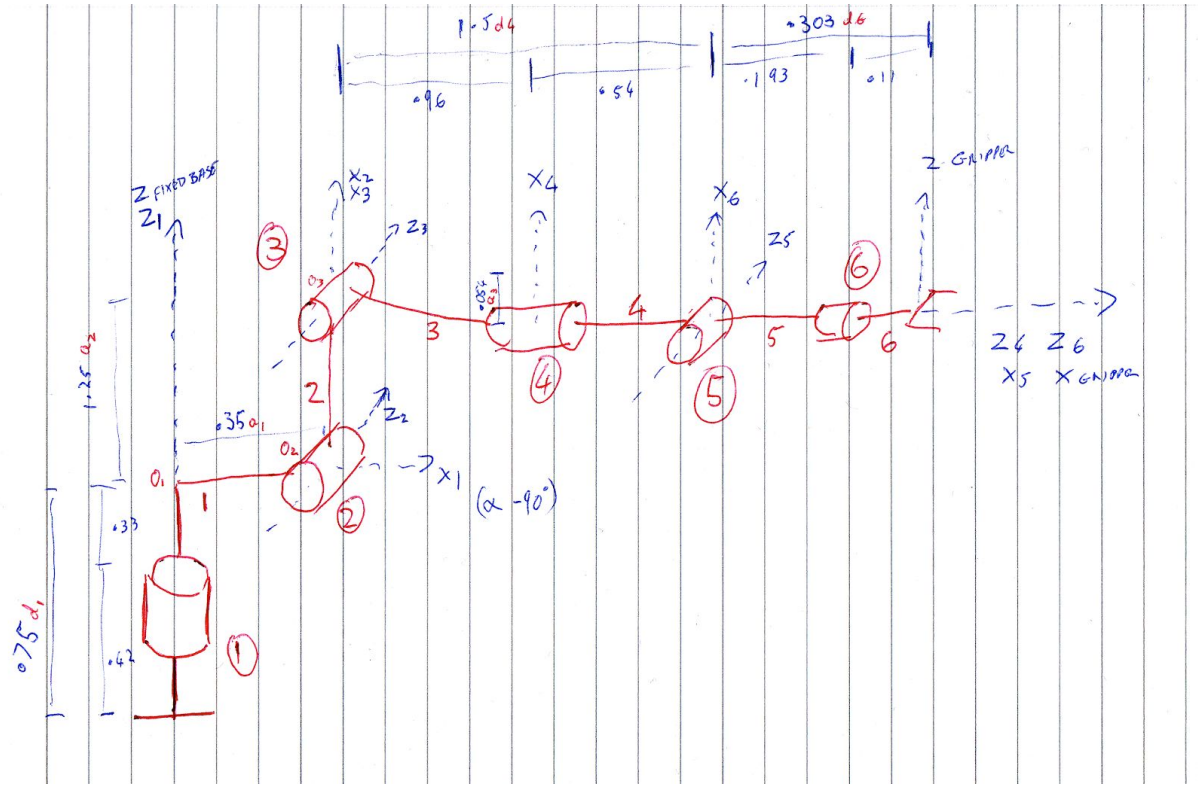
RoboND - Kinematics Project

Fergal Toohey

Kinematic Analysis

1. Run the forward_kinematics demo and evaluate the kr210.urdf.xacro file to perform kinematic analysis of Kuka KR210 robot and derive its DH parameters.



Links	α (i-1)	a(i-1)	d(i)	θ (i)
0 -> 1	0	0	.75	q1
1 -> 2	- π /2	0.35	0	- π /2 + q2
2 -> 3	0	1.25	0	q3
3 -> 4	- π /2	-.056	1.5	q4
4 -> 5	π /2	0	0	q5

5 -> 6	- π /2	0	0	q6
6 -> EE	0	0	.303	0

The DH table was created from the RViz model, and the URDF file, according to the modified DH parameter convention used for this course.

Adjustments of note from the URDF:
Origins of Joints 4, 5, and 6 were set to be coincident, at Joint 5, thereby facilitating a spherical wrist model.
The gripper link (6->EE) was set therefore set to be the full length from Joint 5 to the gripper (.303)
Origin of Joint 1 was moved up its Z axis, so that be level with Joint 1
The rotation at Joint 2 was adjusted by -90 degrees to bring it in line with the DH convention.

