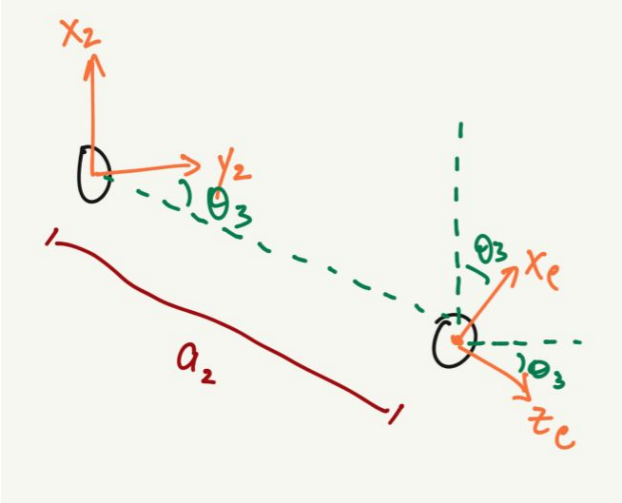
Pre-Lab

1. The equation for the red dot can be calculated as follows:

   We define the DH parameters for links 1 and 2 to be:

Link	a_i	d_i	α_i	θ_i
1	0	$-\frac{\pi}{2}$	$-d_1$	θ_1
2	$-a_2$	0	0	$\theta_2 + \frac{\pi}{2}$

   We can derive the transformation matrices for T_1^0 and T_2^1 using the standard DH matrix and substituting the DH parameters.



   Based on the figure above, the transformation matrix for link 3 is then calculated as

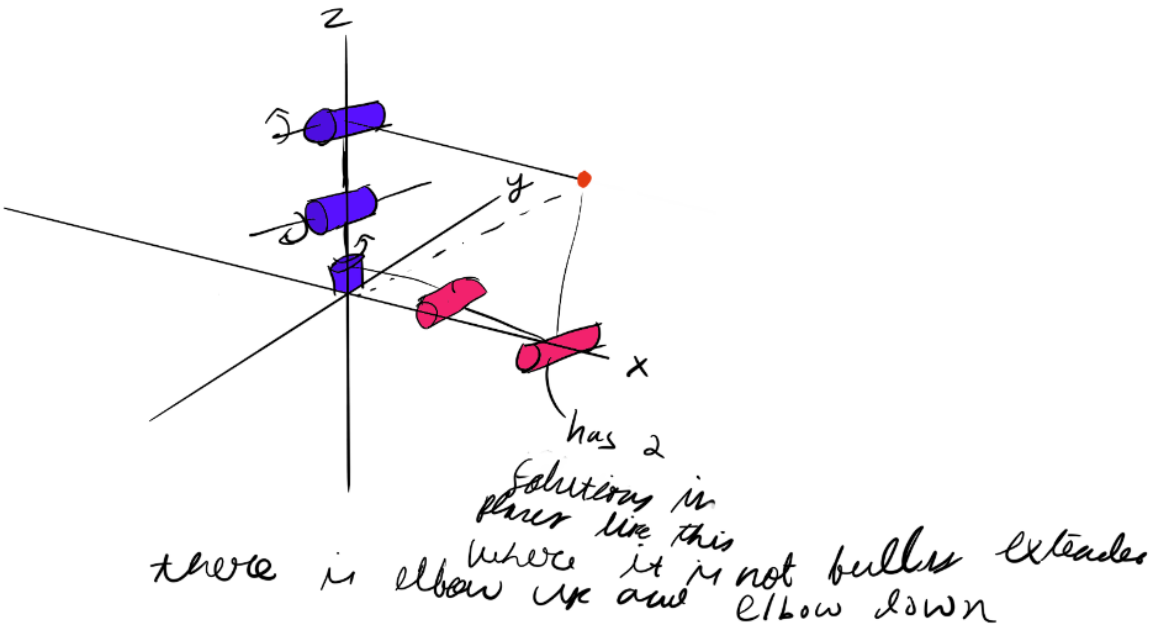
$$R_3^0 = \begin{bmatrix} c_{\theta_3} & 0 & -s_{\theta_3} & -a_2 c_{\theta_3} \\ s_{\theta_3} & 0 & c_{\theta_3} & a_2 s_{\theta_3} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

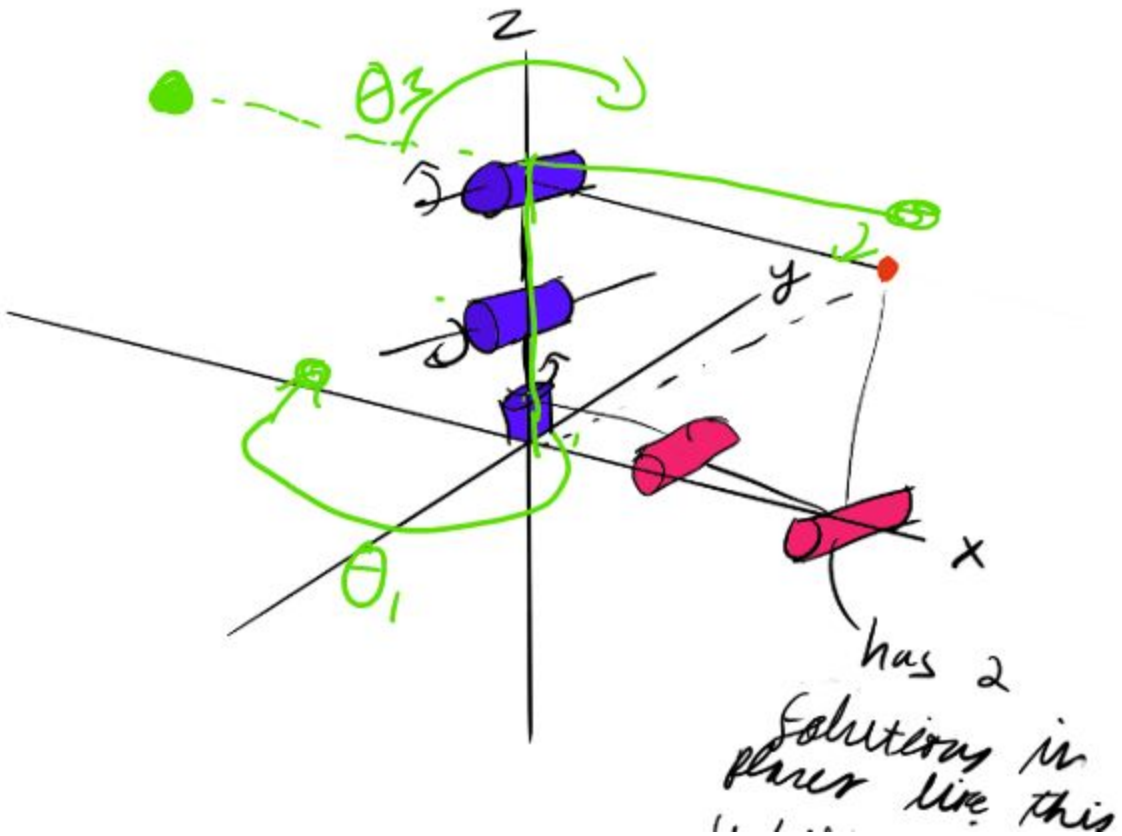
   We then calculate $T_e^0 = T_1^0 T_2^1 T_e^2$

