

Lab 4 Report - Group 18

Qinghua He - qh2297

Jaisel Singh - js6897

Huan Gu - hg2721

Assignment 1

This node extracts the pose and velocity of the base_footprint frame and publishes them.

Code:

https://github.com/jaiselsingh1/prob_rob_labs_ros_2_group18/commit/494708fc23f9ccc8bf78f01d316787f2130aea32

Command to launch the ground-truth publisher:

```
ros2 launch prob_rob_labs ground_truth_from_link_states_launch.py ref_frame:=odom
```

Assignment 2

Command to make the robot move:

```
ros2 topic pub /cmd_vel geometry_msgs/Twist '{linear: {x: 1.0}, angular: {z: 0.0}}'
```

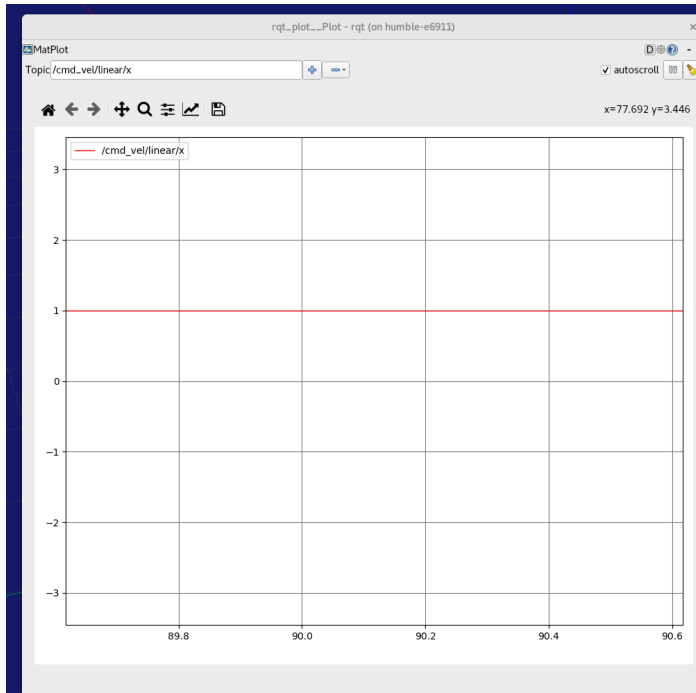
Commands to visualize the velocities:

```
ros2 run rqt_plot rqt_plot /cmd_vel/linear/x
```

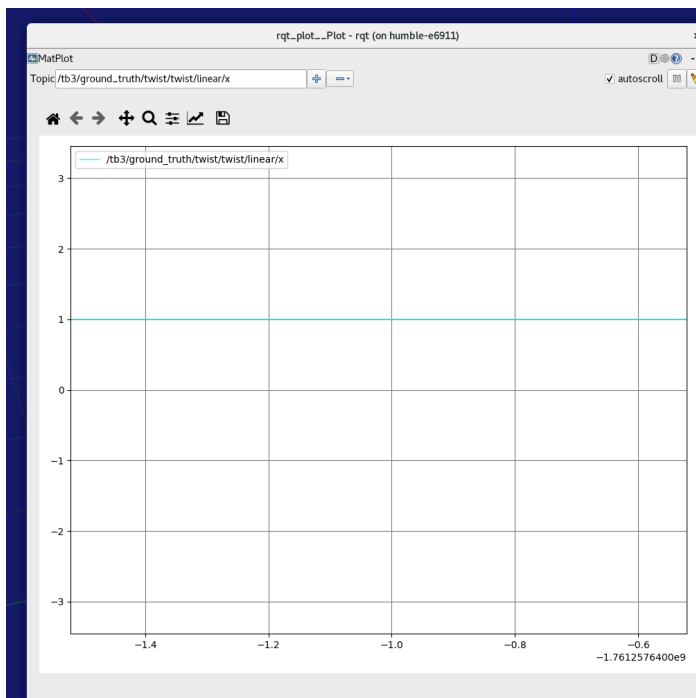
```
ros2 run rqt_plot rqt_plot /tb3/ground_truth/twist/twist/linear/x
```

The following screenshots show that it is achieving the requested velocity.

