MAE 204 FINAL PROJECT

NANDAN SESHADRI - A59018150

Report summary:

Software-
Matlab was used to program milestone1, milestone 2, function of the feedforward + feedforward control and the wrapper script on live script while commenting out the other tasks.
Gazebo was used to simulate the UR3e robot motion for 3 different situations- best case, overshoot and new task.

Approach -

- Reference Trajectory Generator(Milestone 1) -
  We divided the trajectory generator function into 6 different trajectories using CartesianTrajectory and ScrewTrajectory from the modern robotics textbook functions while keeping a track of the gripper's open and close configuration. The paths are -

1. Trajectory 1 - Gripper is open
2. Trajectory 2 - Gripper is open
3. Trajectory 3 - Gripper is closed at the beginning of the trajectory
4. Trajectory 4 - Gripper is closed
5. Trajectory 5 - Gripper is closed
6. Trajectory 6 - Gripper is opened at the beginning of the trajectory

   The end effector trajectories (containing the rotation and position i.e 12 values) were converted into a 1x12 vector and stored into a csv and then run through the provided trajectory test function to plot the end effector trajectories for iteration.
   End Effector Trajectory Simulation: trajectory.avi

- Kinematics Simulator(Milestone 2) -

  The robot kinematics is simulated using the current joint angles and maximum joint velocities. A simple Euler equation was used for the computation-

  The NextState function was provided with three inputs - the current joint angles, joint velocities, maximum joint velocities and the timestep. The next waypoint's joint angles were calculated by summing the initial joint angles and product of joint velocities and time steps.
  A test loop was created to run for 100 iterations and the next joint angle and gripper position for the robot joints were calculated. Then the value of current joint angles was updated with the next joint angles for the next iteration. All the new joint angle values and gripper position were stored and exported into a csv file. The csv file was simulated in gazebo. The initial joint angle was set based on the values provided in the report instruction i.e. [θ1... θ6] = [-pi/6, -pi/2, -pi/2,-pi/2, 5pi/6]. The maximum joint velocities

were set using the values provided in the UR3 specification sheet i.e. max_v = [pi,pi,pi,2pi,2pi,2pi] rad/s. A zero gripper position value was assigned and outputted by the function. For simplification only 1 joint velocity was provided to the robot to ensure that the robot moved in the right direction.

- Feedforward + Feedback Control

  Using the task-space motion control to control the UR3e robot-

$$\nu_b(t) = \left[Ad_{X^{-1}X_d}\right]\nu_d(t) + K_p X_e(t) + K_i \int_0^t X_e(t)dt$$

