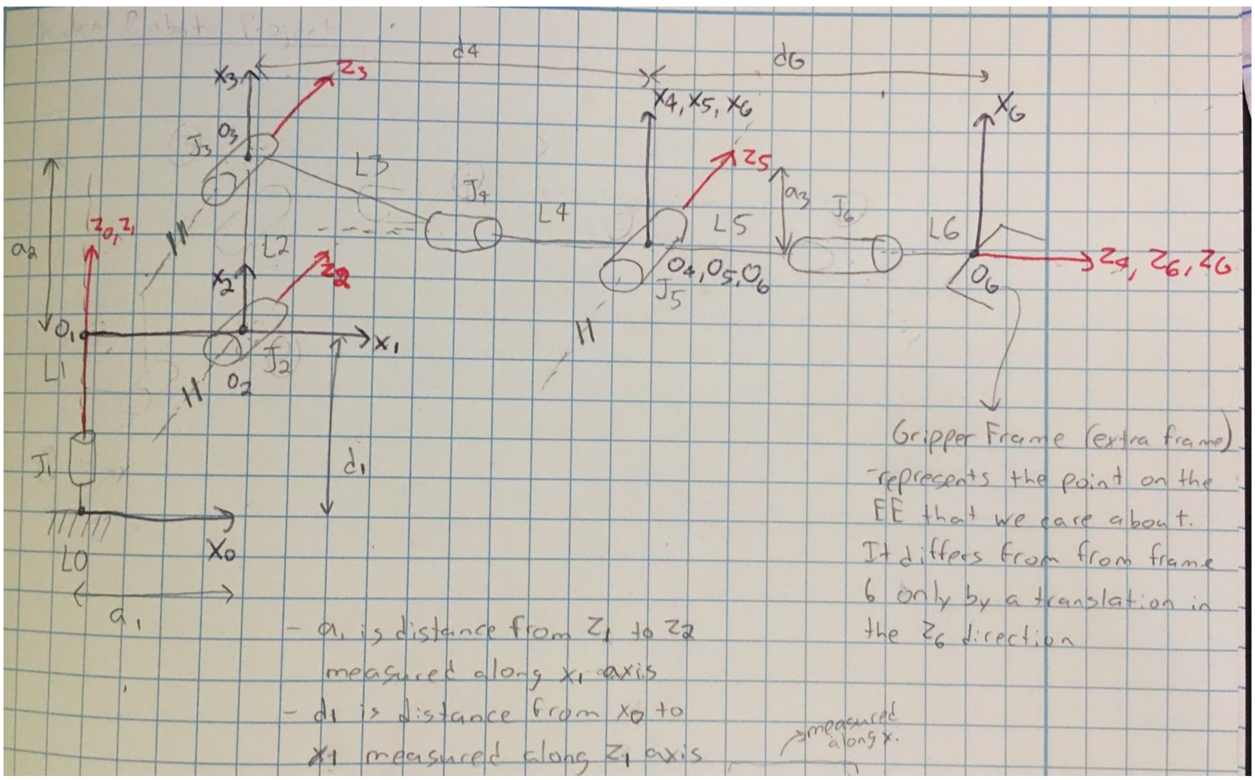
Project 2 - Kuka KR210 Arm

Forward Kinematics

The forward kinematics problem is to determine the position and orientation of the end-effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints. We can solve the forward kinematics problem by using the Denavit Hartenberg (DH) approach. In this, rather than finding 6 unknown parameters (x, y, z, yaw, pitch, roll), we only need to find 4 parameters (a_i link length, $\boldsymbol{\alpha}_i$ link twist, d_i link offset, and Θ_i joint angle). We only need 4 parameters because have freedom in choosing the origin and the coordinate axes of the frame, so by choosing clever placements of each frame, we eliminate 2 extra parameters. The following diagram shows the 6 DOF Kuka KR210 arm.