The video in the following link proves that our publisher is working.



```
local_js6897@humble-e6911:/run/host/workdir/local_js6897/ros2_ws
File Edit View Search Terminal Tabs Help
local_js6897_... local_js6897_... local_js6897_... local_js6897_...
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #59: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #60: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #61: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #62: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #63: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #64: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #65: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[]
```



```
local_js6897@humble-e6911:/run/host/workdir/local_js6897/ros2_ws
File Edit View Search Terminal Tabs Help
local_js6897_... local_js6897_... local_js6897_... local_js6897_...
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #103: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #104: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #105: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #106: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #107: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #108: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #109: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[]
```

Video:

https://github.com/jaiselsingh1/prob_rob_labs_ros_2_group18/blob/master/pictures/l4a2.webm

Assignment 3

Commands to publish to cmd_vel topic:

```
ros2 topic pub /cmd_vel geometry_msgs/Twist '{linear: {x: 2.0}, angular: {z: 0.0}}'
```

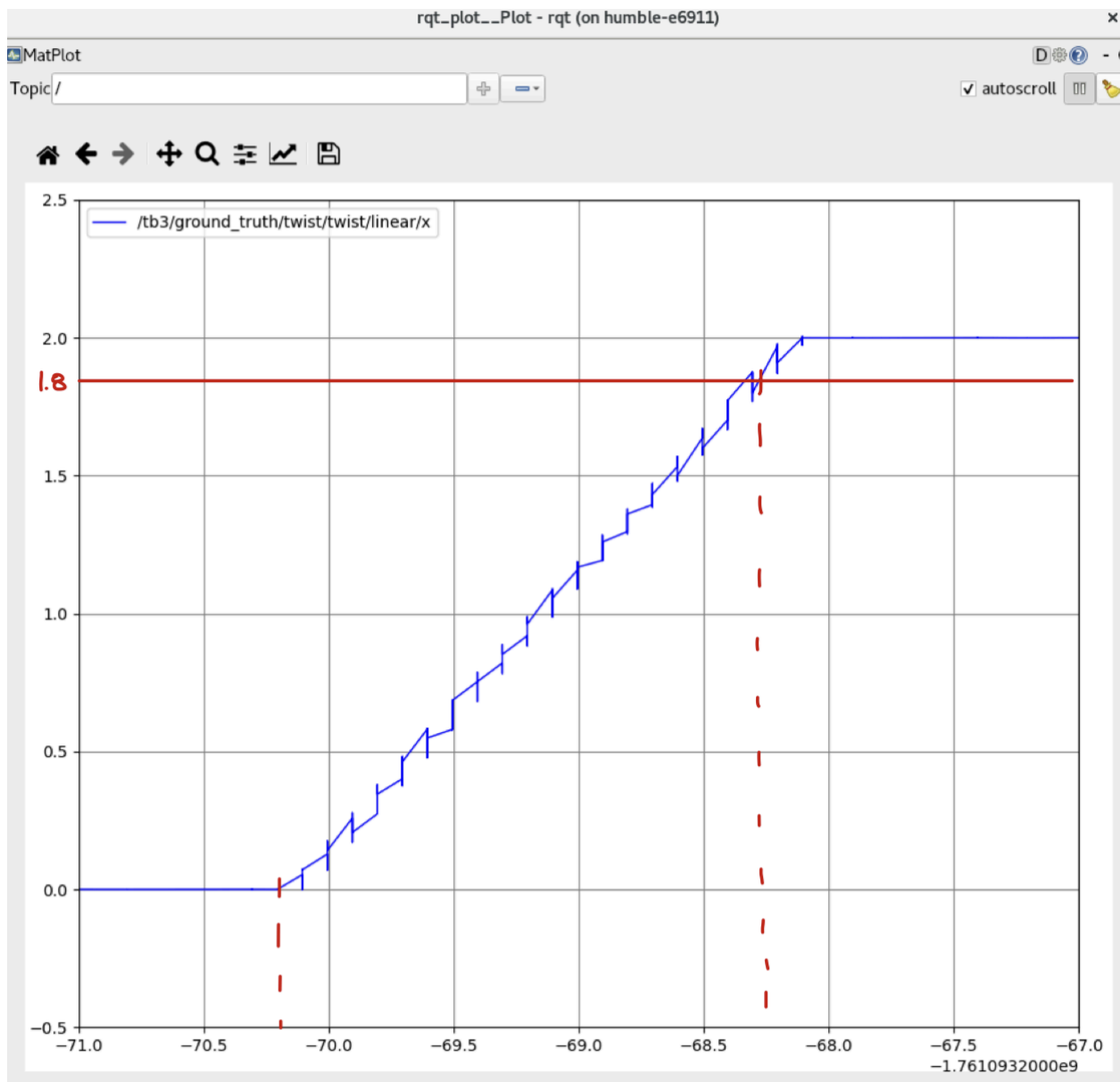
```
ros2 topic pub /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0}, angular: {z: 0.3}}'
```

