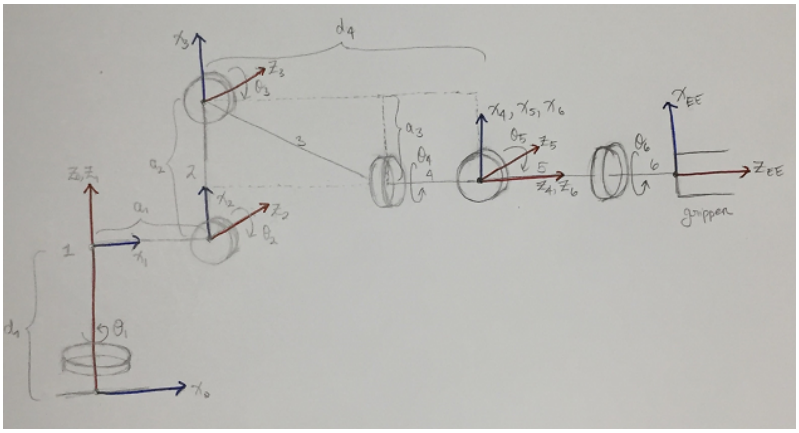
# Robotic Arm Pick and Place Project: Forward Kinematics and Inverse Kinematics of the Kuka KR210

January 5, 2018

## DH parameters

In forward kinematics, we construct homogenous matrices, $T_i^{i-1}$, for each joint i=1,2,..n whose composition encodes a matrix that gives the end effector's position and orientation relative to the frame corresponding to the base link. To derive each matrix, $T_i^{i-1}$, we draw a diagram of the links and joints of the robot and assign frames to each joint to help us find the DH parameters that compose $T_i^{i-1}$. The following is a diagram illustrating the nonzero DH parameters: ($\theta_i$, $d_i$, $a_{i-1}$, $\alpha_{i-1}$) assigned to the 6 degrees of freedom robot: Kuka KR210. Note that the choice of direction of which the positive $Y_i$ axis points for frames corresponding to the links $i$ determines the direction in which $\theta_i$ turns. In this case, $Y_i$ for $i = 1, 4, 6$ is pointing into the page so that the corresponding $\theta_i$ are positive when they turn counter-clockwise about $Z_i$. Similarily, the $Y_i$-axis for $i = 2, 3, 5$ are pointing to the right so that $\theta_i$ are positive when turning in the counter-clockwise direction about $Z_i$.




The corresponding table lists the values of the DH parameters. The constants $d_1$, $a_1$, $a_2$, $a_3$, $d_4$, and $d_G$ are obtained from measurements of the robot and $\theta_i$ for $i = 1, 2, 3, 4, 5, 6$ are joint angle variables.

| link i | $T_i^{i-1}$ | $\alpha_{i-1}$ | $a_{i-1}$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|---|
| 1 | $T_1^0$ | 0 | 0 | $d_1$ | $\theta_1$ |
| 2 | $T_2^1$ | $-\pi/2$ | $a_1$ | 0 | $\theta_2 - \pi/2$ |
| 3 | $T_3^2$ | 0 | $a_2$ | 0 | $\theta_3$ |
| 4 | $T_4^3$ | $-\pi/2$ | $-a_3$ | $d_4$ | $\theta_4$ |
| 5 | $T_5^4$ | $\pi/2$ | 0 | 0 | $\theta_5$ |
| 6 | $T_6^5$ | $-\pi/2$ | 0 | 0 | $\theta_6$ |
| 7 | $T_{EE}^6$ | 0 | 0 | $d_G$ | 0 |

# Forward Kinematics

The goal of forward kinematics is to obtain the pose for all links in any given coordinate frame. In particular, one wants to find the pose for the end effector with respect to the base frame. This is given by the matrix $T_{EE}^0$ which can be computed as the composition of the intermediate transformations $T_i^{i-1}$ associated to each link $i$:

$$T_{EE}^0 = T_1^0 * T_2^1 * T_3^2 * T_4^3 * T_5^4 * T_6^5 * T_{EE}^6 \tag{1}$$

Each transformation matrix is defined by:

$$T_i^{i-1} = R_x(\alpha_{i-1}) * D_x(a_{i-1}) * R_z(\theta_i) * D_z(d_i) \tag{2}$$

where ($\alpha_{i-1}$, $a_{i-1}$, $\theta_i$, $d_i$) are the DH parameters associated to each link $i$ and

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_x(a) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_z(a) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence, a simple computation gives:

$$T_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ cos(\alpha_{i-1})sin(\theta_i) & cos(\alpha_{i-1})cos(\theta_i) & -sin(\alpha_{i-1}) & -d_i sin(\alpha_{i-1}) \\ sin(\alpha_{i-1})sin(\theta_i) & cos(\alpha_{i-1})cos(\theta_i) & cos(\alpha_{i-1}) & d_n cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given the DH parameters, we can therefore compute each $T_i^{i-1}$ using the above formula:

$$T_1^0 = \begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 0 & 0 \\ sin(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} sin(\theta_2) & cos(\theta_2) & 0 & a_1 \\ 0 & 0 & 1 & 0 \\ cos(\theta_2) & -sin(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 & a_2 \\ sin(\theta_3) & cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} cos(\theta_4) & -sin(\theta_4) & 0 & -a_3 \\ 0 & 0 & 1 & d_4 \\ -sin(\theta_4) & -cos(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} cos(\theta_5) & -sin(\theta_5) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ sin(\theta_5) & cos(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^5 = \begin{bmatrix} cos(\theta_6) & -sin(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin(\theta_6) & -cos(\theta_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{EE}^6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_G \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, we can construct a homogenous transform matrix from base link to end effector link by only using the position of the end effector link:

$$EE = (EE_x, EE_y, EE_z)$$

and the orientation of the end effector link:

$$R_{EE}^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

by combining the two in the following formula:

$$T_{EE}^0 = \left[\begin{array}{ccc|c} & R_6^0 & & \begin{matrix} EE_x \\ EE_y \\ EE_z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & EE_x \\ r_{21} & r_{22} & r_{23} & EE_y \\ r_{31} & r_{32} & r_{33} & EE_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

