

Udacity Robotics Software Engineer

Project 2

Robotic Arm – Pick and Place

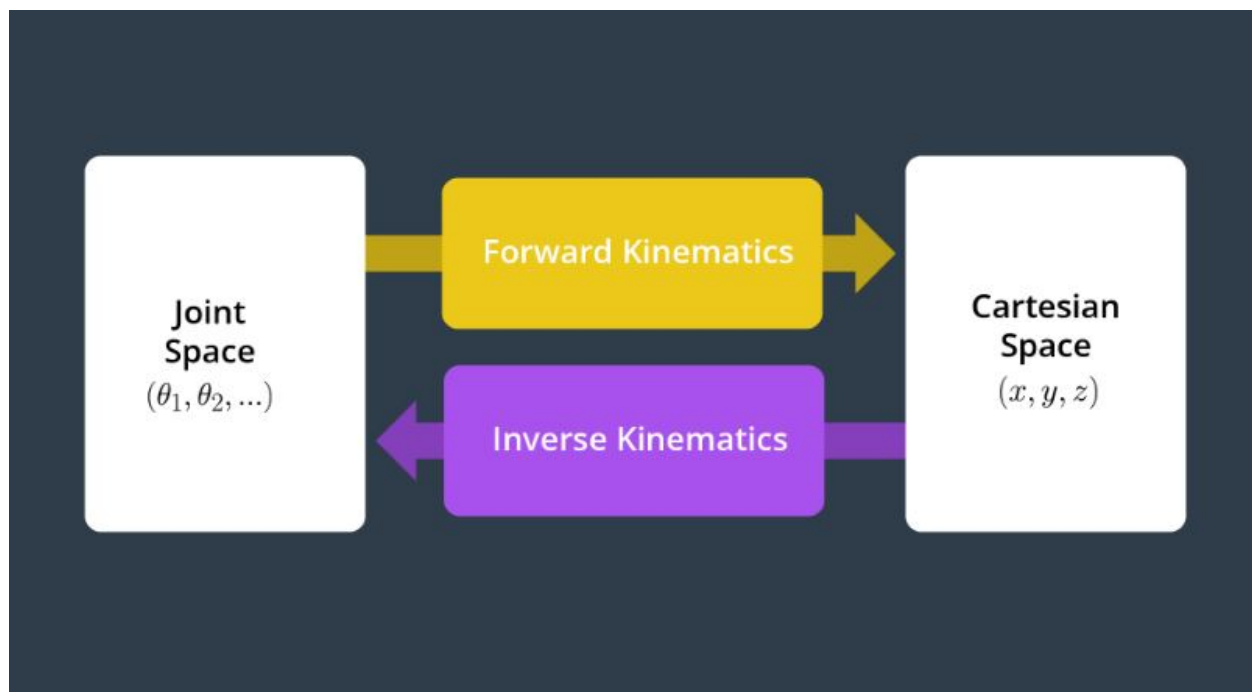
By Leroy Friesenhahn



This was the most difficult project I have worked on in a long time. Understand deriving DH parameter table was my greatest challenge. I realize I have a long way to go to understand all there is to know about Forward Kinematics and Inverse Kinematics.

Kinematics

According to Wikipedia, Kinematics is a branch of classical mechanics that describes the motion of points, bodies and system of bodies without considering the mass of each of the forces that caused the motion. In this project, we will use Forward and Inverse Kinematics to solve the pick and place of the Kuba KR210 robot.



Forward Kinematics uses joint variable to determine the position and orientation of the KR210 gripper arm in 3D space (Cartesian Space).

The process uses composition of homogeneous transforms to solve this problem. Starting at the base link and moving link by link to the end effector. Denavit-Hartenberg (DH) convention is used in solving kinematic analysis for the KR210 robot. Using four parameter instead of twelve parameters help to simplify the analysis process. The DH parameters are used to build each transform

