

Project 2 Inverse kinematics of kr210 Serial Manipulator

Introduction

The basic operation of serial manipulators such as the KR210 shown in Figure 1 are conceptually easy to grasp. One can see that by setting the angle at each joint, the end effector (ee) can be placed in a continuous set of positions and orientations bounded by the physical geometry of the manipulator. The aggregation of position and orientation defined as the *pose* and fully defines the configuration of a joint or link in space. The reachable workspace for a manipulator is the volume bounded by the set of all points that the robot can reach with its end effector. Within reachable workspace exists the dexterous workspace which is defined by the reachable points that the effector can approach in any orientation. Conceptually, one can view the reachable workspace as the volume bounded by all points one can reach with just a fingertip, while the dexterous workspace represents the volume bounded by all points one could actually grasp and manipulate with unconstrained movement at the wrist.

With the basic operation and constraint as stated above, then the task of setting or determining the ee's pose in the workspace can be approached in two ways depending on which problem is being solved.

The first problem to be solved, is to determine the ee position given the values of the joint angles. The value of knowing the ee pose given joint angles is obvious, at robot initialization to determine pose prior to any movements, or following a movement, to calculate the error between the commanded and actual pose. This



Figure 1 Kuka KR210 Serial Manipulator

problem is commonly referred to as the *forward kinematics* problem and of the two is the easier to solve because for a given set of joint angles there exists one and only one possible solution for the pose of the ee.

The second problem to be solved is the reverse of the first problem. Rather than knowing the joint angles it is the ee pose that is known and the joint angles are to be determined. This is commonly referred to as the *inverse kinematics* problem. The primary complication with this problem being that there does not exist one and only one solution. Many, none or infinite solutions may exist depending on the physical robot configuration and the pose provided. In the simplest example, for the revolute joints in the kr210, one can imagine that in addition to a joint angle θ , there are infinite equivalent angles at $\theta \pm 2\pi$ that will (at least mathematically) satisfy a required pose. Beyond that, there are also solution sets that are not linear combinations of 2π , such as those shown Figure 2 from lesson 11 indicating how an 'elbow up' and 'elbow down' configuration coupled with a left hand and right hand orientation at the robot's waist can result in 4

valid solutions that will yield the same ee position. Note that in figure the orientations are not equivalent, but if one considers the ee in the image to instead be the wrist center as would be the case for our KR210 model, then the orientation is irrelevant and they would provide equivalent solutions.

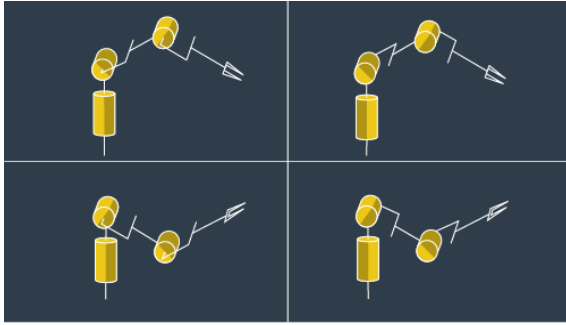


Figure 2 Anthropomorphic manipulator from lesson 11.18

For this project we will be solving for both the forward and inverse kinematics problems. The approach to each for the KR210 will be described in the individual sections. The successful submission will accurately describe solutions to both problems and test them on a simulated kr210 in the virtual sandbox tool Gazebo.

Forward Kinematics

The forward kinematic problem can be solved by starting at the base frame of the robot and then applying successive rotations (based off of the given joint angles) and translations (based off of the dimensions of the manipulator links). The individual rotation and translation at each joint can be combined into a single matrix called the homogenous transform which is described below in Equation 1 from lesson 11.10.

Equation 1 Generalized form of homogenous transformation

$$\begin{bmatrix} {}^A\mathbf{r}_{P/A_o} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^A_B R & {}^A\mathbf{r}_{B_o/A_o} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{r}_{P/B_o} \\ 1 \end{bmatrix}$$

The vectors and matrices labeled in Equation 1 are described below in Table 1.

Table 1 Homogeneous Transformation definitions

${}^A\mathbf{r}_{P/A_o}$	Vector (3x1) to point P from origin of A-frame described in A-frame
${}^A_B R$	Rotation matrix (3x3) from B to A (B basis vectors expressed in A)
${}^A\mathbf{r}_{B_o/A_o}$	Vector (3x1) from A-origin to B-origin
${}^B\mathbf{r}_{P/B_o}$	Vector(3x1) from B-origin to the point P

In addition to combining the translation and rotation into a single matrix, the homogenous matrices for the individual joints can be successively post multiplied to develop a single homogenous transformation that will define the composition of multiple rotations from frame 0 to frame N or as shown in lesson 11.12:

Equation 2 Composition of homogeneous translations

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

However, as pointed out in the lesson, each transformation ${}^{N-1}_N T$ as currently described would require 6 independent parameters, meaning that for our KR210 analysis, we would need 30 parameters to describe the orientation of every joint. However, it is intuitively evident that the joints of our robot are not truly independent and are constrained by the axis which each joint can rotate about and the physical geometry of the links. For example, joint 2's position is constrained to a fixed circular path .75m above the base with a radius of .35m about the joint1 axis. Jacques Denavit and Richard Hartenberg leveraged these constraints to reduce the number of independent parameters per joint to 4, which are commonly referred to as DH parameters. Each DH parameter represents a geometric angle or distance between successive links with

the definition as follows and shown in Figure 3 from lesson 11-12:

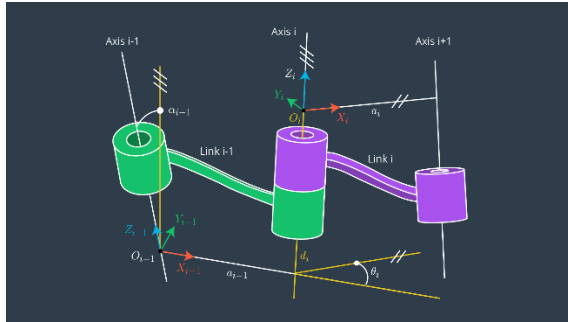


Figure 3 DH parameters with respect to a revolute joint

Table 2 DH Parameters and their definitions

DH Parameter	Description
α_{i-1} (twist angle)	Angle between z_{i-1} and z_i as measured about the x_{i-1} axis according the RH rule
a_{i-1} (link length)	Distance from z_{i-1} to z_i as measured along the x_{i-1} axis
d_i (link offset)	Signed distance from x_{i-1} to x_i measured along the z_i axis
θ_i (joint angle)	Angle between x_{i-1} and x_i measured about the z_i axis

It is worth noting that α , a , and d for the kr210 are all fixed (non-variable) parameters that are defined by the robot geometry. θ is the only variable DH parameter for this manipulator.

To determine the DH parameters for any robot the following steps should be followed as shown in Table 3. These steps were followed for the kr210 utilizing the parameters listed in the urdf file found in [the project repository](#). Most of the DH parameters were trivial to map to the associated values in the urdf with 3 exceptions. The first being that at joint 2, when θ_2 is 0, the link is physically rotated at $-\pi/2$ radians. Secondly, the axis for joints 4, 5 and 6 are coincident at joint 5 for the DH kinematic analysis but physically separate, making d_4 a sum of two urdf values for separate links (.96 + .54). Lastly, the urdf file contains multiple entries for end effectors including a vacuum and gripper style ee. It is important to realize this and choose the parameters for the gripper style end effector.

Table 3 Steps to identify DH parameters (Duplicated from Lesson 11-13)

1. Label all joints from $\{1, 2, \dots, n\}$.
2. Label all links from $\{0, 1, \dots, n\}$ starting with the fixed base link as 0.
3. Draw lines through all joints, defining the joint axes.
4. Assign the Z-axis of each frame to point along its joint axis.
5. Identify the common normal between each frame Z^{i-1} and frame Z^i .
6. The endpoints of "intermediate links" (i.e., not the base link or the end effector) are associated with two joint axes, $\{i\}$ and $\{i+1\}$. For i from 1 to $n-1$, assign the X^i to be ...
 - a. For skew axes, along the normal between Z^i and Z^{i+1} and pointing from $\{i\}$ to $\{i+1\}$.
 - b. For intersecting axes, normal to the plane containing Z^i and Z^{i+1} .
 - c. For parallel or coincident axes, the assignment is arbitrary; look for ways to make other DH parameters equal to zero.
7. For the base link, always choose frame $\{0\}$ to be coincident with frame $\{1\}$ when the first joint variable (θ_1 or d_1) is equal to zero. This will guarantee that $\alpha_0 = a_0 = 0$, and, if joint 1 is a revolute, $d_1 = 0$. If joint 1 is prismatic, then $\theta_1 = 0$.
8. For the end effector frame, if joint n is revolute, choose X_n to be in the direction of X_{n-1} when $\theta_n = 0$ and the origin of frame $\{n\}$ such that $d_n = 0$.

A hand diagram was made of the robot per the instructions above and used to derive the DH parameters for the kr210 in Table 4. The hand drawing follows in Figure 4.

Table 4 DH parameters for kr210

	α_{i-1}	a_{i-1}	d_i	θ_i
T_1^0	0.	0.	.75	θ_1
T_2^1	$-\pi/2$.35	0.	$\theta_2 - \pi/2$
T_3^2	0.	1.25	0.	θ_3
T_4^3	$-\pi/2$	-.054	1.5	θ_4
T_5^4	$-\pi/2$	0.	0.	θ_5
T_6^5	$-\pi/2$	0.	0.	θ_6
T_7^6	0.	0.	.303	0