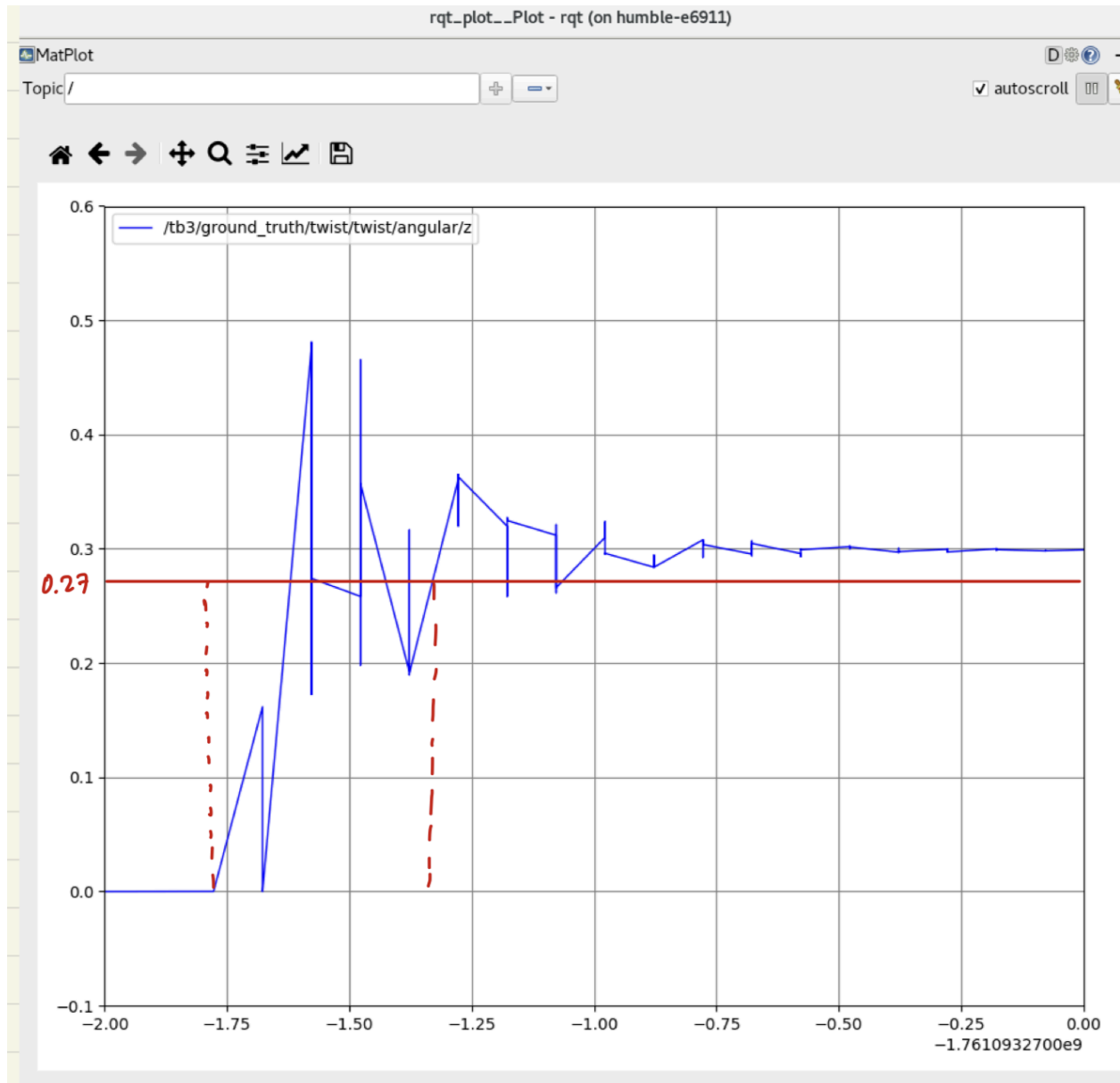
Commands to plot the /tb3/ground_truth/twist topic using rqt_plot:

```
ros2 run rqt_plot rqt_plot /tb3/ground_truth/twist/twist/linear/x
```

```
ros2 run rqt_plot rqt_plot /tb3/ground_truth/twist/twist/angular/z
```

The annotated screenshots are as follows.