$${}^0_N T = {}^0_1 T {}^1_2 T {}^2_3 T \dots {}^{N-1}_N T$$

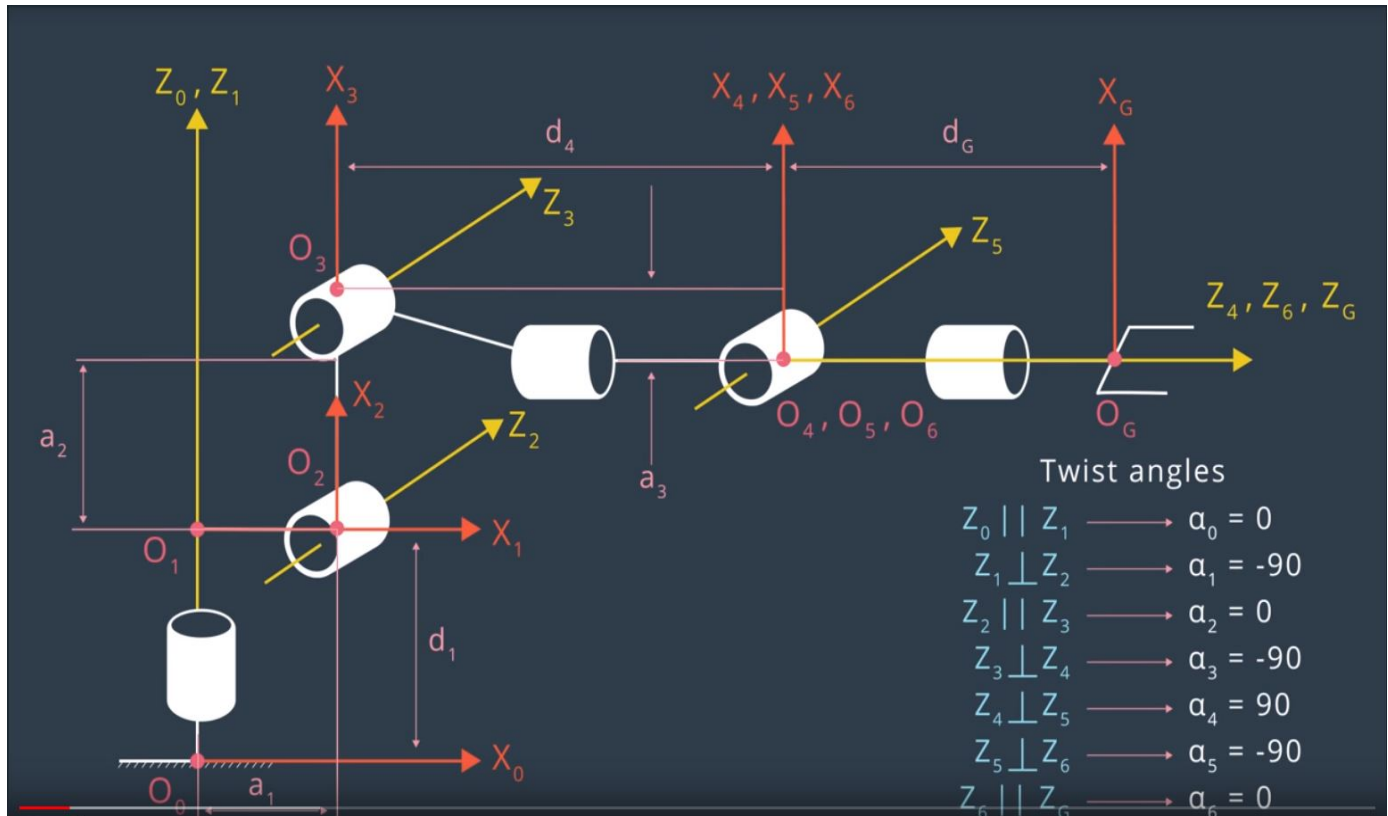
Total transform between adjacent links

$${}^{i-1}_i T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Each link consist of four individual transforms: 2 rotations and 2 translation, performed in the order shown below.

$${}^{i-1}_i T = R_X(\alpha_{i-1}) D_X(a_{i-1}) R_Z(\theta_i) D_Z(d_i)$$

The analysis begins by drawing the diagram for the KR210 robot. Assigning the Z axis, X axis and gives us the necessary information to derive the values needed for the DH parameter table.



From the above diagram the following DH parameter table is derived.

