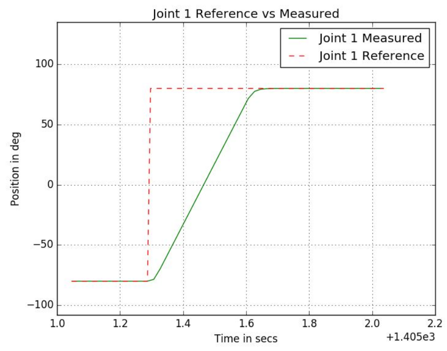
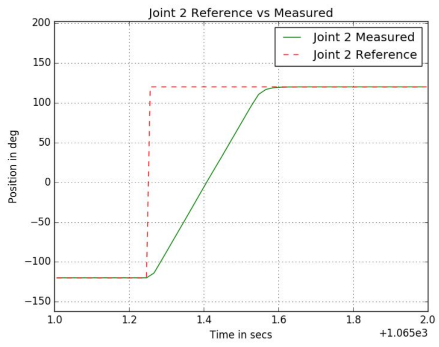
# RBE 500 : FINAL ASSIGNMENT - PART 3 REPORT

Team 7 : Neel Dhanaraj, Sabhari Natarajan, Sidharth Sharma

1) Velocity Level Kinematics : For this a node was written, with two services, one for Forward Velocity Kinematics and other for Inverse Velocity Kinematics. The Jacobian was calculated from the joint states read from gazebo. The forward velocity kinematics service was used to get the End-Effector cartesian velocities from a given set of joint velocities. The inverse velocity kinematics service was used to get the joint velocities for a required End-Effector cartesian velocities. As this is a 3 DOF arm, we can only control 3 dimensions in the 3D space (i.e. $v_x, v_y, v_z$), $\omega_x, \omega_y, \omega_z$ cannot be controlled. Hence, only the first three rows (linear component) are considered for the Jacobian in inverse velocity kinematics. This gives us a square matrix for which inverse is defined.

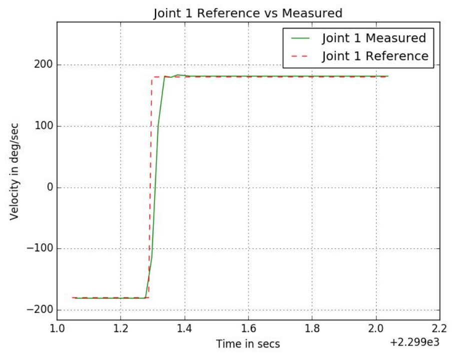
2) The position controller was extended to Joint 1 and Joint 2. The velocity limits for Joint 1 and 2 were set to 8 and 12 rad/sec respectively (This was based on a FANUC SCARA robot). After individual tuning the values need to be tuned more when both joints were moved simultaneously, to negate the effect of one joint on another. The following images show the response after PID tuning.

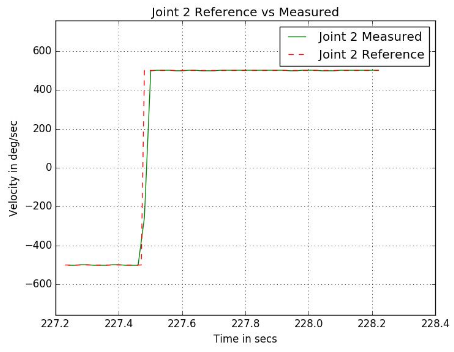




Joint 1 : p: 1000.0, i: 0.00, d: 20.0 Joint 2 : p: 500.0, i: 0.00, d: 10.0

3) Similarly, the velocity controller was also tuned and below are the results.

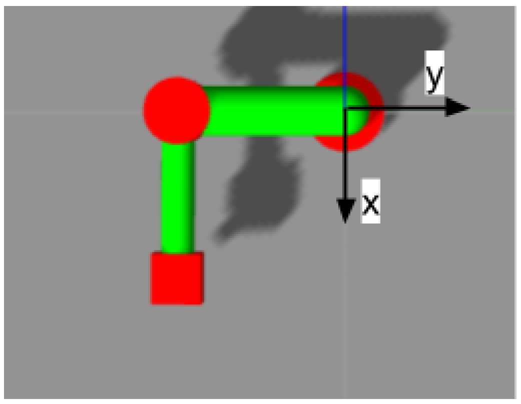




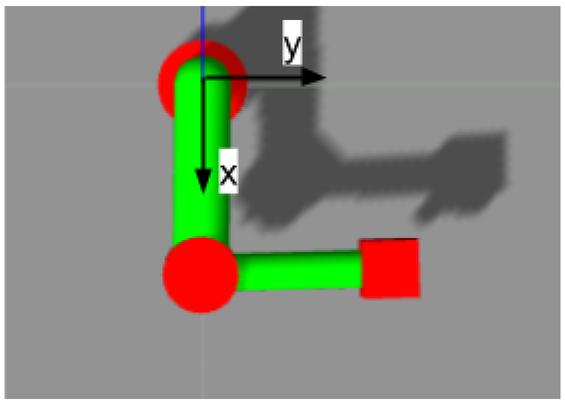
Joint 1 : p: 40.0, i: 0.00, d: 0 Joint 2 : p: 6.0, i: 0.00, d: 0

4) With the position and velocity controller tuned, both of these were integrated. The constraint in using them is that, at a given instant only one controller can be running. So, for this we load both the controllers from the launch file, but start only the position controller. During runtime we call a service called ‘switch_controller’ that helps us switch from one controller to another.