  Where -

  $\left[Ad_{X^{-1}X_d}\right]\nu_d(t)$ - Feedforward control

  $K_p X_e(t) + K_i \int_0^t X_e(t)dt$ - Feedback control

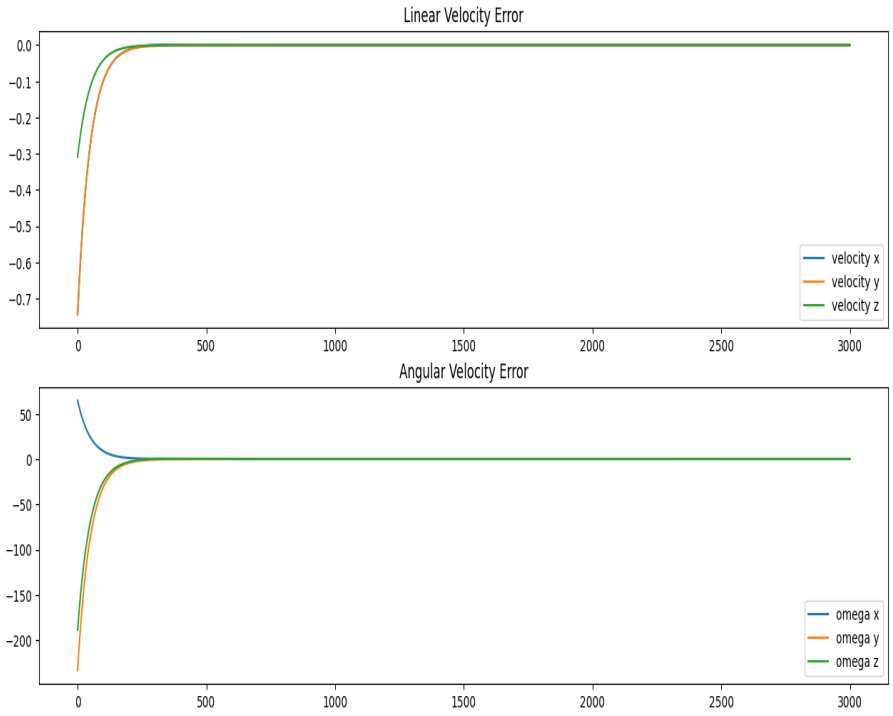
  The Feedforward + Feedback control approach uses a proportional-derivative (PD) controller to decrease the deviation between the robot's desired and actual positions. The function takes the actual, desired and the next desired transformation matrix of the robot, the control measures - Kp and Ki, the body Jacobian axis, timestep between each configuration as well as time to calculate the error twist and joint velocities for the robot. The error twist is calculated using the equation provided above (v_b(t)) and the joint velocities were calculated by multiplying the pseudoinverse of the body Jacobian screw axis with the error twist. The Feedforward + Feedback control function helps to minimize the error between the current and desired positions, and help the end effector perform efficiently.

- Full Program / Wrapper Script
  The wrapper script combines all the functions created above to simulate the robot to pick up and move the cube to the desired position. In the beginning a reference trajectory is generated using the trajectory generator function. A loop is run for N = time/timestep iterations. The current, desired and next position trajectory values are assigned from the reference trajectory, and the error twist and joint velocities are calculated using the Feedback Control function. These joint velocities are then used to calculate the next joint velocities using the NextState function. The trajectories and current joint angles are updated for the next iteration. The joint velocities obtained from NextState function are stored into a csv file and fed into gazebo to simulate the robot.

Results-

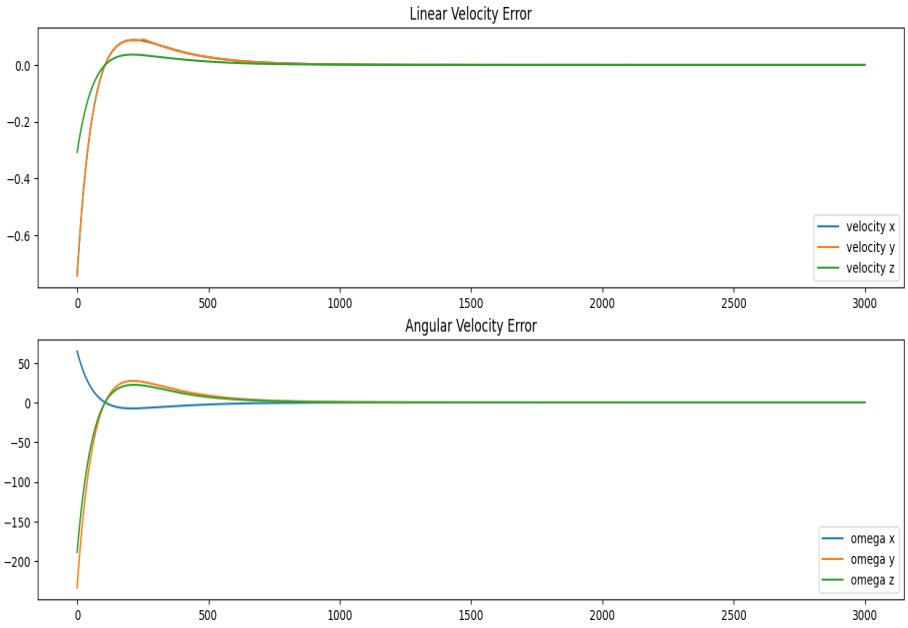
1. Best case - The best case plot shows that the errors converge smoothly without any overshoot.

