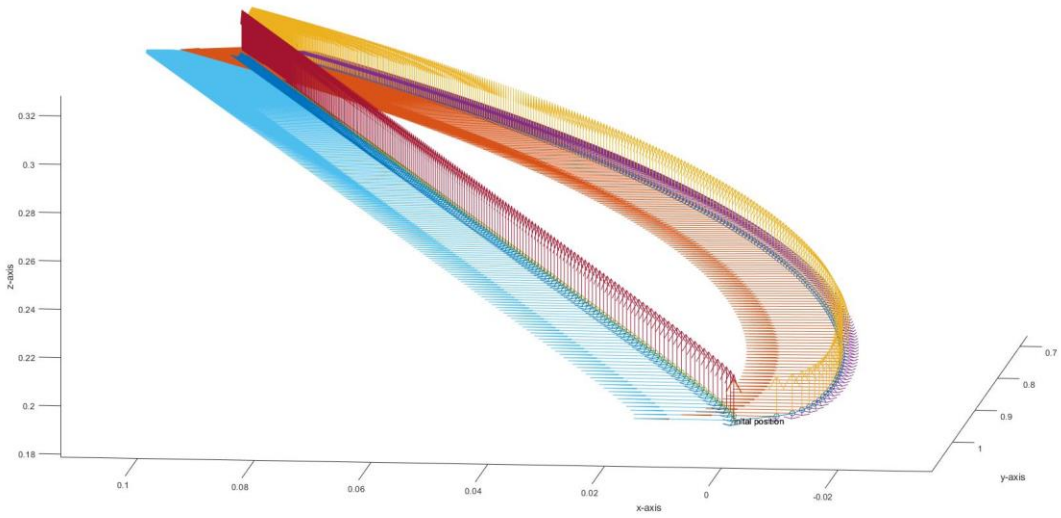


Figure. 3 Difference between Pure Translation and Trans/Rot mix.

The pure translational path is colored in blue and red, and the translational and rotational is in orange and yellow. Both of these end up in the same final position but different orientations. The end effector was set to rotated about the z-axis at 45 degrees. The final position for the x, y, and z points was set to 100mm, 700mm, and 300mm, respectfully. There were no joint constraints in this plot.

C. Pure Translation

For pure translation, when theta=0 for the end effector, there were modifications made to the program as described in section 3 page 9.

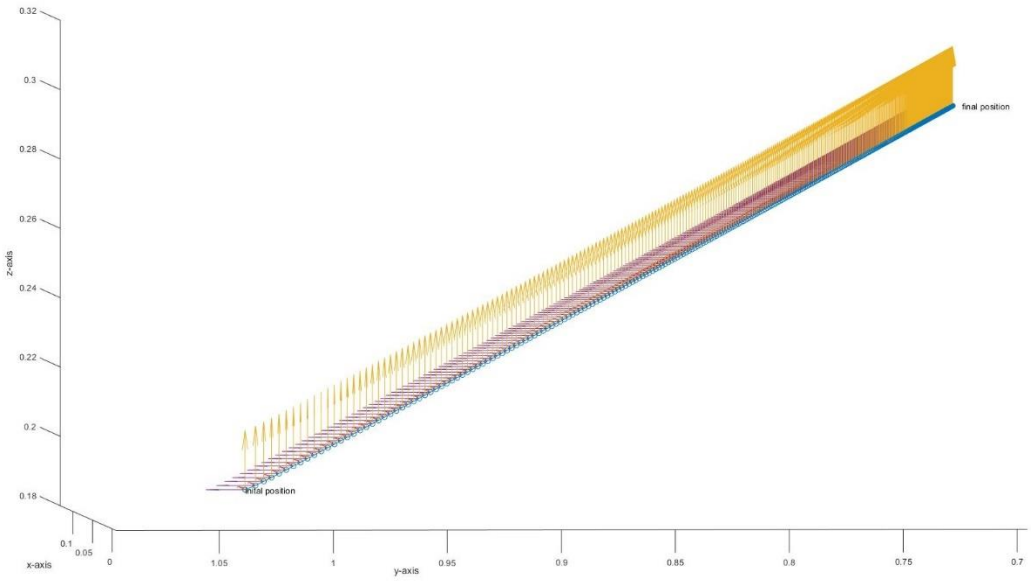


Figure. 4 Pure Translation and no change in end effector orientation.

The final position for the x, y, and z points was set to 100mm, 700mm, and 300mm, respectfully.

Results when the orientation of the end effector is constant and the final position changes:

```
K>> gst_step(1:4,1:4,1)

ans =

    1.0000         0         0         0
         0    1.0000         0    1.0372
         0         0    1.0000    0.1914
         0         0         0    1.0000
g_final =

    1.0000         0         0    0.1000
         0    1.0000         0    0.7000
         0         0    1.0000    0.3000
         0         0         0    1.0000
```

```
K>> gst_step(1:4,1:4,i)

ans =

    1.0000   -0.0000   -0.0000    0.0991
    0.0000    1.0000    0.0000    0.7030
    0.0000   -0.0000    1.0000    0.2990
         0         0         0    1.0000
```

Based on the results above, any simulation that is within the limits of the robot and position of the end effector frame are not constant, but the orientation is constant, then it will be successful even when the joint limits were accounted for.