The first step is to draw out the six revolute joints and label them J1-6. Next draw in the axis of rotation which is represented by the dashed line. Next draw the z axis along this axis of rotation for each joint. Here, Z0 and Z1 axis coincide, then Z2, Z3, and Z5 are parallel, and Z4, Z6, and ZG coincide. ZG is the z axis for the gripper frame which represents the point on the end effector that we wish to move. Next, I drew in the x axis using the right hand rule. In this diagram, X4, X5, and X6 share the same x axis location. To make our analysis easier for the inverse kinematics portion of the project, we require this setup for the spherical wrist (more discussed later). Lastly, I label in a1, a2, a3, d4, and dG lengths. From this diagram, I get the following DH parameters table

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	0	0	0.33 + 0.42 = 0.75	Θ_1
2	-90	0.35	0	Θ_2-90°
3	0	1.25	0	Θ_3
4	-90	-0.054	0.96 + 0.54 = 1.5	Θ_4
5	90	0	0	Θ_5
6	-90	0	0	Θ_6
G	0	0	0.193 + 0.11= 0.303	0

α_{i-1} → angle between Z_{i-1} and Z_i measured about X_{i-1}
a_{i-1} → distance from Z_{i-1} to Z_i measured along X_{i-1} where X_{i-1} is perpendicular to both Z_{i-1} to Z_i
d_i → signed distance from X_{i-1} to X_i measured along Z_i
Θ_i → angle between X_{i-1} to X_i measured about Z_i

The values for the DH table were obtained from the URDF provided. In the URDF file, it outlines all joint information (lengths, joint type, parent name, child name, etc). From this, I can determine the DH parameter values. Example, bolded shows relevant joint info needed:

```
<joint name="joint_2" type="revolute">
  <origin xyz="0.35 0 0.42" rpy="0 0 0"/>
  <parent link="link_1"/>
  <child link="link_2"/>
  <axis xyz="0 1 0"/>
  <limit lower="${-45*deg}" upper="${85*deg}" effort="300" velocity="${115*deg}"/>
</joint>
```

Once the DH parameter table is obtained, the transformation from each joint to the next can be formed using the following formula:

T = * Rot_x,α_{i-1} * Trans_x,a_{i-1} * Rot_z,Θ_i * Trans_z,d_i

This results in the following homogenous transform matrix:

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

This matrix represents the transformation required to move from one joint to the next joint. It moves the reference frame in link i-1 to be exactly coincident with the reference frame in link i. I then created a script called FK_server.py. At line 33, I begin to build the individual homogeneous transformations using the above matrix and the parameters from the DH table. Each individual homogeneous transformation matrix is as follows:

$$
T_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_0 \\ \sin(\theta_1)\cos(\alpha_0) & \cos(\theta_1)\cos(\alpha_0) & -\sin(\alpha_0) & -\sin(\alpha_0)d_1 \\ \sin(\theta_1)\sin(\alpha_0) & \cos(\theta_1)\sin(\alpha_0) & \cos(\alpha_0) & \cos(\alpha_0)d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
T_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_1 \\ \sin(\theta_2)\cos(\alpha_1) & \cos(\theta_2)\cos(\alpha_1) & -\sin(\alpha_1) & -\sin(\alpha_1)d_2 \\ \sin(\theta_2)\sin(\alpha_1) & \cos(\theta_2)\sin(\alpha_1) & \cos(\alpha_1) & \cos(\alpha_1)d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
T_3^2 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_2 \\ \sin(\theta_3)\cos(\alpha_2) & \cos(\theta_3)\cos(\alpha_2) & -\sin(\alpha_2) & -\sin(\alpha_2)d_3 \\ \sin(\theta_3)\sin(\alpha_2) & \cos(\theta_3)\sin(\alpha_2) & \cos(\alpha_2) & \cos(\alpha_2)d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
T_4^3 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & a_3 \\ \sin(\theta_4)\cos(\alpha_3) & \cos(\theta_4)\cos(\alpha_3) & -\sin(\alpha_3) & -\sin(\alpha_3)d_4 \\ \sin(\theta_4)\sin(\alpha_3) & \cos(\theta_4)\sin(\alpha_3) & \cos(\alpha_3) & \cos(\alpha_3)d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$T_5^4 = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & a_4 \\ \sin(\theta_5)\cos(\alpha_4) & \cos(\theta_5)\cos(\alpha_4) & -\sin(\alpha_4) & -\sin(\alpha_4)d_5 \\ \sin(\theta_5)\sin(\alpha_4) & \cos(\theta_5)\sin(\alpha_4) & \cos(\alpha_4) & \cos(\alpha_4)d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^5 = \begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6) & 0 & a_5 \\ \sin(\theta_6)\cos(\alpha_5) & \cos(\theta_6)\cos(\alpha_5) & -\sin(\alpha_5) & -\sin(\alpha_5)d_6 \\ \sin(\theta_6)\sin(\alpha_5) & \cos(\theta_6)\sin(\alpha_5) & \cos(\alpha_5) & \cos(\alpha_5)d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_G^6 = T_7^6 = \begin{bmatrix} \cos(\theta_7) & -\sin(\theta_7) & 0 & a_6 \\ \sin(\theta_7)\cos(\alpha_6) & \cos(\theta_7)\cos(\alpha_6) & -\sin(\alpha_6) & -\sin(\alpha_6)d_7 \\ \sin(\theta_7)\sin(\alpha_6) & \cos(\theta_7)\sin(\alpha_6) & \cos(\alpha_6) & \cos(\alpha_6)d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Next, I build the total transformation from the base to the end effector by multiplying all the individual transformations together.