   Extracting the position vector from T_e^0 gives:

$$d_e^0 = \begin{bmatrix} -a_2 c_{\theta_1} c_{(\theta_2+\frac{\pi}{2})} - a_3 c_{\theta_1} c_{\theta_3} c_{(\theta_2+\frac{\pi}{2})} - a_3 c_{\theta_1} s_{(\theta_2+\frac{\pi}{2})} s_{\theta_3} \\ -a_2 s_{\theta_1} c_{(\theta_2+\frac{\pi}{2})} - a_3 s_{\theta_1} c_{\theta_3} c_{(\theta_2+\frac{\pi}{2})} - a_3 s_{\theta_1} s_{(\theta_2+\frac{\pi}{2})} s_{\theta_3} \\ d_1 + a_2 s_{(\theta_2+\frac{\pi}{2})} + a_3 c_{\theta_3} s_{(\theta_2+\frac{\pi}{2})} - a_3 s_{(\theta_2+\frac{\pi}{2})} s_{\theta_3} \end{bmatrix}$$

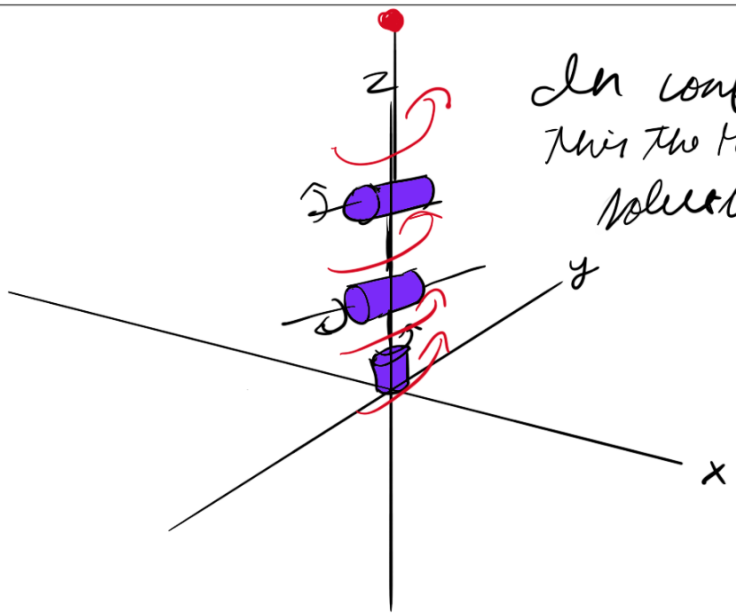
2. There are 8 solutions: two for elbow up and elbow down. For each of those configurations, you can add $\pi \; or \; -\pi$ to θ_1 and reflect the joint behind the manipulator, which creates 4 solutions.This is shown by the green section in the diagram below.



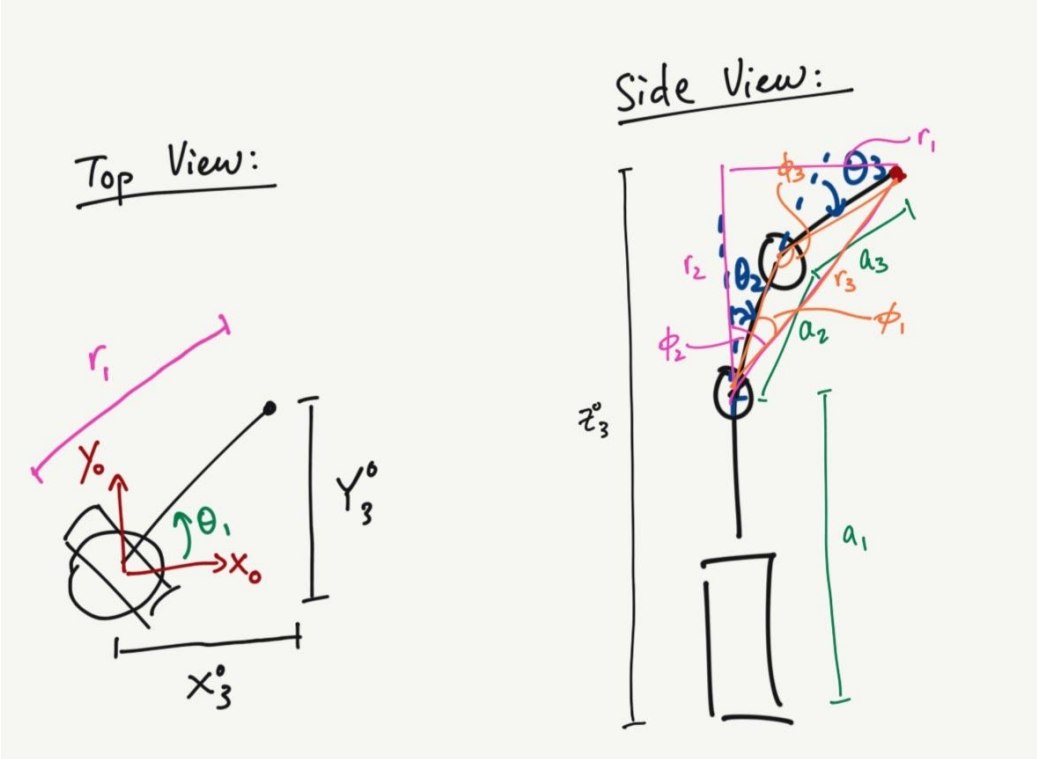


The green rotation shows the effect of rotating the manipulator behind itself into the red dot. This rotation can also be performed by adding pi to theta1.