Where :

Alpha = the robot arm twist angle

a = arm link length

d – arm link offset

q – arm joint angle

Note: Do to lack of subscripts usage in the creation of this document, the following ledger applies

Alpha(i-1) – alpha value for link I -1

$a(i-1)$ – a value for link $i-1$

$d(i)$ – d value for link i

$q(1)$ - theta value for designated link

DH Parameter Table

Link	Alpha (i-1)	a (i-1)	d(i)	Theta
0 to 1	0	0	d1	$q(1)$
1 to 2	-90	a1	0	$q(2) - 90$
2 to 3	0	a2	0	$q(3)$
3 to 4	-90	a3	d4	$q(4)$
4 to 5	90	0	0	$q(5)$
5 to 6	-90	0	0	$q(6)$
6 to G	0	0	d7	$q(7)$

Note: 7 is the Gripper

Using the Rviz Display for each of the links the following variables values can be obtained:

$d1 = \text{link2}[z] = 0.75$

$a1 = \text{link2}[x] = 0.35$

$a2 = \text{link3}[z] = 1.25$

$$a3 = \text{link3}[z] - \text{link5}[z] = 1.9464 - 2 = -0.054$$

$$d4 = \text{link5}[x] - \text{link3}[x] = 1.85 - 0.35001 = 1.5$$

$$\text{Gripper: } d7 = \text{link_gripper}[x] - \text{link5}[x] = 2.153 - 1.85 = 0.303$$

See view below:

Displays	
Update Interval	0
Frame Timeout	15
▼ Frames	
All Enabled	<input checked="" type="checkbox"/>
▶ base_footprint	<input checked="" type="checkbox"/>
▶ base_link	<input checked="" type="checkbox"/>
▼ gripper_link	<input checked="" type="checkbox"/>
Parent	link_6
▶ Position	2.153; -3.2702e-07; 1.946
▶ Orientation	7.574e-08; 3.0915e-06; -7.5945e-08; 1
▶ Relative Position	0.11; 0; 0
▶ Relative Orientation	0; 0; 0; 1
▶ left_gripper_finger_link	<input checked="" type="checkbox"/>
▶ link_1	<input checked="" type="checkbox"/>
▼ link_2	<input checked="" type="checkbox"/>
Parent	link_1
▶ Position	0.35; -5.3162e-08; 0.75
▶ Orientation	1.6499e-13; 2.1725e-06; -7.5945e-08; 1
▶ Relative Position	0.35; 0; 0.42
▶ Relative Orientation	0; 2.1725e-06; 0; 1
▼ link_3	<input checked="" type="checkbox"/>
Parent	link_2
▶ Position	0.35001; -5.3162e-08; 2
▶ Orientation	2.265e-13; 2.9824e-06; -7.5945e-08; 1
▶ Relative Position	0; 0; 1.25
▶ Relative Orientation	0; 8.0998e-07; 0; 1
▼ link_4	<input checked="" type="checkbox"/>
Parent	link_3
▶ Position	1.31; -1.9898e-07; 1.946
▶ Orientation	2.1822e-08; 2.9824e-06; -7.5945e-08; 1
▶ Relative Position	0.96; 0; -0.054
▶ Relative Orientation	2.1822e-08; 0; 0; 1
▼ link_5	<input checked="" type="checkbox"/>
Parent	link_4
▶ Position	1.85; -2.81e-07; 1.946
▶ Orientation	2.1822e-08; 3.0915e-06; -7.5945e-08; 1
▶ Relative Position	0.54; 0; 0
▶ Relative Orientation	0; 1.0905e-07; 0; 1
▶ link_6	<input checked="" type="checkbox"/>
▶ right_gripper_finger_link	<input checked="" type="checkbox"/>
▶ world	<input checked="" type="checkbox"/>
▼ Tree	
▼ world	
▶ base_footprint	

Using Python code, create the transformation matrix:

```
# Creates Homogeneous Transform Matrix
```

```
# from DH parameters
```

```
def TF_Matrix(q, d, a, alpha):
```

```
    T = Matrix([[cos(q),          -sin(q),          0,          a          ],
                [ sin(q)*cos(alpha), cos(q)*cos(alpha), -sin(alpha), -sin(alpha)*d],
                [ sin(q)*sin(alpha), cos(q)*sin(alpha),  cos(alpha),  cos(alpha)*d],
                [          0,          0,          0,          1]])

    return T
```

The homogeneous_transform is call for each individual joint to create each transform

```
# Generate each individual transformation matrices
```

```
# Transform from joint 0 to 1
```

```
T0_1 = TF_Matrix(alpha0,a0,d1,q1).subs(DH)
```

```
# Transform from joint 1 to 2
```

```
T1_2 = TF_Matrix(alpha1, a1, d2, q2).subs(DH)
```

```
# Transform from joint 2 to 3
```

```
T2_3 = TF_Matrix(alpha2, a2,d3,q3).subs(DH)
```

```
# Transform from joint 3 to 4
```

```
T3_4 = TF_Matrix(alpha3,a3,d4,q4).subs(DH)
```