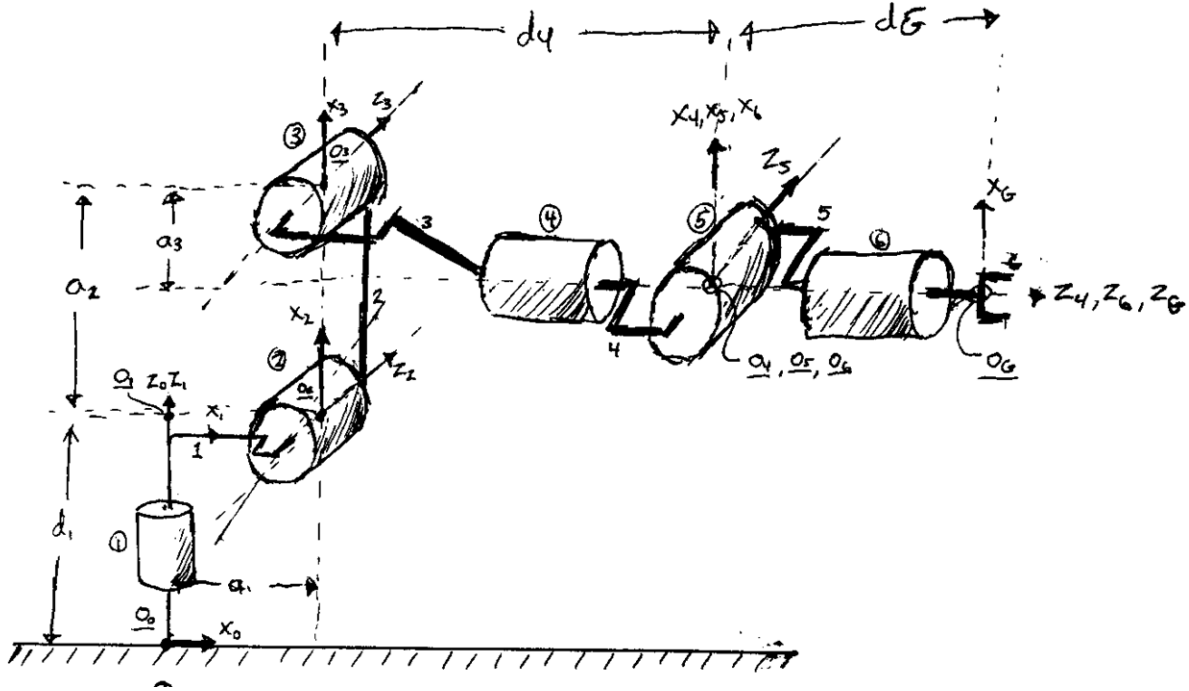


Figure 4 Illustration of kr210 with joint axis and DH parameters indicated after following steps 1-8. Note that this drawing indicates the gripper axis as identified by subscript 6 but subscript 7 was selected to identify the gripper for all code and other references.

Once the DH parameters have been determined, work can commence developing the individual DH transforms between each joint. The DH transformation matrix itself is just the composition of 4 elementary rotations and translations that when post multiplied result in the generalized DH homogenous transform shown in Equation 3.

Equation 3 the four basic rotations whose composition results in the DH Transformation matrix shown below.

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

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

Substituting for our DH parameters determined previously yields the individual transforms for each i-1 to i joint. As expected, the only variables present are the θ terms (represented by the 'q's).

Table 5 DH transformation matrices at each joint

T_1^0	$\begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0.0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
T_2^1	$\begin{bmatrix} \cos(q_2 - 0.5\pi) & -\sin(q_2 - 0.5\pi) & 0 & 0.35 \\ 0 & 0 & 1 & 0.0 \\ -\sin(q_2 - 0.5\pi) & -\cos(q_2 - 0.5\pi) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
T_3^2	$\begin{bmatrix} \cos(q_3) & -\sin(q_3) & 0 & 1.25 \\ \sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
T_4^3	$\begin{bmatrix} \cos(q_4) & -\sin(q_4) & 0 & -0.054 \\ 0 & 0 & 1 & 1.5 \\ -\sin(q_4) & -\cos(q_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
T_5^4	$\begin{bmatrix} \cos(q_5) & -\sin(q_5) & 0 & 0.0 \\ 0 & 0 & -1 & 0 \\ \sin(q_5) & \cos(q_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
T_6^5	$\begin{bmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0.0 \\ 0 & 0 & 1 & 0.0 \\ -\sin(q_6) & -\cos(q_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
T_7^6	$\begin{bmatrix} 1 & 0 & 0 & 0.0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.303 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Lastly, we can derive a single transformation, T_7^0 , from the composition of these individual transformations. This equation represent the solution to the forward kinematics problem but It is too unwieldy to display in its entirety here, but with the definition below it is easily recreated:

Equation 4 Definition of the transformation from base frame to end effector

$$T_7^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6$$

While T_7^0 represents the theoretical solution to the problem with respect to the hand drawn model in Figure 4 it will not produce results that match with GViz or Gazebo. The reason being that the manipulators urdf file defines the axis in a different orientation. To correct this, and get the urdf / GViz / Gazebo coordinates sytem to align with the results of our forward kinematic solver, a correctional frame rotation needs to be performed of π radians about the z axis, followed by a rotation of $-\pi/2$ radians about the y axis. The symbolic matrix representing the correctional Rotation transform (cRot) is shown below:

$$\begin{bmatrix} \cos(\theta_y) \cos(\theta_z) & -\sin(\theta_z) & \sin(\theta_y) \cos(\theta_z) & 0 \\ \sin(\theta_z) \cos(\theta_y) & \cos(\theta_z) & \sin(\theta_y) \sin(\theta_z) & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By substituting in the rotational corrections we now have the final computed form of our correction transform.

Equation 5 Rotation correction homogenous transform

$$T_{cRot} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 6 Forward kinematic solution (specific to this project) for the kr210

$$T_7^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6 T_{cRot}$$

Using Equation 6 several test cases were run to check the validity and error of the approach:


Test Cases for Forward Kinematics:

Table 6 Forward Kinematics Test case 1, zero / initial manipulator configuration

Variables	Expected Result	Result
$\theta_1 = 0$	Translation: [2.153, 0, 1.946]	$\begin{bmatrix} 1 & 0 & 0 & 2.153 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1.946 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$\theta_2 = 0$	Rotation:	
$\theta_3 = 0$	No Rotation (3x3 identity)	
$\theta_4 = 0$	Source: urdf file / geometry of manipulator	
$\theta_5 = 0$		Error (X,Y,Z) (0.000, 0.000,0.000)
$\theta_6 = 0$		Error (R,P,Y) (0,0,0)

Table 7 Forward Kinematics Test Case 2, random manipulator configuration

Variables	Expected Result
$\theta_1 = 0.58$ $\theta_2 = -0.56$ $\theta_3 = -1.84$ $\theta_4 = -4.64$ $\theta_5 = 1.11$ $\theta_6 = -5.99$	Translation(x,y,z): [-1.404, -.0593, 2.935] Rotation (r,p,y): [-0.574, -0.254 ,2.539] Source: tf_echo base_link gripper_link

Position Result (X,Y,Z) [-1.400, -0.594, 2.939] Position Error(abs) (0.004, 0.001, 0.004) Rotation Result (R,P,Y) (-0.570, -0.255, 2.545) Rotation Error(abs) (0.004, 0.001,0.006)	
---	---

Inverse Kinematics

As stated previously, the inverse kinematics problem is less tractable due to there being no closed form solution to the problem that will deliver the desired end effector pose.

