AA274A Section 6: Rosbag

Brian Dobkowski bdobkows@stanford.edu
AA274A, Stanford University

1. Problem 1

For the α, δ, and ρ messages, we used the Float64 message type from the ROS std_msg library.

```
import numpy as np
import rospy
from utils import wrapToPi
from std_msgs.msg import Float64

# command zero velocities once we are this close to the goal
RHO_THRES = 0.05
ALPHA_THRES = 0.1
DELTA_THRES = 0.1

alpha_publisher = rospy.Publisher('/controller/alpha', Float64, queue_size
    =10)
delta_publisher = rospy.Publisher('/controller/delta', Float64, queue_size
    =10)
rho_publisher   = rospy.Publisher('/controller/rho', Float64, queue_size=10)

class PoseController:
    """ Pose stabilization controller """
    def __init__(self, k1, k2, k3, V_max=0.5, om_max=1):
        self.k1 = k1
        self.k2 = k2
        self.k3 = k3

        self.V_max = V_max
        self.om_max = om_max

    def load_goal(self, x_g, y_g, th_g):
        """ Loads in a new goal position """
        self.x_g = x_g
        self.y_g = y_g
        self.th_g = th_g

    def compute_control(self, x, y, th, t):
        """
        Inputs:
            x,y,th: Current state
            t: Current time (you shouldn't need to use this)
        Outputs:
```

```
            V, om: Control actions

        Hints: You'll need to use the wrapToPi function. The np.sinc function
        may also be useful, look up its documentation
        """

        rho   = np.sqrt(np.power(self.x_g - x,2) + np.power(self.y_g - y,2))
        alpha = wrapToPi(np.arctan2(self.y_g - y, self.x_g - x) - th)
        delta = wrapToPi(np.arctan2(self.y_g - y, self.x_g - x) - self.th_g)

        alpha_publisher.publish(alpha)
        delta_publisher.publish(delta)
        rho_publisher.publish(rho)

        ########## Code starts here ##########
        V = self.k1 * rho * np.cos(alpha)
        om = self.k2 * alpha + self.k1 * np.cos(alpha) * np.sinc(alpha/np.pi)
    * (alpha + self.k3 * delta)

        ########## Code ends here ##########

        # apply control limits
        V = np.clip(V, -self.V_max, self.V_max)
        om = np.clip(om, -self.om_max, self.om_max)

        return V, om
```

2. Problem 2

Command used to record requested topics to a file:
rosbag record -O our_bag /controller/alpha /controller/delta /controller/rho

3. Problem 3

Rosbag info:

```
path:        our_bag.bag
version:     2.0
duration:    27.0s
start:       Dec 31 1969 16:19:07.80 (1147.80)
end:         Dec 31 1969 16:19:34.80 (1174.80)
size:        112.2 KB
messages:    1626
compression: none [1/1 chunks]
types:       std_msgs/Float64 [fdb28210bfa9d7c91146260178d9a584]
topics:      controller/alpha   542 msgs    : std_msgs/Float64 (2 connections
    )
             controller/delta   542 msgs    : std_msgs/Float64 (2 connections
    )
```

```
                controller/rho      542 msgs    : std_msgs/Float64 (2 connections
    )
```

4. Problem 4

When we run rosbag play, the topics that were recorded start being published on the ROS server. Roscore needs to be started before using rosbag play because rosbag is actually publishing the topics to ROS, not just printing them out.

5. Problem 5

Rqt plot for α, δ, ρ:

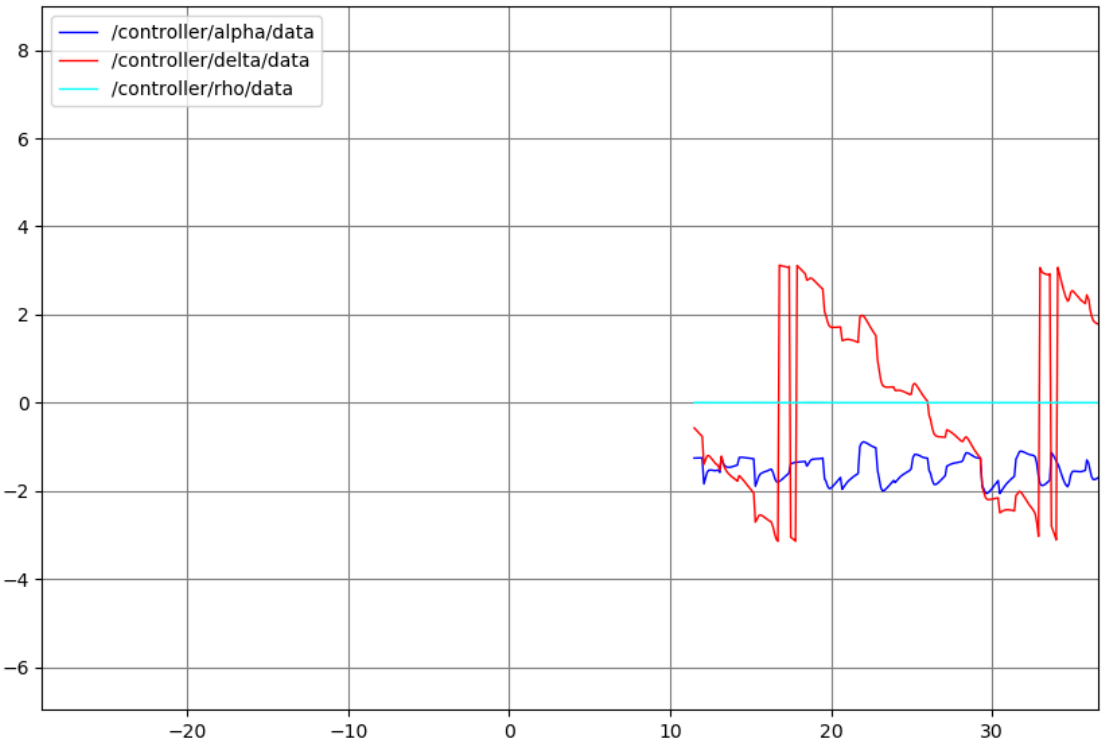

Figure 1: RQT Plot Output

6. Problem 6

We didn't get to this problem in Section.