ME449 Final Project README

Ruotong Jia student number: 3222201

1 Notations and Definitions

The notations used in this report are listed as below

- $[\phi]$ - Orientation of the chassis
- x, y - Cartesian position of the chassis
- $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ - Joint angles of the robotic arm
- $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ - Joint angles of the robotic arm
- $\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, \dot{\theta_5}$ - Joint angular velocities of the robotic arm
- $\psi_1, \psi_2, \psi_3, \psi_4$ - Wheel angles
- $\dot{u_1}$, $\dot{u_2}$, $\dot{u_3}$, $\dot{u_4}$ - Wheel angular velocities

2 How to use the code

Run the input script that contains inputs for all cases: run_inputs.py. Please note that to turn on or off joint_limiting feature, in each code, simply change IF_JOINT_LIMIT to True or False.

Once the input script finishes running, two files will be generated: trajectory.csv and configurations.csv. trajectory.csv is the same desired trajectory file as in Milestone 2. configuration.csv is the final csv file for visualizing the robot behaviour in scene 6.

3 Controller Tuning

3.1 System Setup

The position error of the end effector is $x = -1.0$, $y = -1.0$, $theta_z = \frac{\pi}{6.0}$ In this section, an initial position error is added to the configuration chassis, and the chassis configuration is

$$[\phi, x, y, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \psi_1, \psi_2, \psi_3, \psi_4] = [\frac{\pi}{6.0}, -1.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$$

Two singularity avoidance methods are applied: zero tolerance in Jacobian and joint limits. The Jacobian zero tolerance for singularity avoidance is 10^{-3}

Joint Limit Feature
For implementing the full controller, joint limits and speed limits are added later to mimic real life behaviour of the robot. The speed limits are

$$[wheel\_speed\_lim, joint\_speed\_lim] = [30.0, 60.0]$$

Joint limits are implemented when a set of joint and wheel velocities are generated. If the set of joint and wheel velocities will make the robot break certain joint limits, a new set of joint and wheel velocities will be generated by setting an all zero column in the Jacobian, such that the violating joint or wheel velocities remaining the same. The joint limits (from joint 1 to joint 5) are

$$[joint\_limits] = [[-2.297, 2.297], [-1.671, 1.853], [-1.6, -0.2], [-1.717, -0.2], [-2.89, 2.89]]$$

3.2 Tuning Process

(a) Using feedforward term only

The correctness of the code and system setup is ensured by using only the feedforward term when the initial error is zero. The behaviour of the robot on Vrep is close to the desired trajectory with slight position errors. When the specified position error is applied, however, there is always a visible error and the robot never reaches the goal location.

(b) Using K_p + feedforward and increase K_p With the specified initial robot configuration and no joint limits, the controller corrects its initial error and creates a reasonable trajectory as K_p increases. When K_p becomes too high, the robot shows significant oscillations and overshoot. One thing worth noting is that after adding joint limits and speed limits, there are steady state errors from the first stand off point to the second stand off point. This shows the restricting effect of these limits.

(c) Using K_p and K_i terms and increase K_i values When the robot has steady state error as specified in (b), small K_i values will help reduce the steady state errors. However, as K_i increases, oscillation is introduced to the system before the steady state errors are eliminated.

(d) **Special note on controller gains versus joint and speed limits**
When joint limit is taken off, the controller gains need to be made smaller

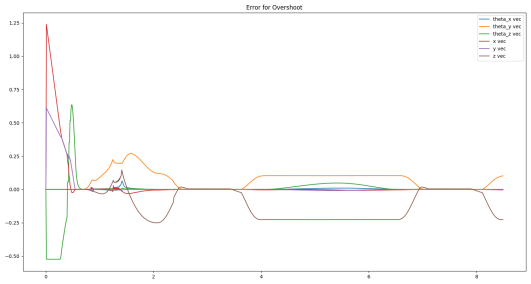

Figure 1: Error Dynamics of the Overshoot Case

so that the robot does not demonstrate a non-converging behaviour. Reason for that is with a set of joint limit, the robot might need to move its joint faster within the speed limits to compensate for not being able to achieve the previous end-effector position. Therefore, higher controller gains are required in that case.

4 Result Explanation

4.1 Overshoot

A P controller is used for introducing overshoots and oscillations. This P controller solves the pick and place problem but cannot fully eliminate steady state errors as shown in Figure 3.

$$K_p = \begin{bmatrix} 30.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 20.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 30.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 200.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 200.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 30.0 \end{bmatrix}$$

4.2 Best Result

This is a feedforward + PIcontroller, which solves the pick and place problem but cannot fully eliminate steady state errors as shown above. When K_i terms

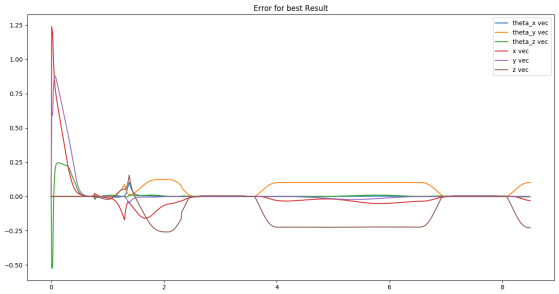

Figure 2: Error Dynamics of the Best Case

increase, the steady state errors are reduced but more oscillations will come in before they are fully eliminated.

The error dynamics is illustrated in Figure 2

$$K_p = \begin{bmatrix} 30.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 20.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 30.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 20.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 30.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 30.0 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 \end{bmatrix}$$

4.3 New Case

The new initial position of the block is $x = 1.5m$, $y = 0m$, $\theta = 0rad$ The controller is the same as in best case. The performance of the controller is shown in Figure **??**

$$K_p =$$

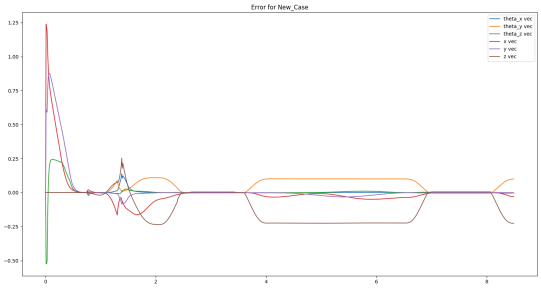

Figure 3: Error Dynamics of the New Case

$$\begin{bmatrix} 30.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 20.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 30.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 20.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 30.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 30.0 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 \end{bmatrix}$$

4.4 Future Work

Currently the best case controller works with an initial configuration that is vertical to the plane. When a horizontal initial configuration is applied, the controller's error will not converge during the time frame that is used to generate the desired trajectory. Therefore, more tuning could be done to resolve this issue.

Another potential improvement is the smoothness of the best case controller. Even though the controller does not intersect itself, and most of the time it demonstrates a smooth behaviour, there are still two "jerky" moments when it

reaches down to the first standoff position. Successful future tuning might be implemented to improve the smoothness of the controller.