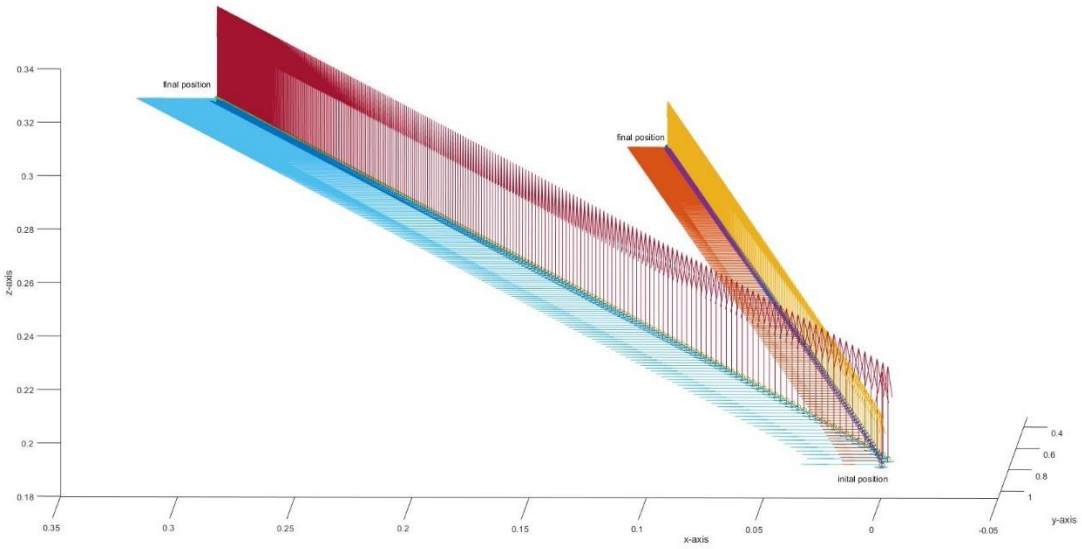


Figure. 5 Comparing between two purely translational instances

To compare between two instances, with the x, y, and z points of 100mm, 700mm, 300mm to 300mm, 300mm, 300mm, respectfully. The orientation does not change at all and it is only translating the end effector. It should be noted that neither of these instances had the joint limits exceeded. The end effector successfully reached the goal with the proper orientation.

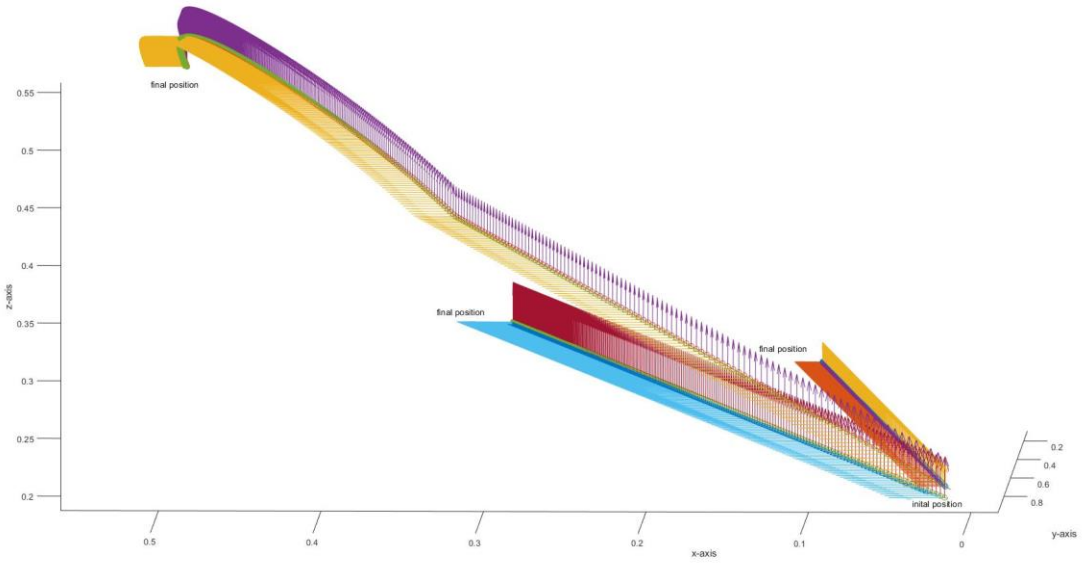


Figure. 6 Comparing for translation with and without joint limits exceeding.

