

**4.a)**Question:

**Unicycle Model:** A unicycle has forward velocity  $v$  and yaw rate  $\omega$ . In the body frame, there is no lateral motion ( $v_y = 0$ ) so  $v_x = v$ .

- Write the body-frame twist for the unicycle
- Expand the KE to derive the ODEs
- For constant  $v, \omega > 0$ : What shape is the trajectory? What is the radius?

Solution:

The body frame twist is

$$\xi = \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} = \begin{pmatrix} v \\ 0 \\ \omega \end{pmatrix}$$

Deriving the equations of motion

$$\begin{aligned} \dot{X} &= X\xi^\wedge = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -\omega & v \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} -\omega \sin(\theta) & -\omega \cos(\theta) & v \cos(\theta) \\ \omega \cos(\theta) & -\omega \sin(\theta) & v \sin(\theta) \\ 0 & 0 & 0 \end{pmatrix} \\ \implies \dot{x} &= v \cos(\theta), \dot{y} = v \sin(\theta), \dot{\cos}(\theta) = -\omega \sin(\theta), \dot{\sin}(\theta) = \omega \cos(\theta) \end{aligned}$$

$$\dot{\cos}(\theta) = -\sin(\theta)\dot{\theta} = -\omega \sin(\theta) \implies \dot{\theta} = \omega$$

$$\dot{\sin}(\theta) = \cos(\theta)\dot{\theta} = \omega \cos(\theta) \implies \dot{\theta} = \omega$$

$$\boxed{\dot{x} = v \cos(\theta), \dot{y} = v \sin(\theta), \dot{\theta} = \omega}$$

For constant  $v, \omega > 0$  the trajectory follows a circle with a radius found by computing the following integral

$$r = \int_0^{\frac{\pi}{2\omega}} v \cos(\omega t) dt = \frac{v}{\omega} \int_0^{\frac{\pi}{2}} \cos(\theta) d\theta = \frac{v}{\omega}$$

The trajectory is a circle with radius  $\frac{v}{\omega}$

**4.b)**Question:Solution:

## Code Snippet 1: SE(2) Integrator Implementation

```

1 import numpy as np
2 from Code.SE2.SE2_maps import se2_compose, se2_exp, se2_wedge
3
4 def se2_euler_integrator(X_k, xi_k, dt):
5     """
6     Integrate using first order Euler
7
8     :param X_k: group element at t_k
9     :param xi_k: twist at t_k
10    :param dt: timestep
11    """

```

```

12     X_kp1 = se2_compose(X_k, np.eye(3) + dt * se2_wedge(xi_k))
13
14     return X_kp1
15
16 def se2_Lie_group_integrator(X_k, xi_k, dt):
17     """
18     Integrate using exponential map (assuming twist is constant across timestep)
19
20     :param X_k: group element at t_k
21     :param xi_k: twist at t_k
22     :param dt: timestep
23     """
24     X_kp1 = se2_compose(X_k, se2_exp(dt * xi_k))
25
26     return X_kp1

```

## Code Snippet 2: SE(2) Integrator Test

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from Code.SE2.SE2_maps import se2_compose, se2_exp, se2_log, se2_wedge, se2_ee, se2_inverse
4 from Code.SE2.SE2_integrators import se2_euler_integrator, se2_Lie_group_integrator
5
6 ## Unicycle sim
7
8 # Initialize
9 X_0 = np.eye(3)
10 v = 1 # [m / s]
11 omega = 0.5 # [rad / s]
12 xi_k = np.array([[v], [0], [omega]])
13 dt = 0.1 # [s]
14 tf = 20
15 time = np.arange(0, tf + dt, dt)
16
17 # Simulation loop
18 X_k_euler = X_0
19 X_k_Lie = X_0
20
21 positions_euler = np.zeros([2, time.size])
22 positions_Lie = np.zeros([2, time.size])
23
24 error_euler = np.zeros_like(time)
25 error_Lie = np.zeros_like(time)
26
27 for k in range(time.size - 1):
28     X_k_euler = se2_euler_integrator(X_k_euler, xi_k, dt)
29     X_k_Lie = se2_Lie_group_integrator(X_k_Lie, xi_k, dt)
30
31     positions_euler[:, k + 1] = X_k_euler[0:2, 2]
32     positions_Lie[:, k + 1] = X_k_Lie[0:2, 2]
33
34     R_euler = X_k_euler[0:2, 0:2]
35     R_Lie = X_k_Lie[0:2, 0:2]
36
37     error_euler[k + 1] = np.linalg.norm(R_euler.T @ R_euler - np.eye(2), "fro")
38     error_Lie[k + 1] = np.linalg.norm(R_Lie.T @ R_Lie - np.eye(2), "fro")
39
40 # Plot
41 # Trajectories
42 plt.plot(positions_euler[0, :], positions_euler[1, :], label = "Euler")
43 plt.plot(positions_Lie[0, :], positions_Lie[1, :], label = "Lie Group")
44 plt.xlabel("X Position [m]")
45 plt.ylabel("Y Position [m]")
46 plt.title("Unicycle Trajectories vs Integrator")
47 plt.legend()
48 plt.grid(True)
49 plt.gca().set_aspect('equal', adjustable='box')
50 plt.savefig("./HW2/Parts/Q4/AAE590LGM_HW2_Q4_traj.png")

```