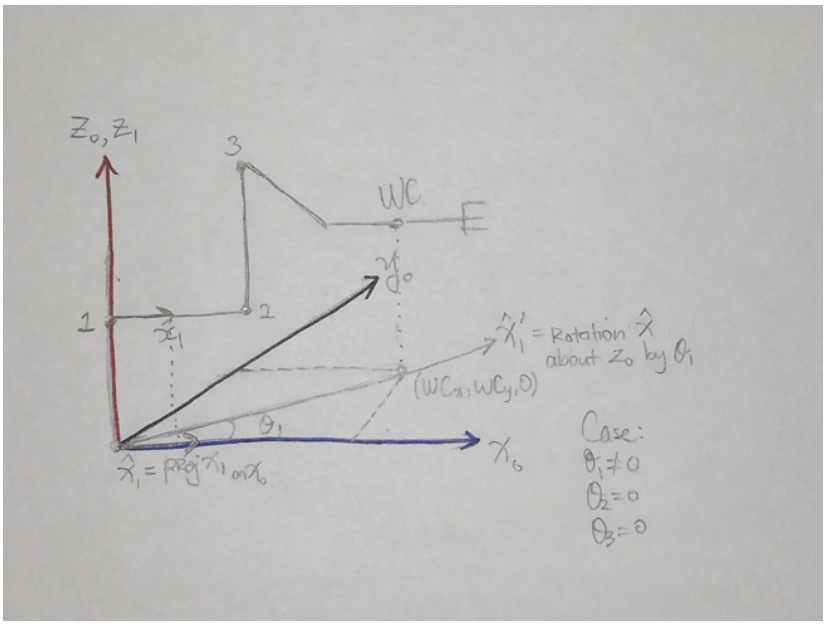



Figure 1

## Geometric Inverse Kinematics

The goal of inverse kinematics is the following: given the transformation matrix $T^0_{EE}$ obtained by forward kinematics and specifying the desired position and orientation of the end effector, solve for the joint variables $\theta_i$ for $i = 1, 2, 3, ...n$. Solutions to $\theta_i$ then provide a configuration of the joints whose end effector is in the desired pose. By setting the transformation matrix equal to a matrix that encodes the final pose of the end effector, one obtains a system of 12 equations to solve for the values of the joint variables. But this is often hard to do. Instead, we use the desired position of the end effector and geometry of the robot to obtain equations that solve for the first three variables: $\theta_1$, $\theta_2$, and $\theta_3$. Then we use the desired orientation of the end effector together with the inverted matrix of the orientation of the third joint with respect to the base frame to obtain equations for the end effector poised with respect to the fourth frame which are easier to solve for $\theta_4$, $\theta_5$ and $\theta_6$.

To see how this works, lets start with deriving $\theta_1$. Consider the base frame, $\{X_0, Y_0, Z_0\}$ as colored blue, black, and red respectively in Figure 1: 1 and consider the frame associated to link 1: $\{X_1, Z_1\}$. Project $X_1$ onto the $X_0$ axis and call this axis $\hat{X}_1$. Note that $X_1$ rotation about $Z_1$ is the same as $\hat{X}_1$ rotation about $Z_1$ since $\hat{X}_1$ is a parallel translate of $X_1$ along $Z_1$. Rotate $\hat{X}_1$ about the $Z_1$ axis to a new axis $\hat{X}_1'$. The angle between $\hat{X}_0$ and the new axis $\hat{X}_1'$ is equivalent to $\theta_1$.

To solve for the value of $\theta_1$, we write it as a formula in terms of desired

coordinates of the end effector. The desired coordinates are:

$$\left[\begin{array}{ccc|c} & & & EE_x \\ & R_6^0 & & EE_y \\ & & & EE_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

where $R_6^0$ are the coordinates for a 3x3 rotation matrix specifying the orientation of the end effector and $(EE_x, EE_y, EE_z)$ is the position of the end effector. Observe that the wrist center is given by the formula:

$$WC = \begin{bmatrix} EE_x \\ EE_y \\ EE_z \end{bmatrix} - d_G * R_6^0 * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Here, the position of the end effector subtracts off the change between the wrist center and the end effector. In particular, this change is given by the translation along the end effector's z-axis from wrist center to end effector by $d_G$ units with respect to the base frame. Since the coordinates of the wrist center: WC $= (WC_x, WC_y, WC_z)$ can be totally described by the coordinates of the end effector pose, then it is enough to solve $\theta_1$ by recasting it in terms of the coordinates of WC. In fact, if we project the wrist center position down to the $X_0 - Y_0$ axis, we see that $(WC_x, WC_y, 0)$ intersects $\hat{X}_1'$. From this we can obtain a right triangle for which the opposite side of $\theta_1$ has length $WC_y$ and the adjacent non-hypoteneuse side of $\theta_1$ has length $WC_x$. Therefore we can recast $\theta_1$ as:

$$\theta_1 = arctan2(WC_y, WC_x)$$

Notice that we could have used the end effector's y and x coordinates in place of the wrist centers y and x coordinates since the projection of the end effector's coordinates onto the $X_0 - Y_0$ plane also lies on the $\hat{X}_1'$ axis, but we focus all calculations centering on the wrist center because it is the origin of intersection for the frames of three revolute joints which determine the end effector's orientation.