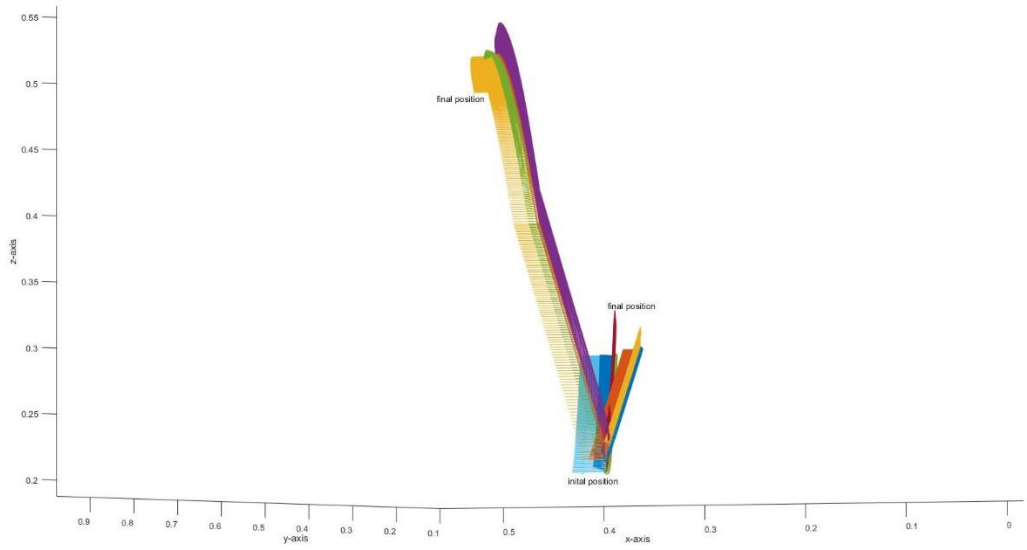


Figure. 7 Another view of showing joint limits exceeding.

When the joint limits are exceeded, as shown by the curve in the path of the yellow and purple arrows, it forces the robot to go another path, but it never changes its orientation. This is compared to the other two instances of when the joint limits were not exceeded. The second the slope starts to change is when the joint limits have been exceeded in one of the joints and it takes on a new path. The error was within 0.1%.

D. Pure Rotation (Only about the y-axis)

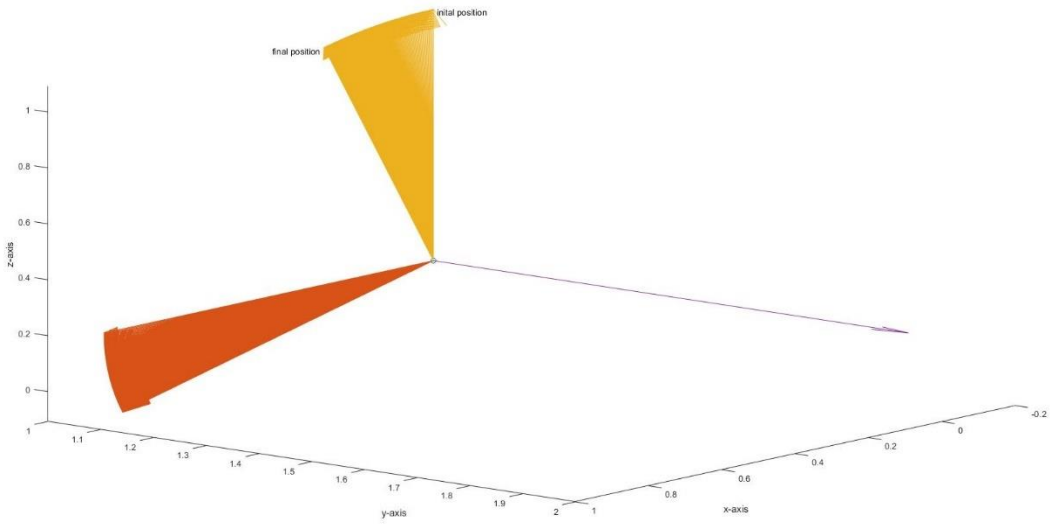


Figure. 8 Pure Rotation about y-axis of 20 degrees

This is a pure rotational situation where p_inital = p_final, but the change in theta is 20-degrees.

