Robot Manipulation Capstone

Overview

The capstone project in Robot Manipulation aims to integrate concepts from the Modern Robotics software in an automated manner. It consists of three main components: **Milestone 1**, **Milestone 2**, and **Milestone 3**.

Milestone 1 is responsible for receiving the requested speed from the controller. Using this input and the robot's current configuration, it calculates the configuration for the next state.

Milestone 2 focuses on generating the desired trajectory for the entire procedure at the start. The total time of each trajectory segment is crucial: a shorter total time results in higher speeds for the same path length, which can lead to unstable trajectories. Proper consideration of segment durations ensures the trajectory remains smooth and achievable.

Milestone 3 implements a **feedforward + feedback (FF+FB)** control strategy. At every time step, it gathers the desired trajectory and the current configuration. Using the FF+FB control law, it computes the required twist for the robot to guide the end-effector to the desired configuration.

To achieve this, the *end-effector Jacobian* is constructed at each time step since the Jacobian depends only on the configuration. The Jacobian is then used to convert the desired end-effector twist into a list of wheel speeds and arm joint speeds. These speeds are passed to Milestone 1, which calculates the next configuration.

By repeating this process 1525 times, the robot's full trajectory can be obtained. In this implementation:

- Total time: **15.26 seconds**,
- Time step (Δt): **0.01 seconds**,
- Trajectory scaling factor (K): **1**.

Joint Limit

This part is created to avoid the self-collision with the chassis and other joints in a realatively coarse way. The highest probability of the collision appears when the end-effector is moving towards the chassis. To avoid I constrain the joint 1 within a range from -2.93 to 2.93, which there is a slightly

gap from the angle where the end-effector touch the two sides of the chassis. As the range of the $joint_1$ decreased, we can less decrease the range of the other joints because we do not need to consider the constrain given by the chassis collision. The range for $joint_2$ would be $[-2, 2]$. For the $joint_3$ and $joint_4$, the porblem of singularity need to be taken into account. I set the upper bound of the range for both joints to be -0.02, which is used to play a role as "0". The lower bound of the both joints is -2.5. The $joint_5$ seems not to change that much among all these cases, therefore we just set it within the range$[-1, 1]$.

The function" TestJointLimit" would firstly extract the current joint angle from the current robot configuration, then compare with their limit one by one, and finally returns a list showing which joint is out of the limit bound. With the information, we can shut down the corresponding joint from moving in the next iteration by setting its jacobian colume to be all zero.

Result

Three cases are provided to test the robot's performance, according to the theorem of PI control law, the relation between K_i and K_p needs to be $\frac{K_p^2}{4} = K_i$ to reach the critically damping model. Besides that, because we added the joint limit and excute the limit to the robot dynamics. The

move which should be down by the joint out of the limit needs to be compensated by the other joints, which will introduce new unstability. In this case, it's risky to use either large K_p or K_i. Therefore we choose $K_p = 1$ and $K_i = 0.25$ fot both the "Best" and the "New Task" case. For the "Overshoot" case, we will use$K_p = 3$ and $K_i = 2$. An interesting thing is no matter what value K_i is, a large K_p would also introduce the oscillation, the reason for that might be the joint limit stuff we mentioned above.