```

51 plt.show()
52
53 # Error
54 plt.semilogy(time, error_euler, "o", label = "Euler")
55 plt.semilogy(time, error_Lie, "o", label = "Lie Group")
56 plt.xlabel("Time [s]")
57 plt.ylabel("Rotation Error")
58 plt.title("Rotation Error vs Time for Both Integrators")
59 plt.legend()
60 plt.grid(True)
61 plt.savefig("./HW2/Parts/Q4/AAE590LGM_HW2_Q4_error.png")
62 plt.show()
63
64 # Compare with Exp map
65 X_f = se2_compose(X_0, se2_exp(tf * xi_k))
66 print(np.linalg.norm(X_f - X_k_Lie, "fro"))

```

Figure 1: Trajectories of both integrators for unicycle

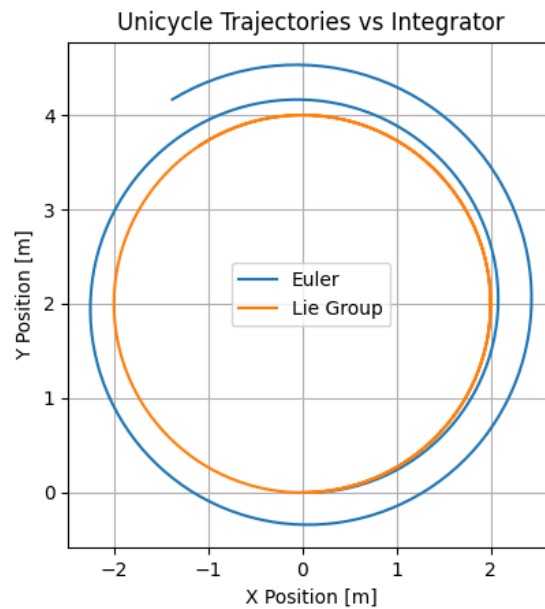
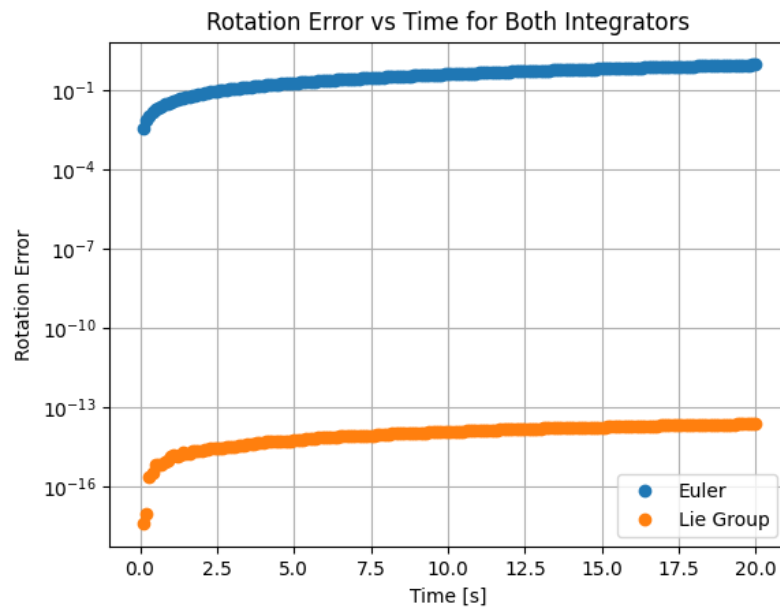


Figure 2: Error over time for both integrators



The error between the exact propagation using the exponential map and the Lie group integrator at the final time is  $1.8289049996250573e-14$ . This means that within numerical precision they are the same which makes sense given that the Lie group integrator uses the exponential map.