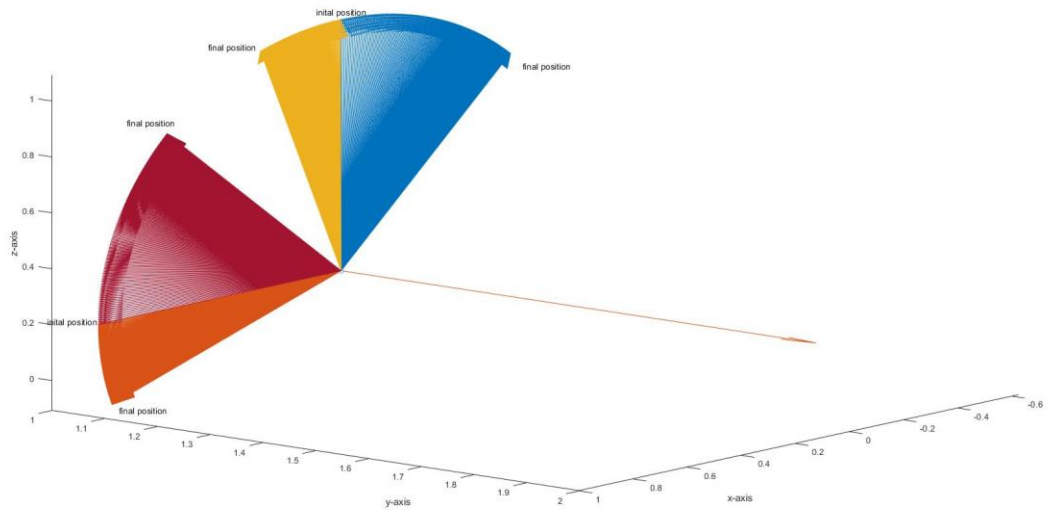


Figure. 9 Pure rotation of -45 and +20 degrees both from initial position

This is a pure rotational situation where this is comparing between a (+ve) 20-degree rotation with a (-ve) 45-degree rotation about the y-axis. It should be noted that the Baxter robot is limited to only a rotation about the end effector axis, which is the y-axis in this situation, and it can not pure rotate about any other axis. When I tried simulating any other axis, the end effector would need to move its position which then makes it not purely rotational.

Results when it rotates about the y-axis by 45 degrees and the end effector is constant:

```
K>> gst_step(1:4,1:4,1)

ans =

    1.0000         0         0         0
         0    1.0000         0    1.0372
         0         0    1.0000    0.1914
         0         0         0    1.0000

K>> g_final

g_final =

    0.7071         0   -0.7071         0
         0    1.0000         0    1.0372
    0.7071         0    0.7071    0.1914
         0         0         0    1.0000
```

```
K>> gst_step(1:4,1:4,i)

ans =

    0.7101    0.0000   -0.7041   -0.0000
   -0.0000    1.0000    0.0000    1.0372
    0.7041    0.0000    0.7101    0.1914
         0         0         0    1.0000
```

V. Animation Results

A full video of the animation and explanation can be found on YouTube at: https://youtu.be/m_UQjZQDWMc. The video only of the animation will be submitted with the report.

For the animation I chose that the orientation of the end effector will be rotated -45 degree about the z-axis and the initial point is [0;0;0] to a final [0.7;0.1;0.4] position. In the plots below, I achieved that goal only for the plots based on the position from the forward kinematics that was derived in class. The position function that was given from the Robotics System Toolbox which I inputted all of the joint angles that were found through computation of the iterative motion planning program did not produce the same position as shown in the figures of Baxter.
From my understanding, this has to do with the robot's initial configuration that I choose compared to what the developer chose for Baxter. For instance, the developer's rotational axes are not the same as mine except for the z-direction. Also, the developer's initial position is not the same as mine shown in the the figure below, as opposed to mine it is in the [0;1.372;0.1914] position. The only thing trait that I feel we both chose to be the same is the z-axis direction because for both plots, it did change the orientation -45 degrees about the z-axis which is also shown below.