For the plot for v_x , the 90% ramp-up time (from 0.0 to 1.8), $\tau_v \approx (-68.35) - (-70.15) = 1.80$ s.
 For the plot for w_z , the 90% ramp-up time (from 0.0 to 0.27), $\tau_w \approx (-1.35) - (-1.80) = 0.45$ s.
 So, we can determine the parameters a_v and a_w as a function of dt , and $G_v = G_w = 1$.

$$a_v = 0.1 \frac{\Delta t}{1.80}$$

$$a_w = 0.1 \frac{\Delta t}{0.45}$$

Assignment 4

Here are the calculations to formulate the state-update law for the Extended Kalman filter, and derive the Jacobian and B-matrix for state transformation.

Assignment 4

$$\theta[n+1] = \theta[n] + \omega \Delta t$$

$$x[n+1] = x[n] + v \Delta t \cos \theta[n]$$

$$y[n+1] = y[n] + v \Delta t \sin \theta[n]$$

$$v[n+1] = a_v \cdot v[n] + G_v \cdot (1 - a_v) u_v[n]$$

$$\omega[n+1] = a_\omega \cdot \omega[n] + G_\omega \cdot (1 - a_\omega) u_\omega[n]$$

$$\mathbf{x} = \begin{bmatrix} \theta \\ x \\ y \\ v \\ \omega \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_v \\ u_\omega \end{bmatrix}$$

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n, \mathbf{u}_n)$$

$$f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \theta + \omega \Delta t \\ x + v \Delta t \cos \theta \\ y + v \Delta t \sin \theta \\ a_v v + G_v (1 - a_v) u_v \\ a_\omega \omega + G_\omega (1 - a_\omega) u_\omega \end{bmatrix}$$

$$J = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t \\ -v \Delta t \sin \theta & 1 & 0 & \Delta t \cos \theta & 0 \\ v \Delta t \cos \theta & 0 & 1 & \Delta t \sin \theta & 0 \\ 0 & 0 & 0 & a_v & 0 \\ 0 & 0 & 0 & 0 & a_\omega \end{bmatrix}$$

$$B = \frac{\partial f}{\partial \mathbf{u}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ G_v (1 - a_v) & 0 \\ 0 & G_\omega (1 - a_\omega) \end{bmatrix}$$

Assignment 5

Here are the calculations to formulate the measurement model and derive the C-matrix.

Assignment 5

$$v = \frac{r_\omega}{2}(\omega_r + \omega_l),$$
$$\omega = \frac{r_\omega}{2R}(\omega_r - \omega_l),$$

$$\omega_r = \frac{1}{r_\omega}v + \frac{R}{r_\omega}\omega,$$
$$\omega_l = \frac{1}{r_\omega}v - \frac{R}{r_\omega}\omega.$$

The measurement vector:

$$\mathbf{z} = \begin{bmatrix} \omega_r \\ \omega_l \\ \omega_g \end{bmatrix},$$

The measurement model:

$$\mathbf{z} = \mathbf{C} \mathbf{x} + \mathbf{w},$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{r_\omega} & \frac{R}{r_\omega} \\ 0 & 0 & 0 & \frac{1}{r_\omega} & -\frac{R}{r_\omega} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Given:

$$r_w = 33 \text{ mm} = 0.033 \text{ m}, \quad R = \frac{143.5}{2} \text{ mm} = 0.07175 \text{ m}.$$

$$\frac{1}{r_w} = 30.303, \quad \frac{R}{r_w} = 2.174.$$

$$\mathbf{C} \approx \begin{bmatrix} 0 & 0 & 0 & 30.303 & 2.174 \\ 0 & 0 & 0 & 30.303 & -2.174 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