Fortunately under certain criteria the problem can be simplified and decomposed into three smaller problems given *one* of the following is true, as seen in lecture notes 11-18:

1. Three neighboring joint axes intersect at a single point.
-- or --
2. Three neighboring joint axes are parallel.

In the case of the kr210, the first condition is satisfied as can be seen in Figure 4 where the axis for joint 4, joint 5, and joint 6 all intersect. This location on the kr210 is functionally analogous to a human wrist and as such is called the spherical wrist of the robot. The location where the 3 axis intersect, in this case at joint 5, is called the wrist center (wc). With this condition met, the inverse kinematics problem can be broken into three separate tasks given a desired pose for the end effector:

1. **Determine the position of the wrist center** using the given pose of the ee and the known geometry of the manipulator.
2. **Determine the angles theta1, theta2, theta3** at the first three joints that will locate the wrist center in the location found in step 1.
3. **Determine theta4, theta5, theta6** by leveraging Equation 7 and solving for individual Euler angles:

Equation 7 Matrix equation for deriving the Euler angle equations at the wrist center for joints 4, 5 and 6.

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

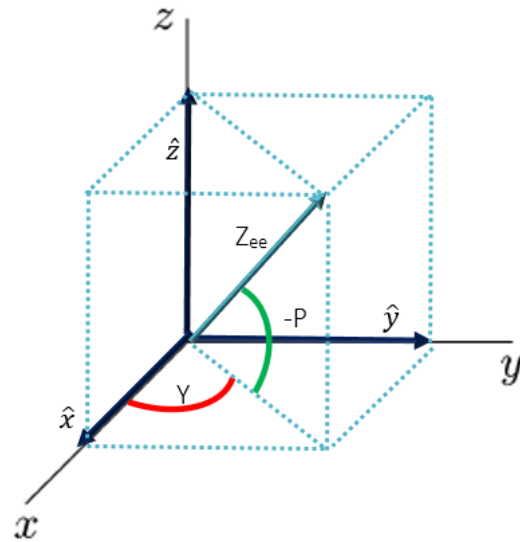
Determining the wrist center

The approach taken to find the wrist center was a purely geometric / trigonometric one rather

than what was presented in the lecture although the end result is the same. The approach consisted of:

1. Determining the unit vectors of the local Z_{ee} axis of the end effector expressed in the base frame using given yaw and pitch.
2. Then multiply the resulting unit vector by d_7 , the distance from ee to the wrist center to get the vector from wrist center to ee in world coordinates.
3. Subtract the vector in step 2 from the given ee position vector to obtain the wc vector in world coordinates.

Determining the unit vectors of Z_{ee} in world frame unit vectors:



From FIGURE we can derive the following:

$$\cos Y = \frac{\hat{x}}{\cos P} \xrightarrow{\text{yields}} \hat{x} = \cos Y \cos P$$

$$\sin Y = \frac{\hat{y}}{\cos P} \xrightarrow{\text{yields}} \hat{y} = \sin Y \cos P$$

$$\hat{z} = \sin(-P) \xrightarrow{\text{yields}} \hat{z} = -\sin(P)$$

Equation 8 follows from the above results and was used to determine wc . Note that since it was approached purely geometrically, and the analysis already references the base frame, there is no need to add an additional correction.

Equation 8 Equation for wc in base frame position given ee position

$$\overline{wc} = \overline{ee} - d_7 \begin{bmatrix} \cos Y \cos P \\ \sin Y \cos P \\ -\sin P \end{bmatrix}$$

Table 8 Test case for wc location, given random ee pose.

Variables	Expected Result	Result
$P = -0.302$ $Y = 1.724$ $\overline{wc} =$ $\begin{bmatrix} -1.571 \\ -.055539 \\ 3.1283 \end{bmatrix}$	Translation: $[-1.5268, 0.84125, 3.0381]$ Source: Rviz link 5 position	Result (X,Y,Z) : $[-1.52685, -0.84129, 3.0382]$ Error (X,Y,Z) $(.0001, .00004, .0001)$

Determining $\Theta_1, \Theta_2, \Theta_3$

The second task for the IK problem is determining the angles for joint 1, 2, 3 of the manipulator such that the wrist center is placed at the location determined in the previous step. For the kr210 we can determine closed form solutions for these angles using geometry, and the law of cosines. Θ_1 is the easiest to determine of the three as it just requires simple trigonometry. As is seen from Figure 5

$$\tan \theta_1 = \frac{wc_y}{wc_x}$$

Θ_2 and Θ_3 are somewhat less tractable but can be determined with the use of the law of cosines.