5) Now, we have the ability to switch controllers. When we start the code we first move away from a non-singular configuration using position controller. After this we switch to the velocity controller and give it a command to move with $v_x= 0$ ,$v_y = 0.2$ ,$v_z = 0$ m/s for a duration of 2 sec. Following images show the change in position of SCARA arm after the command

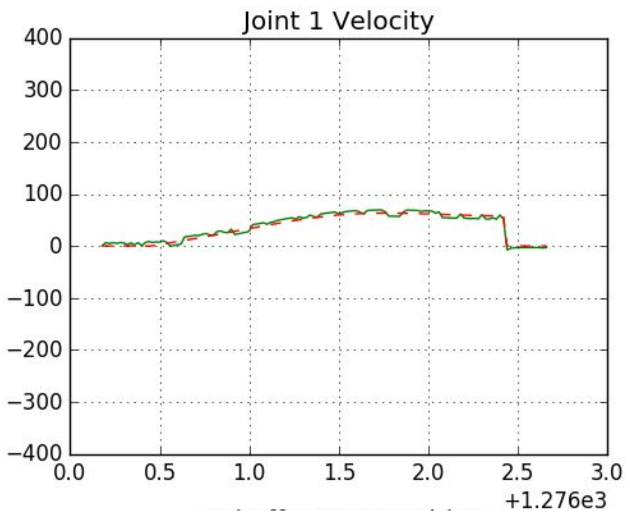


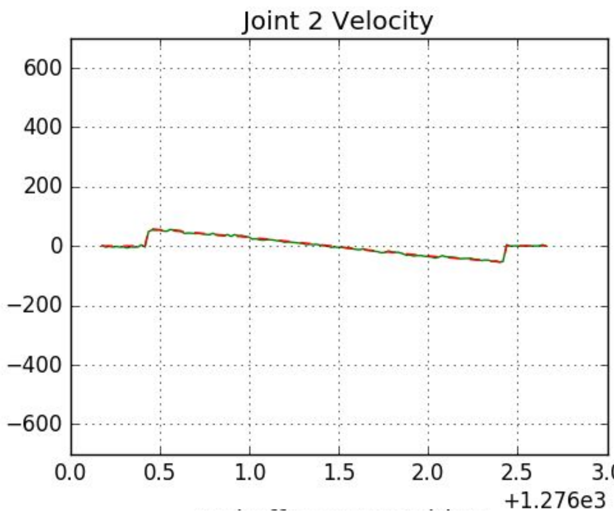
Start Position
(q1 : -90°, q2 = 90°)



End Position

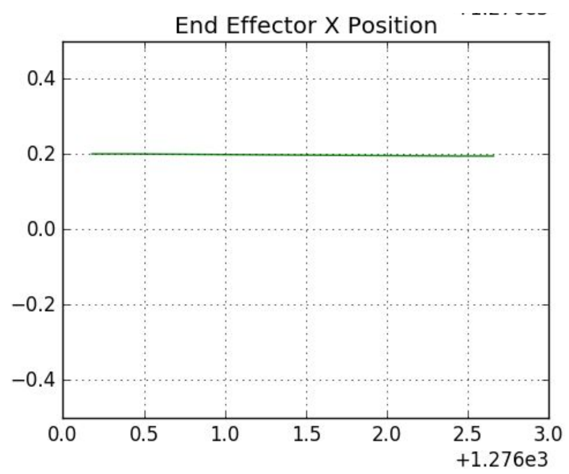
Following graphs show the Joint Velocity Reference (Red dotted line) vs Joint Velocity Measured (green solid line) for the above command. X-axis is the time in seconds and Y-axis is the velocity in deg/sec.

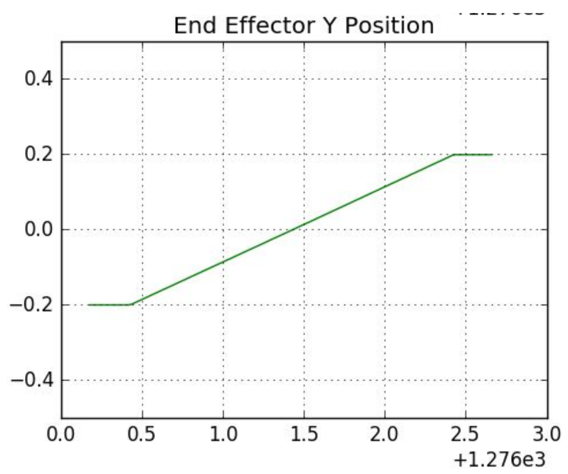




In these graphs we can see that the measured velocity closely matches with the provided reference velocity.

Below is a set of graphs showing the change in End-effector position for the given command. X-axis is the time in seconds and Y-axis is the position in m.





We can see from the graph that the X-Position remains constant and Y-Position changes at a constant rate of 0.2 m/s for a duration of 2 sec. This reaffirms that the velocity controller is working as required. This graph was plotted using the Forward Kinematics server which was built in the first part.

Attachments : final_project.zip

1) Scripts - C1_PC_Server.py, C2_Set_Reference.py, C3_VK_Server.py, C4_Velocity_Controller
2) Robots - C_SCARA.urdf.xacro
3) Config - C_SCARA_config.yaml
4) Launch - C_Load_SCARA.launch
5) Srv - PC_srv.srv, VK_srv.srv, FK_srv.srv