Project Report

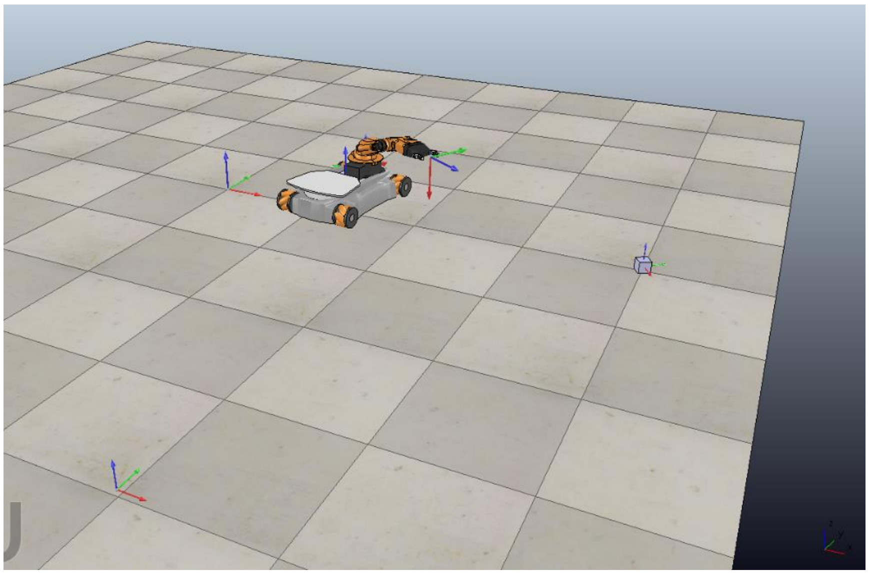
1. Summary

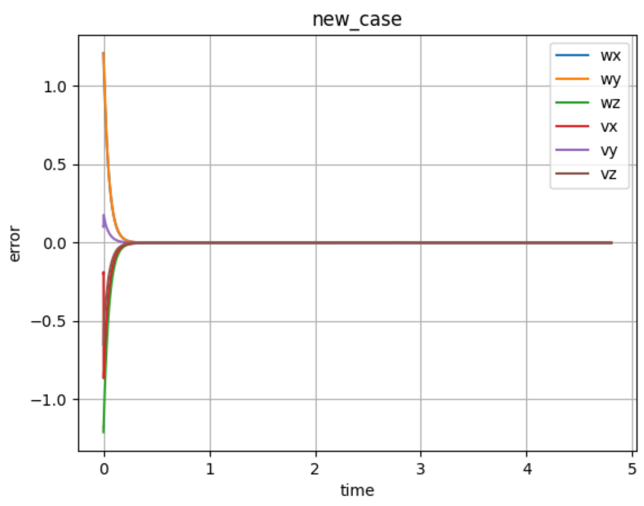
   In this project we are using python to code a software that can generate certain csv file which can manipulate youBot. YouBot is a robot that has a 5R arm and a base with 4 wheels and our task is to make it relocate a cube. All the youBot movement will be simulated by the CoppeliaSim simulator. The whole project is divided into 4 part and each part has a component.

   1) For component one, we are writing a function that input the current state and current velocity generate the next state. Here in this project state is defined as follow:
      The velocity is defined as follow:
   2) As for second part we are trying to build up a software that can generate a trajectory just for the end-effector to grab and release the cube. The overall trajectory can be break down into eight segments and each can be composed with N small trajectory.
   3) Component 3 it's about feedback control. we input the current real end-effector configuration, current desire end-effector configuration and the next step desire end-effector configuration. Then compute the end-effector twist and the commanded speed of weed and joints.
   4) The last part, is just wrap everything above up. Beyond that we will have the feedback control with PI gain matrices to help control the robot movements

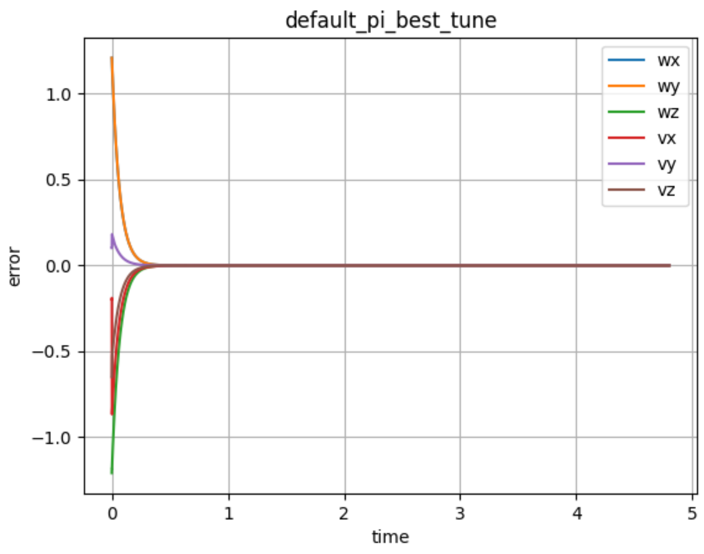
2. Result

   1) The new task here we set the initial configuration of cube as: (-pi/3,2,0), and the goal as: (0,1,-2)。For pi control we set Ki at 15 (on a 6 by 6 the diagonal matrix)