2. Using the DH parameter table you derived earlier, create individual transformation matrices about each joint. In addition, also generate a generalized homogeneous transform between base_link and gripper_link using only end-effector(gripper) pose.

The homogenous transform from frame *i*-1, to frame *i* under the DH convention is defined as:

$$
{}^{i-1}_{i}\mathrm{T} = \begin{bmatrix} cos(q_i) & -sin(q_i) & 0 & a_{i-1} \\ sin(q_i)cos(\alpha_{i-1}) & cos(q_i)cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & -sin(\alpha_{i-1})d_i \\ sin(q_i)sin(\alpha_{i-1}) & cos(q_i)sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Thus, the transformation matrixes for each individual joint, using the parameters from the DH table, are as follows:

$$
{}^{0}_{1}\mathrm{T} = \begin{bmatrix} cos(q_1) & -sin(q_1) & 0 & a_0 \\ sin(q_1)cos(\alpha_0) & cos(q_1)cos(\alpha_0) & -sin(\alpha_0) & -sin(\alpha_0)d_1 \\ sin(q_1)sin(\alpha_0) & cos(q_1)sin(\alpha_0) & cos(\alpha_0) & cos(\alpha_0)d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{1}_{2}\mathrm{T} = \begin{bmatrix} cos(q_2) & -sin(q_2) & 0 & a_1 \\ sin(q_2)cos(\alpha_1) & cos(q_2)cos(\alpha_1) & -sin(\alpha_1) & -sin(\alpha_1)d_2 \\ sin(q_2)sin(\alpha_1) & cos(q_2)sin(\alpha_1) & cos(\alpha_1) & cos(\alpha_1)d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{2}_{3}\mathrm{T} = \begin{bmatrix} cos(q_3) & -sin(q_3) & 0 & a_2 \\ sin(q_3)cos(\alpha_2) & cos(q_3)cos(\alpha_2) & -sin(\alpha_2) & -sin(\alpha_2)d_3 \\ sin(q_3)sin(\alpha_2) & cos(q_3)sin(\alpha_2) & cos(\alpha_2) & cos(\alpha_2)d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

...

$$
{}^{6}_{EE}\mathrm{T} = \begin{bmatrix} cos(q_7) & -sin(q_7) & 0 & a_6 \\ sin(q_7)cos(\alpha_6) & cos(q_7)cos(\alpha_6) & -sin(\alpha_6) & -sin(\alpha_6)d_7 \\ sin(q_7)sin(\alpha_6) & cos(q_7)sin(\alpha_6) & cos(\alpha_6) & cos(\alpha_6)d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Generalized homogenous transform

The generalized homogenous transform between the base pose and the End Effector pose is equal to the end effector roll, pitch, and yaw supplied by ROS (*r, p, y*), post-multiplied by correction rotations about the z axis (180 degrees) and y axis (-90 degrees), to remove the discrepancy between Gazebo and the DH parameters. This corrected rotation matrix was created in python. [In IK_server.py, it is the sympy matrix referred to as R_EE]

A homogenous transformation can then be derived by combining the corrected rotation matrix with the translation from End Effector position relative to the base (pos_{xyz}):

$$
{}^{0}_{EE}\mathrm{T} = \begin{bmatrix} s(p)c(r)c(y) + s(r)s(y) & -s(p)s(r)c(y) + s(y)c(r) & c(p)c(y) & pos_x \\ s(p)s(y)c(r) - s(r)c(y) & -s(p)s(r)s(y) - c(r)c(y) & s(y)c(p) & pos_y \\ c(p)c(r) & -s(r)c(p) & -s(p) & pos_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

3: Decouple Inverse Kinematics problem into Inverse Position Kinematics and inverse Orientation Kinematics; doing so derive the equations to calculate all individual joint angles.

The last three joints of the Kuka 210 define a Spherical Wrist, as they are revolute and their joint axes intersect at a single point, Joint 5, the Wrist Centre.

