

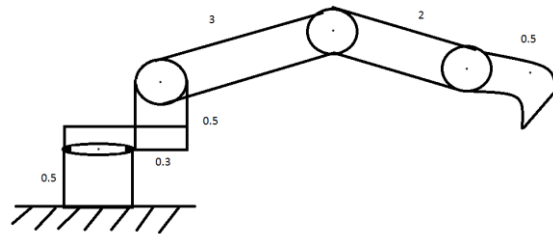
EE 209 AS lab report: Inverse and Forward kinematics of Robotics design

By ZHUOQI LI 504135743, Weining Zhou

Introduction:

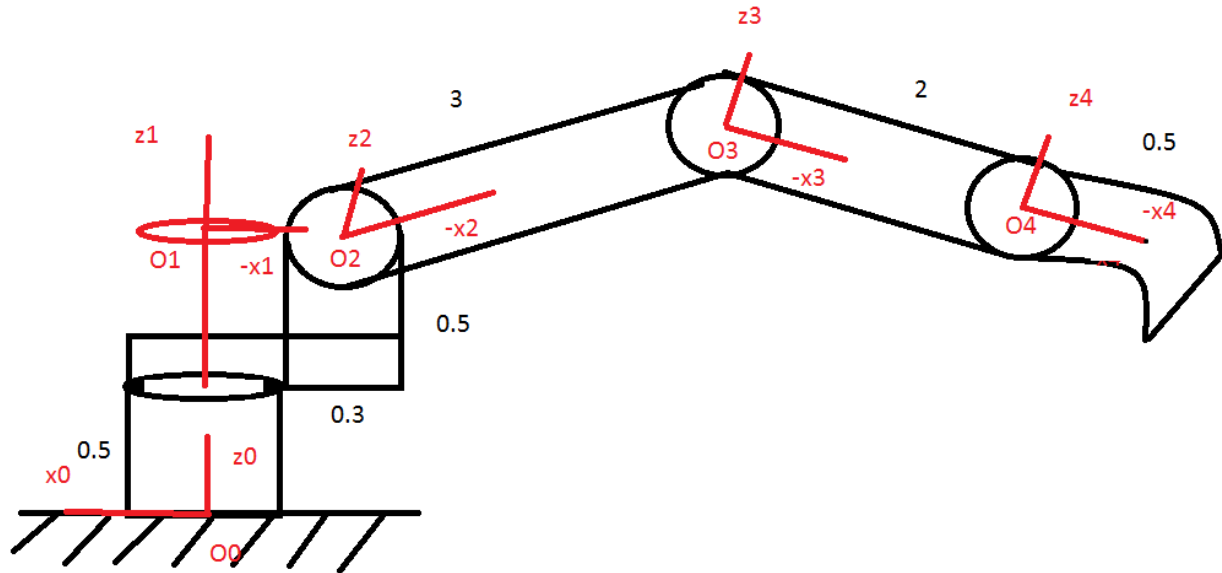
The goal of this lab is to understand forward and inverse kinematics of robotics design. In this lab, we need to first find an object in the real world that resembles the kinematic chain robot with 4 or more joints (1DOF). By analyzing the modified DH parameters of this object, we could further calculate the coordinates of its end effector in the world frame using forward kinematics. Also, using inverse kinematics realized by the Jacobian method, we could deduce joint parameters (theta angle) from the desired end effector position.

The kinematic chain object we found in the real world is an excavator. It has four joints; three of them have z-axis pointing out of page and one pointing upward as shown in the draft on the right. The end effector in our case is the claw attached to the fourth joint, which is located 0.5 meter to the right of the last joint's origin. Since our kinematic chain has 4 1-DOF joints, the total DOF of our chain is 4. Also note that since we have one Z axis pointing upward, our end effector can move in 3-D space limited by the arm length. The linkage between joint one and joint two is fixed, therefore the longest distance the arm can reach is 6.5 m, which happens when the arm is upright. The intended function of this linkage, just like any excavator, is to move the claw from one place to another. In this lab, we will analyze the linear motion of the claw using forward/inverse kinematics and compare it with the straight line motion.



