**Theoretical Part**

1. Due to the rotation being with respect to the fixed frame (frame O) we need to apply the transformations in reverse.
2. In this part we will reverse the order of applying the transformations in the part a. As shown below, the result is not the same as the result of the part a because the order of applying is important.
3. With this part being fixed frame too, the transformations should be applied in reverse.

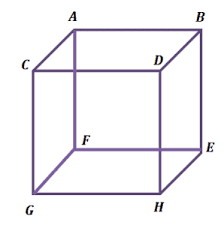
The result makes it clear that the order of applying the transformation matrices is directly influential for the result.

1. In this case, the rotation is with respect to the current frame so we apply the transformation matrices in the order they are being implemented.
2. As done in the Question 1.b, the order of applying the transformation matrices is important and applying them from right to left is different from applying them from left to right.
3. The order of applying the matrices is different in the questions 1.a and 1.b due to the respect of rotation being to the fixed frame vs. current frame.

The source of this difference is in the rotation performance being about a fixed frame or being performed continuously about the frames in order.

The first situation (the performance being about a fixed frame) is called fixed frame and in the result, the transformation matrices should be implemented in reverse the order but the second situation (with the frame being chased by the previous frame) is called current frame and the order of transformation matrices must be kept the way it is.

1. For the prismatic joints (linear joints) the z axis should be in the direction of the linear relative motion and for the revolute joints (rotary joints) the z axis is in the same direction of the rotation axis.
2. Joint parameters described as (for revolute joints) and (for prismatic joints) are parts of the D-H parameters. is the angle between and about the axis and the is the distance between and along the axis.
3. Yes! Skew lines are lines which have no intersection but also, they are not parallel. These lines only exist in 3 or more dimensions and must be placed in different planes. There can be a perpendicular line to both and a simple example can be shown below.



and are two skew lines and is perpendicular to both of them.

1. When determining the we need to pay attention to the and . must be perpendicular to both and and cannot be parallel to .
2. Link parameters are associated with the physical parts of the robot and describes the twist of the link and the describes the length of the link.

is the angle between and about the axis and the is the distance between and along the axis.

1. The difference between the two naming frames, and is not that much! With both the transformation is from previous frame (frame 0) to the current frame (frame 1) and the only difference is the placement of the previous frame’s number.
2. In general, I will explain how z and x axis and each parameter can be calculated and the result of the a and b parts are shown below.

Z: This axis is described as the direction of linear relative motion of prismatic joints and in the direction of rotation axis of revolute joints.

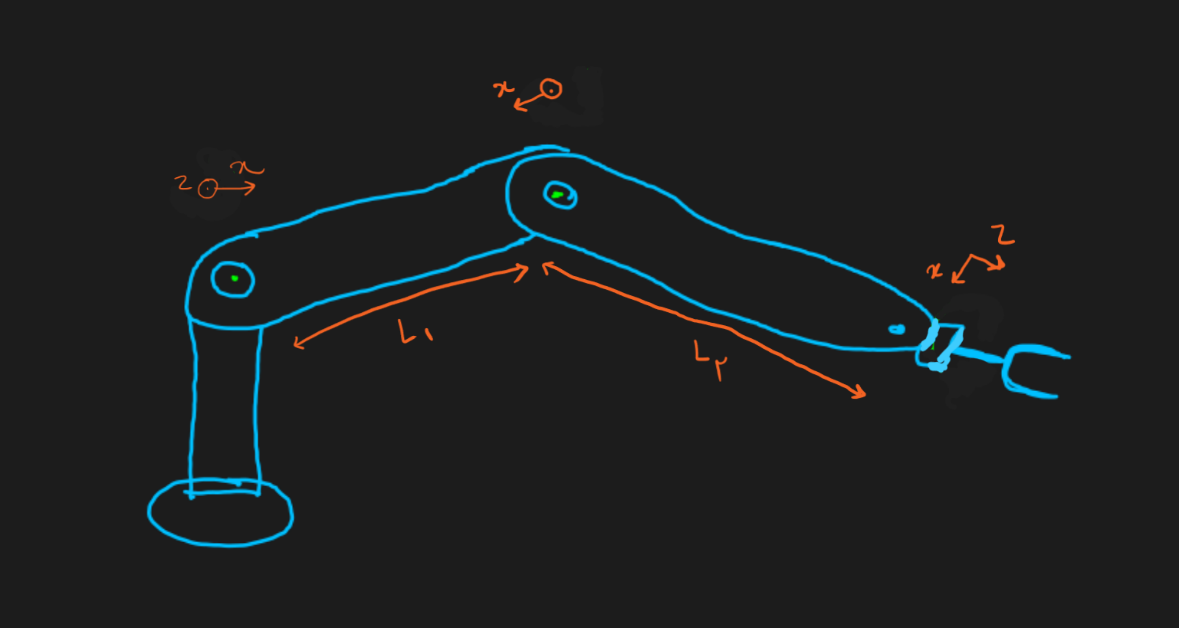
X: This axis can be defined as any direction while it must be perpendicular to current z axis and previous z axis.