This means that the manipulator is solvable in closed form, and the position and orientation of the end effector are kinematically decoupled from one another, into the Inverse Position and Inverse Orientation problems, respectively.

This reduces the values that need to be determined for the last three joints:

- the coordinates of the Wrist Centre, and
- the composition of rotations required to orient the end effector.

Once the Wrist Centre is obtained, the angles of Joints 1-3 can be obtained by trigonometry.

Coordinates of the Wrist Centre:
We have already calculated the End Effector's corrected rotation, so to get the wrist position, we translate back from the End Effector along the End Effector's Z axis, by *d7*, to arrive at Joint 5, the Wrist Centre:

$$WC_x = pos_x - d_7 \cdot n_x$$
$$WC_y = pos_y - d_7 \cdot n_y$$
$$WC_z = pos_z - d_7 \cdot n_z$$

Where *pos* is the End Effector position, *d7* is the value from the DH table, and n_{xyz} is the Z component of the EE's corrected rotation.

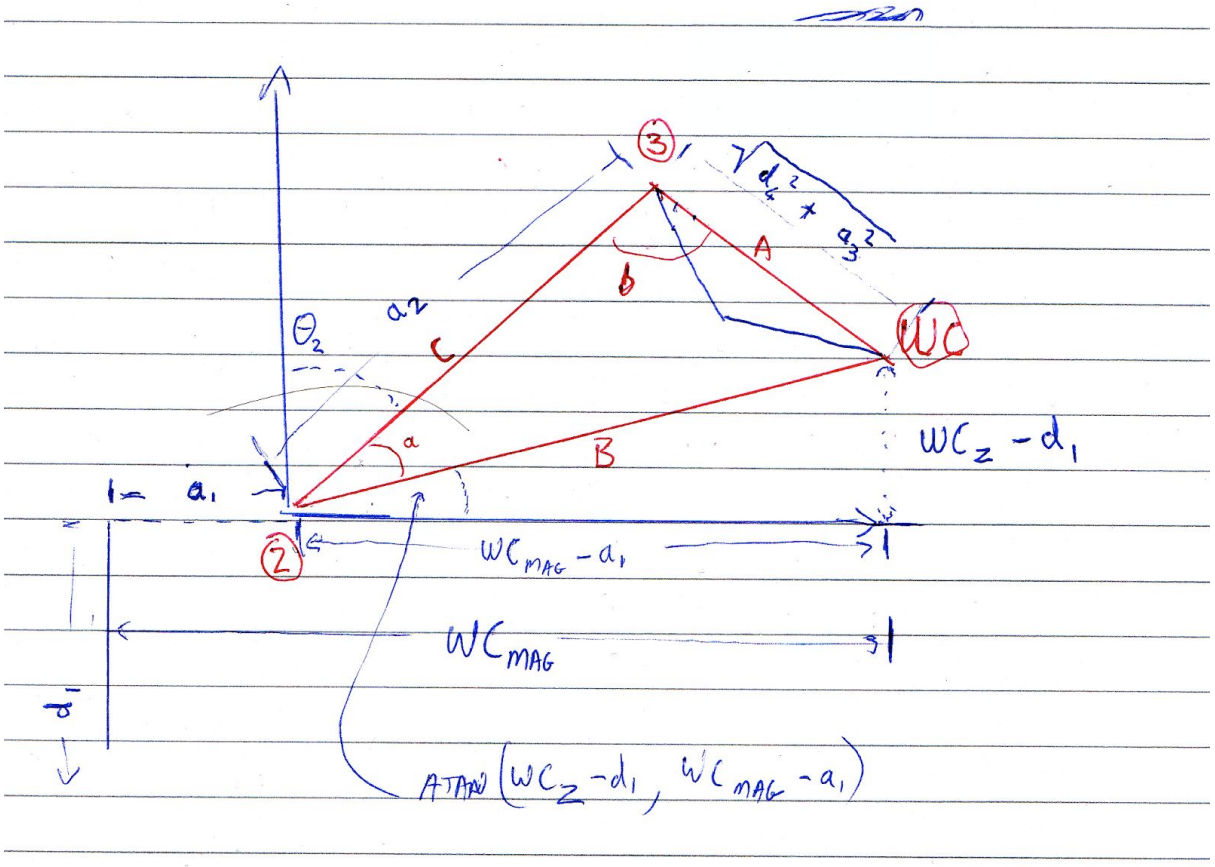
From this, the value for θ_1 can be directly deduced: θ_1 is the horizontal rotation of the base joint, and can be deduced by the XY component of the wrist centre:

$$\theta_1 = atan2(WC_y, WC_x)$$

The distance on the base link's XY plane, from the wrist centre to the origin is required to derive the other angles:

$$WC_{mag} = \sqrt{{WC_x}^2 \cdot {WC_y}^2}$$

Thus the positions of WC and Joint 2 are known, as are the lengths of the links between them. This means that θ_2 and θ_3 can be determined by analysing the triangle created by Joint 2, Joint 3, and the Wrist Centre, as shown below:.



Side A: Distance between Joint 3, and Wrist Centre:

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

Side B: Distance between Wrist Centre and Joint 2:

$$B = \sqrt{(WC_{mag} - a_1)^2 \cdot (WC_z - d_1)^2}$$

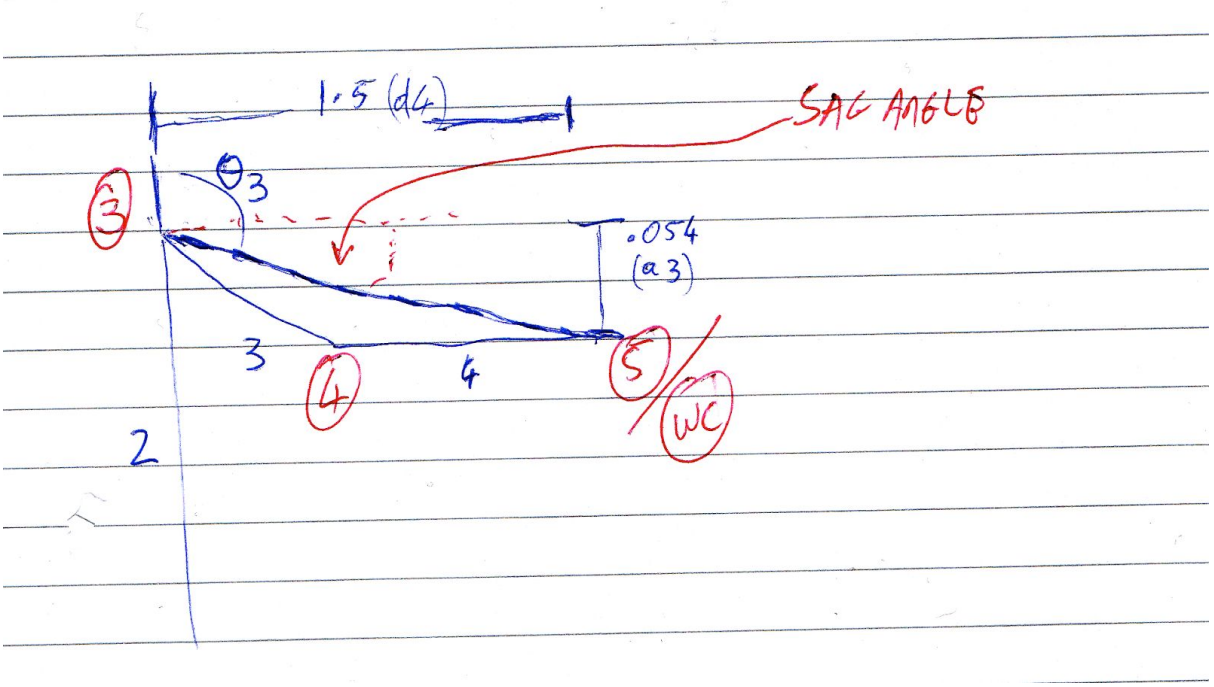
Side C: Distance between Joints 2 and 3:

$$C = a_2$$

As all sides are known, apply Cosine rules to determine angles

$$a = acos((B^2 + C^2 - A^2)/2BC)$$
$$b = acos((A^2 + C^2 - B^2)/2AC)$$

Account for Sag at between Joint 3 and the wrist centre:



$$sag = atan2(a_3, d_4)$$

Use these angles to determine thetas 2 and 3

$$\theta_2 = \pi/2 - a - atan2(WC_z - d_1, WC_{mag} - a_1)$$
$$\theta_3 = \pi/2 - sag - b$$