Method:

To begin with, we start with the forward kinematics which is to deduce the end effector position utilizing the modified DH-parameters of all joints. Compared with IK, FK is relatively easier, and the setup of FK is also crucial for later analysis. In order to measure the modified DH parameters, we first need to construct the coordinate reference for each joint frame. Using rules introduced in class and learned from the Wikipedia page, and start from the base origin which sets the world coordinate frame, coordinate frames are set as shown in the draft:



The base origin and so the world frame is set at the attach point between the object and the world, the x axis is set to point to the left. Other than 3 other joints, the origin of the first joint frame is elevated from the actual joint position to be at the same height as the second joint. This is to simplify the parameter calculation in later steps. For all joints, their x-axis are all pointing to the reverse direction to the next origin position and are perpendicular to both z-axis as I labeled all my axis to be negative in the draft above. The sign of the axis actually does not matter as long as it coincide with signs of corresponding DH parameters.

Using rules learned in the class, modified DH-parameters are obtained as shown below:

a0	0	a1	-0.3	a2	-3	a3	-2
alpha0	0	alpha1	Pi/2	alpha2	0	alpha3	0
D1	1	D2	0	D3	0	D4	0
Theta1	Θ_1	Theta2	Θ_2	Theta3	Θ_3	Theta4	Θ_4

Notice that all theta values are represented in variables. This is because for any end effector pose, the only change in our parameters is the theta values. Other parameters remain the same in all cases due to the physical structure constraint. Also note that the value of D1 is 1, which is cause by the elevation of the joint origin. Before we verge into the calculation of FK, another very important matter is needed to be pointed out: the end effector locates in the coordinate frame of the last joint with the coordinate $(-0.5, 0, 0, 1)$ in homogeneous format. This is very important as it will be used in the last step in FK to transfer the coordinate in its own frame into world frame.

Forward Kinematics:

To test whether our DH-modified parameters are correct, we use the FK to verify. Given a known set of theta angles, we would like to know whether the 'Robot arm' will reach to the expected position. The concept of forward kinematics is very simple: multiply the frame-to-base transform matrix with the frame coordinates to obtain coordinates in the world frame. Since the transform matrix is in

respective to the prior frame, for end effector in our case, the frame-to-base matrix is the concatenation of previous transform matrixes.

$$T_4^0 = T_1^0 * T_2^1 * T_3^2 * T_4^3$$

With each transform matrix in the form of:

$${}^{n-1}T_n = \left[\begin{array}{ccc|c} \cos \theta_n & -\sin \theta_n & 0 & a_{n-1} \\ \sin \theta_n \cos \alpha_{n-1} & \cos \theta_n \cos \alpha_{n-1} & -\sin \alpha_{n-1} & -d_n \sin \alpha_{n-1} \\ \sin \theta_n \sin \alpha_{n-1} & \cos \theta_n \sin \alpha_{n-1} & \cos \alpha_{n-1} & d_n \cos \alpha_{n-1} \\ \hline 0 & 0 & 0 & 1 \end{array} \right]$$

And the world coordinate of the end effector is equal to:

$$q^0 = T_4^0 * q^4, \quad q^4 = [-0.5 \ 0 \ 0 \ 1]^T \text{ (in our case)}$$

Inverse Kinematics:

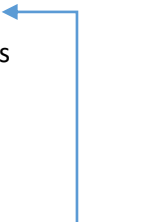
Compared with forward kinematics, inverse kinematics is much harder to compute. This is because given one desired position coordinates, there exit not only one set of theta angles to achieve this position. To calculate the set of theta parameters, we choose the iterative pseudo-inverse jacobian method. This method is derived from Newton's linearization method, and therefore it requires the initial guess to be close to the desired position. The core of this method lie in the solution of the following equation:

$$e = J * \Delta\theta \Rightarrow \Delta\theta = J^\dagger e$$

e – discrepancy between desired and predicted coordinates

J – Jacobian matrix of T_4^0 $\Delta\theta$ – increment in theta values J^\dagger – pseudo – inverse of J