At x=0 y=0 there are infinite solutions because theta 1 can be set to any value. This can be seen in :



3. We start by defining the diagrams for the top view and the side and using the geometric approach to solve for the joint variables (elbow up configuration).



Geometrically, we observe that:

$\theta_2 = \varphi_2 - \varphi_1$

$\theta_3 = 90 - \varphi_3$

$r_2 = z - a_1$

Using the Pythagorean Theorem:

$r_1 = \sqrt{y^2 + x^2}$

$r_3 = \sqrt{(r_1)^2 + (r_2)^2}$

Using trigonometry, observe that:

$\theta_1 = atan2\,(\frac{y}{x})$

$\varphi_2 = atan2(\frac{r_1}{r_2})$

Using the Law of Cosines, we observe that:

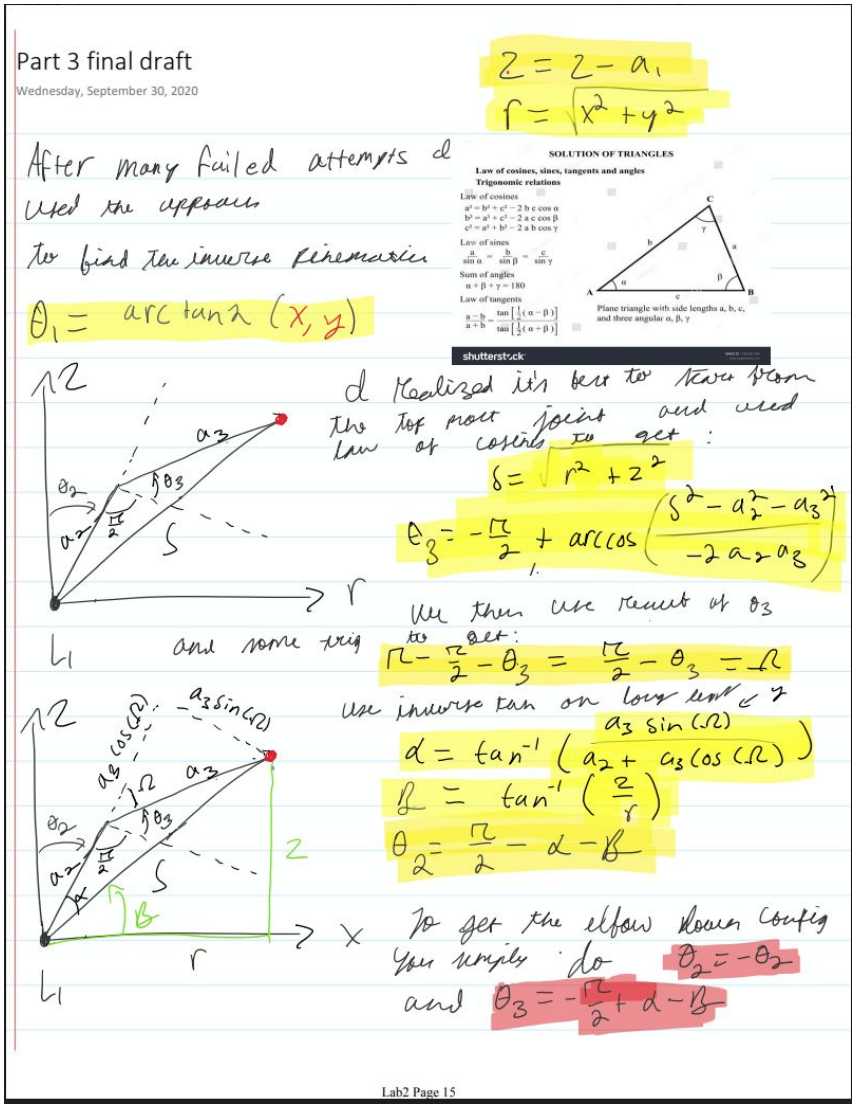
$a_3{}^2 = a_2{}^2 + r_3{}^2 - 2a_2 r_3 cos(\varphi_1) \quad \Rightarrow \quad \varphi_1 = acos(\frac{a_3{}^2 - a_2{}^2 - r_3{}^2}{-2a_2 r_3})$

$r_3{}^2 = a_2{}^2 + a_3{}^2 - 2a_2 a_3 cos(\varphi_1) \quad \Rightarrow \quad \varphi_3 = acos(\frac{r_3{}^2 - a_2{}^2 - a_3{}^2}{-2a_2 a_3})$

Plugging in values, we get:

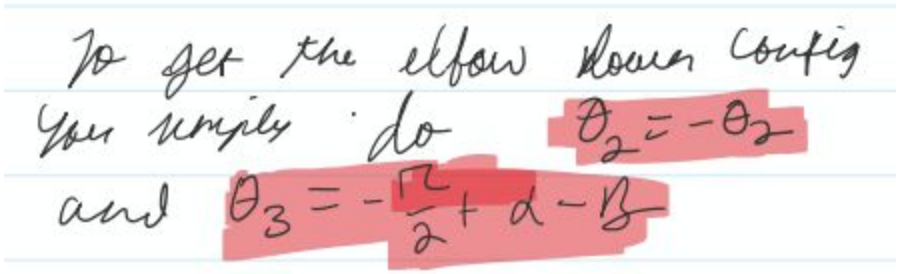
$$\theta_1 = atan2(\frac{y}{x})$$

θ_1 is valid as long as x != 0

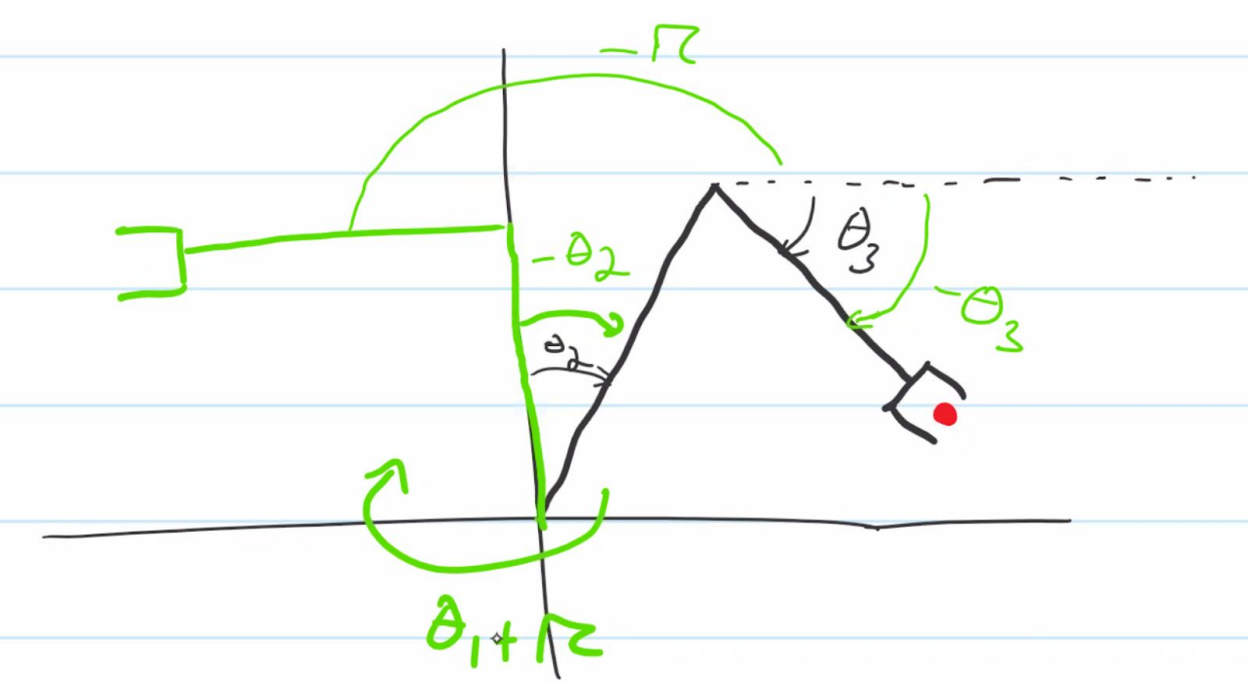


Similarly, for the elbow down configuration, the joint variables can be solved as:

$$\theta_1 = atan2\left(\frac{y_w}{x_w}\right)$$



Like Pre-Lab Q2 states, for each of these two configurations, it is possible to derive four additional solutions: you can add π to θ_1 and reflect the joint behind the manipulator.



Modifying the above equations accordingly gives for elbow up

```
theta1 = atan2( y0c,x0c)+updown*pi;
theta2=-theta2;
theta3=-pi-theta3;
```

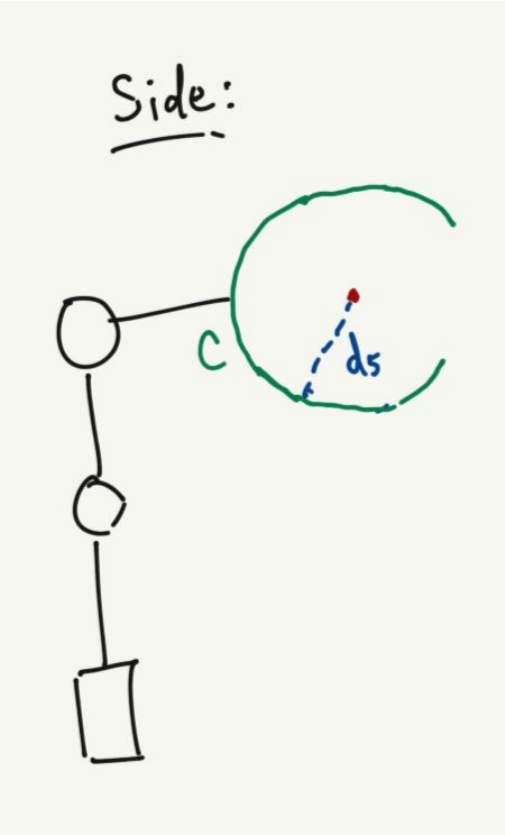
And similarly we get:

```
theta1 = atan2(y0c,x0c)+pi*updown;
theta3=-(theta3+pi);
theta2=-theta2;
```

Updown is either positive or negative one and represents either adding or subtracting pi from theta1, theta3 and theta2 are unchanged by this choice. These in combination with our previous solutions gives 8 total analytical solutions.

Methods

1. Because the robot is not able to yaw from the wrist position and is only able to pitch and roll, this means that in order for the end effector to be at a given position, the wrist can only rotate vertically around the end effector position in an arc shape with radius d_5.



2. Provided that the end effector position is not infeasible, the position of the wrist in order for the end effector frame to be at the provided T_e^0 can be found as follows:

   Given R_e^0 from T_e^0, we need to calculate the position of the wrist center o_c^0

$$o_c^o = o - d_5 \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix}$$

   Next, we use the position of the wrist to solve for the angles θ_1, θ_2, and θ_3 using the IK approach defined in Pre-lab Q3 for either the elbow up or elbow down configuration.