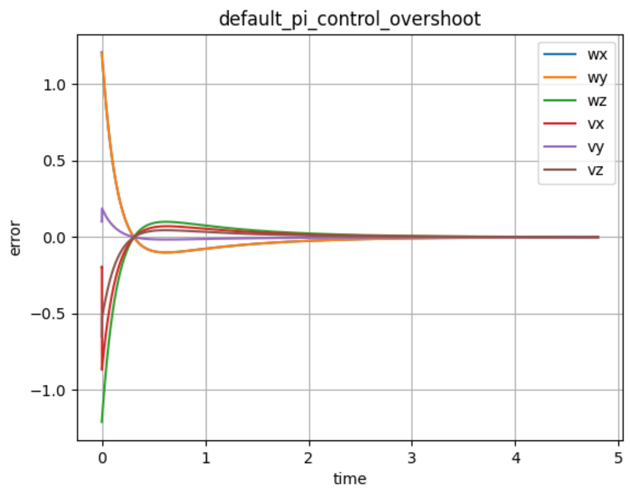


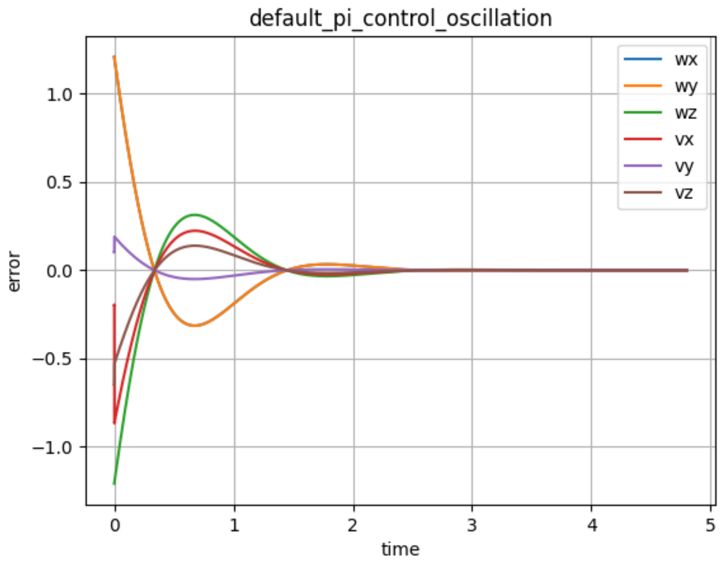


2) As for the best case here, we are using feedforward-plus-P (no integral term),and we set them as 15 (on a 6 by 6 the diagonal matrix). The result of the error plot is as follow which we can see the error decrease very quick and smooth with no overshoot and end up with little error.



3) The third case here we are trying to present a case with less tuned controller which here we have two, one with overshoot but no oscillation and another one with oscillation. We are using feedforward-plus-PI controller for both and the Kp and Ki are as follow.





3. Discussion
   1) Incorporate the system with the integral gain helps the system lower down the eventual error. However, when the proportional gain is big enough, it might be not that effective. As for the disadvantages it has, is that it can create overshoot, even oscillation and slow down the error decrease. The reason why even with very small Ki there still overshoot exist is that the process of integral is accumulate all the value (in this case its error) we have before current step which can led to a very important trait is that it is always lagging. Even after the error it's been eliminated, it will still be trying to "eliminate" the error, which consequently lead to create new error.
   2) One of the most important reasons is that sometimes when we trying to calculate the rotate angle form one to another, we may just simply do subtraction instead of finding the shortest route. For example, when trying to get from 0 degree to 359 degree, we might end up turning 359 degrees counterclockwise while we can simply do it in turning 1 degree clockwise. This would simply increase the joint velocity and end up getting error.

3) Ideally it can happen. But there will be too much computation and taking a unworthily large amount of time, since all of the joints movement are deeply related which can create a tough numerical problem. However, under one condition this could happened. If we only have one revolute joint and the other are all prismatic joints this can easily happen.
4) If we need to implement the torque control to youBot (instead of speed control), we can use the InverseDynamics function from Modern Robotics. The extra input we need will be joints acceleration, spatial force applied by the end-effector, and spatial inertia of the links. Also, the extra pre-defined constants we need will be list of link frames and gravitational acceleration