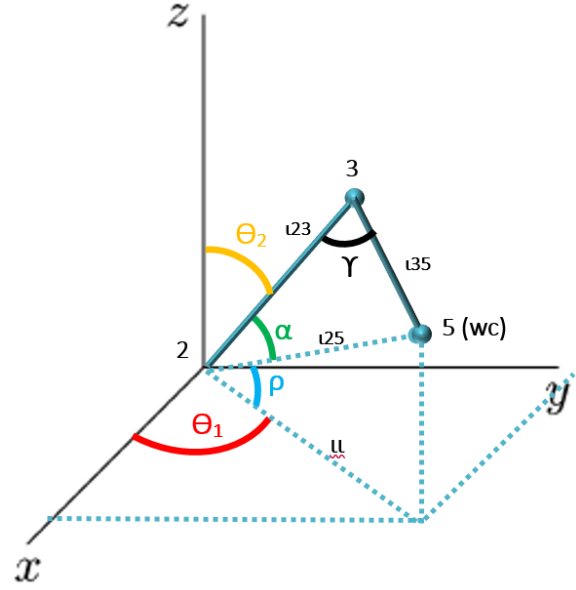


Figure 5 Diagram to aid in visualizing $\Theta_1, \Theta_2, \Theta_3$ derivation. Note that alpha here has no relation to the alpha shown in the DH parameters.

To find Θ_2 we begin by solving for the legs of the triangle formed by joints 2, 3, 5 as well as the projection of that triangle onto the xy plane shown. From modified DH parameter table we can see that:

$$l_{23} = [a_2]$$

$$l_{35} = \sqrt{a_3^2 + d_4^2}$$

l_{25} can be thought of as the subtraction of the joint 2 position vector from the wc position vector (both in base coordinates)

$$\overline{l_{25}} = \overline{wc} - \overline{joint2}$$

$$\overline{l_{25}} = \overline{wc} - \begin{bmatrix} a_1 \cos \theta_1 \\ a_1 \sin \theta_1 \\ d_1 \end{bmatrix}$$

Similarly, the length of u shown projected on the xy plane can be thought of as the subtraction of the joint 2 position from the wc position, but with the z coordinate set to d_1 , since the xy plane is normal to X_2 at the joint origin O_2 as shown in Figure 4.

$$\overline{u} = \begin{bmatrix} wc_x \\ wc_y \\ d_1 \end{bmatrix} - \begin{bmatrix} a_1 \cos \theta_1 \\ a_1 \sin \theta_1 \\ d_1 \end{bmatrix}$$

Now that all necessary lengths are in place, the law of cosines allows determination of ρ and α .

Beginning with the definition of tangent and Pythagorean identity:

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha} \text{ and } \sin^2 \alpha + \cos^2 \alpha = 1$$

$\xrightarrow{\text{yields}}$

$$\tan \alpha = \frac{\sqrt{1 - \cos^2 \alpha}}{\cos \alpha}$$

Then using the law of cosines to determine $\cos \alpha$:

$$\cos \alpha = \frac{l_{35}^2 - l_{23}^2 - l_{25}^2}{-2 l_{23} l_{25}}$$

The combined single equation is not shown but in the code, $\cos \alpha$ is found first then substituted into the equation for $\tan \alpha$.

ρ can be determined by simple application of the definition of the tangent function, and leveraging the u vector determined previously:

$$\tan \rho = \frac{wc_z - d_1}{|ll|}$$

With both ρ and α defined, finding Θ_2 can be determined as follows:

$$\theta_2 = \frac{\pi}{2} - \alpha - \rho$$

Determining Θ_3 proceeds in a similar fashion, but is complicated by the fact that there is an offset in the link that lies between joint 3 and joint 4; indicated by a_3 in Figure 4. Figure 6 shows this in detail, isolating link 2 and 3 for the analysis. From the illustration it can be seen that solutions for the ϕ and γ angles will need to be determined to solve for Θ_3 as:

$$\theta_3 = \frac{\pi}{2} - \gamma - \phi$$

ϕ is relatively easy to determine owing to the fact that it is simply a function of the geometry of link 3. The one nuance being that

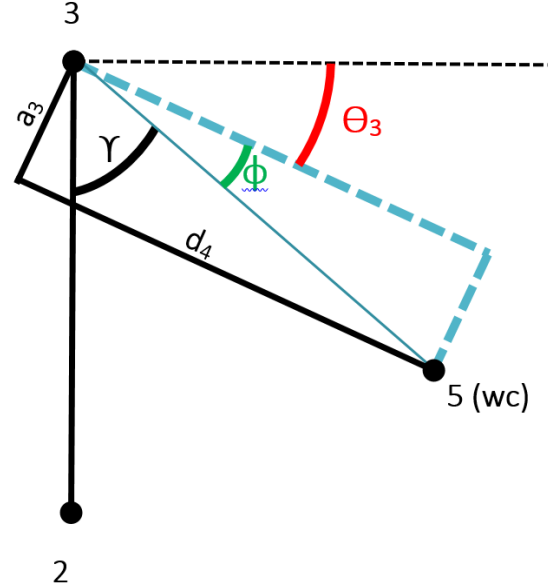


Figure 6 Illustration of link2 and link3 for the process for determining Θ_3

a_3 is negative in the DH table, but for the purposes of determining ϕ we either need to correct for that fact, or use the abs function. A correction was chosen here, hence the leading (-) sign.