: Is the angle between and along .

: Is the distance between and along .

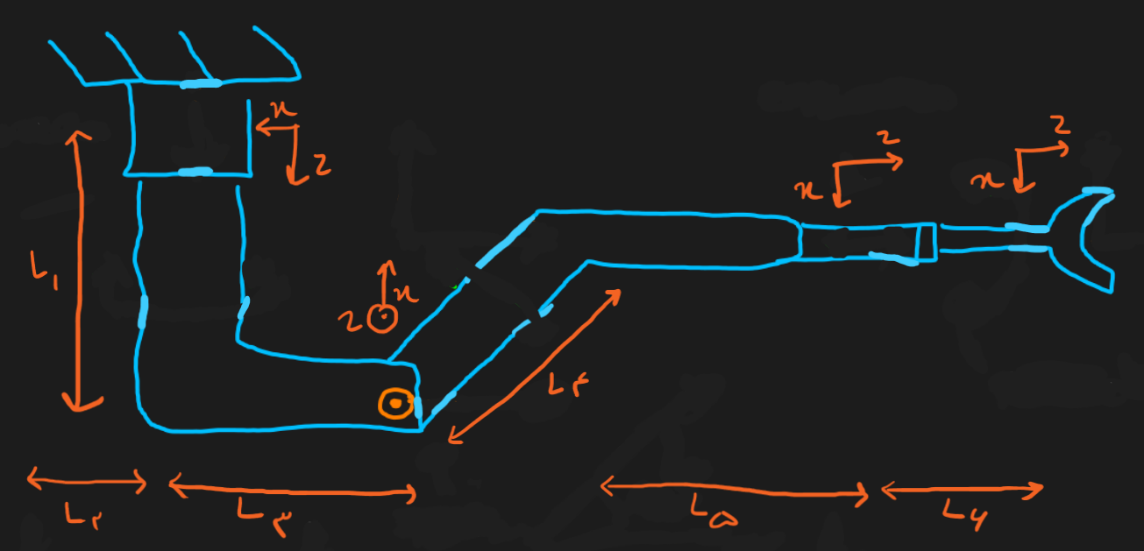
: Is the distance between and along .

: is the angle between and about .



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  |  |  |  |
| 1 | 0 | 0 | 0 |  |
| 2 | 0 |  | 0 |  |
| 3 | 90 | 0 |  |  |

1. B



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  |  |  |  |
| 1 | 0 | 0 |  |  |
| 2 | 90 |  | 0 |  |
| 3 | 90 |  |  |  |
| 4 | 0 | 0 | 0 |  |

* Linear & Angular Motion: Using Jacobian matrix describing the motions in linear world and angular world can be possible.
* Velocity Propagation: Based on the application, in most cases the velocity of end effector must be calculated and controlled. With Jacobian matrix being constrained to time, talking about velocity can be reached.
* Explicit Form
* Static Forces: Force and torque concept can be involved and used with Jacobian matrix.

1. Jacobian matrix is described as below:

In this matrix, the is the linear velocity and the is the angular velocity of the joints of the robot. The matrix is a dimensional matrix with n being the number of joints.