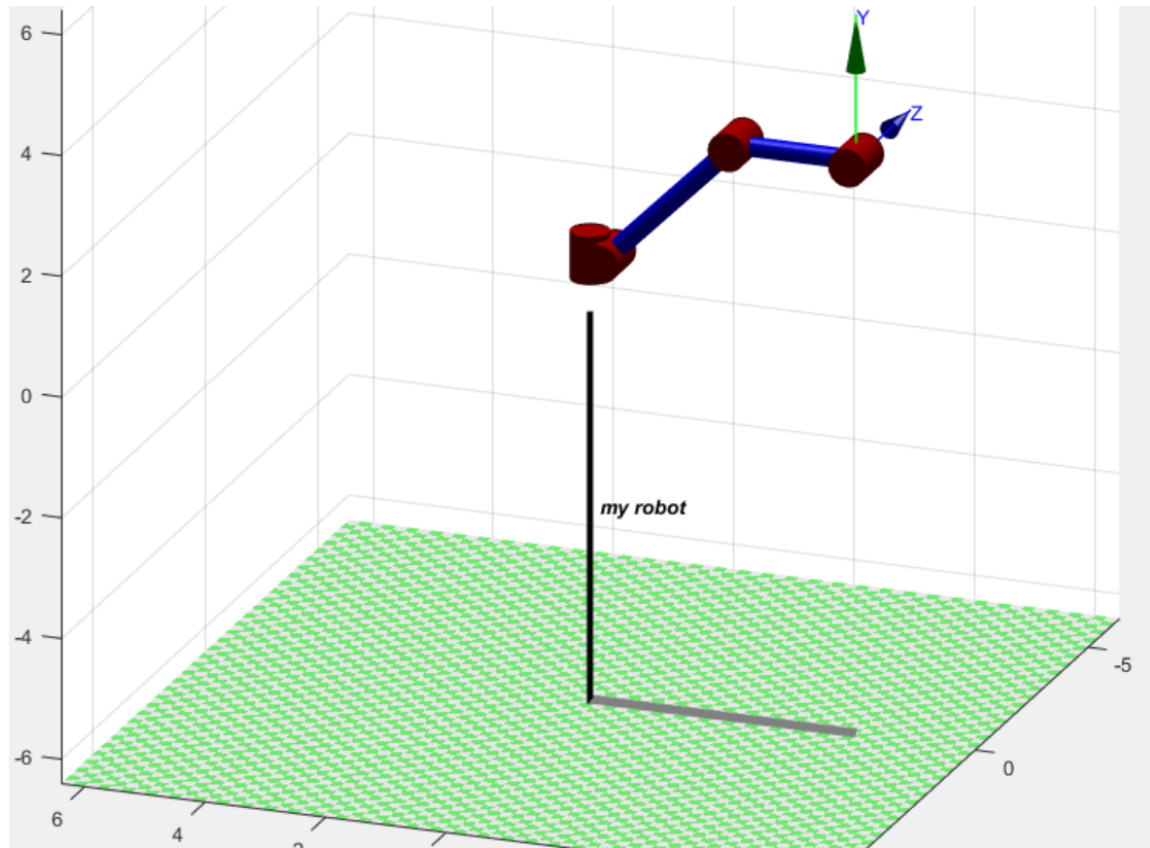
Based on the information above, we deduct an iterative procedure:

1. Make and initial guess on theta values (try to be close to desired values)
 2. Calculate predicted end effector position using FK
 3. Calculate the error between predicated and desired coordinates
 4. Calculate the Jacobian matrix in respect to current theta values
 5. Calculate the pseudo-inverse of the Jacobian matrix
 6. Calculate delta theta by multiply pseudo-inverse T and error
 7. Update theta values by incrementing delta-theta
 8. Resume step 2 and repeat the whole process until it reaches the desired point
- 

Results:

Forward Kinematics:

By choosing a set of theta angles $[0 \ -\pi/4 \ \pi/4 \ 0]$, substitute modified DH parameters into the transform matrix and we get a world coordinate of $(-4.9213, 0, 3.1213, 1)$, which coincides with the result via geometric analysis. And the end effector pose at this set of theta angle is set as below:



(Joint 1 and Joint 2 object are created so large that they seem to overlap each other)

This Matlab diagram generated by inputting our modified DH parameters perfectly coincide the robot arm pose we draw by hand. And the end effector location is exactly the same as our geometric calculation. This shows our choices of DH parameters and transfer function are correct and can be used for further analysis.