Next, we are able to plug in the solved joint variables θ_1, θ_2, and θ_3 into the R_1^0, R_2^1, and R_3^2 matrices that were derived in Lab 1 in order to get R_c^0. These matrices are listed below for reference.

$$R_1^0 = \begin{bmatrix} c_{\theta_1} & 0 & -s_{\theta_1} \\ s_{\theta_1} & 0 & c_{\theta_1} \\ 0 & -1 & 0 \end{bmatrix}$$

$$R_2^1 = \begin{bmatrix} -s_{\theta_2} & -c_{\theta_2} & 0 \\ c_{\theta_2} & -s_{\theta_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_3^2 = \begin{bmatrix} s_{\theta_3} & c_{\theta_3} & 0 \\ -c_{\theta_3} & s_{\theta_3} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_3^0 = \begin{bmatrix} -s_{\theta_2}s_{\theta_3}c_{\theta_1} + c_{\theta_2}c_{\theta_3}c_{\theta_1} & -s_{\theta_2}c_{\theta_3}c_{\theta_1} - c_{\theta_2}s_{\theta_3}c_{\theta_1} & -s_{\theta_1} \\ -s_{\theta_1}s_{\theta_2}s_{\theta_3} + s_{\theta_1}c_{\theta_2}c_{\theta_3} & -s_{\theta_1}s_{\theta_2}c_{\theta_3} - s_{\theta_1}s_{\theta_3}c_{\theta_2} & c_{\theta_1} \\ -s_{\theta_2}c_{\theta_3} - s_{\theta_3}c_{\theta_2} & s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3} & 0 \end{bmatrix}$$

From here, we can solve for $R_e^{3^*}$ with the following equation:

$$R_e^{3^*} = (R_3^0)^{-1} R_e^0$$

Next, we need to derive the R_e^3 matrix using DH convention. Using the DH parameters outlined below, we can see that R_4^3 and R_e^4 are as follows:

Link	a_i	d_i	α_i	θ_i
4	0	0	$-\frac{\pi}{2}$	$\theta_4 - \frac{\pi}{2}$
5	0	d_5	0	θ_5

$$R_4^3 = \begin{bmatrix} c_{\theta_{(4-\frac{\pi}{2})}} & 0 & -s_{\theta_{(4-\frac{\pi}{2})}} \\ s_{\theta_{(4-\frac{\pi}{2})}} & 0 & c_{\theta_{(4-\frac{\pi}{2})}} \\ 0 & -1 & 0 \end{bmatrix}$$

$$R_e^4 = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5} & 0 \\ s_{\theta_5} & c_{\theta_5} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This gives $$R_e^3 = \begin{bmatrix} c_{\theta_{(4-\frac{\pi}{2})}} c_{\theta_5} & -c_{\theta_{(4-\frac{\pi}{2})}} s_{\theta_5} & -s_{\theta_{(4-\frac{\pi}{2})}} \\ s_{\theta_{(4-\frac{\pi}{2})}} c_{\theta_5} & -s_{\theta_{(4-\frac{\pi}{2})}} s_{\theta_5} & c_{\theta_{(4-\frac{\pi}{2})}} \\ -s_{\theta_5} & -c_{\theta_5} & 0 \end{bmatrix}$$

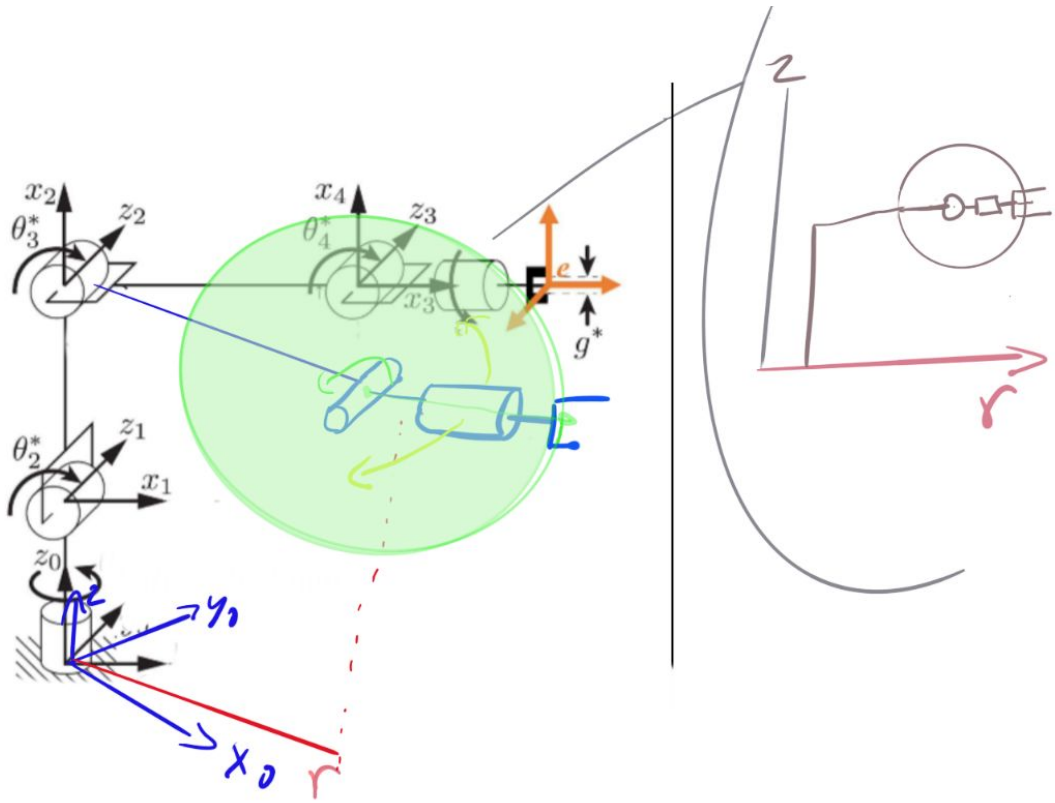
Finally, we recognize that this matrix can be also seen as Euler angles, so we solve for θ_4 and θ_5 by setting the values that we get for $R_e^{3^*}$ equal to R_e^3 .

$$\theta_4 = atan2(\frac{r_{e,23}^3}{r_{e,13}^3})$$

$$\theta_5 = atan2(\frac{-r_{e,31}^3}{-r_{e,32}^3})$$

Lastly, we plug all joint variables back into the derived transformation matrices and post multiply to check that the inputted T_e^0 is the same as the T_e^0 calculated using IK.