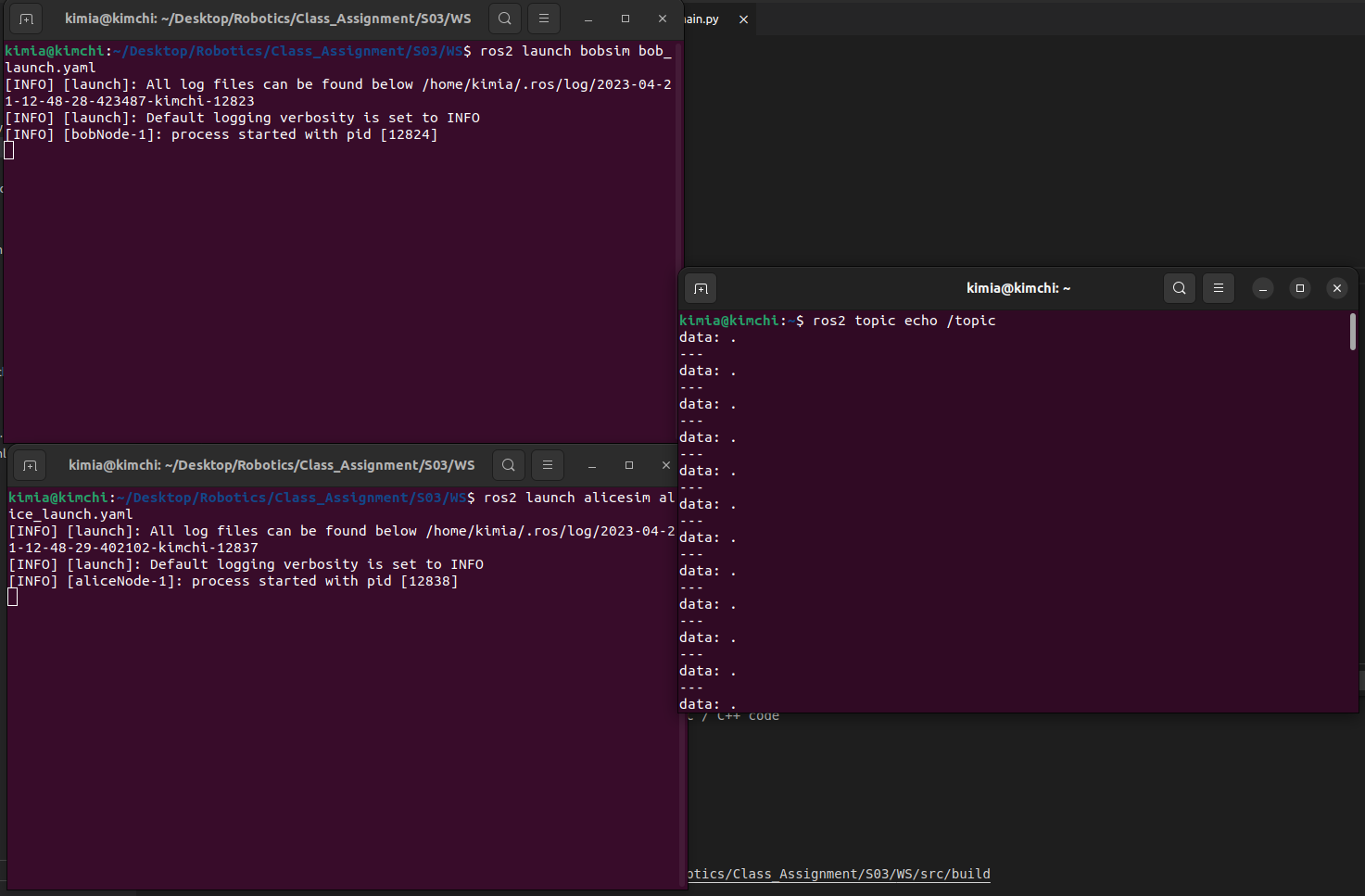
For each joint the linear velocity is described as with being the linear velocity of a prismatic joint and being the linear velocity of a revolute joint. The coefficients ( and ) decide the type of the joint because a joint cannot be prismatic and revolute at the same time and as the result, only one of these values can be replaced by the linear velocity of a joint.

The angular velocity of a joint can only be defined for the revolute joints and once again the coefficient describes the type of the joint.