T0_G = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_G

Lastly, there is a difference in orientation between the DH parameters and the URDF file, so I need to apply a 180 degree rotation about the z axis, and then apply a -90 degree rotation about the y axis.

$$\text{rot\_z} = \begin{bmatrix} \cos(\pi) & -\sin(\pi) & 0 & 0 \\ \sin(\pi) & \cos(\pi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{rot\_y} = \begin{bmatrix} \cos(-\frac{\pi}{2}) & 0 & \sin(-\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-\frac{\pi}{2}) & 0 & \cos(-\frac{\pi}{2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The output of this scripts shows the total homogeneous transformation between the base and each individual link:

```
robond@udacity:~/catkin_ws/src/RoboND-Kinematics-Project/kuka_arm/scripts$ ./FK_server.py
('T0_1 = ', Matrix([
[1.0,   0,   0,    0],
[  0, 1.0,   0,    0],
[  0,   0, 1.0, 0.75],
[  0,   0,   0,  1.0]]))
('T0_2 = ', Matrix([
[  0, 1.0,   0, 0.35],
[  0,   0, 1.0,    0],
[1.0,   0,   0, 0.75],
[  0,   0,   0,  1.0]]))
('T0_3 = ', Matrix([
[  0, 1.0,   0, 0.35],
[  0,   0, 1.0,    0],
[1.0,   0,   0,  2.0],
[  0,   0,   0,  1.0]]))
('T0_4 = ', Matrix([
[  0,    0, 1.0,  1.85],
[  0, -1.0,   0,     0],
[1.0,    0,   0, 1.946],
[  0,    0,   0,   1.0]]))
('T0_5 = ', Matrix([
[  0, 1.0,   0,  1.85],
[  0,   0, 1.0,     0],
[1.0,   0,   0, 1.946],
[  0,   0,   0,   1.0]]))
('T0_6 = ', Matrix([
[  0,    0, 1.0,  1.85],
[  0, -1.0,   0,     0],
[1.0,    0,   0, 1.946],
[  0,    0,   0,   1.0]]))
('T0_G = ', Matrix([
[  0,    0, 1.0, 2.153],
[  0, -1.0,   0,     0],
[1.0,    0,   0, 1.946],
[  0,    0,   0,   1.0]]))
robond@udacity:~/catkin_ws/src/RoboND-Kinematics-Project/kuka_arm/scripts$
```

In RVIZ, the transform from base to link 1, base to link 4, and finally base to link 6 are different than what is produced in my script because these 3 frames (frame 1, 4, and 6) have the origins placed in a different spot in the DH diagram then from the original joint location in the URDF.

Inverse Kinematics

--
-

Please Note: my arm successfully drops the cylinder in the bucket, however, the orientation of the arm is slightly off (it doesn't drop the cylinder straight). I believe the issue lies in somewhere in the last 3 joints of the arm since the orientation of the arm is off, however, I wasn't able to figure this out. If possible, after reading my report, can you provide feedback on how I can go about resolving this? I'd like to try in implement it and learn from this. Thanks for the help! :)

--
-

Most times, you know the desired location of the end effector, and need to find the joint variables (orientation and rotation) of the arm that allow the end effector to reach this desired location.This is called inverse kinematics.

Although solving the inverse kinematics problem is difficult, we can decouple the problem into two simpler problems when dealing with a 6 joint manipulator: 1) inverse position kinematics (first 3 joints) and 2) inverse orientation kinematics (last 3 joints). To do this, the last three joints of the manipulator must intersect at the same point. So in our problem, we have x4, x5, and x6, all intersecting at joint 5, which means the end-effector is kinematically decoupled. With this, we can view the first 3 joints of the arm as the joints that determine the position of the end effector, while the last 3 joints determine the orientation of the end effector. The last 3 joints will intersect at the wrist center, and since these 3 points will not change the position of the end effector, the position of the wrist center is a function of only the first 3 joint variables.