3. The orientations that the Lynx end effector is not able to reach are those that require the wrist to yaw, or rotate about the z axis in frame 3. This means that any rotation of the wrist about the z axis in frame 3 should be equal to 0, which mean $r_{e,33}^3$ should be equal to 0. If this is not equal to 0, then we know that the orientation will be infeasible for the Lynx. The test for whether an orientation is feasible is dotting the desired z end effector with the z0xr where r is the projection of z desired onto the xy plane.



Here the yellow lines represent the infeasible yaw rotation of the robot.

4. Given the desired end effector position in the base frame, we first project the position of the end effector onto the x_e^0 , y_e^0 plane and get the vector for the arm's radius $\hat{r} = [x_e^0 , y_e^0 , 0]$. Next, we find the plane of feasible end effector positions (from now on referred to as the "no-yaw plane" by computing its normal vector $\hat{n}$ as

$$\hat{n} = \widehat{z_0} \times \hat{r}$$

Then we project the x, y, and z components of R_e^0 onto the no-yaw plane described by $\hat{n}$

$$x_p = proj_{\vec{n}}(\vec{x_e^0}) = \vec{x_e^0} - \frac{\vec{x_e^0} \cdot \vec{n}}{(\|\vec{n}\|)^2}\vec{n}$$

The projections of y and z are done in the same fashion, resulting in the projections being non-unit vectors. We want to see which vector was at a greater angle relative to the no-yaw plane:

$$y_{p,loss} = |(\|y_e^0\|) - (\|y_p\|)|$$
$$x_{p,loss} = |(\|x_e^0\|) - (\|x_p\|)|$$

Next, we find the vector normal to the no-yaw plane by taking the cross product of z_0 and z_p. Comparing the $x_{p,loss}$ and $y_{p,loss}$ values, the corresponding x_p or y_p vector with smaller projection loss is used to cross product with z_p to determine the vector perpendicular to z_p in the no-yaw plane. After normalizing the new vectors, we combine them to get R_e^0.

From here, we calculate the final transformation matrix T_e^0 by following the steps outlined in question 2 with the R_e^0 that we just found.

Evaluation

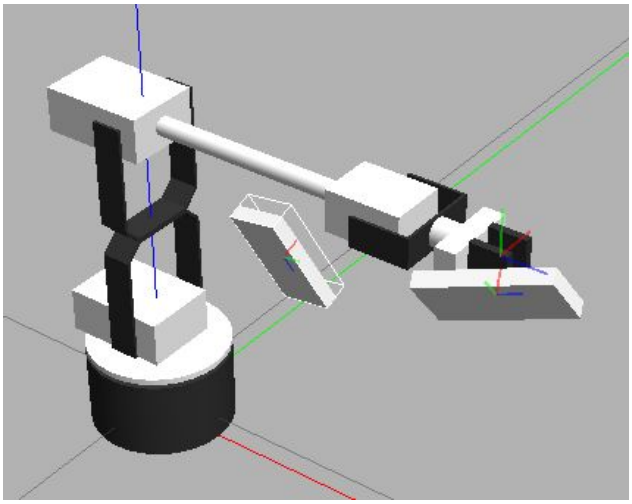
The experimental setup for this lab included a ROS simulation of the Lynx robot. Commands were sent to an Ubuntu virtual machine from MATLAB on the host machine.

The inputs for the experimental setup are a variety of test transformation matrices which align with a set of four targets in the ROS environment. Upon executing each of the test transformation matrices, the Lynx was intended to collide with targets. If the robot arm was able to match the position and orientation of the targets

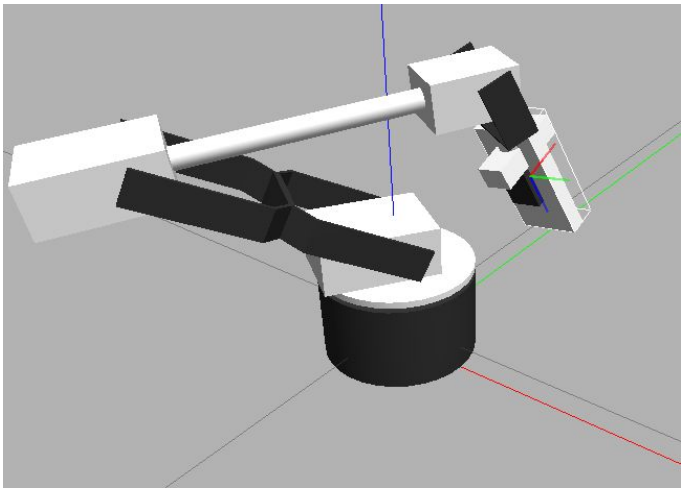
The following displays the results of the inputted target configurations. We also used a preliminary test file that simple acquired a value toe using forward kinematics and then we plugged in this valid Toe into inverse kinematics and used the outputted q to input back into forward kinematics and we compared the two T_e^0's to see if they matched. A match corresponds to the code working correctly.

Zero configuration

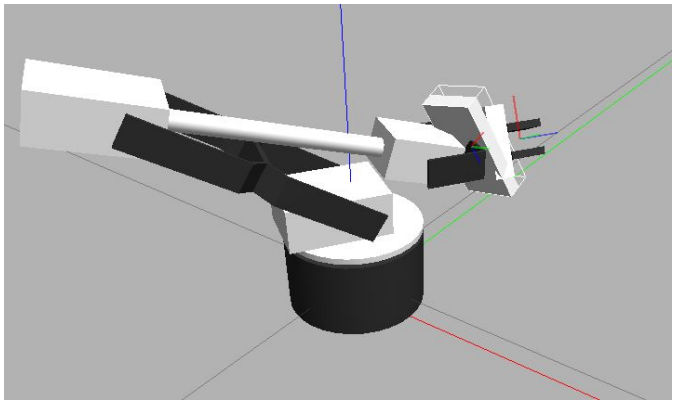
As a benchmark configuration, we computed the results for the