- Controller: Feedforward-plus-PI control
- Gains: Kp = 2 and Ki = 0
- Plots



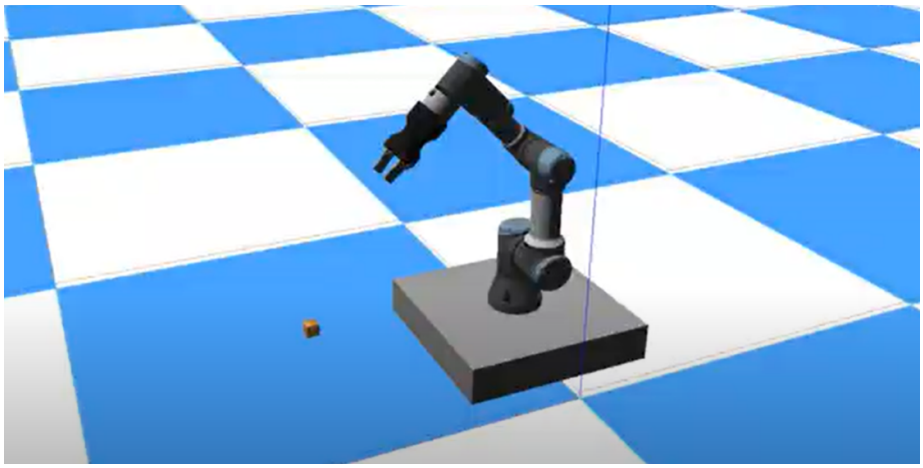
- Gazebo simulation video - [Best_case.mp4]

2. Overshoot - For the best case scenario,. From graph, there is a spike in the steady state in the graph compared to best case.
    - Controller: Feedforward-plus-PI control
    - Gains: Kp = 2 and Ki = 1
    - Plots

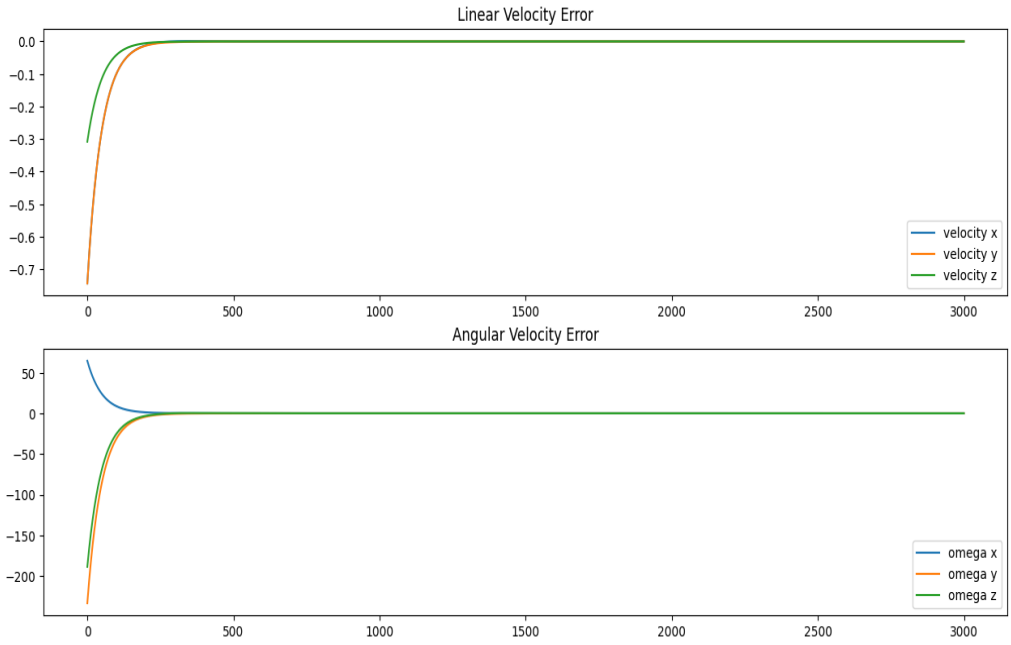


- Gazebo simulation video: [Overshoot.mp4]

