RBE500 Group Assignment – Part 2

Group 10

Team Members: Aniket Patil, Venkatesh Mullur, Febin Fredi

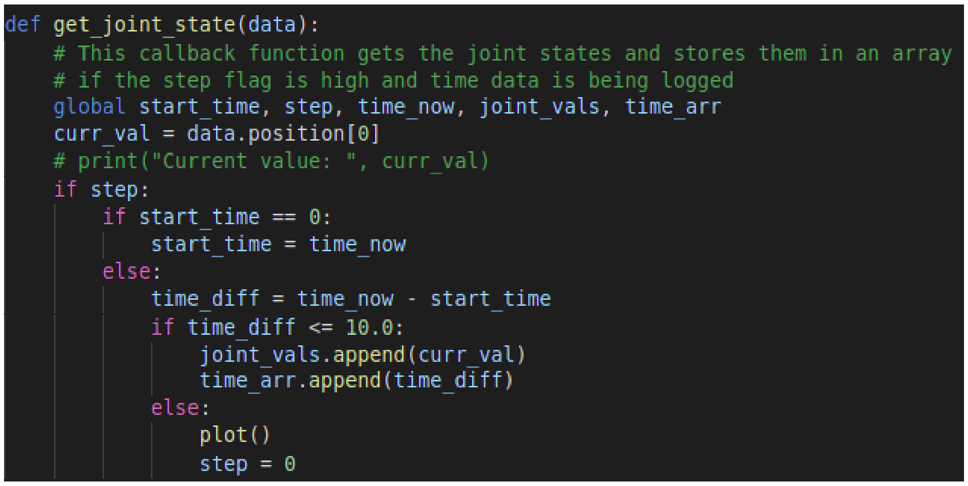
1. To implement the controllers, we have used the **ros_control** package.

- We added the **<transmission>** element in our Xacro file which specifies the type of hardware interface between the controller and actuators.
- We used **hardware_interface/EffortJointInterface**.
- We added **gazebo_ros_control** plugin in the robot.gazebo file, which is also included in the Xacro file.
- We created a robot_config.yaml file which specifies three position controllers of type **"effort_controllers/JointPositionController"** and specifies the P, I and D values for the controller. We modified the values in this file in order to tune our PD controllers for all 3 joints.

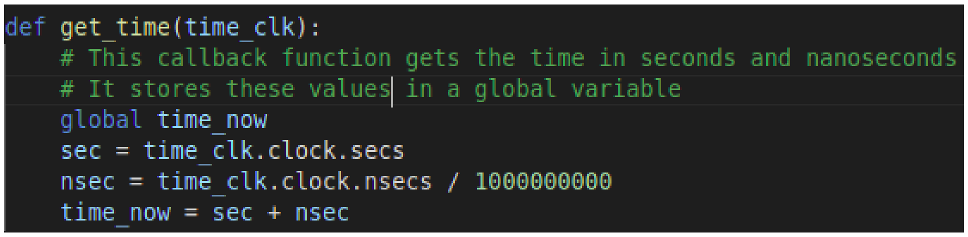
2. We have created a python node which we used for plotting and then tuning the transient response of our robot joints.

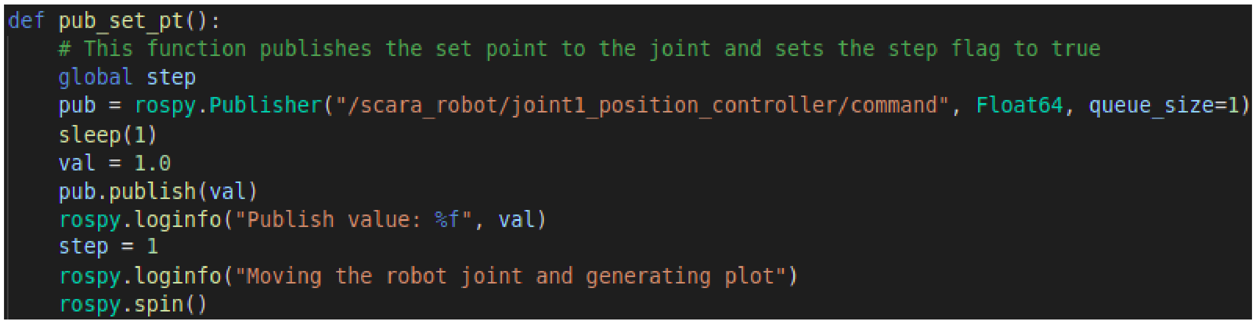
Algorithm for the Python code:

- Publishes a value to the topic **"/scara_robot/command"** that is equal to 1 (step input)
- When this value is changed, a flag (step) is set to 1
- The two Subscribers are continuously subscribing to the topics **"/clock"** and **"/scara_robot/joint_states"**
- When the step flag is high, the time values and joint state values are appended to 2 arrays
- Also, we note the start time at this moment
- At every iteration, we append new values to both these arrays if the time elapsed is less than 10 seconds
- When 10 seconds are over, we stop appending values to the array and plot the values collected in the two arrays
- Another array is created in order to show the reference value (set-point) given to the system



```python
def get_joint_state(data):
    # This callback function gets the joint states and stores them in an array
    # if the step flag is high and time data is being logged
    global start_time, step, time_now, joint_vals, time_arr
    curr_val = data.position[0]
    # print("Current value: ", curr_val)
    if step:
        if start_time == 0:
            start_time = time_now
        else:
            time_diff = time_now - start_time
            if time_diff <= 10.0:
                joint_vals.append(curr_val)
                time_arr.append(time_diff)
            else:
                plot()
                step = 0
```

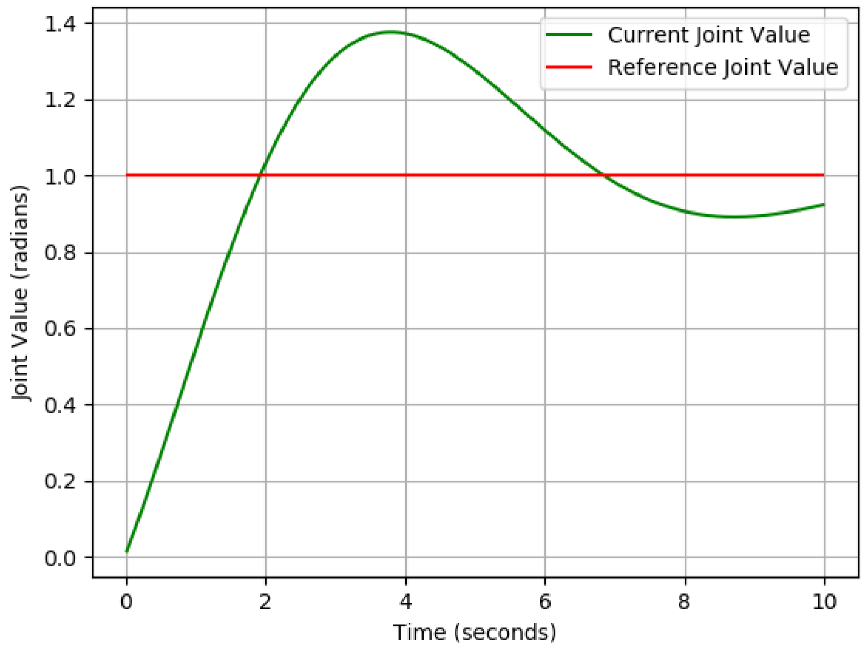

```python
def get_time(time_clk):
    # This callback function gets the time in seconds and nanoseconds
    # It stores these values in a global variable
    global time_now
    sec = time_clk.clock.secs
    nsec = time_clk.clock.nsecs / 1000000000
    time_now = sec + nsec
```


```python
def pub_set_pt():
    # This function publishes the set point to the joint and sets the step flag to true
    global step
    pub = rospy.Publisher("/scara_robot/joint1_position_controller/command", Float64, queue_size=1)
    sleep(1)
    val = 1.0
    pub.publish(val)
    rospy.loginfo("Publish value: %f", val)
    step = 1
    rospy.loginfo("Moving the robot joint and generating plot")
    rospy.spin()
```

Initially joint 1 is set to **continuous** and joint 2 and joint 3 is set as **fixed**. Now when the node runs, using the above algorithm, we see the plot for the untuned values of joint 1(without the interference of the other 2 joints).

This is repeated for joint2 and joint3 where one joint is set continuous and other joints are set as fixed. Thus, getting joint position array for all three joints which is plotted using **matplotlib.pyplot.plot** function.

Joint 1:

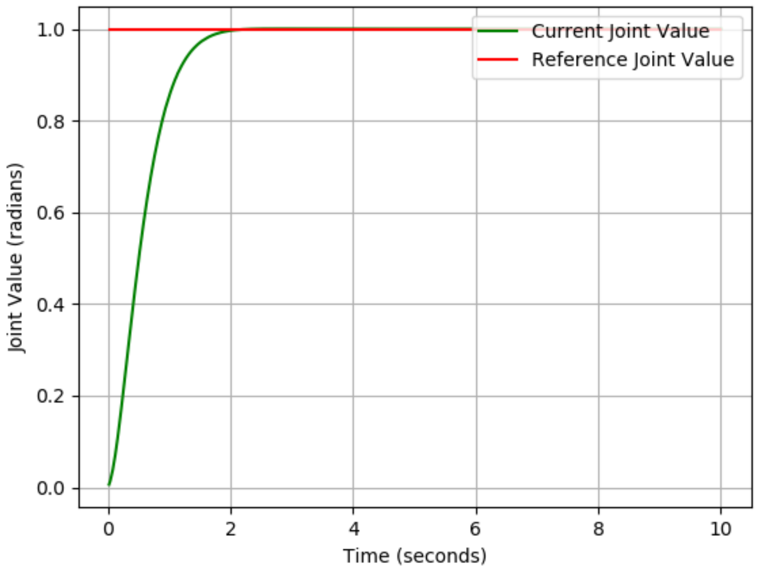


Joint 1 (pre-tune plot) **PID gains: P = 10 D = 10**

Transient Response Characteristics:

Rise Time (Tr): 2 seconds (approx.)

Settling Time (Ts): >10seconds
Peak Overshoot: 0.38 radians (approx.)

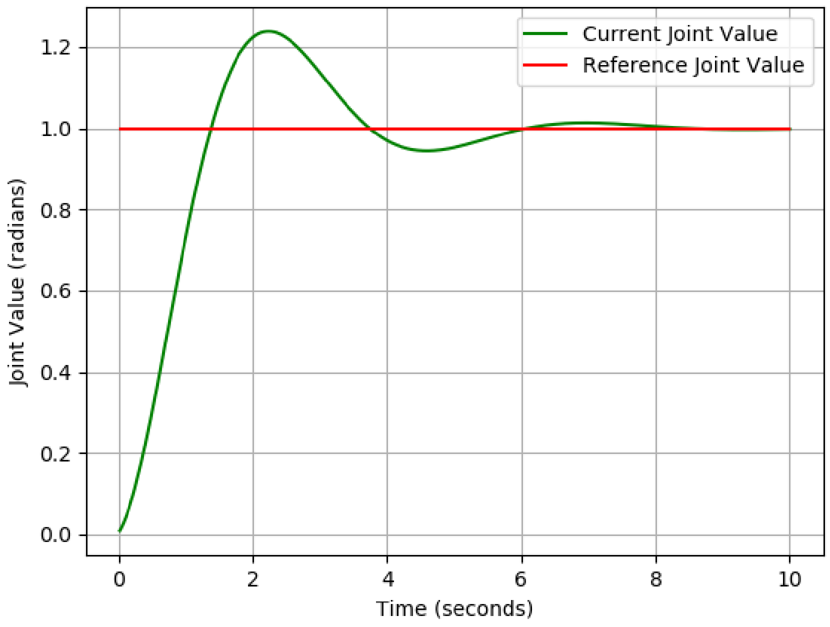


Joint 1 (post-tune plot) **PID gains: P = 200 D = 120**

Transient Response Characteristics:

Rise Time (Tr): 1.4 seconds (approx.)
Settling Time (Ts): 2 seconds
Peak Overshoot: 0.0 radians

Joint 2:

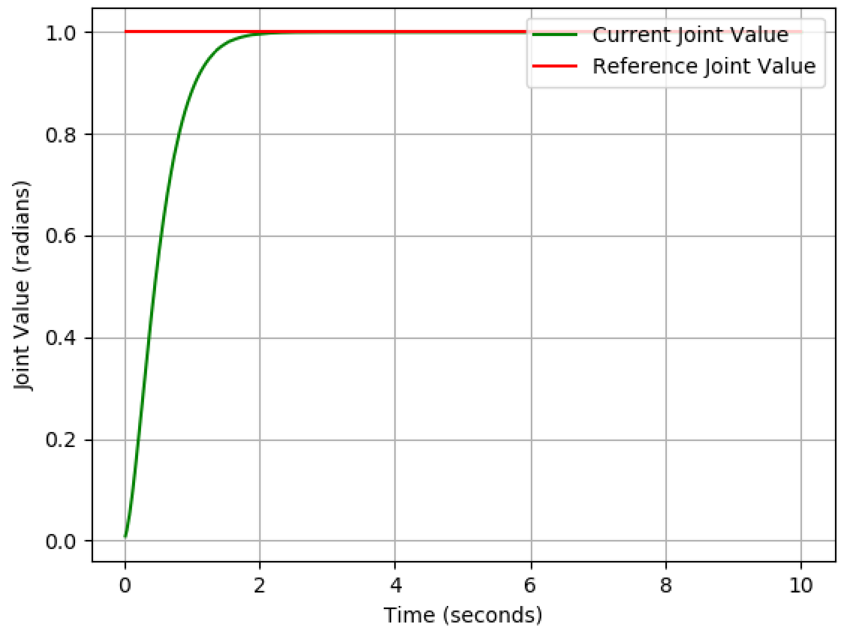


Joint 2 (pre-tune plot) **PID gains: P = 10 D = 5**

Transient Response Characteristics:

Rise Time (Tr): 1.5 seconds (approx.)