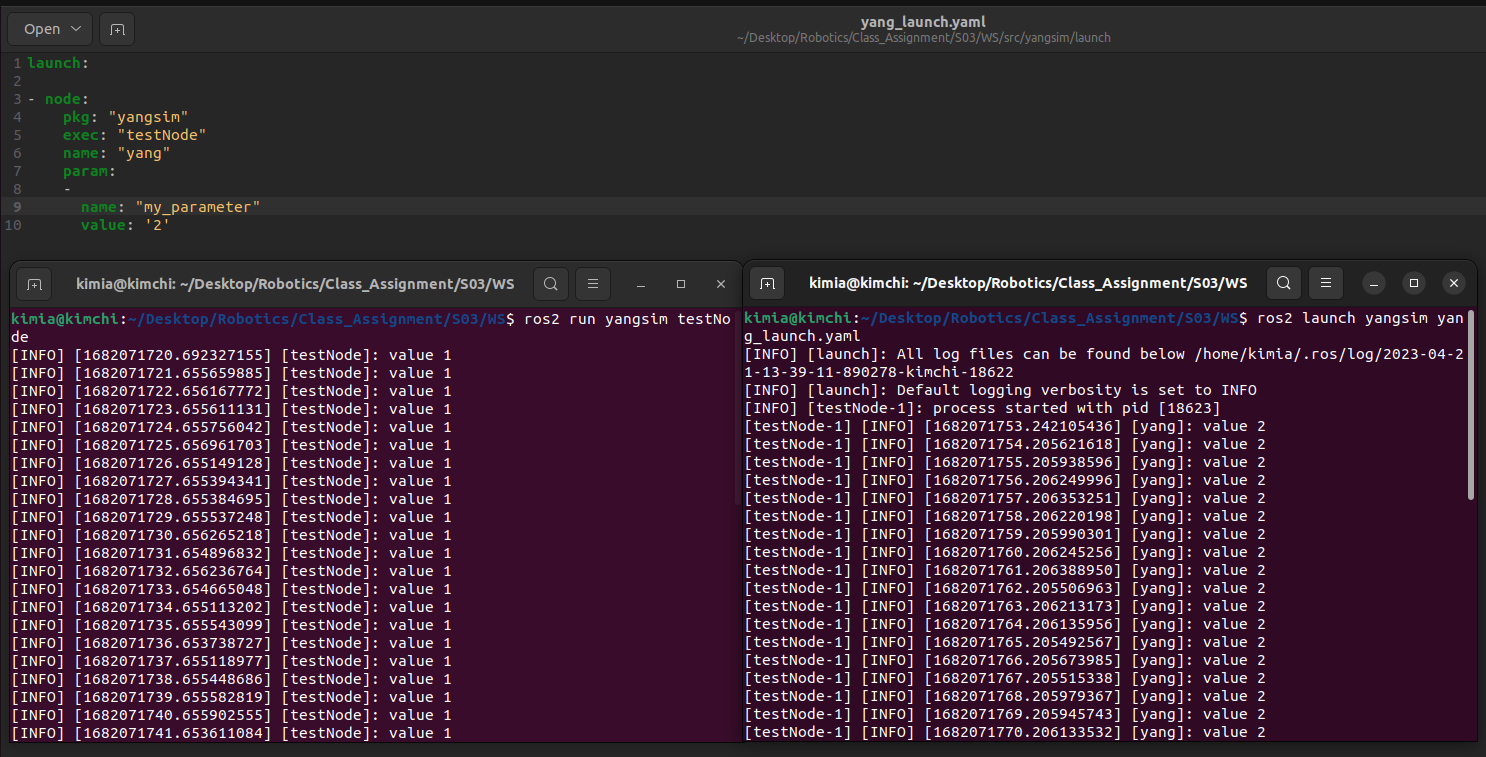
1. Linear velocity can be described as the speed of an object in a straight line while the angular velocity is the change of the angle over time. They both can be described and calculated for a robot as mentioned in the part 6.b.

**Coding Part**

1. Using publishers and subscribers, a dot can be echoed with topics and adding the launch file will result the terminals below.



1. With changing the launch file as below, we can set a parameter in the code.



1. With tf2 using DAG graphs to traverse the target frame to the source frame, it can be ensured that no loop can be created in the tree structure. When moving along in the tree, the package uses parent-child relationship so it never looks up a transform to create a loop. Also, with detecting a loop, the package throws a TransformException so we can wait for the transformation to become available again or skip the transformation altogether.

Additionally with using thread-safe structures and synchronizing mechanisms it can be guaranteed that updating in multithreading programs can be done in a safe manner.

1. Static\_transform\_publisher is a part of tf2 package and publishes a static transformation between target and source frames. Without this tool it can be challenging to calculate transformations between frames (especially in robots with multiple moving parts) and in this case, the programmers have to use their own code to calculate transformation matrices.
2. Static transformations are the transformations that remain fixed with the robot operating vs dynamic transformations that will change over time with parts of the robot moving and operating.