3. New task - The gains used in the best case scenario is re-used in the best task (kp = 2; ki = 0). However, a different cube configuration and the joint angles are selected. They are -
   Px = - 450
   Py = -550
   Pz = 0
   Joint angles = [0, -pi/2, pi/4, -pi/2, -pi/2, 0]



- Controller = Feedforward-plus-PI control
- Gains = kp - 2; ki - 0
- Plots = From the below plot, it looks very similar to the best case since similar gains are used and there will not be a big change in error twist.



- Gazebo simulation video - [New_task.mp4]

From the below plot, it looks very similar to the best case since similar gains are used and there will not be a big change in error twist.

Observations:

While writing the wrapper code, we didn't get any error but our robot was not operating as expected, even though the individual functions were working properly. This error was fixed by updating the index values being used in the wrapper script. In addition, the gripper value that was outputted in the NextState function was removed in the wrapper code, as the reference trajectory was dictating the open and close position of the gripper. Having the additional gripper output was confusing the robot and made the grip open and close between each iteration.

Comprehensive section

Goal: THe UR3e robot is supposed to run in the best case scenario i.e., have gain of kp = 2 and ki = 0.
For both the cases, the initial and final configurations used for Tsc are -

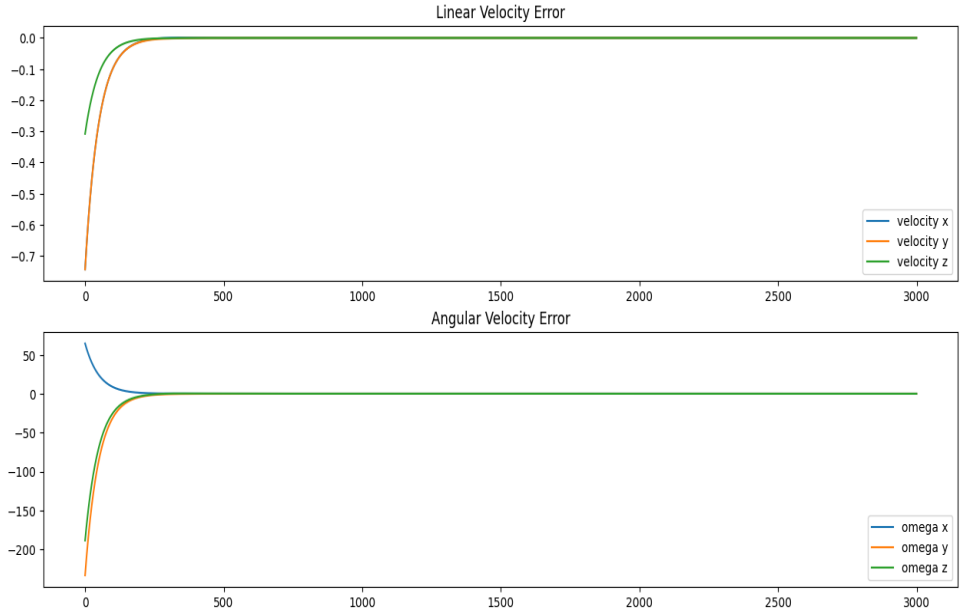
Tsc_initial = ([1, 0, 0, 450]
[0, 1, 0, -300]
[0, 0, 1, 15.25]
[0, 0, 0, 1])

Tsc_final = ([1, 0, 0, -500]
[0, 1, 0, -300]
[0, 0, 1, 15.25]
[0, 0, 0, 1])