Target 1 part 1 and 2



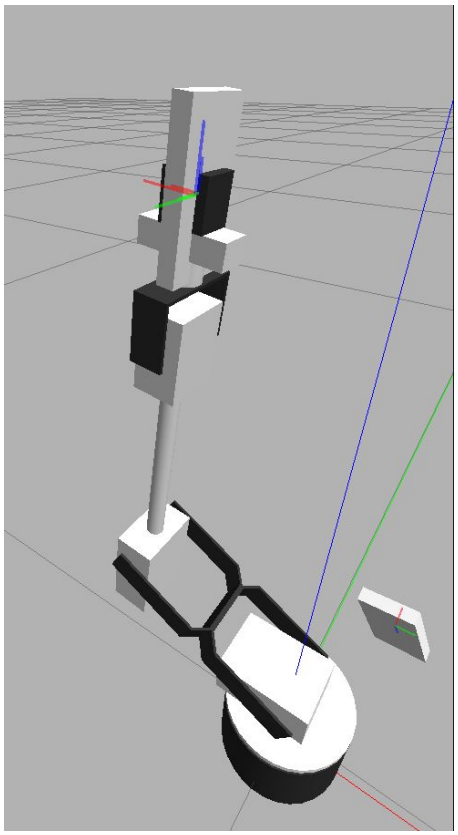
```
Simulation T0e =
    0.0189    0.9692    0.2456   46.9008
    0.9169   -0.1148    0.3823   72.9965
    0.3987    0.2180   -0.8908  100.0097
         0         0         0    1.0000

Target T0e =
    0.0190    0.9690    0.2450   47.0460
    0.9170   -0.1150    0.3820   73.2690
    0.3980    0.2170   -0.8910  100.5470
         0         0         0    1.0000

isPos =
     1

q =
    0.9997   -1.1002    1.0001    1.1998   -0.5003
```



```
Finish the command!
Simulation T0e =
    0.0031    0.8413    0.5405   67.1433
    0.4797    0.4731   -0.7390  104.5020
   -0.8774    0.2616   -0.4021  100.3379
         0         0         0    1.0000

Target T0e =
    0.0190    0.9690    0.2450   47.0460
    0.9170   -0.1150    0.3820   73.2690
    0.3980    0.2170         0  100.5470
         0         0         0    1.0000

isPos =
     0

q =
    0.9997   -1.0022    1.2963   -0.2940   -0.5003
```

Target 2

- Target 2:

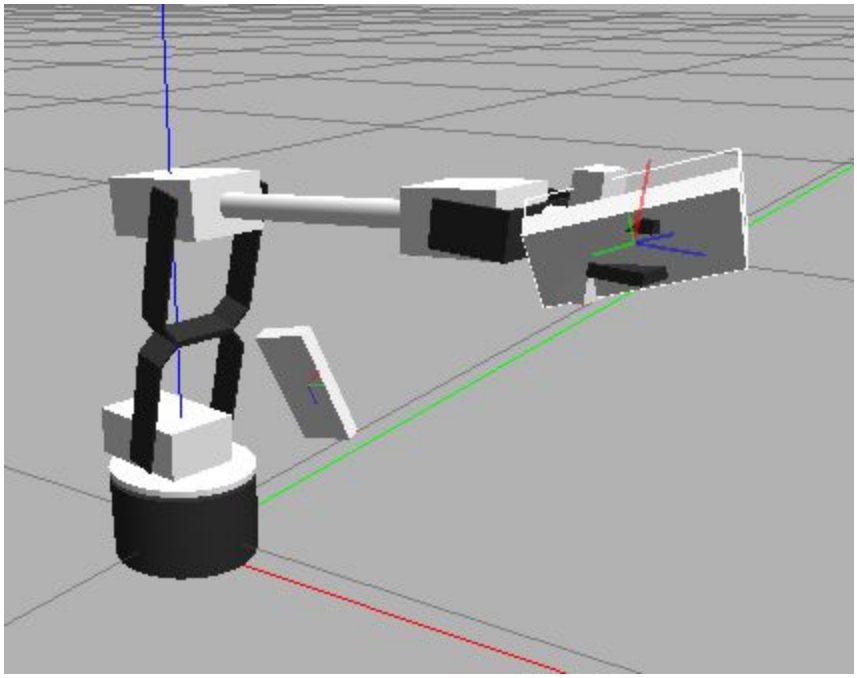


```
Simulation T0e =
   -0.9938   -0.0000    0.1113  -98.8664
    0.0000   -1.0000   -0.0000    0.0001
    0.1113    0.0000    0.9938  401.4872
         0         0         0    1.0000

Target T0e =
   -0.9930         0    0.1190  -96.9360
         0   -1.0000         0         0
    0.1190         0    0.9930  401.2290
         0         0         0    1.0000

isPos =
     1

q =
         0   -1.0591   -0.3923   -0.0001   -0.0000
```

No solution for second config

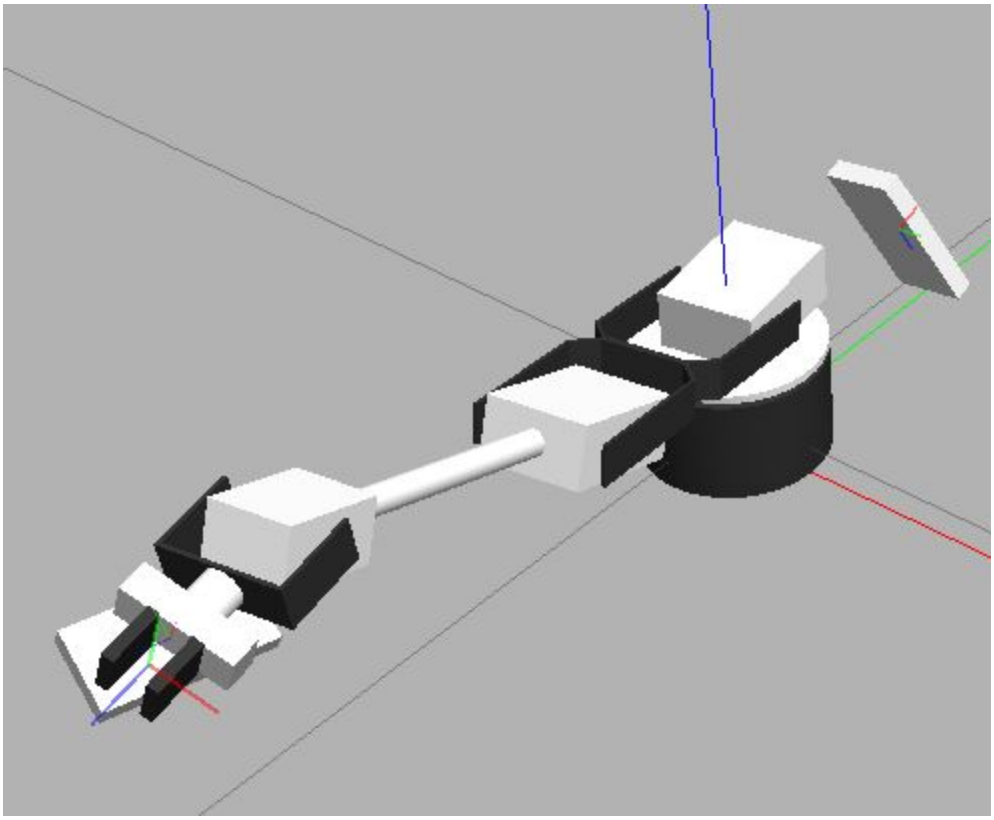
```
Target T0e =
   -0.9930         0    0.1190  -96.9360
         0   -1.0000         0         0
    0.1190         0         0  401.2290
         0         0         0    1.0000

isPos =
     0
```