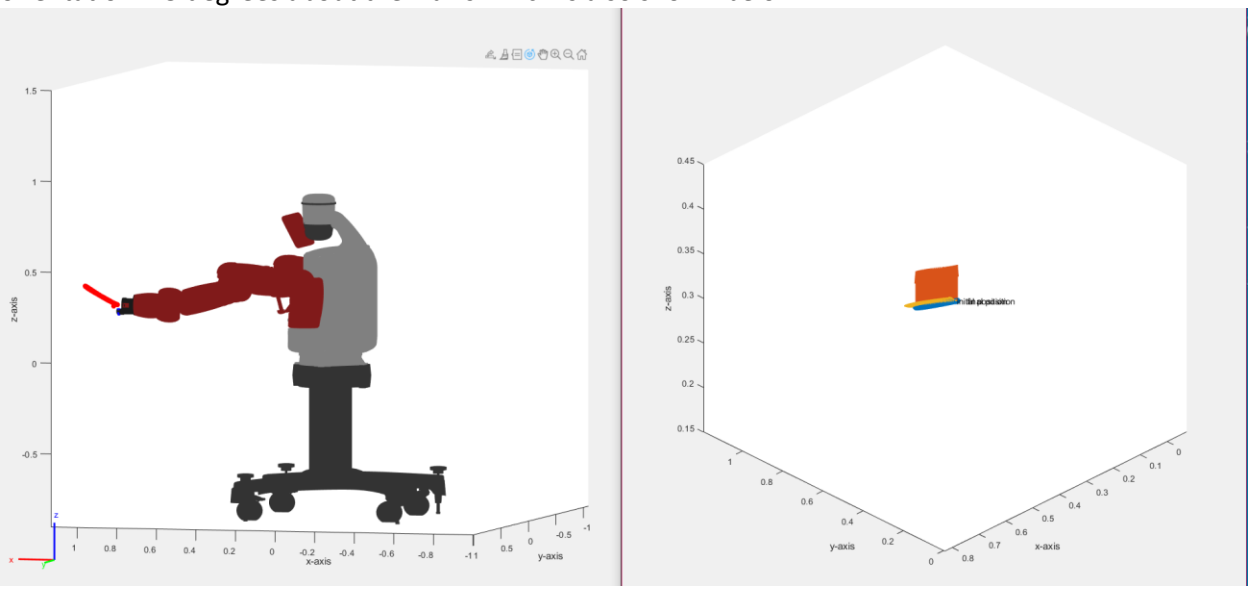


Figure. 10 Comparison between outputs of Baxter

Based on what is shown above, my output is to the right is based on the position from my joint angles and to the left is the output of the position based on my joint angles, but I can only use the Robotic Toolbox to compute the output for the arm and simulate it since the configuration that is default for Baxter cannot be modified. It can be shown that my joint angles output from my program are used.

```
for i=1:length(trajTimes)
    configNow = [0,thetaplus1(1:7,1,i)',zeros(1,7)]; %%USE MY VALUES OF THETA
    poseNow = getTransform(robot,configNow,endEffector); %%FUNCTION FROM ROBOT TOOLBOX
    show(robot,configNow,'PreservePlot',false,'Frames','off');%%PLOTS BAXTER MOVEMENT
    plot3(poseNow(1,4),poseNow(2,4),poseNow(3,4),'r.','MarkerSize',20) %%PLOTS POINT
    drawnow;
end
```