$$\tan \phi = -\frac{a_3}{d_4}$$

γ is the same γ from Figure 5, and can be determined again using the definition of the tangent, the Pythagorean identity, and law of cosines:

$$\cos \gamma = \frac{l_{25}^2 - l_{35}^2 - l_{23}^2}{-2 l_{35} l_{23}}$$

$$\tan \gamma = \frac{\sqrt{1 - \cos^2 \gamma}}{\cos \gamma}$$

Determining $\Theta_4, \Theta_5, \Theta_6$

The last 3 angles are determined by leveraging the knowledge that the 3 joints at the wrist center are used to set orientation of the ee and that the provided yaw, pitch and roll correspond to rotations about the z, y and x axis at the wrist (R_{rpy}). Furthermore, the forward kinematics *rotation* from joint 1 -> joint 6 should equal R_{rpy} . This is the fundamental

concept illustrated in Equation 7 which is then used to solve for the individual angles.

$$R_{06} = R_{rpy}$$

However, R_{06} contains all Θ angles, and for our purposes here we are only interested explicit solutions for $\Theta_4, \Theta_5, \Theta_6$. From the project notes, section 15 we see this can be done by pre-multiplying both sides by $(R_{03})^{-1}$.

This leaves R_{36} on the left hand side of the equation containing only trigonometric functions for the variables $\Theta_4, \Theta_5, \Theta_6$.

$$R_{36} = (R_{03})^{-1} R_{rpy}$$

However we know from lecture 11.8, that given the orthonormality of rotation matrices the inverse of a 3x3 rotation is equal to its transpose giving:

Equation 9 Equation for determining $\Theta_4, \Theta_5, \Theta_6$ (restated from Equation 7)

$$R_{36} = (R_{03})^T R_{rpy}$$

Yielding an operation that is much more advantageous to solve both manually and computationally as solving for the transpose is a trivial exercise when compared with inverting a matrix.

The rotation matrix for R_{rpy} is defined as shown in Equation 10.

Equation 10 Rrpy definition

$$R_{rpy} = Zrot(\theta_z = y) Xrot(\theta_y = p) Yrot(\theta_x = r) R_{cRot}$$

where: $y = \text{yaw}, p = \text{pitch}, r = \text{roll}$

$(R_{03})^T$ can be determined by using our homogeneous transformations, extracting only the transformed rotation portion:

$$(R_{03})^T = ([T_1^0 \ T_2^1 \ T_3^2][1:3,1:3])^T$$

Equation 11 symbolically evaluated R36 matrix for LHS of Equation 9. Note that $q_4, q_5, q_6 = \Theta_4, \Theta_5, \Theta_6$.

$$\begin{bmatrix} -\sin(q_4)\sin(q_6) + \cos(q_4)\cos(q_5)\cos(q_6) & -\sin(q_4)\cos(q_6) + \sin(q_6)\cos(q_4)\cos(q_5) & -\sin(q_5)\cos(q_4) \\ \sin(q_5)\cos(q_6) & -\sin(q_5)\sin(q_6) & \cos(q_5) \\ -\sin(q_4)\cos(q_5)\cos(q_6) + \sin(q_6)\cos(q_4) & \sin(q_4)\sin(q_6)\cos(q_5) - \cos(q_4)\cos(q_6) & \sin(q_4)\sin(q_5) \end{bmatrix}$$

Note that in Equation 10 the rotation (rot) matrices are our elementary rotations arranged to perform an intrinsic rotation and R_{cRot} is the 3x3 rotation matrix from our correctional transformation defined earlier in Equation 5.

$$\begin{aligned} Xrot &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \\ Yrot &= \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \\ Zrot &= \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Table 9 Elementary rotation matrices

The left side of Equation 9, R_{36} can be determined using our homogeneous transformations from Table 1 and extracting only the rotation portion of the rotation. Which yields the following for the left hand side [LHS] of Equation 9:

$$R_{36} = [T_4^3 \ T_5^4 \ T_6^5][1:3,1:3]$$

By substituting our $\Theta_1, \Theta_2, \Theta_3$ values previously calculated into our $(R_{03})^T$ term and multiplying it with our R_{rpy} rotation, the right hand side [RHS] of Equation 9 becomes entirely numerical with no variables. The trigonometric functions of $\Theta_4, \Theta_5, \Theta_6$ the left hand side of the equation can now be compared element-wise to their corresponding elements in the **RHS** matrix to develop closed form solutions.

Evaluation of Equation 10 yields the following closed form solutions for $\Theta_4, \Theta_5, \Theta_6$ shown in Table 10.

Table 10 Closed form solutions for $\Theta_4, \Theta_5, \Theta_6$

$\tan \theta_4 = \frac{LHS[3,3]}{-LHS[1,3]} = \frac{\sin \theta_4 \sin \theta_5}{\sin \theta_5 \cos \theta_4} = \frac{\sin \theta_4}{\cos \theta_4} = \frac{RHS[3,3]}{-RHS[1,3]}$
$\tan \theta_5 = \frac{\sqrt{LHS[3,3]^2 + LHS[1,3]^2}}{LHS[2,3]} = \frac{\sqrt{(\sin \theta_4 \sin \theta_5)^2 + (\sin \theta_5 \cos \theta_4)^2}}{\cos \theta_5} = \frac{\sin \theta_5}{\cos \theta_5} = \frac{\sqrt{RHS[3,3]^2 + RHS[1,3]^2}}{RHS[2,3]}$
$\tan \theta_6 = \frac{-LHS[2,2]}{LHS[2,1]} = \frac{\sin \theta_5 \sin \theta_6}{\sin \theta_5 \cos \theta_6} = \frac{\sin \theta_6}{\cos \theta_6} = \frac{-RHS[2,2]}{RHS[2,1]}$

Test Cases for Inverse Kinematics:

Table 11 Inverse kinematic test case #1: Zero configuration

Given Pose	Calculated Joint Angles
eePos: [2.153, 0, 1.946] Rpy: [0., 0., 0.]	$\theta_1 = 0$ $\theta_2 = 6e-17$ $\theta_3 = 2e-16$ $\theta_4 = 3.14159$ $\theta_5 = 1e-16$ $\theta_6 = -3.14159$
Position Result (X,Y,Z) [2.153, 4e-33, 1.946] Position Error(abs) (0.000, 4e-33, 0.000) Source: urdf file / manipulator geometry	

Table 12 Inverse kinematic test case #2: Random configuration

Given Pose	Calculated Joint Angles
eePos: [2.7584, -0.88758, 1.699] Rpy: [-0.053, -0.021, 0.084]	$\theta_1 = -0.35584$ $\theta_2 = 0.66398$ $\theta_3 = -0.67212$ $\theta_4 = 1.60282$ $\theta_5 = 0.43998$ $\theta_6 = -1.64931$
Position Result (X,Y,Z) [2.7584, -0.88758, 1.699] Position Error(abs) (0.000, 0.0000, 0.000) Source: T0_7 forward kinematics transform	

Simulated environment results

The methods described previously were implemented in the provided IK_server.py script and tested in the Gazebo environment. The performance of the kinematics script was acceptable. On 11 successive plans, the script solved for pose 262 times with an average solve time of 0.109 seconds per solution. Maximum time was found to be 0.134 seconds per solution on the smallest pose set (10) likely due to the script overhead being relatively constant regardless of the size of the requested trajectory set. The gripper position was consistently accurate as expected from the previous test cases, only failing to pick up cans(positioned correctly, but gripper just slides past can when corrected) due to peculiarities of VM environment which were solved by modifying trajectory_sampler.cpp:

- The duration in line 327 from 1.0 to 2.0
- The gripper close joint positions on lines 532 & 533 from .02 to .03.

Lastly, the manipulator appeared to be making many redundant movements, especially about joint axis 4 and joint axis 5. While not a complete remedy, an attempt was made to prevent these extraneous movements, among the several tried, simply ignoring Θ_4 and Θ_5 by making them static following the first pose, up until the last 10 poses (i.e. pose 2 through n-10 have Θ_4 and Θ_5 equal to their initial values at pose 1 in a trajectory) appeared to improve much of the unnecessary movement happening mid trajectory. This solution is application A video of the manipulator performing movements of cans from the shelf to the bucket can be viewed at <https://youtu.be/YYKrgsCJ0mY> (Note that the video is sped up 4.5x).

Appendix

IK_server.py code

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

# Copyright (C) 2017 Electric Movement 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
from mpmath import *
from sympy import *

q1, q2, q3, q4, q5, q6, q7 = symbols('q1:8')
d1, d2, d3, d4, d5, d6, d7 = symbols('d1:8')
a0, a1, a2, a3, a4, a5, a6 = symbols('a0:7')
alpha0, alpha1, alpha2, alpha3, alpha4, alpha5, alpha6 = symbols('alpha0:7')
#
# Create Modified DH parameters
#
s = {alpha0: 0., a0: 0., d1: .75, q1: q1,
     alpha1:-pi/2., a1: .35, d2: 0., q2: q2-pi/2.,
     alpha2: 0., a2: 1.25, d3: 0., q3: q3,
     alpha3:-pi/2., a3:-.054, d4: 1.5, q4: q4,
     alpha4: pi/2., a4: 0., d5: 0., q5: q5,
     alpha5:-pi/2., a5: 0., d6: 0., q6: q6,
     alpha6: 0., a6: 0., d7: .303, q7: 0.}
#
# Define Modified DH Transformation matrix
#
### Since the DH matrix is symbolically the same for each Tn-1 -> Tn transform
### rather than explicitly express it each time, just create some generic symbols
### and an expression to represent the DH matrix. Then use that to create our
### individual transformation matrices via substitution:

#
# Create the generic DH parameter symbols:
#
theta, alpha, a, d = symbols('theta, alpha, a, d')
#
# Create the generalized DH matrix using generic symbols:
#
DH_Matrix = Matrix([
    [cos(theta), -sin(theta), 0, a],
    [sin(theta)*cos(alpha), cos(theta)*cos(alpha), -sin(alpha), -sin(alpha)*d],
    [sin(theta)*sin(alpha), cos(theta)*sin(alpha), cos(alpha), cos(alpha)*d],
    [0, 0, 0, 1]
])
#
# Create individual transformation matrices
# Define the elementary rotation matrices :
theta_x, theta_y, theta_z = symbols('theta_x, theta_y, theta_z')

xRot = Matrix([[1, 0, 0],
               [0, cos(theta_x), -sin(theta_x)],
               [0, sin(theta_x), cos(theta_x)]])

yRot = Matrix([[cos(theta_y), 0, sin(theta_y)],
               [0, 1, 0],
               [-sin(theta_y), 0, cos(theta_y)]])

zRot = Matrix([[cos(theta_z), -sin(theta_z), 0],
```

```

        [ sin(theta_z), cos(theta_z), 0 ],
        [ 0, 0, 1 ]])

# Create correction Rotation matrix
cRot = zRot * yRot
cRot = cRot.col_insert(3,Matrix([0,0,0]))
cRot = cRot.row_insert(3,Matrix([0,0,0,1]))
cRotn = cRot.subs({theta_y: -pi/2, theta_z:pi})