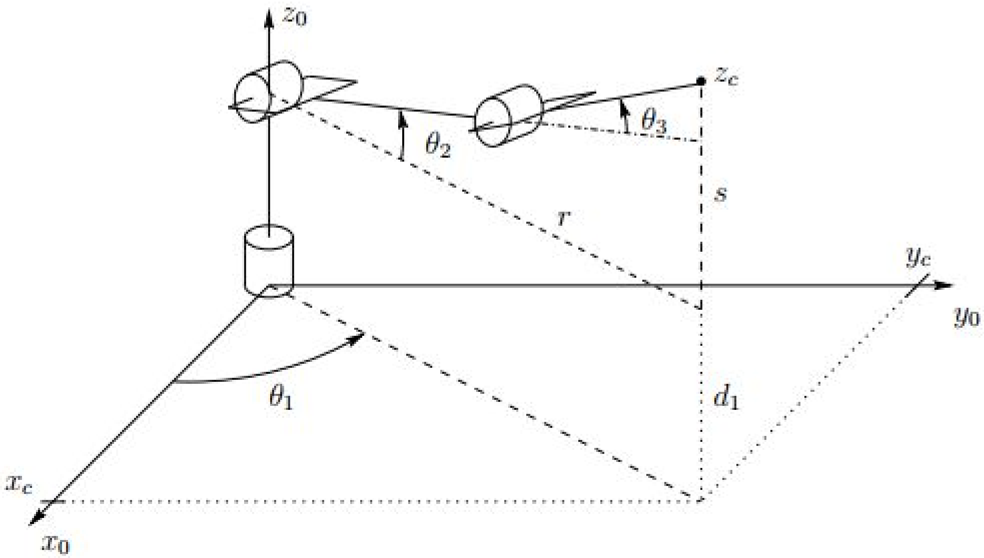
The first step is to find the wrist center located at x4, x5, and x6 (joint 5). The origin of the end effector is found by a translation along z6 axis, which corresponds to d6 in the DH parameter table. Point px, py, and pz represents the position of the end effector and can be obtained from the transformation matrix from the base to the gripper frame (it is the fourth column):

$$
{}_{EE}^{0}T = \left[\begin{array}{ccc:c} & {}_{6}^{0}R & & {}^{0}\boldsymbol{r}_{EE/0} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

We also have to account for the length of the end effector which is 0.303 as specified in the URDF. So to get from the end effector position to the wrist position, we use the following equations

wx = px - (d6 + eeLength)*r13
wy = py - (d6 + eeLength)*r23
wz = pz - (d6 + eeLength)*r33

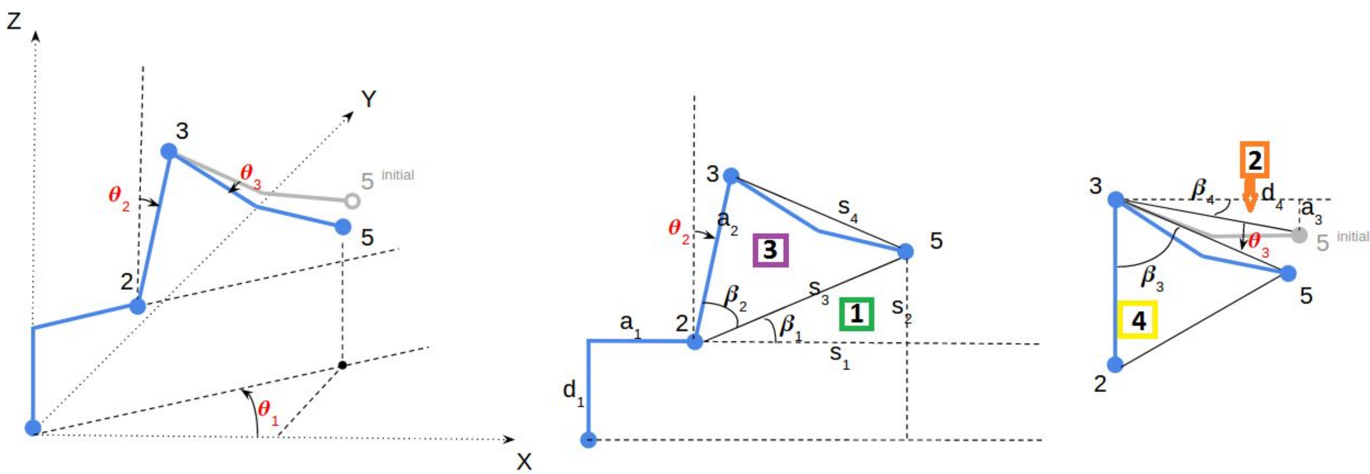
Now that we have the wrist center, we start by finding Θ1, Θ2, and Θ3 joint values. I used the geometric approach to solve for these values. The following diagram shows where Θ1 is located