Figure. 11 Code to plot simulation of Baxter

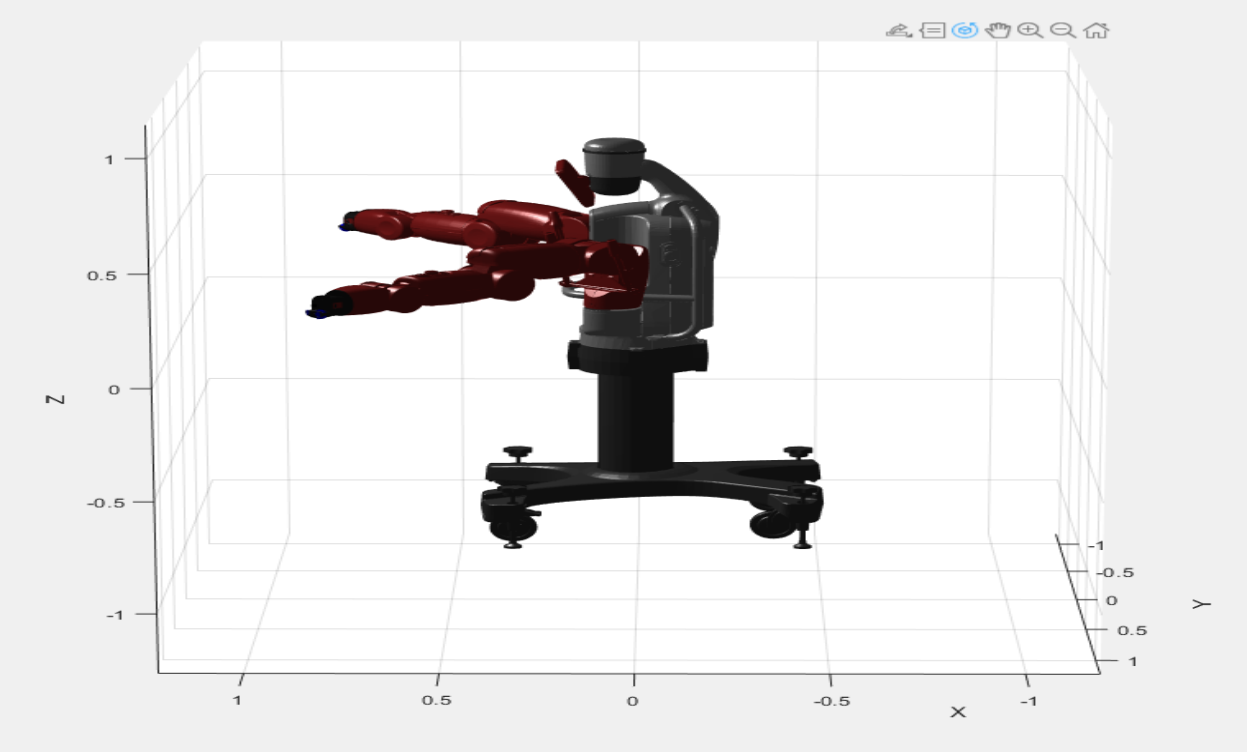


Figure. 12 Configuration of Baxter provided in Robotic System Toolbox

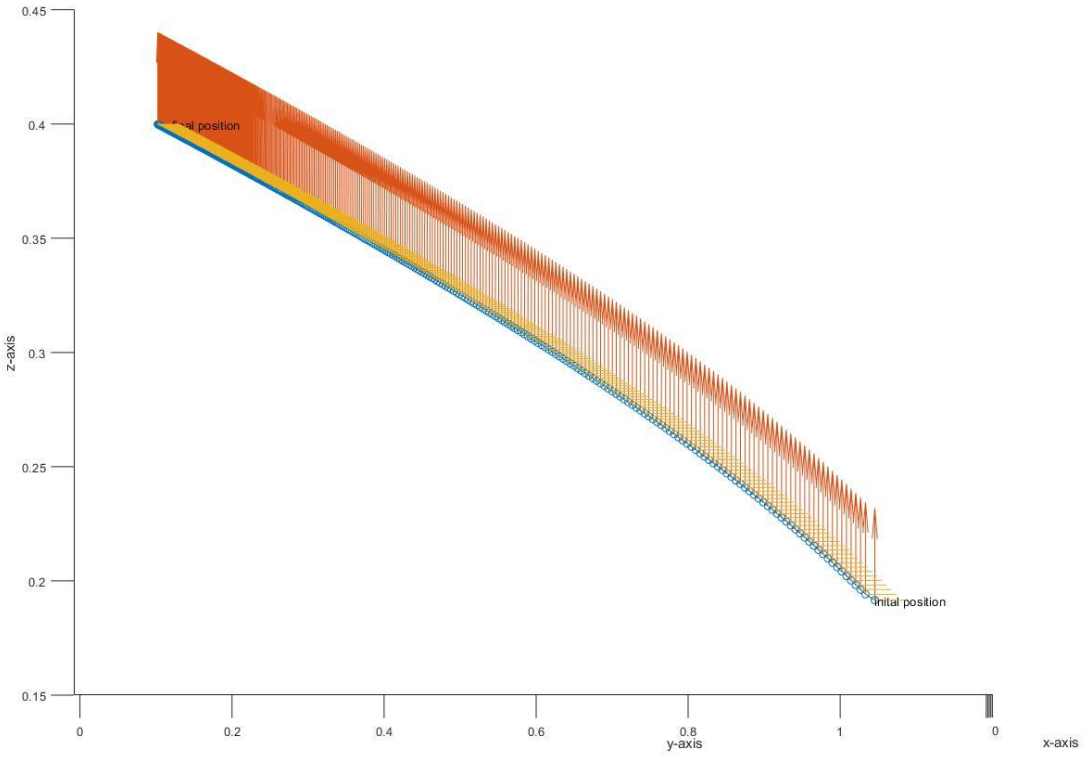


Figure. 13 My output of orientation and position of end effector in space.

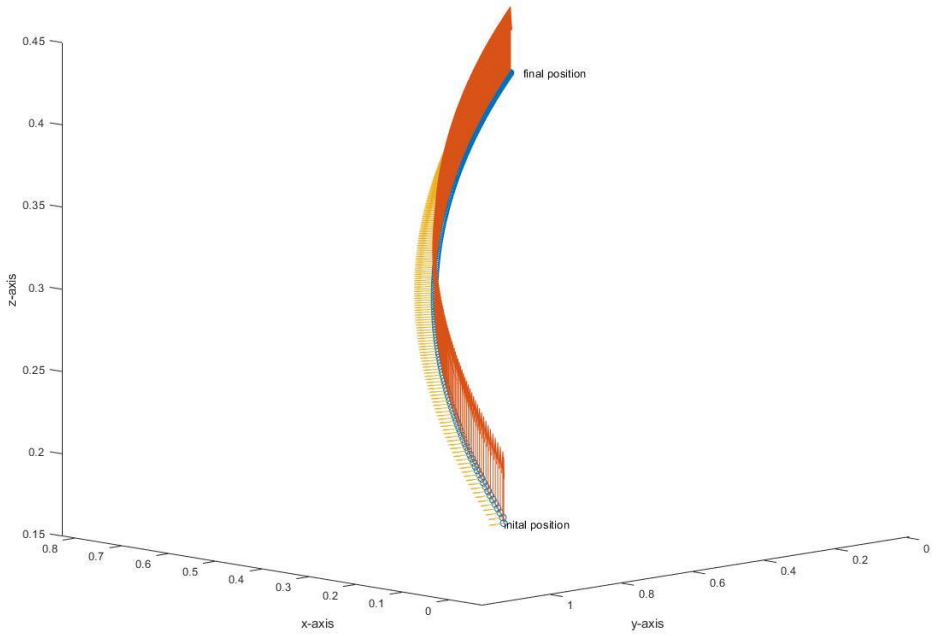


Figure. 14 My output of Orientation and Position of end effector in space.