Transform from joint 4 to 5

$T_{4_5} = \text{TF_Matrix}(\alpha_4, a_4, d_5, q_5).subs(DH)$

Transform from Joint 5 to 6

$T_{5_6} = \text{TF_Matrix}(\alpha_5, a_5, d_6, q_6).subs(DH)$

Transform from joint 6 to Gripper

$T_{6_G} = \text{TF_Matrix}(\alpha_6, a_6, d_7, q_7).subs(DH)$

Setting up the Transformation Matrix using the DH parameters yields the following transforms:

$${}^0T^1 = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1T^2 = \begin{bmatrix} \sin(q_2) & \cos(q_2) & 0 & 0.35 \\ 0 & 0 & 1 & 0 \\ \cos(q_2) & -\sin(q_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2T^3 = \begin{bmatrix} \cos(q_3) & -\sin(q_3) & 0 & 1.25 \\ \sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^3T^4 = \begin{bmatrix} \cos(q_4) & -\sin(q_4) & 0 & -0.54 \\ 0 & 0 & 1 & 1.5 \\ -\sin(q_4) & -\cos(q_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^4T^5 = \begin{bmatrix} \cos(q_5) & -\sin(q_5) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(q_5) & \cos(q_5) & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$${}^5T^6 = \begin{bmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin(q_6) & -\cos(q_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^6T^6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.303 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This information give us all we need to calculate the transform from 0 to the end effector using the following:

$${}^0T_N = {}^0T_1 {}^1T_2 {}^2T_3 \dots {}^{N-1}T_N$$

Final Forward Kinematics is written as:

$$T0_2 = T0_1 * T1_2$$

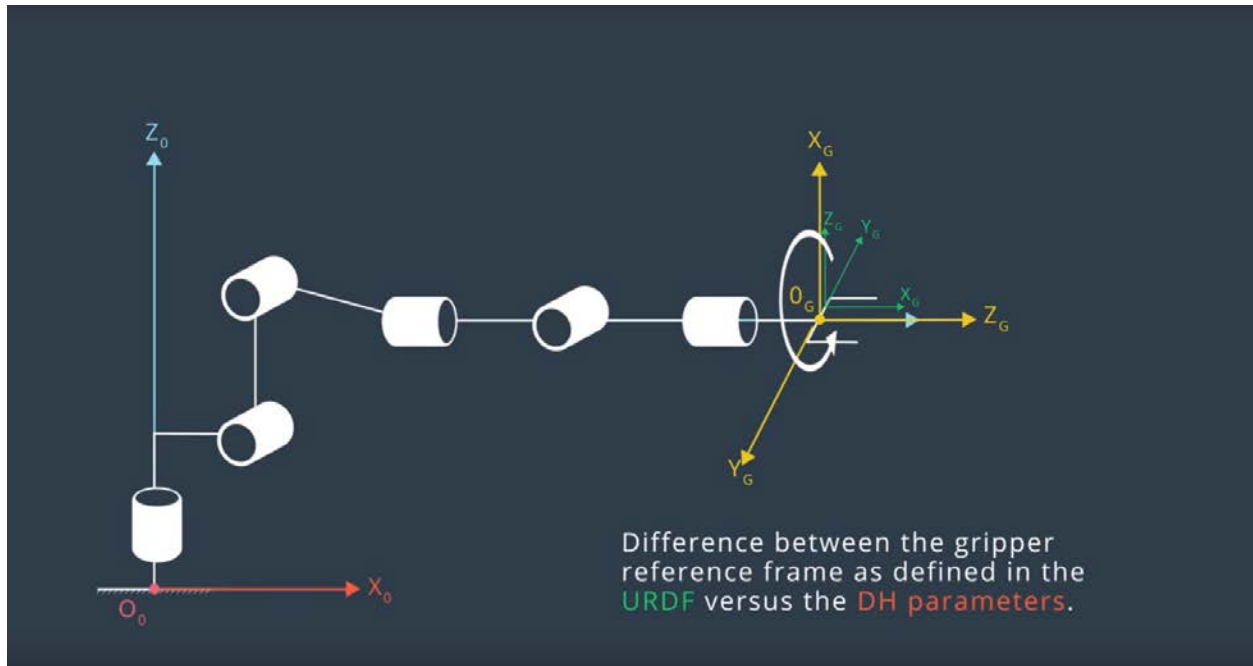
$$T0_3 = T0_2 * T2_3$$

...