Inverse Kinematics:

Using the iterative method introduced in the method section, to test whether our IK iterative method work or not, we choose a random point (within the arm length limit) $(-0.5 \ 0 \ 5 \ 1)$ as the desired point and start to perform our IK on it. To analyze how the initial guess affects our IK calculation, we choose two set of initial guesses to start with, one close to the desired pose and the other far from it.

The following table shows the predicated point coordinates in the iteration process:

Start position: around (-5, 0, 3)

Desired position: (-0.5, 0, 5)

Predicated position 1:

Predicated position 2:

iteration	X	Y	Z	iteration	X	Y	Z
1	-4.9213	0	3.12132	1	-0.50979	0	3.36175
2	1.49969	0	-0.1051	2	0.50143	0	4.83132
3	-3.14667	0	2.14576	3	-0.4931	0	4.85079
4	-0.12269	0	2.33339	4	-0.49741	0	4.99753
5	0.30025	0	4.65496	5	-0.50000	0	4.99999
6	-0.5423	0	4.90645	6	-0.49999	0	4.99999
7	-0.4982	0	4.99789	7	-0.50000	0	4.99999
8	-0.5000	0	4.99999	8	-0.50000	0	4.99999
9	-0.4999	0	4.99999	9	-0.50000	0	4.99999
10	-0.5000	0	5	10	-0.50000	0	4.99999

iteration	Theta1	Theta2	Theta3	Theta4	iteration	Theta1	Theta2	Theta3	Theta4
1	0	-0.785	0.785	0	1	4.035	2.477	4.769	4.0354
2	0	-2.869	4.140	-1.271	2	3.828	1.643	5.811	3.8280
3	0	-1.170	2.111	-0.941	3	4.099	1.707	5.476	4.0994
4	0	-2.673	2.683	-0.010	4	4.124	1.622	5.536	4.1245
5	0	-2.409	1.439	0.969	5	4.125	1.621	5.536	4.1256
6	0	-2.161	1.375	0.785	6	4.125	1.621	5.536	4.1256
7	0	-2.149	1.311	0.837	7	4.125	1.621	5.536	4.1256
8	0	-2.148	1.310	0.837	8	4.125	1.621	5.536	4.1256
9	0	-2.148	1.310	0.837	9	4.125	1.621	5.536	4.1256
10	0	-2.148	1.310	0.837	10	4.125	1.621	5.536	4.1256

As we can see from the two table above, the initial guess does have large impact on the end effector pose predication. For a guess value far from the desired point, although our method does get to the desired point after 7 iteration, as we can see from the table, before it gets there it did not follow any certain pattern but rather jump around to some random coordinates. As for the second group of iteration where initial guess is close to the desired point, it gets to the desired point after only four iterations and follows a very smooth linear motion pattern. This is expected as the iterative pseudo-inverse method is supposed to work best when initial guess is close to the desired point. Also, to our surprise, our method still works when the initial guess is far from the final answer and it only take less than 10 iterations. Another thing to mention that our iterative algorithm is also very fast. It could finish 1000 iterations in less than 1 second!

Motion trajectory modification:

To generate the trajectory in the operational space we can use the IK method discussed above. But as revealed from the experimental data above depending on the initial point the trajectory using current method could be as smooth as straight-line motion only if two points (initial guess point/start point and the desired end point) are close enough. Or the robot arm will 'jump all over the place' before