Next, we find a formula for $\theta_2$ in terms of wrist center coordinates and fixed value DH parameters. Observe the following figure 2: 2

The value that we are interested in finding is the angle between the $X_2'$ axis and the $X_1$ axis where $X_2'$ is the $X_2$ axis rotated at an angle $\theta_2$ about the $Z_2$ axis in the clockwise direction. In fact, this should be equal to $\theta_2 - \pi/2$ since at $\theta_2 = 0$, the angle between the $X_2$ axis and the $X_1$ axis is $\pi/2$. To find $\theta_2$, we see from the figure that

$$\theta_2 = \pi/2 - a - d$$

To obtain angle $d$, observe from figure 2: 2 that we can construct a right triangle where the opposite side to the angle d has height $WC_z - d_1$ and an adjacent side of the angle d has length $WC_x - a_1$. But the side of length $WC_x - a_1$ is restricted by the fact that we are assuming in the figure that $\theta_1 = 0$ and therefore $WC_y = 0$. To account for non-zero $\theta_1$ and therefore non-zero $WC_y$, observe that in figure 1: 1 the length of the rotated line segment with end points $(WC_x, WC_y, 0)$ and the origin of the base frame lying along $\hat{X}_1'$ represents the new $WC_x$ when $\theta_1$

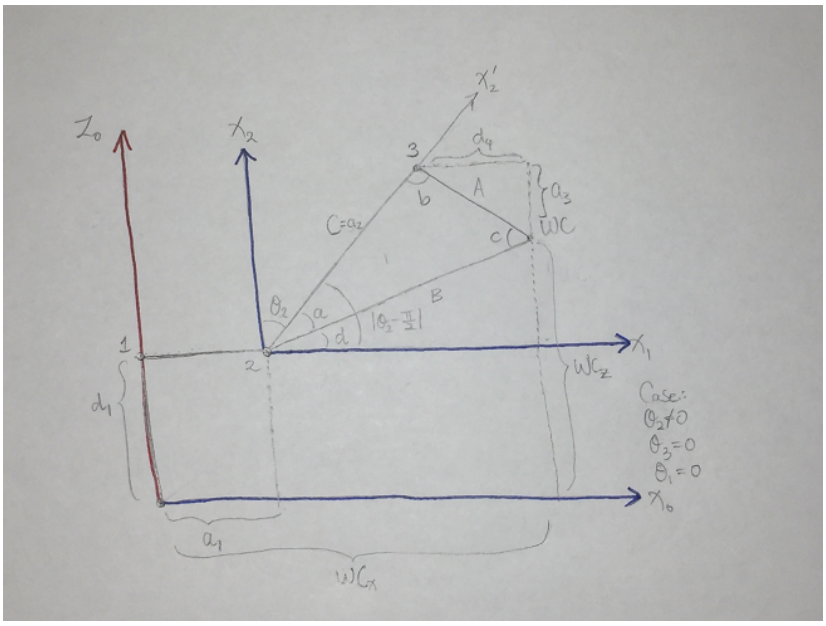



Figure 2

is nonzero. Therefore, we can replace $WC_x$ with the generalized case where the length of the hypoteneuse of the right triangle with side lengths $WC_x$ and $WC_y$ is what we are interested in. By pythogrean theorem, we obtain $\sqrt{WC_x^2 + WC_y^2}$ as the replacement for $WC_x$ for nonzero $\theta_1$. This allows us to write $d$ as follows:

$$d = arctan2(WC_z - d_1, \sqrt{WC_x^2 - WC_y^2} - a_1)$$

Also, observe that the geometry of figure 2: 2 does not take into consideration non-zero $\theta_3$. We can check to see that we do not need to make any modifications to our equation by observing in figure 3: 3 that the equations remain invariant under the new geometric construction when a perturbation by $\theta_3$ yields a new WC coordinate. Further, $\theta_2$ and $\theta_3$ operate under similar constraints being confined to the $X_0 - Z_0$ axis so the coordinates of WC maintain its generality upon this restriction. To obtain angle $a$, observe from figure 2: 2 that there is a triangle with sides A,B,C that we can apply the law of cosines to obtain a formula for the angle $a$ in terms of the sides of the triangle. The law of cosines is given by:

$$A^2 = B^2 + C^2 - 2BCcos(a)$$

Solving for the angle $a$, we obtain:

$$a = cos^{-1}(\frac{B^2 + C^2 - A^2}{2BC})$$

Now we must find A,B, and C in terms of known parameters. C is simply the DH parameter $a_2$. B is the hypoteneuse of the right triangle with sides of length

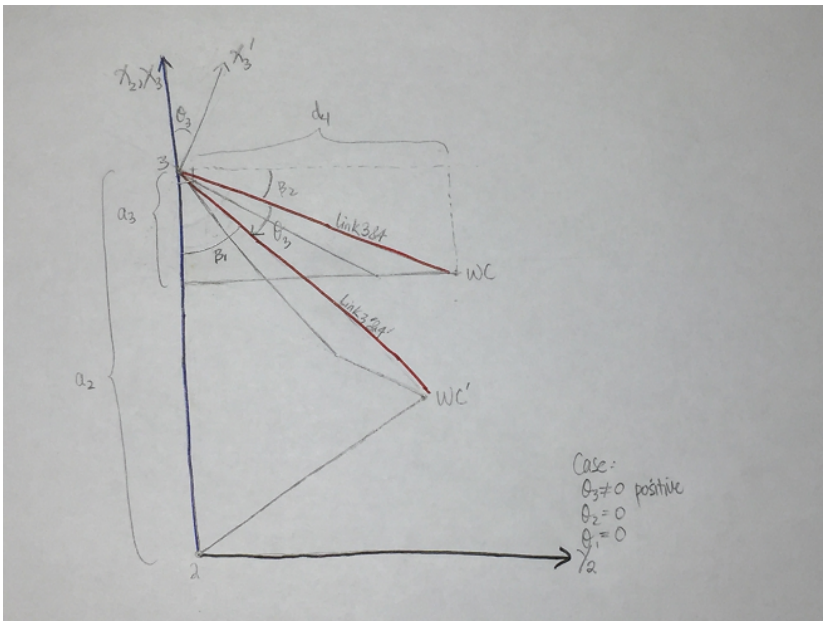



Figure 3

$WC_z - d_1$ and $\sqrt{WC_x^2 - WC_y^2} - a_1$ where these lengths were found when solving for angle $d$, so by the pythagorean theorem:

$$B = \sqrt{(WC_z - d_1)^2 + (\sqrt{WC_x^2 - WC_y^2} - a_1)^2}$$

Finally, A is the hypoteneuse of the right triangle with sides of length $d_4$ and $a_3$. Hence, by pythagorean theorem:

$$A = \sqrt{d_4^2 + a_3^2}$$

Furthermore, notice that the equation for $\theta_2$ is restricted to account for a subspace of the workspace that link 2 can extend. In fact, the geometry to obtain $\theta_2$ sets limits for the parameter by taking $a \in (0, \pi)$ and $d \in [0, \pi/2 - a]$. So we can conclude that $\theta_2 \in [0, \pi/2]$. Therefore, $\theta_2$ is solved to move only within a quandrant of the fixed frame.

To solve for $\theta_3$ observe that by definition, $\theta_3$ is the angle between the $X_2$ axis and $X_3'$ axis where $X_3'$ is the rotation of the $X_3$ axis about the $Z_3$ axis by $\theta_3$. This is marked in figure 3: 3

Notice that there is also another angle $\theta_3$ formed between the segment labeled link3&4 and the segment labeled link3′&4′ that we also claim is equivalent to $\theta_3$. We reason that we can regard link3&4 similarily to $X_3$ as a fixed axis at joint 3 and any rotation of $X_3$ by an angle $\theta_3$ to its new position $X_3'$ would likewise rotate link3&4 by an angle of $\theta_3$ to its new position at link3′&4′. Therefore, as indicated by figure 3: 3, $\theta_3$ has the form:

$$|\theta_3| = \pi/2 - \beta_2 - \beta_1$$

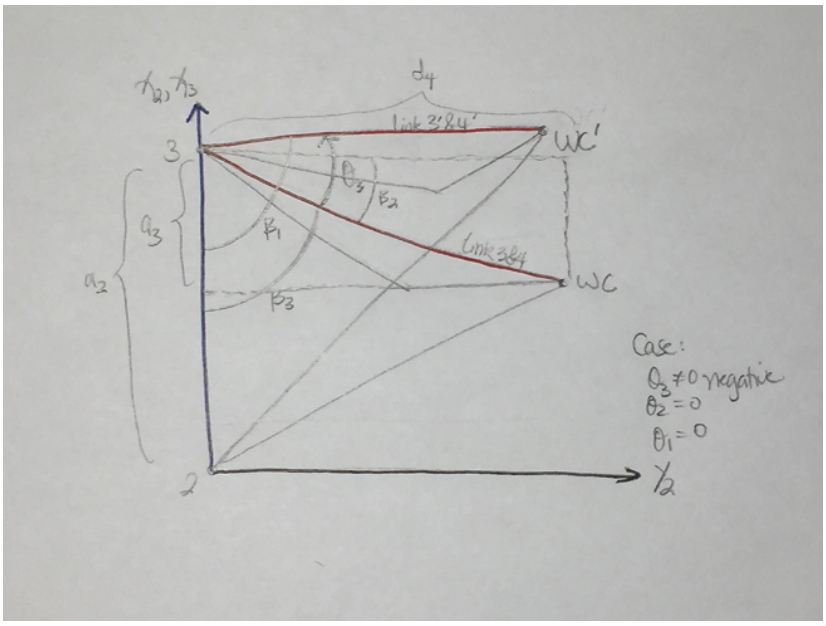



Figure 4

$\beta_1$ is equal to angle $b$ in figure 2: 2 so we can obtain $\beta_1$ by the law of cosines:

$$\beta_1 = b = cos^{-1}(\frac{A^2 + C^2 - B^2}{2AC})$$

where A,B,C are from figure 2: 2 which were already solved. Further, $\beta_2$ is given by forming the inverse tangent of $a_3$ with $d_4$:

$$\beta_2 = arctan2(|a_3|, |d_4|)$$

Observe that the equation derived for $|\theta_3|$ has restricted geometry where the movement of the angle of the third link is constrained to a quarter quandrant of the cartesian plane. Notice that $\beta_1 \in [0, \pi/2 - \beta_2]$ and $\beta_2$ is fixed so $|\theta_3| \in [0, \pi/2 - \beta_2]$. And finally, since $\theta_3$ is turning in the positive direction, then $\theta_3 = |\theta_3|$. This finishes a constrained solution for $\theta_3$. Further, we can extend $\theta_3$ to the interval $[-\pi/2 + \beta_2, 0)$ by turning $\theta_3$ in the negative direction. Observe from figure 4: 4 that we have the following formula for $|\theta_3|$:

$$|\theta_3| = \beta_1 - \beta_3$$
$$\beta_3 = \pi/2 - \beta_2$$

Once again, we have the same formula for $\beta_1$ and $\beta_2$:

$$\beta_1 = b$$
$$\beta_2 = arctan2(|a_3|, |d_4|)$$

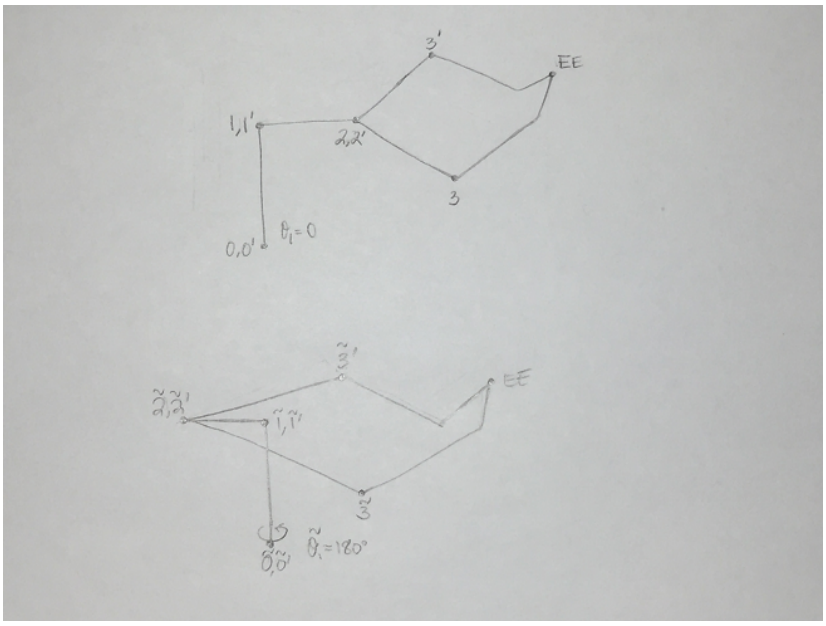



Figure 5

Therefore, we obtain the formula:

$$|\theta_3| = \beta_1 - \pi/2 + \beta_2$$

But since we are turning in the negative direction, we have:

$$\theta_3 = -|\theta_3| = \pi/2 - \beta_2 - \beta_1$$

which turns out to be the same formula for $\theta_3$ in the region $[0, \pi/2 - \beta_2]$. Hence, we have extended the equation for $\theta_3$ into the region $[-\pi/2 + \beta_2, 0)$.

In addition to this set of solutions for $\theta_1$, $\theta_2$, and $\theta_3$, we can take a further look at the symmetries of the robot to find that multiple solutions to a given end effector position also exist when we fix $\theta_4$, $\theta_5$ and $\theta_6$ and vary $\theta_1$, $\theta_2$ and $\theta_3$. In fact, restricted to planar movement, links 2 and 3 mimic joint angle solution sets of the RR robot arm. This gives two solutions of the robot when $\theta_1 = 0$ as shown in the first sketch of figure 5: 5.

On the other hand, if we stay in the same plane, we may turn $\theta_1$ around by $180°$ and re-orient $\theta_2$ and $\theta_3$ in a similar way to when $\theta_1 = 0$ to reach for the end effector's pose. This is shown in the second sketch of figure 5: 5. However, for the purposes of our pick and place task at hand, we stick to the formula which model the first solution as diagramed in figure 5, sketch 1, with joint labels $0', 1', 2', 3'$. This allows for the most maneuverability within its workspace to reach it's targets.

To obtain solutions to $\theta_4$, $\theta_5$ and $\theta_6$, we consider the orientation part of the end effector: $R_6^0$. This has a decomposition:

$$R_6^0 = R_3^0 * R_6^3$$

Hence, taking the inverse of $R_3^0$, we have:

$$R_6^3 = (R_3^0)^{-1} * R_6^0$$

$R_3^0$ is a matrix valued function in the variables $\theta_1$, $\theta_2$, and $\theta_3$ of which we already solved for in terms of the desired pose of the end effector and constant DH parameters, $R_6^0$ is given initially as apart of the end effector's desired orientation, and $R_6^3$ is a matrix valued function in the variables $\theta_4$, $\theta_5$, and $\theta_6$ for which we need to solve for by the above equation. Observe that if we calculate the homogenous matrix $T_6^3$ via:

$$T_6^3 = T_4^3 * T_5^4 * T_6^5$$

the orientation part of $T_6^3$, $R_6^3$ will have the following entries in the matrix. Denote $R_6^3(i,j)$ to be the i-th row and j-th column of the matrix $R_6^3$, then we have the functions:

$$\begin{aligned}
R_6^3(1,3) &= -sin(\theta_5)cos(\theta_4) \\
R_6^3(3,3) &= sin(\theta_4)sin(\theta_5) \\
R_6^3(2,2) &= -sin(\theta_5)sin(\theta_6) \\
R_6^3(2,1) &= sin(\theta_5)cos(\theta_6) \\
R_6^3(2,3) &= cos(\theta_5)
\end{aligned}$$

To find $\theta_4$, we use the first two equations and nonzero $\theta_5$ to deduce the following two cases. We also use the function arctan2() instead of arctan() to distinguish double solutions when using arctan(). If $sin(\theta_5) > 0$ then we have:

$$\begin{aligned}
arctan2(R_6^3(3,3), -R_6^3(1,3)) &= arctan2(sin(\theta_4)sin(\theta_5), sin(\theta_5)cos(\theta_4)) \\
&= arctan2(\operatorname{sgn}(sin(\theta_5))sin(\theta_4), \operatorname{sgn}(sin(\theta_5))cos(\theta_4)) \\
&= arctan2(sin(\theta_4), cos(\theta_4)) \\
&= \theta_4
\end{aligned}$$

If $sin(\theta_5) < 0$ then we have:

$$\begin{aligned}
arctan2(-R_6^3(3,3), R_6^3(1,3)) &= arctan2(-sin(\theta_4)sin(\theta_5), -sin(\theta_5)cos(\theta_4)) \\
&= arctan2(-\operatorname{sgn}(sin(\theta_5))sin(\theta_4), -\operatorname{sgn}(sin(\theta_5))cos(\theta_4)) \\
&= arctan2(sin(\theta_4), cos(\theta_4)) \\
&= \theta_4
\end{aligned}$$

To find $\theta_5$, we observe:

$$\begin{aligned}
R_6^3(1,3)^2 + R_6^3(3,3,)^2 &= sin^2(\theta_5)cos^2(\theta_4) + sin^2(\theta_4)sin^2(\theta_5) \\
&= sin^2(\theta_5)(cos^2(\theta_4) + sin^2(\theta_4)) \\
&= sin^2(\theta_5)
\end{aligned}$$