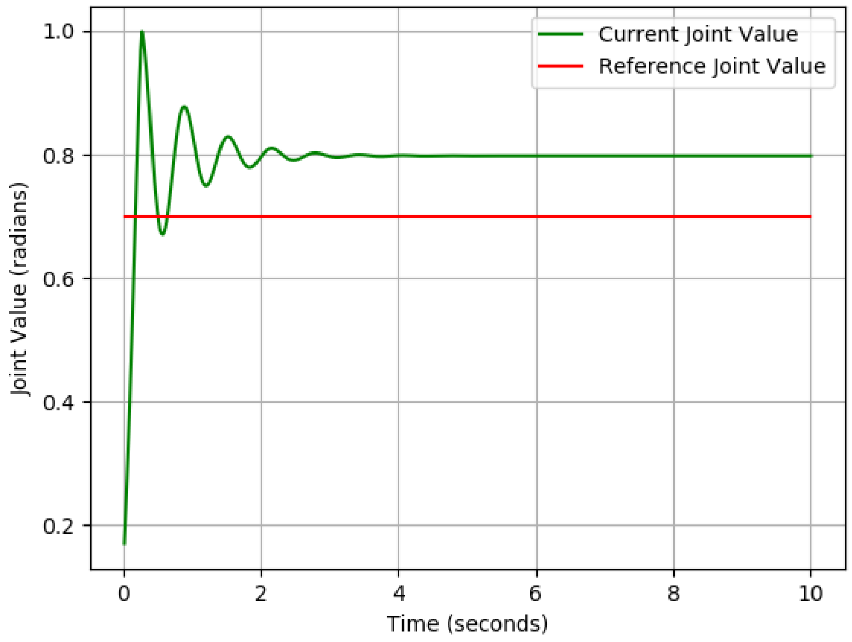
Settling Time (Ts): 10 seconds (approx.)
Peak Overshoot: 0.24 radians (approx.)



Joint 2 (post-tune plot) **PID gains: P = 55 D = 30**

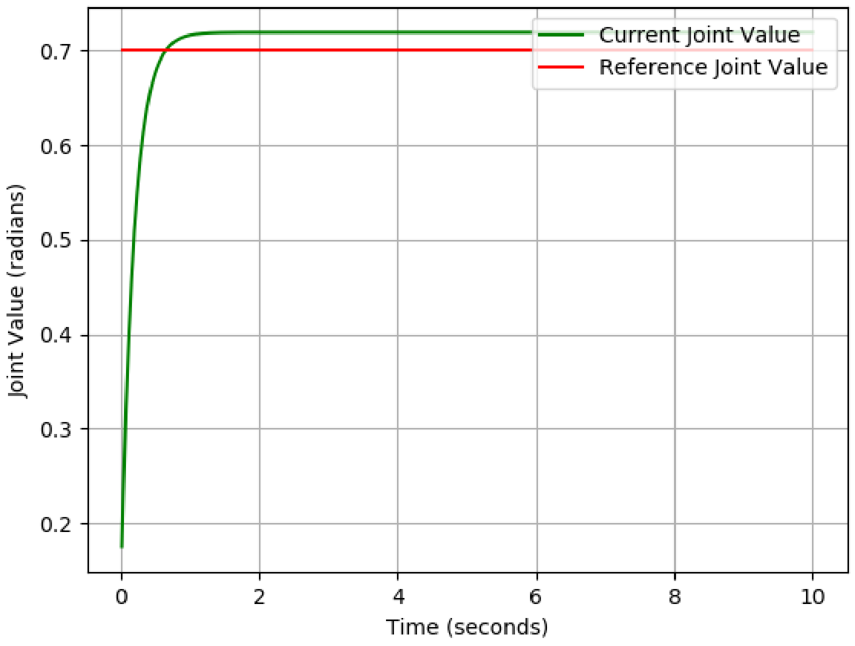
Transient Response Characteristics:
Rise Time (Tr): 1.4 seconds (approx.)
Settling Time (Ts): 2 seconds (approx.)
Peak Overshoot: 0.0 radians

Joint 3:



Joint 3 (pre-tune plot) **PID gains: P = 10 D = 0.3**

Transient Response Characteristics:
Rise Time (Tr): 0.2 seconds (approx.)
Settling Time (Ts): 4 seconds (approx.)
Steady state error: 0.1 radians
Peak Overshoot: 0.3 radians



Joint 3 (post-tune plot) **PID gains: P = 50 D = 10**

Transient Response Characteristics:
Rise Time (Tr): 0.7 seconds (approx.)
Settling Time (Ts): > 10 seconds
Steady state error: 0.02 radians
Peak Overshoot: 0.02 radians