Composition of extracted rotation matrices for rotations 0 -> 3

$${}^{0}_{3}R = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T$$

Get rotation of spherical wrist joints by applying end effector rotation matrix to inverse of R0_3

$${}^{3}_{6}R = {}^{0}_{3}R^{-1} \cdot {}^{0}_{EE}R$$

Extract euler angles:

To identify the symbolic relationship of the Euler angles, R3_EE was calculated symbolically in Python.

$${}^{3}_{EE}R = {}^{3}_{4}R \cdot {}^{4}_{5}R \cdot {}^{5}_{6}R \cdot {}^{6}_{EE}R$$

Producing the following matrix:

$$
{}^{3}_{EE}R = \begin{bmatrix} -s(\theta_4)s(\theta_6) + c(\theta_4)c(\theta_5)c(\theta_6) & -s(\theta_4)c(\theta_6) - s(\theta_6)c(\theta_4)c(\theta_5) & -s(\theta_5)c(\theta_4) \\ s(\theta_5)c(\theta_6) & -s(\theta_5)s(\theta_6) & c(\theta_5) \\ -s(\theta_4)c(\theta_5)c(\theta_6) - s(\theta_6)c(\theta_4) & s(\theta_4)s(\theta_6)c(\theta_5) - c(\theta_4)c(\theta_6) & s(\theta_4)s(\theta_5) \end{bmatrix}
$$

Which can be solved for $\theta_{4,5,6}$ using arctan2:

$$
\theta_4 = atan2(r_{3,3}, r_{1,3})
$$
$$
\theta_5 = atan2(\sqrt{{r_{1,3}}^2 + {r_{3,3}}^2}, r_{2,3})
$$
$$
\theta_6 = atan2(r_{2,2}, r_{2,1})
$$

Project Implementation

Detailed discussion of the implementation can be found in the comments of IK_server.py. The implementation essentially mirrors the maths outlined above. The code was first implemented in IK_debug.py, and relevant code transferred to IK_server.py on completion. Irrelevant code for the final code (such as that used to generate the symbolic representation of R3_EE discussed above) is omitted.

One difference from the suggested code in the lessons is R0_3.transpose() instead of R0_3.inv('LU'). As pointed out by other students in Slack, .inv('LU') leads to unstable results in this scenario, often leading to a ~.30 offset on EE position. On replacing this code with transpose(), which in this instance is equivalent, EE offset stays robustly at 0, as shown in the Test Case 1 output:

> Wrist error for x position is: 0.00000046
> Wrist error for y position is: 0.00000032
> Wrist error for z position is: 0.00000545
> Overall wrist offset is: 0.00000548 units
>
> Theta 1 error is: 0.00093770
> Theta 2 error is: 0.00178633
> Theta 3 error is: 0.00206506
> Theta 4 error is: 0.00172809
> Theta 5 error is: 0.00198404
> Theta 6 error is: 0.00252923
>
> End effector error for x position is: 0.00000000
> End effector error for y position is: 0.00000000
> End effector error for z position is: 0.00000000
> Overall end effector offset is: 0.00000000 units

The Robot successfully picks up the object from all tested positions, generating sensible paths. However, calculating the inverse kinematics is noticeably slow, with each step taking many seconds. Measures were taken to speed up this calculation, by removing the simplify() operations, which reduced calculation time by roughly 50%, however further work is needed, particularly to analyse the order of the Sympy operations, to maximise the amount performed outside the main loop.

The robot also has a tendency to apply a seemingly unnecessary 360 degree rotation to the gripper when making even the slightest adjustments, such as reaching for the middle centre object from the start point. Further analysis is warranted to identify if this is the nature of the robot, or a quirk of this Euler angle implementation.