So, we have:

$$\pm\sqrt{R_6^3(1,2)^2 + R_6^3(3,3)^2} = \pm|sin(\theta_5)|$$

And therefore:

$$arctan2(\pm\sqrt{R_6^3(1,2)^2 + R_6^3(3,3)^2}, R_6^3(2,3)) = arctan2(\pm|sin(\theta_5)|, cos(\theta_5)) = \pm\theta_5$$

Here, the sign of $\theta_5$ will depend on the sign of $sin(\theta_5)$. Finally, $\theta_6$ is found taking $\theta_5$ nonzero and simplifying for the following two cases similar to $\theta_4$. If $sin(\theta_5) > 0$ then we have:

$$\begin{aligned} arctan2(-R_6^3(2,2), R_6^3(2,1)) &= arctan2(sin(\theta_5)sin(\theta_6), sin(\theta_5)cos(\theta_6)) \\ &= arctan2(\operatorname{sgn}(sin(\theta_5))sin(\theta_6), \operatorname{sgn}(sin(\theta_5))cos(\theta_6)) \\ &= arctan2(sin(\theta_6), cos(\theta_6)) \\ &= \theta_6 \end{aligned}$$

If $sin(\theta_5) < 0$, then we get:

$$\begin{aligned} arctan2(R_6^3(2,2), -R_6^3(2,1)) &= arctan2(-sin(\theta_5)sin(\theta_6), -sin(\theta_5)cos(\theta_6)) \\ &= arctan2(-\operatorname{sgn}(sin(\theta_5))sin(\theta_6), -\operatorname{sgn}(sin(\theta_5))cos(\theta_6)) \\ &= arctan2(sin(\theta_6), cos(\theta_6)) \\ &= \theta_6 \end{aligned}$$

In the particular case when $\theta_5 = 0$ there is a redundancy in solution choices in selecting $\theta_4$ and $\theta_6$ to be in such a configuration. However, in any case that $\theta_5 = 0$, for simplicity, we choose $\theta_4 = 0$ and although $\theta_6$ can produce unique end effector poses for varying $\theta_6$ depending on the shape of the end effector, for ease we also assume $\theta_6 = 0$ as well.

Observe figure 6: 6, sketch #3 to see what is meant by the multiple solutions that occur when $\theta_5 = 0$. In this case, varying $\theta_4$ and $\theta_6$ from $0°$ to $360°$ rotates the end effector in a circle without moving the position of the end effector.

Furthermore, we can obtain more solutions by setting $\theta_1 = \theta_4$ and rotating by this angle the robot outlined by connecting the joints $2, 3, EE$ in figure 5: 6, sketch #1 around the axis created by joining 2 and 3. We can also take $\theta_2$ and $\theta_3$ nonzero as illustrated by the robot outlined by connecting $2', 3', EE'$ and rotate this robot about the axis created by joining 2 and 3 and observe another circle of solutions. Finally, as illustrated in figure 5: 6, sketch #2, (also illustrated in sketch #1), we can make a maneuveur of $\theta_1$ by a maximum of $180°$ angle and modify the other angles respectively to line up with the end effector intitial position. However, we maintain the use of a single set of equations as dictated by the previous formulas to create our workspace.

Now that we have found a subset of solutions for the joint angles, we can go to the next step and code it up.

## Code in IK_server.py

The following is code that calculates a set of joint angles as a service to translate the body of the robot to end effector trajectory points. It also produces a comparison graph of the error between the desired trajectory for the robot to follow and the actual path taken which is calculated by the inverse kinematics joint angles evaluated in the forward kinematics equations. Note that the code does not use sympy and uses only numpy and any use of the inverse for rotation

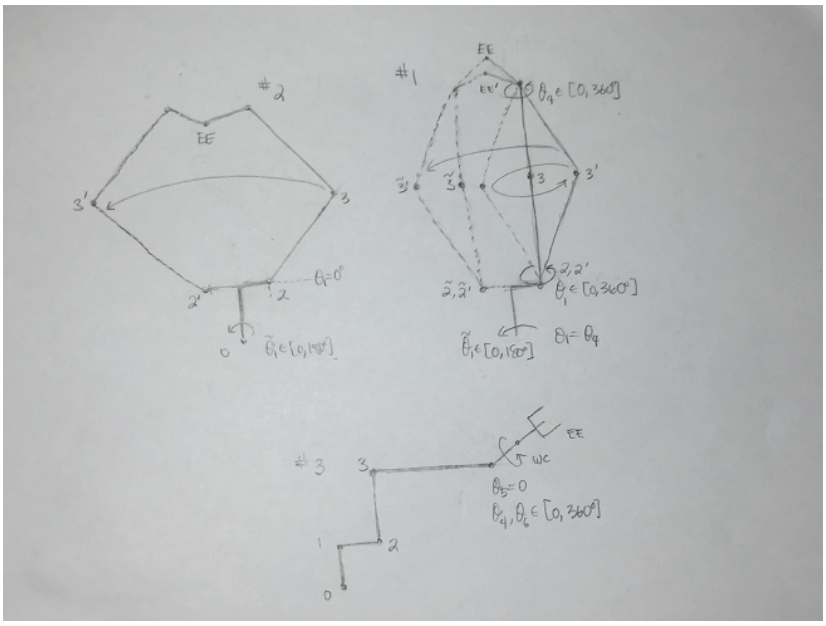



Figure 6

matrices is replaced with the mathematically equivalent transpose of the rotation instead. Both contribute to improving accuracy and speed of computation. On another note, float64 is used throughout the code to maintain accuracy.

```
#!/usr/bin/env python

# Copyright (C) 2017 Udacity Inc.
#
# This file is part of Robotic Arm: Pick and Place project for
    Udacity
# Robotics nano-degree program
#
# All Rights Reserved.

# Author: Harsh Pandya

# import modules
import rospy
import tf
from kuka_arm.srv import *
from trajectory_msgs.msg import JointTrajectory,
    JointTrajectoryPoint
from geometry_msgs.msg import Pose
import numpy as np
from numpy import *
from math import atan2, acos
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Define transformation matrices