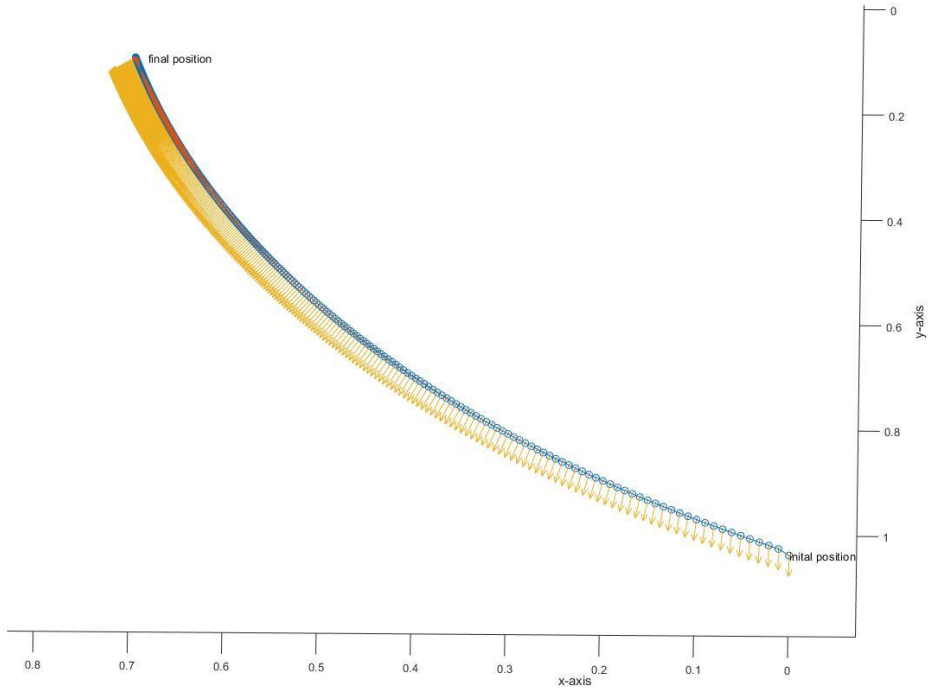


Figure. 15 My output of Orientation and Position of end effector in space

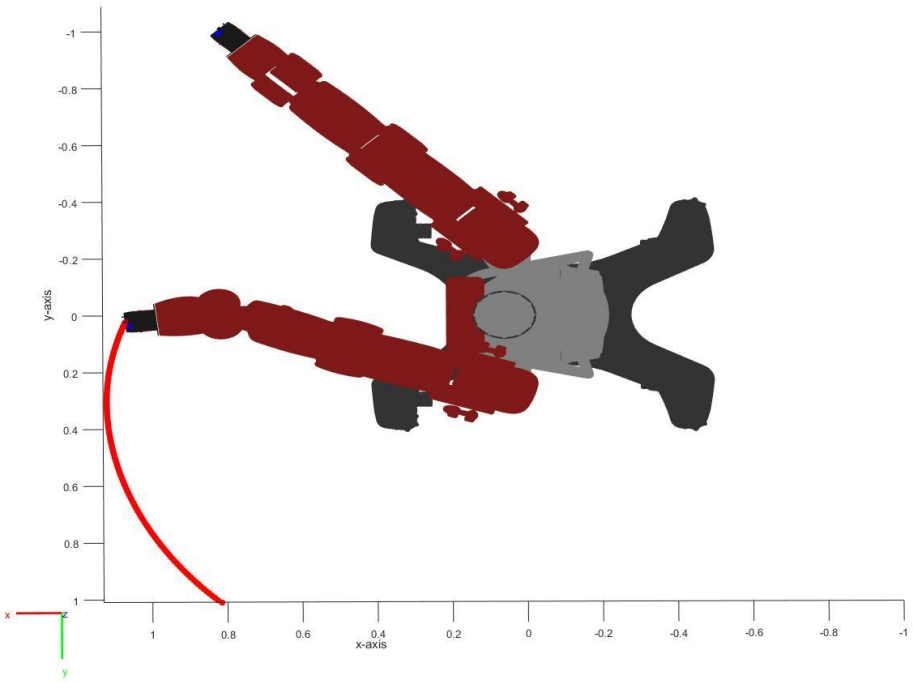


Figure. 16 Plotting Baxter's Movements based on my joint angles.