Target 3:



```
Simulation T0e =
    0.0447   -0.1682    0.9847  290.0037
   -0.0076   -0.9858   -0.1680  -49.4858
    0.9990    0.0000   -0.0453  221.8053
         0         0         0    1.0000

Target T0e =
   -0.3409   -0.1075    0.9339  282.9600
    0.7843   -0.5803    0.2195  -48.3020
    0.5184    0.8073         0  235.0710
         0         0         0    1.0000

isPos =
     0

q =
   -0.1691    0.2241   -0.3121    0.0880   -0.0000
```

- Target 4: if we force the end effector to go to the closest config we get:



  - Simulation T0e =
  - 0.0959 -0.9809 0.1692 65.3084
  - 0.9051 0.0152 -0.4249 -378.5581
  - 0.4142 0.1939 0.8893 167.4490
  - 0 0 0 1.0000
  -
  - Target T0e =
  - 0.5054 -0.8371 -0.2095 -45.0000
  - -0.0306 0.2253 -0.9738 -300.0000
  - 0.8623 0.4986 0.0883 63.0000
  - 0 0 0 1.0000
  - q = -1.4000 1.4000 -1.8000 0.4960 -2.0000 10.0000

But this point is actually outside the reachable workspace so we get:

```
    0.5054   -0.8371   -0.2095  -45.0000
   -0.0306    0.2253   -0.9738 -300.0000
    0.8623    0.4986    0.0883   63.0000
         0         0         0    1.0000

Target T0e =
    0.5054   -0.8371   -0.2095  -45.0000
   -0.0306    0.2253   -0.9738 -300.0000
    0.8623    0.4986    0.0883   63.0000
         0         0         0    1.0000

isPos =
     0
```

Analysis

1. We were unable to fix a great deal of the bugs in our program so we are unable to say for sure what the true differences between ik and resulting sims were. We will try to resubmit this is as far as we were able to get in the allotted time. After two days we have been able to work out the majority of the bugs and the code is working satisfactorily. We found that our IK function agreed nicely with our simulation with an error of about 1*10^(-4) between the two. This was found by analysing the dot product of known feasible orientations desired z with the yaw component.
2. After determining whether the joint configuration was feasible we chose the joint that was closest to the base configuration to minimize the movement of the joints. This was done by taking the L_2 norm of the the q of each possible analytical solution and picking the one with the smallest norm. We would change how we defined the norm in reality. We might want to define the norm to minimize stress on the robot by choosing paths that require little energy. We might also consider hazard of an orientation by potential collisions this would also be factored into the norm in some way.
3. *Explain modifications to FK/IK eqns to improve accuracy of results to account for real physics (gravity, torque limits, friction, etc)*
    a. *what are major factors that affect accuracy.*

        One factor was how much yaw a desired target had. More yaw resulted in a greater discrepancy between the reached target and the desired one. We would ideally perform a single value decomposition to turn 3 degree rotations into 2 degree rotations/

    b. *what is impact of factors on joint variables qi needed to reach a position and orientation*

        This is determined by whether the end effector is within the reachable workspace. And it is determined by if the orientation is feasible.

c. *how much data or what additional info do you need to implement proposed approach?*

   We would most likely need to implement a rotary encoder into each of the joints to establish a high level of rotational sensing accuracy. Additionally, we would need to be able to implement the Single Value Decomposition in order to get the closest feasible orientation and position.

Conclusions

In reality, the Lynx robot will have overshoot and shocks whenever an impulse is applied to the motors. This will impact the simulation as ROS does not account for this to a large degree.

Inverse kinematics is an extremely complex topic. The crucial point is deciding if a solution exists and from there we can determine what solution to use. Currently there is no approximation for positions outside the reachable workspace and that would be a desirable addition to the code.

How the Concepts Are Incorporated Into Our Code

- Lines 11-37 in getO.m are used to calculate the closest feasible position and orientation and project the desired zaxis onto the feasible workspace of the end effector provided that an infeasible orientation or position are given.
- Lines 45-47 in getO.m are used to calculate the wrist's joint variable values θ_4 and θ_5 using the derived Euler angle matrix R_e^4
- The function GetxC.m is used to calculate the position of the wrist center when given a transformation matrix T_e^0.
- The function getP.m is used to calculate the values of joint variables 0-3 when given the wrist center position.
- The function calculateFK_R03.m is used to calculate the rotation matrix R_3^0 given the set of joint variable angles $\theta_1 - \theta_3$. This is needed in order to calculate R_e^3 which is used to calculate the wrist's joint variables θ_4 and θ_5.
- The function project.m is used to calculate the projection of a vector u onto a plane described by normal vector n.
- The function tester is used to test that Calculate IK is working as expected without providing joint limits.