```

```
def T0_1(q):
    T = np.array([[cos(q), -sin(q), 0, 0],
                  [sin(q), cos(q), 0, 0],
                  [0, 0, 1, .75],
                  [0, 0, 0, 1 ]], dtype=np.float64)
    return T

def T1_2(q):
    T = np.array([[sin(q), cos(q), 0, .35],
                  [0, 0, 1, 0],
                  [cos(q), -sin(q), 0, 0],
                  [0, 0, 0, 1]], dtype=np.float64)
    return T

def T2_3(q):
    T = np.array([[cos(q), -sin(q), 0, 1.25],
                  [sin(q), cos(q), 0, 0],
                  [0, 0, 1, 0],
                  [0, 0, 0, 1]], dtype=np.float64)
    return T

def T3_4(q):
    T = np.array([[cos(q), -sin(q), 0, .054],
                  [0, 0, 1, 1.5],
                  [-sin(q), -cos(q), 0, 0],
                  [0, 0, 0, 1]], dtype=np.float64)
    return T

def T4_5(q):
    T = np.array([[cos(q), -sin(q), 0, 0],
                  [0, 0, -1, 0],
                  [sin(q), cos(q), 0, 0],
                  [0, 0, 0, 1]], dtype=np.float64)
    return T

def T5_6(q):
    T = np.array([[cos(q), -sin(q), 0, 0],
                  [0, 0, 1, 0],
                  [-sin(q), -sin(q), 0, 0],
                  [0, 0, 0, 1]], dtype=np.float64)
    return T

def T6_EE(q):
    T = np.array([[1, 0, 0, 0],
                  [0, 1, 0, 0],
                  [0, 0, 1, .303],
                  [0, 0, 0, 1]], dtype=np.float64)
    return T

def Rot_x(q):
    T = np.array([[1,0,0],
                  [0, cos(q), -sin(q)],
                  [0, sin(q), cos(q)]], dtype=np.float64)
    return T

def Rot_y(q):
    T = np.array([[cos(q), 0, sin(q)],
                  [0, 1, 0],
                  [-sin(q), 0, cos(q)]], dtype=np.float64)
    return T

def Rot_z(q):
```

```
    T = np.array([[cos(q), -sin(q), 0],
                  [sin(q), cos(q), 0],
                  [0, 0, 1]], dtype=np.float64)
    return T

def Rot_extrinsic_XYZ(z,y,x):
    T = np.array([[cos(z)*cos(y), cos(z)*sin(y)*sin(x)-sin(z)*cos(x), cos(z)*sin(y)*cos(x) + sin(z)*sin(x)],
                  [sin(z)*cos(y), sin(z)*sin(y)*sin(x)+cos(z)*cos(x), sin(z)*sin(y)*cos(x) - cos(z)*sin(x)],
                  [-sin(y), cos(y)*sin(x), cos(y)*cos(x)]], dtype=np.float64)
    return T

def handle_calculate_IK(req):
    rospy.loginfo("Received %s eef-poses from the plan" % len(req.poses))
    if len(req.poses) < 1:
        print "No valid poses received"
        return -1
    else:

        # Initialize service response
        joint_trajectory_list = []
        PX = []
        PY = []
        PZ = []
        PXcheck = []
        PYcheck = []
        PZcheck = []
        for x in xrange(0, len(req.poses)):
            # IK code starts here
            joint_trajectory_point = JointTrajectoryPoint()

      # Extract end-effector position and orientation from request
      # px,py,pz = end-effector position
      # roll, pitch, yaw = end-effector orientation
            px = req.poses[x].position.x
            py = req.poses[x].position.y
            pz = req.poses[x].position.z

            (roll, pitch, yaw) = tf.transformations.euler_from_quaternion(
                [req.poses[x].orientation.x, req.poses[x].orientation.y,
                    req.poses[x].orientation.z, req.poses[x].orientation.w])

            EE = np.array([px,py,pz], dtype = np.float64)

            Rot_EE = Rot_extrinsic_XYZ(yaw,pitch,roll)
            #R_zR_y = np.dot(Rot_z(yaw), Rot_y(pitch))
            #Rot_EE = np.dot(R_zR_y, Rot_x(roll))

            ### Your IK code here
      # Compensate for rotation discrepancy between DH parameters and Gazebo
          Rot_Error = np.dot(Rot_z(np.pi), Rot_y(-np.pi/2))
              Rot_EE = np.dot(Rot_EE, Rot_Error)
      #
      # Calculate joint angles using Geometric IK method
        # Calculate wrist center
```

```
            WC = EE - 0.303*Rot_EE[:,2]
        # project wrist center coordinates to the base frame to obtain theta_1
            theta1 = atan2(WC[1],WC[0])
        # find sides of projected triangle connecting joint 2 and 3 and WC for theta2 and theta3
            side_a = 1.501
            side_b = sqrt(pow((sqrt(WC[0] ** 2 + WC[1] ** 2) - 0.35), 2) + pow((WC[2] - 0.75), 2))
            side_c = 1.25
        # solve for angles a and b with inverse cosine
            angle_a = acos((side_b ** 2 + side_c ** 2 - side_a ** 2) / (2 * side_b * side_c))
            angle_b = acos((side_a ** 2 + side_c ** 2 - side_b ** 2) / (2 * side_a * side_c))

            theta2 = np.pi/2 - angle_a - atan2(WC[2] - 0.75, sqrt(WC[0] ** 2 + WC[1] ** 2) - 0.35)
            theta3 = np.pi/2 - (angle_b + 0.036)
        # Find theta4, theta5, and theta6 by using the equations of the orientation part of T3_6
            R0_2 = np.dot(T0_1(theta1)[0:3, 0:3], T1_2(theta2)[0:3, 0:3])
            R0_3 = np.dot(R0_2, T2_3(theta3)[0:3, 0:3])
            R3_6 = np.dot(R0_3.T, Rot_EE)

            if EE[2]-WC[2] >= 0.:
                theta5 = atan2(sqrt(R3_6[0,2] ** 2 + R3_6[2,2] ** 2), R3_6[1,2])
            elif EE[2]-WC[2] < 0.:
                theta5 = atan2(-sqrt(R3_6[0,2] ** 2 + R3_6[2,2] ** 2), R3_6[1,2])

            if sin(theta5) >= 0.:
                theta4 = atan2(R3_6[2,2], -R3_6[0,2])
                theta6 = atan2(-R3_6[1,1], R3_6[1,0])
            elif sin(theta5) < 0.:
                theta4 = atan2(-R3_6[2,2], R3_6[0,2])
                theta6 = atan2(R3_6[1,1], -R3_6[1,0])
            ###

            # Populate response for the IK request
            # In the next line replace theta1,theta2...,theta6 by your joint angle variables
          joint_trajectory_point.positions = [theta1, theta2, theta3, theta4, theta5, theta6]
          joint_trajectory_list.append(joint_trajectory_point)
            print("Calculated %s of %s" % (x + 1, len(req.poses)))

            # Can comment out below for faster run-time. The following makes a plot for the error.
            T0_2 = np.dot(T0_1(theta1), T1_2(theta2))
            T0_3 = np.dot(T0_2, T2_3(theta3))
            T0_4 = np.dot(T0_3, T3_4(theta4))
            T0_5 = np.dot(T0_4, T4_5(theta5))
            T0_6 = np.dot(T0_5, T5_6(theta6))
            T0_EE = np.dot(T0_6, T6_EE(0.))
            PX.append(px)
            PY.append(py)
            PZ.append(pz)
            PXcheck.append(T0_EE[0,3])
            PYcheck.append(T0_EE[1,3])
```