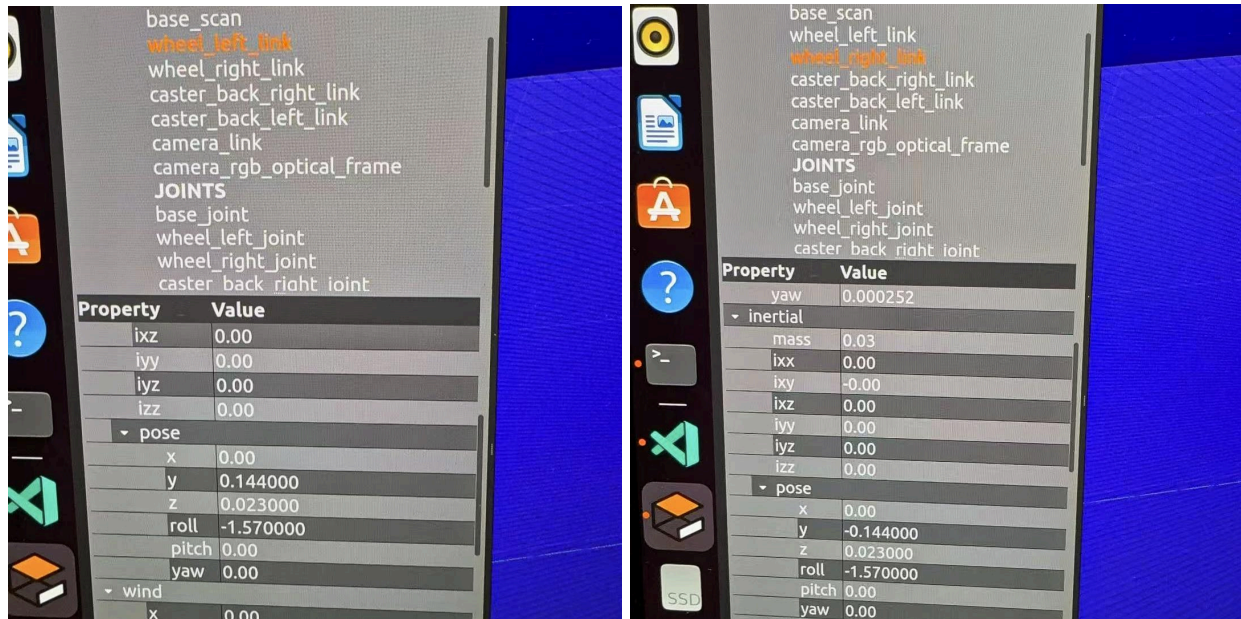
Assignment 6

Code to implement the odometry-tracking node using the model we derived:

https://github.com/jaiselsingh1/prob_robot_labs_ros_2_group18/commit/585a48ed8e6ab7f6b8481df30a031581dd83de9b

Command to publish the odometry messages to /ekf_odom topic:

```
ros2 launch prob_robot_labs odometry_tracking_launch.py
```



However, as we were tuning the parameters, we found from the pose-y of wheel_left_link and wheel_right_link that the provided 143.5mm is exactly the robot radius R we want and not the wheel-separation as described in Assignment 5, so we don't have to make it half again.

We prefer not to modify the calculations above in Assignment 5 and just illustrate here to show our thinking process for this problem.

Assignment 7

Our odometry pose and ground-truth pose are moving together with reasonably small error.

The video showing that our tracker is working:

https://github.com/jaiselsingh1/prob_robot_labs_ros_2_group18/blob/master/pictures/r4a7.webm

The blue arrow is the ground-truth pose, while the red arrow is the pose from our EKF.

Assignment 8

Code to subscribe to the ground truth and our odometry topic, and calculate the euclidean distance between estimated robot positions & angular difference in robot orientation:

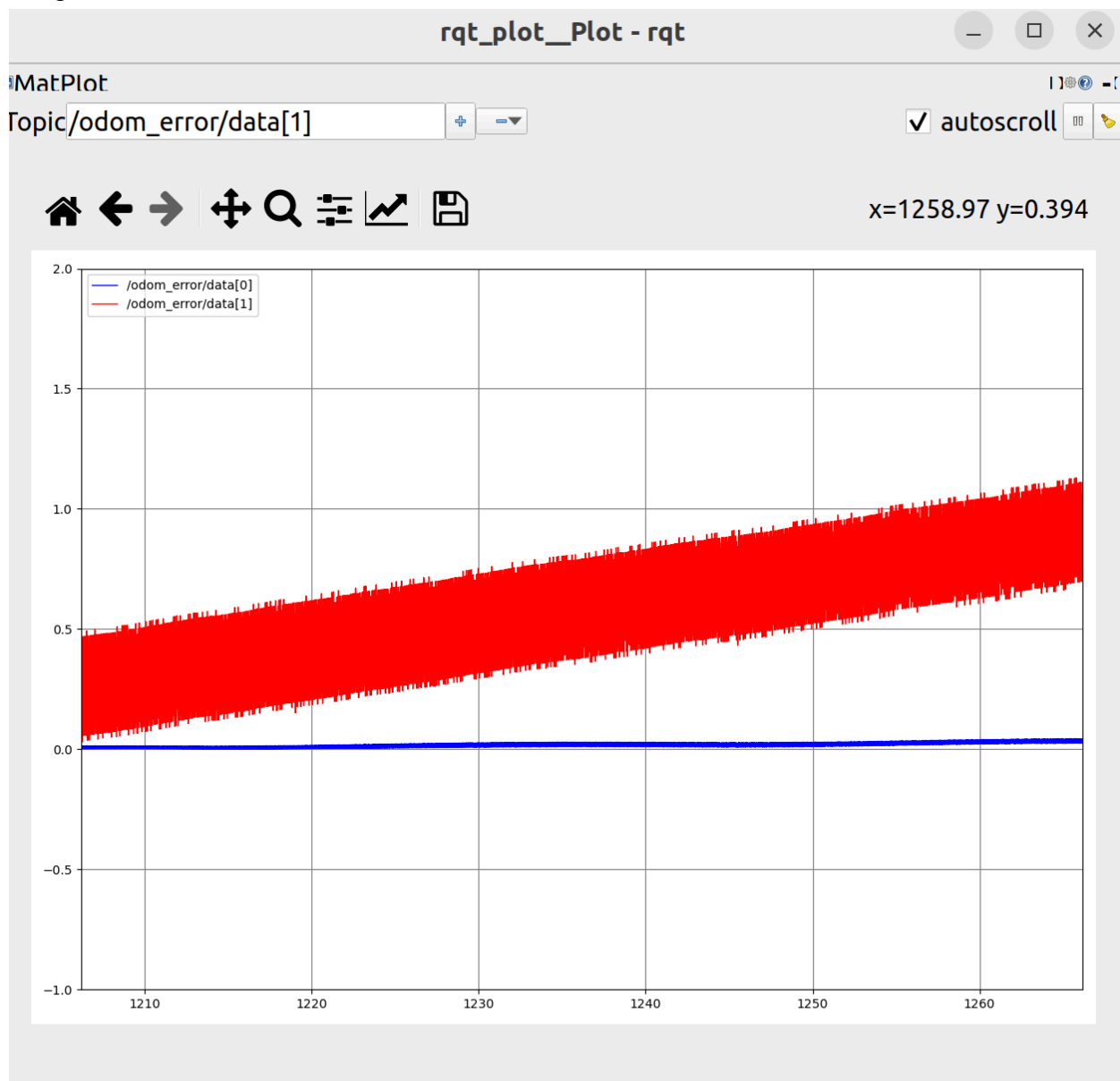
https://github.com/jaiselsingh1/prob_robot_labs_ros_2_group18/commit/98b0dfa738bd2710997e9222e419f81036b93f51