$$T0_EE = T0_6 * T6_EE$$

Inverse Position Kinematics

In robotics, inverse kinematics uses kinematics equations to determine the joint parameters that provide the desired position of the robot's end-effectors. For this to occur with the KR210, one needs to account for the discrepancy between the DH parameters and the Gazebo URDF:



This can be corrected by rotating the Z axis by 180 degrees followed by a rotation of the Y axis of -90 degrees

Using the values: r – rotate, p – pitch, y – yaw, one can rotate the gripper to reflect the URDF file instead of the DH parameter using the following Matrix

$$R_{rpy} = \text{Matrix}([[\cos(y) \cdot \cos(p), \\ \cos(y) \cdot \sin(p) \cdot \sin(r) - \sin(y) \cdot \cos(r), \cos(y) \cdot \sin(p) \cdot \cos(r) + \sin(y) \cdot \sin(r)], \\ [\sin(y) \cdot \cos(p), \sin(y) \cdot \sin(p) \cdot \sin(r) + \cos(y) \cdot \cos(r), \\ \sin(y) \cdot \sin(p) \cdot \cos(r) - \cos(y) \cdot \sin(r)], \\ [-\sin(p), \cos(p) \cdot \sin(r), \cos(p) \cdot \cos(r)]]])$$

And $R_{corr} = \text{Matrix}([[0,0,1],$
 $[0,-1,0],$
 $[1,0,0]])$

The final rotational correction becomes: $R_{rpy} = R_{rpy} * (R_{corr}.T)$

The wrist center (WC) can be calculated from the Extender Matrix

$EE = \text{Matrix}([px],$
 $[py],$
 $[pz])$

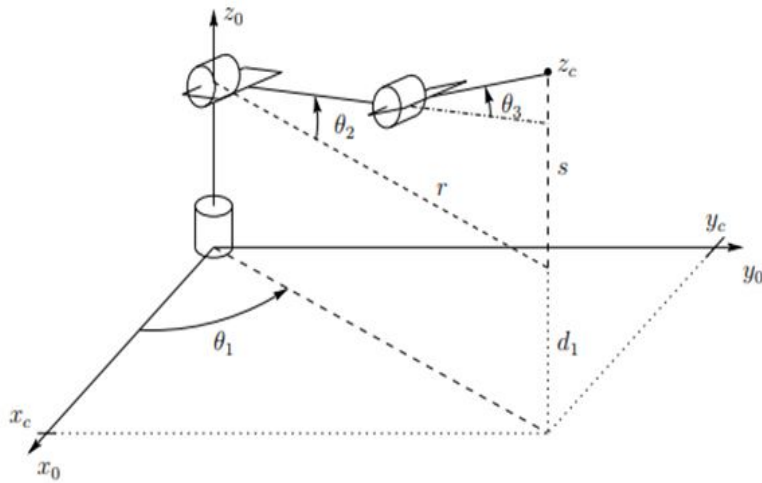
Where px , py , py are the roll, pitch, and yaw for the end effector position

Therefore the WC becomes

$WC = EE - (0,303) * R_{rpy}[:,2]$

Solving the inverse kinematics for the Kuka KR210 consists of dividing the robot into two parts:

Part 1 – joints 1 thru 3.



Theta1 is obtained from the equation:

$$\theta_1 = \text{atan2}(y_c, x_c)$$

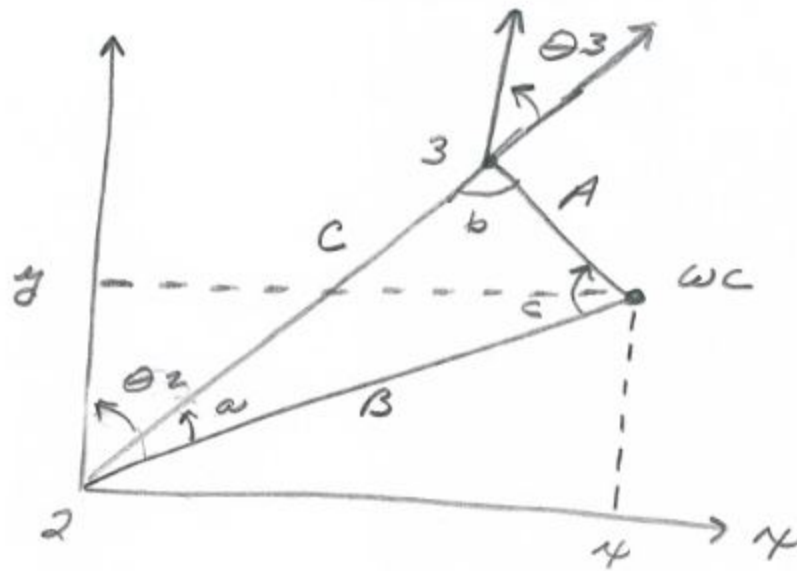
Using the values from our DH parameters to Gazebo yields:

$$\theta_1 = \text{atan2}(WC[1], WC[2])$$

The remaining angles can be obtained by using the Cosine Law:

$$C^2 = A^2 + B^2 - 2AB\cos(\gamma)$$

$$\cos(\gamma) = \frac{A^2 + B^2 - C^2}{2AB}$$



The following code applies the Cosine Law to obtain theta 2 and theta3

```
side_a = 1.25 # from measurements
side_b = 1.501 # from calculations
side_c = sqrt(pow((sqrt(WC[0] * WC[0] + WC[1] * WC[1]) - 0.35), 2) + pow((WC[2] - 0.75), 2))

# get the angle by taking the inverse cos
angle_a = acos((side_c * side_c + side_a * side_a - side_b * side_b) / (2 * side_c * side_a))
angle_b = acos((side_b * side_b + side_a * side_a - side_c * side_c) / (2 * side_b * side_a))
angle_c = acos((side_c * side_c + side_b * side_b - side_a * side_a) / (2 * side_c * side_b))

theta2 = pi/2 - angle_a - atan2((WC[2] - 0.75), (sqrt(WC[0] * WC[0] + WC[1] * WC[1]) - 0.35))
theta3 = pi/2 - angle_b - 0.036 # 0.036 accounts for sag in Link4 of -0.054
```

Part 2 – joint 4 thru 6 (wrist with gripper) : The previous three joint angles can be used to obtain the wrist center. Using the previous information, the following rotation matrix is used to solve for the remaining three angles

$${}^3_6R = ({}^0_3R)^{-1} {}^0_6R = ({}^0_3R)^T {}^0_6R$$

This results in:

$$\begin{aligned} {}^A_B R_{XYZ} &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \end{aligned}$$

The Rotation Matrix can be coded as:

RO_1 = T0_1[0:3,0:3]

RO_2 = T0_2[0:3,0:3]

...

RO_EE = T0_EE[0:3,0:3]

The final information needed to determine theta 4 thru 6 is

1. The Rotation matrix from 0 to 3


```
R0_3_calc = R0_3.evaf(subs={q1,: theta1, q2: theat2,  
q3: theta3})
```

2, The transformation matrix from 3 to 6

```
R3_6 = R0_3_calc.T * Rrpy
```

Calculating the values of theta4, theta5, and theta 6 is as follows:

```
theta4 = atan2(R3_6[2,2], -R3_6[0,2])  
theta5 = atan2(sqrt(R3_6[0,2]*R3_6[0,2] + R3_6[2,2]*R3_6[2,2]), R3_6[1,2])  
theta6 = atan2(-R3_6[1,1], R3_6[1,0])
```

IK_server.py

Coding the IK_server.py module was time consuming. After testing the code in using the IK_debutg.py program, I still found error in some of the transform matrix even though the results from the IK_debug .py program appeared to be correct. I was concerned about the amount of rotation of the gripper. This could have been corrected by limiting the maximum angles of rotation. My slow workstation did not allow the gripper to close before moving to the drop off point. I stepped thru the process and the problem was no longer as issue. I took screen shots of each run and successfully performed the pick and place greater than or equal to 8 out of 10 time.

Conclusion:

The code completed its task. It successfully grabbed the cylinder and placed into the barrel. I used the Windows snipping tool to take screen shot's of 8 successfully runs. The slowness of my computer prevented me from using any video capture software to record a video of the project(unbearably slow). I used the next button to step through the process to allow the computer ample time to perform the simulation. The continue button failed to allow sufficient time for the gripper fingers to close.

This was a very difficult project for me. I identified weakness in my matrix operations skills as also in my ability to identify and assign DH parameters. My slow workstation hindered my development, debugging and testing phases. I would like to thank the countless individual who posted videos, documents, and help hints on the internet including U-Tube, which helped me in understand DH Parameters determination, Matrix manipulations and KR210. The 40+ hours spend on these items only scratch the surface in letting me know area I need additional assistance. Thank you all.