Figure 7

```
            PZcheck.append(T0_EE[2,3])


        rospy.loginfo("length of Joint Trajectory List: %s" % len(joint_trajectory_list))
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(PX, PY, PZ, c='r', marker='o', label='EE desired position')
        ax.scatter(PXcheck, PYcheck, PZcheck, c='b',marker='^', label='IK calculated position')
        plt.legend(loc=2)
        ax.set_xlabel('X axis')
        ax.set_ylabel('Y axis')
        ax.set_zlabel('Z axis')
        plt.show()
        return CalculateIKResponse(joint_trajectory_list)


def IK_server():
    # initialize node and declare calculate_ik service
    rospy.init_node('IK_server')
    s = rospy.Service('calculate_ik', CalculateIK, handle_calculate_IK)
    print "Ready to receive an IK request"
    rospy.spin()

if __name__ == "__main__":
    IK_server()
```

Finally, we present evidence of the result of a successful pick and place task. In figure 7 is the initial setup of the robot in the pick and place task with the robot ready to receive commands to pick-up the blue object in the upper right-hand corner of the shelf.

After *IK_server.py* has been rosrun and a request from Rviz with planned trajectory end effector points has been taken, the service calculates a set of joint angles which is then turned to action in the gazebo simulation environment where the robot follows the calculated trajectory given by the inverse kinematic

Figure 8

equations. Pictured in figure 8, the robot has reached its target object for pick-up.

Further, a plot of the error between desired end effector trajectory and calculated end effector trajectory is given in figure 9

Now we employ the calculate IK service once more where Rviz supplies a planned trajectory for the end effector of the robot and the service responds with a set of calculated joint angles ready to be used to plan the poses for the robot in the gazebo simulated environment. Once the robot has received the calculated trajectory, it acts to move to the drop-off location. Pictured in figure 10 is the final state of the robot positioned over the bucket to release its target object.

We also have a final analysis of the error plot between the expected trajectory and the computed trajectory as shown in figure 11

Note that commenting out the code that makes the error plot allows for faster computational time which leaves us with a fast inverse kinematics calculation service that finishes the pick and place project!

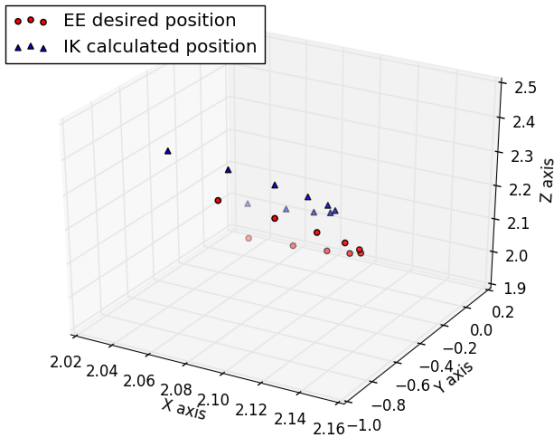



Figure 9

Figure 10

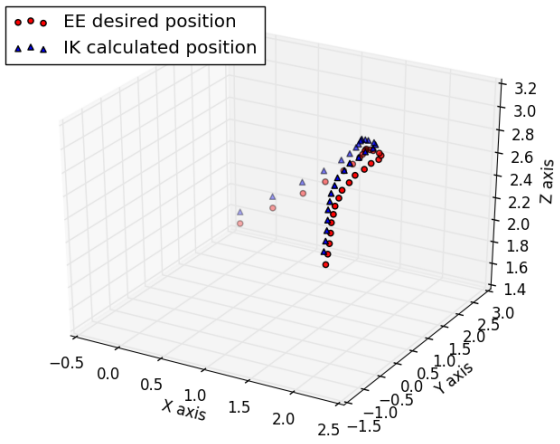

EE desired position
IK calculated position
X axis
Y axis
Z axis

Figure 11