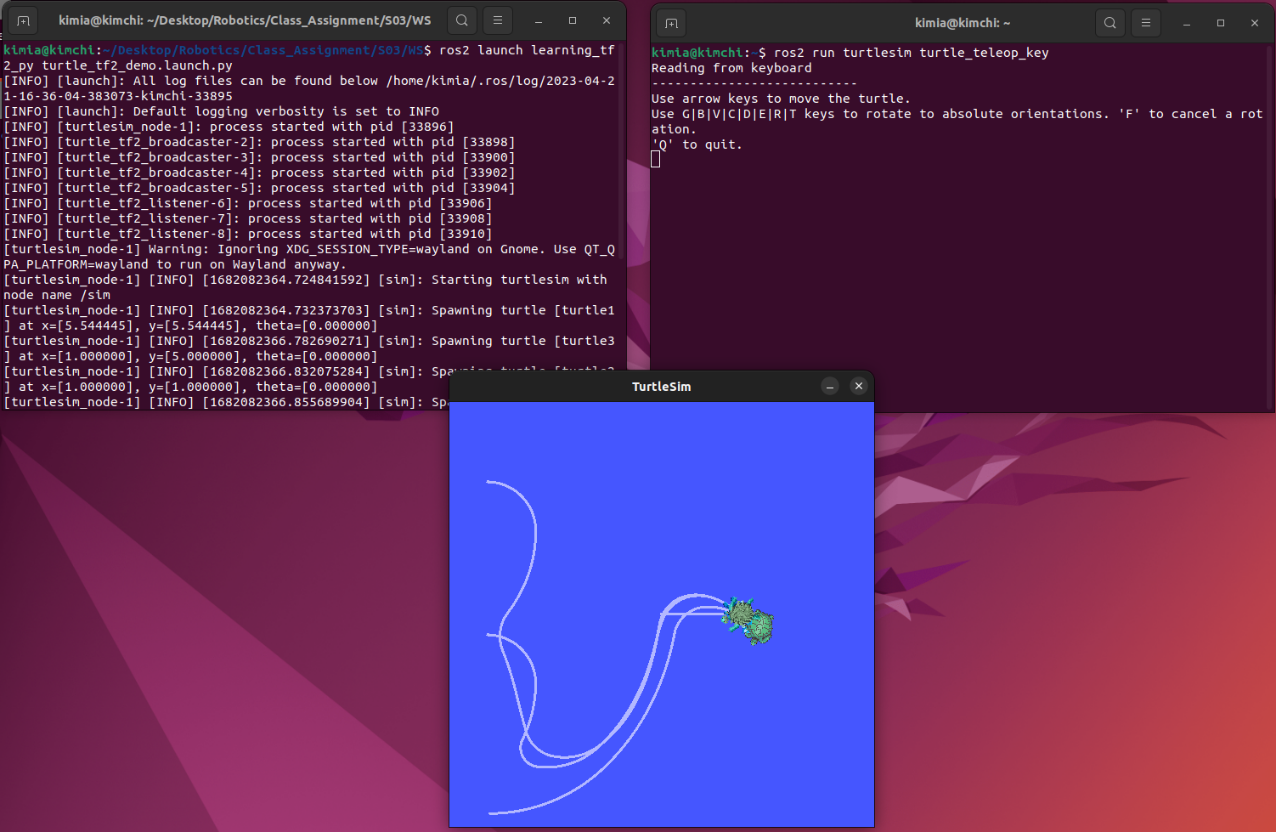
Static transformations are typically used for the parts of the robot that doesn’t move such as sensors or fixed parts and the parts that are fixed relative to the robot like the outside world.

On the other hand, dynamic transformations are used for the components of the robot that moves throughout the operation of the robot, like joints and end-effector and calculating transformation between frames that are calculated with the entry data such as cameras or sensors data.

1. Tf2 trees represent the coordinate frames and the transformations between them with nodes being the coordinates and the edges being the transformations while the URDF trees describe the physical structure of the robot with nodes being links or joints and edges being transformations between links and joints.

The relation between these two trees can be defined as that the coordinate frames of the tf2 trees are created using physical descriptions of the robot described in URDF trees.

1. We can see the result of using tf2 library and launch files to achieve our goal.



1. Using ROS-Gazebo bridge, the result and terminals can be shown below.