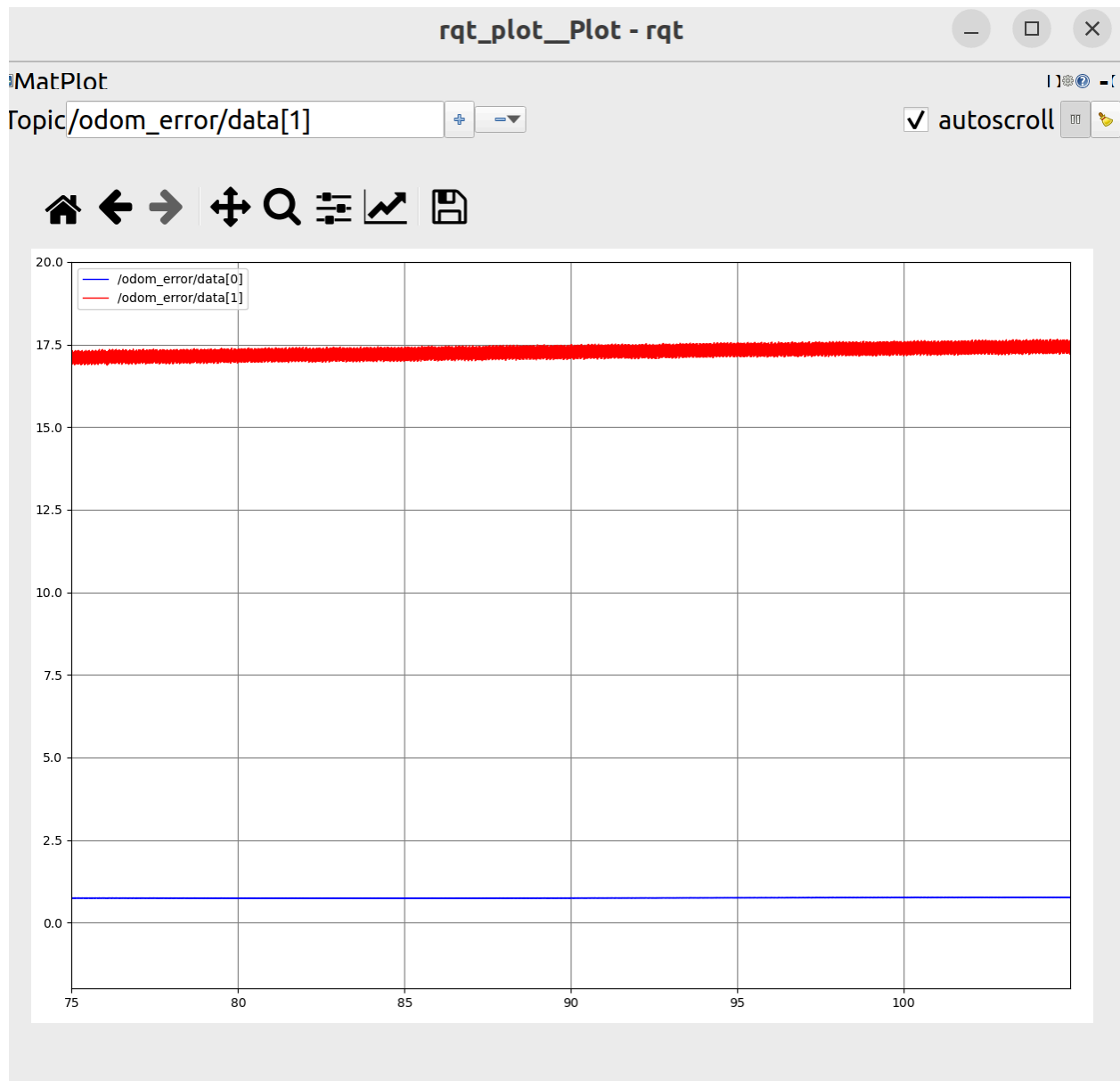
Command to visualize the error and move the robot around for about a minute:

```
ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}"
```

```
ros2 launch prob_robot_labs odom_error_evaluator_launch.py
```

The plots are as follows.





Data[0] is the position error.

Data[1] is the yaw angular error.

The first plot shows the errors at the beginning.

The second plot shows the errors after a really long time, so the errors keep accumulating.

Assignment 9

Our covariance matrix:

▼ ✓ /ekf_odom	nav_msgs/msg/Odometry	unknown	29.40	
▶ header	std_msgs/Header			
child_frame_id	string			'base_footprint'
▼ pose	geometry_msgs/PoseWithCovariance			
▶ pose	geometry_msgs/Pose			
				array([2398.25876673, 1000. , 1000. , 1000. , 1000. , 1000. , 1000. , 5246.62075797, 1000. , 526.36470196])
covariance	double[36]			
▼ twist	geometry_msgs/TwistWithCovariance			
▶ twist	geometry_msgs/Twist			
				array([5.42862343e-05, 1.00000000e+03, 6.98177629e-04])
covariance	double[36]			

The covariance matrices of pose and twist both contain 36 elements and are 6x6 in size.

From the covariance matrix of pose, theta is stable and x and y are growing out of bound.

From the covariance matrix of twist, both v and w are stable.

That's because ekf is less certain about the robot's absolute pose from odometry, but more confident for the rotation and velocities.