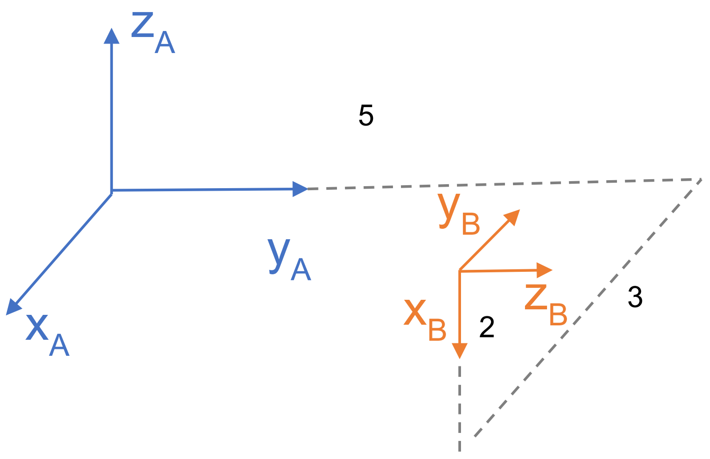
ENN 581 – Semester 2 – 2023

Week 2 – Forward Kinematics of Manipulator Arms

Krishna Manaswi Digumarti

Section 1 – Theory

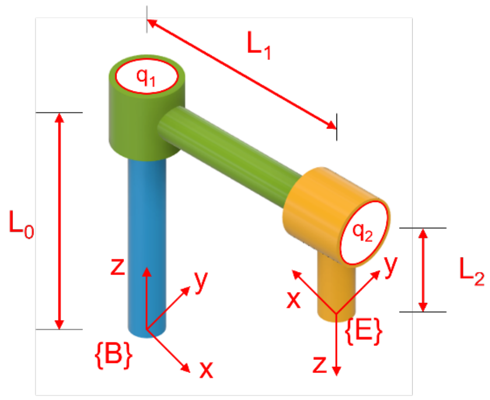
1. Given the following two frames {A} and {B}, determine the homogeneous transformation matrix ${}^{A}T_{B}$ between them.



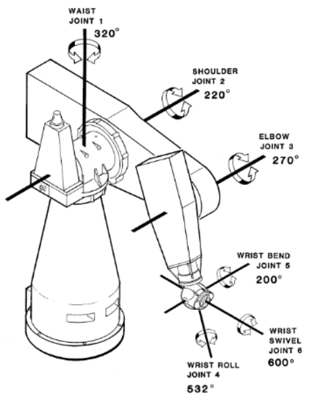
2. Frames {A} and {B} are related through an intrinsic compound rotation about z-x-z by α, β and γ. Determine the transformation matrix ${}^{A}T_{B}$.

3. Given a rotation matrix of the form $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ that corresponds to the above intrinsic rotation of z-x-z, determine the values of the Euler angles in terms of the elements of the rotation matrix.

4. For a Furuta pendulum, as shown below, verify that the transformation matrix relating the base frame and the end effector frame is the same when using the elementary transform sequence and proximal DH notation.



5. Write down the transformation matrix relating the base and end effector frames for the PUMA 560 arm shown below. Practice ETS and Proximal-DH methods.



Section 2 – Simulation

This week, we will continue exploring the features of CoppeliaSim and use it to plot the tool position of a robot manipulator arm.

Download the scene file from Canvas called **lite6_kinematic_starter.ttt**. Note that this scene file will not work out of the box. You must make appropriate changes as you work through.

A skeleton code **forward_kinematics_skeleton.py** has been provided to get started on the practical. A completed code will be provided next week. Refer to the CoppeliaSim user manual for reference on all the functions mentioned in this document. (CoppeliaSim User Manual (coppeliarobotics.com))

This scene has a model describing the UFACTORY Lite 6 robot arm. Expand the hierarchy of the robot to see the various joints and links. The joints have already been assigned their appropriate 3D pose. We will learn to move the joints in a kinematic simulation.

Task 1: Assign all joints to kinematic mode. This means that they do not take part in a dynamic simulation. They can be moved instantaneously using the sim.setJointPosition() command.

Task 2: Make sure that all the links are static objects. If they are left in dynamic mode, they are under the influence of gravity and will fall.

Task 3: Write a script that sets the joint positions to 10, 20, 30, 40, 50 and 60 degrees.

Try the following:

- Get a handle to the joint using sim.getObject()
- Assign a position to the joint using sim.setJointPosition()

Task 4: How would you modify the code if you wanted to visualise the motion in smaller increments? Hint: look at the *while* loop in the skeleton code.

Graphs

Task 5: Let us plot the position of the tip. A dummy object called ToolTip is attached to the last link. This will help us with plotting. A graph object called Graph has also been added to the scene.

Try the following:

- Get a handle to the Graph
- Get a handle to the ToolTip
- Tell the graph object that you are giving it a new stream of data by using the sim.addGraphStream() function. Make separate streams for x, y and z positions.
- In the *while* loop add data into the stream using the sim.setGraphStreamValue() function.

For 3D graphs, you need to add a 3D curve using the sim.addGraphCurve() function and update the stream data as before using sim.setGraphStreamValue()

Collision detection

Task 6: detect collision between the Base and the Tool links. You can use the sim.checkCollision() function for that and pass in handles to the Base and Tool.

Try the following:

- Get a handle to the Base and Tool links.
- In the *while* loop, check for collision.
- Print a message when collision is occurring.

Workspace of the manipulator

Assign limits of your choice to the joints. Plot the workspace of the manipulator.

Try the following:

- Assign limits of your choice to the joints.
- Iterate over joint angles within these limits. You may choose to restrict motion to one or two interesting joints. Choose an appropriate increment in angle to have a reasonable set of points.
- For each position, store the location of the ToolTip. For example, you can write it to a text file as comma separated x, y, z values.
- Consider whether you want to record the point or not in the event of a collision.
- Plot the points from the stored file using matplotlib.

You may use the code plotFromCSV.py to assist you in plotting the workspace.

Bonus: Figure out how to make a 3D boundary surface of the workspace.