In the diagram, zc represents the position of the wrist center, d1 is the distance from joint 1 to joint 2 and is taken from the DH table. We find Θ1 by projecting the length of the arm from joint 2 to the wrist center onto the x0 and y0 plane. From this, we know where wx and wy (in the picture xc and yc) sit in this plane. So we can determine Θ1 as follows:

Θ1 = atan2(wy, wx)

Next, I used the following diagram to help find Θ2 and Θ3



To find Θ2, I first find s1 and s2, which represent the distance from joint 2 to joint 5 in x and z respectively. The value s1 is simply the difference between wx and a1, and s2 is simply the difference between wz and d1:

s1 = wx - a1

s2 = wz - d1

Next, I can find β1 using the tangent of s2 and s1 in triangle 1 marked green:

β1 = atan2(s2, s1)

Next, to find β2, I need s3 and s4. The value s3 is simply the hypotenuse of triangle 1 marked in green.

$s3 = \text{sqrt}(s1^2 + s2^2)$

And s4 is simply the hypotenuse of triangle 2 marked in orange.

$s4 = \text{sqrt}(d4^2 + a3^2)$

Now, using cosine law in triangle 3 marked purple, I can find β2:

$\beta 2 = \cos^{-1}((a2^2 + s3^2 - s4^2) / (2*a2*s3))$

Now, I can find Θ2 by subtracting β2 and β1 from 90 degrees

$\Theta 2 = (pi/2) - \beta 2 - \beta 1$

Next, I need to find β3 and β4 to find Θ3. β3 is found using cosine law in triangle 4 marked yellow.

$\beta 3 = \cos^{-1}((s4^2 + a2^2 - s3^2) / (2*a2*s4))$

Then β4 is found using the tangent of a3 and d4 in triangle 2 marked orange.

$\beta 4 = \text{atan2}(a3, d4)$

Lastly, I can find Θ3 by subtracting β3 and β4 from 90 degrees

$\Theta 3 = (pi/2) - \beta 4 - \beta 3$

Now I need to find the remaining 3 angles, Θ4, Θ5, and Θ6 to find the wrist orientation in space. Since the joint variables in the transform from the base to joint 3 are independent from the joint variables in the transform from joint 3 to joint 6, we should find the orientation of the end effector by solving the rotation from joint 3 to 6. These angles can be found by solving the following equation

$${}^{3}_{6}R = \left({}^{0}_{3}R\right)^{-1} {}^{0}_{6}R = \left({}^{0}_{3}R\right)^{T} {}^{0}_{6}R$$

I already know the rotation from the base to joint 3 from the forward kinematics portion of this project, and I know the total rotation from the base to the joint 6 in terms of Θ4, Θ5, and Θ6. By using this above equation, I can determine the orientation of the final 3 joints. Once I have this matrix R3_6, I use the inverse tangent to determine Θ4, Θ5, and Θ6. When doing the math for Θ5, I take a square root. The square root naturally provides two answers + or -. These two answers determine different orientations in the arm.

I wasn't able to get the orientation correct for my arm, and I believe this is due to an issue with my Θ4, Θ5, and/oa Θ6 values. This is something I'd still like to improve on. Another improvement I'd be interested in is reducing the IK calculation times. It takes a bit of time to calculate the inverse kinematics, and I read on Slack that the sympy library is the cause of this. I had moved all my sympy calls outside the main loop, however, it was mentioned that by rewriting the sympy calls in numpy, the calculation time is significantly reduced. Sometimes, I also receive a SerializationError, but not always. I'm not sure how to go about resolving this.