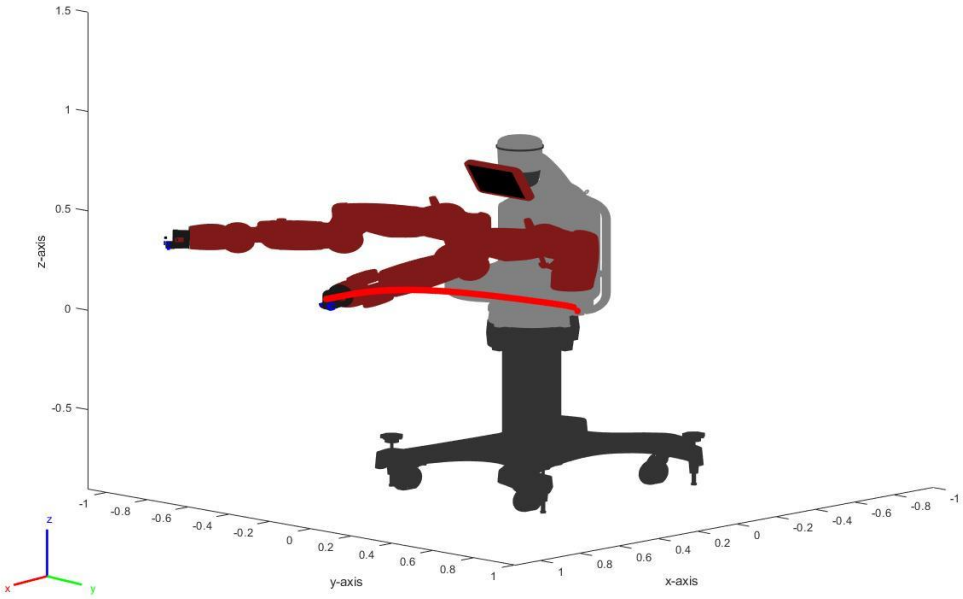


Figure. 17 Plotting Baxter's Movements based on my joint angles.

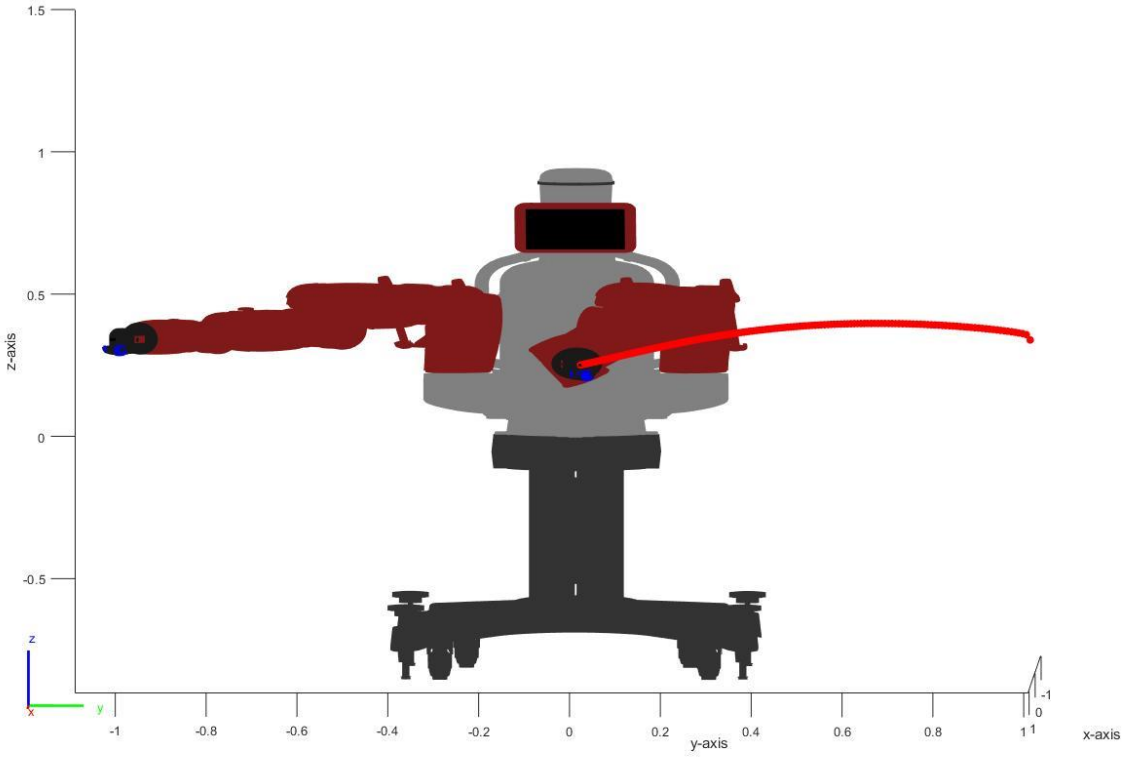


Figure. 18 Plotting Baxter's Movements based on my joint angles.

VI. Conclusion

Overall, the project is a success. The algorithm that was created successfully plots the initial and final configuration when theta does not change, the position does not change, or when everything changes as it accounts for the joint limits of the robot.

Unfortunately, the only part that was not a success was making an animation based on the joint angles I obtained to produce the exact same result from what I plotted from the forward kinematics methods derived in class. Based on my justifications made, as in the configuration the developer used and what I used differ except for the z-axis, I feel the animation would have produced the exact same result if I was able to use his configuration - rotational axis and the points on those axes.

Since the initial and desired final position were successfully obtained using everything that was taught in class, but the only issues was inputting the joint angles based on a different configuration for the simulation to produce the same result, the overall the project was a success.