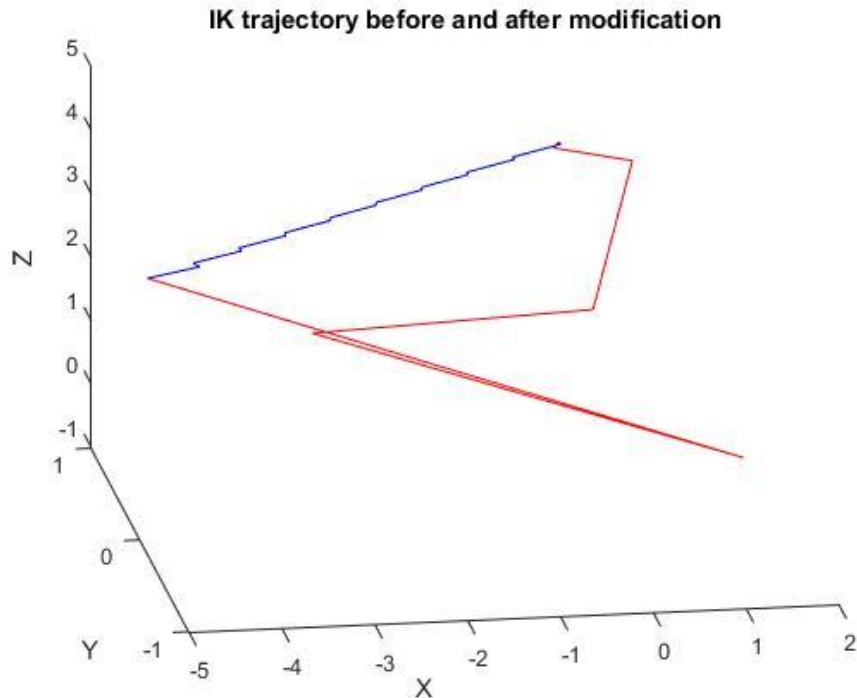
it reaches close enough to the desired point, which is definitely unacceptable in the real world application. To generate more accurate trajectory, we come up with one solution:

1. Draw a straight line between start point and end point
2. Break this straight line into multiple points, make sure adjacent points are close to each other
3. Set initial guess close to the start point, use the IK method to approach the next point along the straight line repeatedly until it reaches the desired point

The following table shows the predicated points along the trajectory between the same two points we used in IK method:

iteration	X	Y	Z	iteration	X	Y	Z
1	-4.92132	0	3.12132	17	-2.95627	0	3.95628
2	-4.92129	0	3.12129	18	-2.45677	0	4.12403
3	-4.35787	0	3.26997	19	-2.46462	0	4.16497
4	-4.42728	0	3.32967	20	-2.46502	0	4.16502
5	-4.43004	0	3.33004	21	-1.96281	0	4.33513
6	-4.43004	0	3.33004	22	-1.97349	0	4.37364
7	-3.90896	0	3.49117	23	-1.97377	0	4.37377
8	-3.93794	0	3.53881	24	-1.46603	0	4.54566
9	-3.93878	0	3.53878	25	-1.48239	0	4.58230
10	-3.93878	0	3.53878	26	-1.48251	0	4.58251
11	-3.43199	0	3.70292	27	-0.96845	0	4.75441
12	-3.44697	0	3.74757	28	-0.99129	0	4.79096
13	-3.44753	0	3.74753	29	-0.99126	0	4.79126
14	-3.44753	0	3.74753	30	-0.47239	0	4.96042
15	-2.94681	0	3.91331	31	-0.50017	0	4.99961
16	-2.95580	0	3.95628	32	-0.50000	0	5.00000

As we can see, although there are more points in the trajectory, but the increment between adjacent points are much smoother!



According to the above diagram comparing two trajectories (blue for modified method, red for original method), after our modification, the trajectory almost follow a straight line pattern in spite of noticeable step increment. This could be improved by enhance the resolution – break the straight line into more points. Still, the modification brings considerable enhancement compared with the previous irregular trajectory pattern.

Conclusion:

In this lab, by implementing the inverse and forward kinematics by hand, we had built a good understanding of what is forward and inverse kinematics and how do them apply in robotics design. This lab takes me about 15 hours to do. A large chunk of the time was spent on reading documents online to learn about inverse/forward kinematics. To be honest, this lab is hard at the beginning as we don't exactly understand what are we supposed to do, and inverse kinematics were not well covered in class so it took us at least 5 hours to learn online. Personally the most fun part of this lab would be the moment I found a toolbox online of which it enables us to graphically display the robot and speedup the IK algorithm by a thousand time (its Jacobian calculation function is so fast). And the least fun part will be the very beginning of the lab when we have no idea where to start nor do we quite understand the concept of IK. In general this lab is okay for me; although I personally dislike linear algebra, but I do learned something new and practical. I hope next time there will be more hands-on practice.