#
# Define the Homogeneous tranforms from each joint to the next
# and composition tranfforms from 0-->7 for forward kinematics
# solver and 0-->6 for use in inverse kinematics solver.
#
T0_1 = DH_Matrix.subs({theta: q1, alpha: alpha0, a: a0, d: d1})
T0_1 = T0_1.subs(s)
T1_2 = DH_Matrix.subs({theta: q2, alpha: alpha1, a: a1, d: d2})
T1_2 = T1_2.subs(s)
T2_3 = DH_Matrix.subs({theta: q3, alpha: alpha2, a: a2, d: d3})
T2_3 = T2_3.subs(s)
T3_4 = DH_Matrix.subs({theta: q4, alpha: alpha3, a: a3, d: d4})
T3_4 = T3_4.subs(s)
T4_5 = DH_Matrix.subs({theta: q5, alpha: alpha4, a: a4, d: d5})
T4_5 = T4_5.subs(s)
T5_6 = DH_Matrix.subs({theta: q6, alpha: alpha5, a: a5, d: d6})
T5_6 = T5_6.subs(s)
T6_7 = DH_Matrix.subs({theta: q7, alpha: alpha6, a: a6, d: d7})
T6_7 = T6_7.subs(s)
T0_6 = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6
T0_7 = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_7

# Define gamma and its symbols, which is the unit vector pointing
# from the wrist center to the ee. Will use it in our IK solver
ROLL,PITCH,YAW = symbols('ROLL PITCH YAW')
gamma = Matrix([cos(YAW)*cos(PITCH), sin(YAW)*cos(PITCH), -sin(PITCH)])

R0_3 = (T0_1*T1_2*T2_3)[0:3,0:3]
R0_3T = R0_3.T
Rrpy = zRot * yRot * xRot

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:

        #Moved FK and IK code out of this function as it does not need to run
        #every time a request is recieved, it just needs to run once

        # Initialize service response

        joint_trajectory_list = []
        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])

            ##Gripper Rotation
            rpy = {ROLL:roll, PITCH:pitch, YAW:yaw}
            gamma.subs(rpy)
            ##Gripper Position
            pos = Matrix([px, py, pz])

```

```

##Wrist center calculation
wc = pos - s[d7] * gamma.subs(rpy)

## Theta1 #####
thetal = atan2(wc[1],wc[0])
#-----

## Theta2 #####
## Calculate the l35, l23, l25 lengths, and V25 vector using wrist center - joint2 position,
## and then take it's norm to find length
l35 = sqrt(s[a3]*s[a3] + s[d4]*s[d4])
l23 = s[a2]
v25 = wc - Matrix([s[a1]*cos(thetal), s[a1]*sin(thetal), s[d1]])
l25 = sqrt(v25[0]*v25[0]+v25[1]*v25[1]+v25[2]*v25[2])

## NOW CALCULATE rho and alpha (from diagram 5) so that we can calculate theta2
## angle using the formula theta2 = pi/2-alpha-rho. Note that in the writeup for
## determining l1 the z terms were d1, but since they were both d1, I just use 0 below.
l1 = Matrix([wc[0],wc[1],0]) - s[a1] * Matrix([cos(thetal), sin(thetal),0])
rho = atan2(wc[2]-s[d1], sqrt(l1[0]*l1[0] + l1[1]*l1[1] + l1[2]*l1[2]))

# Determine cos_alpha and use it to derive alpha. Note that this alpha is
# NOT the same alpha used in the DH table.
cos_alpha = ((l35*l35 - l23*l23 - l25*l25) / (-2 * l23 * l25))
alpha = atan2(sqrt(1 - cos_alpha*cos_alpha), cos_alpha)

# Now that we have rho and alpha, we can calculate theta2
theta2 = pi/2 -alpha -rho
#-----

## Theta3 #####
# Now calculate gamma (The angle between l23 and l35)
cos_gamma = (l25*l25 - l35*l35 - l23*l23) / (-2 * l35 * l23)
# Then calculate theta3. The formula represented below is pi/2 - gamma - phi
theta3 = pi/2 - atan2(sqrt(1 - cos_gamma*cos_gamma), cos_gamma) + atan2(s[a3],s[d4])
#-----

## Spherical Wrist Euler Angles #####
R0_3Tn = R0_3T.subs({q1:thetal, q2:theta2, q3:theta3})
Rrpy = Rrpy.subs({theta_z: rpy[YAW], theta_y: rpy[PITCH], theta_x: rpy[ROLL]})
# Compensate for rotation discrepancy between DH parameters and Gazebo
Rrpy = Rrpy * cRotn[:3,:3]
RHS=(R0_3Tn*Rrpy)
theta4 = atan2(RHS[2,2],-RHS[0,2])
theta5 = atan2(sqrt(RHS[2,2]*RHS[2,2] + RHS[0,2]*RHS[0,2]), RHS[1,2])
theta6 = -atan2(RHS[1,1],RHS[1,0])

# Populate response for the IK request
# In the next line replace thetal,theta2...,theta6 by your joint angle variables
if x and (len(req.poses)-x)>4:
    if abs(theta6-prev_theta6)>1.and abs(theta4-prev_theta4)>1. and abs(theta6 + theta4) <
.1:
        print("Theta4 / Theta6 Cancel",theta4,theta6)
        theta4 = prev_theta4
        theta6 = prev_theta6
    prev_theta4, prev_theta6 = theta4,theta6
    joint_trajectory_point.positions = [thetal, theta2, theta3, theta4, theta5, theta6]
    joint_trajectory_list.append(joint_trajectory_point)

rospy.loginfo("length of Joint Trajectory List: %s" % len(joint_trajectory_list))
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()

```