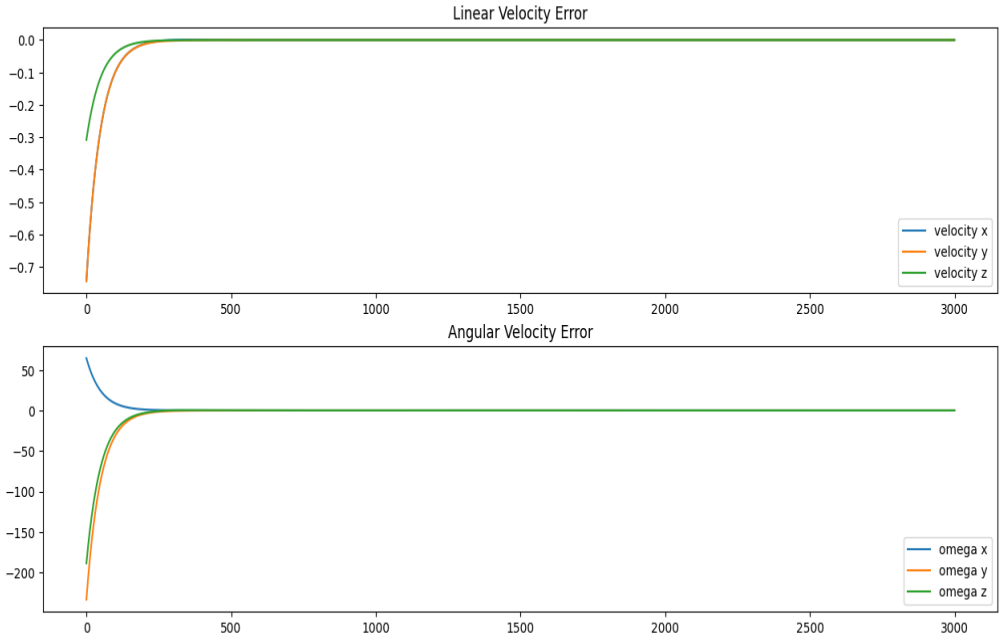
Modification:

- The trajectory generation function was modified to obtain the desired configuration. In the main project, **CartesianTrajectory** and **ScrewTrajectory** functions were used.
- However the above functions take in the interpolation of the initial and final configuration of the end effector pose.
- Thus, the **JointTrajectory** function is used to solve this problem. This function takes in the interpolation of the joint angles.
- We ran into multiple singularity problems while selecting the required joint angles. However, the angles were rotated to obtain the desired configurations for different cases.

1. When the cube's final configurations : $(x, y, \theta) = (-500\ mm, -300\ m, \pi/2\ rad)$
    - Angles were assumed and an initial guess was taken and passed through the **IKinSpace** space function. These angles were used and 180 degrees were added to joint 1(base joint) to obtain the required configuration.
    - Finally the FKinSpace function was used to obtain transformation matrices for all the joints.
    - Gazebo simulation: Part 1 simulation
    - CSV file: Comp_1_csv
    - Error twist plots-



2. When the cube's final configurations : (x, y, θ) = (− 500 mm, − 300 m, π/2 rad) while the first joint has limits between [-pi/6 pi/6]
    - Since joint limits are implemented for this part, we need to take a different approach.
    - To solve this issue, we divide the simulation into 2 parts.
    - In the first part, the robot moves from initial position to middle position. In the second part, the robot moves from middle to final position.
    - From the initial to middle position, we implement -180 deg to joint 3. And finally, we implement +180 degs to joint 5 to move from middle to final position where the robot places the cube in the desired position.
    - Gazebo simulation: Part 2 simulation
    - CSV file : Comp_2_csv
    - Error twist plots-



Acknowledgements: I would like to mention Sharath Matada and Mihir Kulkarni who helped me under the concepts needed to tackle this comprehensive exam better. We also took academic integrity seriously.