Lab1: Building a Simulator

Yiwen(Robert) Wu

https://github.com/Robert1124/cse460

I. Exercise 1

In the pursuit of commanding a point robot to trace an elliptical path in a two-dimensional plane, a control policy $u(t)$ was formulated. This policy is rooted in the parametric representation of an ellipse, traditionally given by:

$$x(t) = a\cos(\omega t + \phi), \tag{1}$$
$$y(t) = b\sin(\omega t + \phi), \tag{2}$$

where a and b denote the semi-major and semi-minor axes of the ellipse, ω is the angular velocity, and ϕ is the phase offset. For an ellipse centered at the origin and aligned with the axes, ϕ is taken to be zero.

To derive the control inputs, we consider the time derivative of the parametric equations, yielding the velocity components of the robot at any instant:

$$\frac{dx}{dt} = -a\omega\sin(\omega t), \tag{3}$$
$$\frac{dy}{dt} = b\omega\cos(\omega t). \tag{4}$$

These derivatives reflect the instantaneous velocity vector $u(t)$ required to maintain the elliptical trajectory. For the specific case of uniform motion along the ellipse, ω is set to 1, simplifying the control policy to:

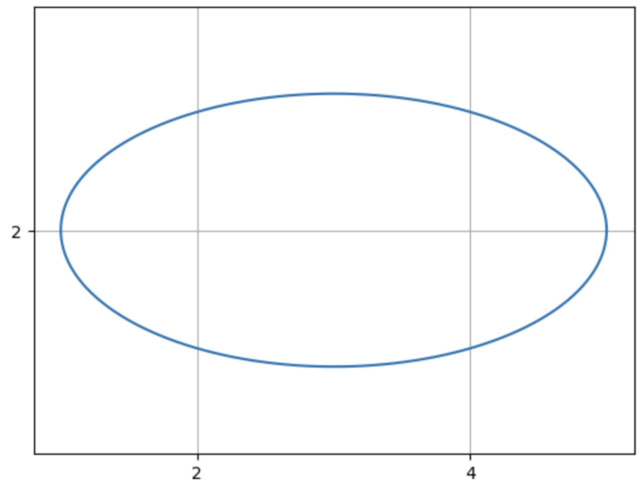
$$u_x(t) = -a\sin(t), \tag{5}$$
$$u_y(t) = b\cos(t). \tag{6}$$

The negative sign in u_x ensures that the motion proceeds in the counterclockwise direction when viewed from the positive z-axis, aligning with the conventional orientation for positive rotation.

The simulation framework utilizes a discrete-time model with a time step Δt, iterating over a complete period $T_f = 2\pi$, corresponding to the fundamental period of the sine and cosine functions. The initial conditions place the robot at the rightmost extremity of the ellipse by offsetting the center's x-coordinate by the semi-major axis length, while the y-coordinate remains unchanged. Specifically, with the ellipse's center at $(3, 2)$ and a semi-major axis of 2, the initial position is computed as:

$$\mathbf{x}_0 = \begin{bmatrix} 3+a \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}. \tag{7}$$

Through an iterative simulation, the robot's state is updated according to the control policy, accurately reflecting the velocity vectors that are tangential to the desired elliptical path at every time step. The resulting data points can be visualized to verify adherence to the elliptical trajectory.



II. Exercise 2

To achieve the elliptical trajectory with a thirty-degree rotation, the control policy from the first exercise is modified. The modification involves rotating the control input velocity vector to align with the rotated ellipse.

A rotation matrix R is employed to transform the velocity vector. The matrix for a counterclockwise rotation by an angle θ in 2D is given by:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{8}$$

This matrix is applied to the velocity vector $\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ to obtain the rotated velocity vector.

The control policy function is defined as:

$$\text{control\_2}(t, y) = \text{rotate}\left(\begin{bmatrix} -a\sin(t) \\ b\cos(t) \end{bmatrix}, \theta\right) \tag{9}$$

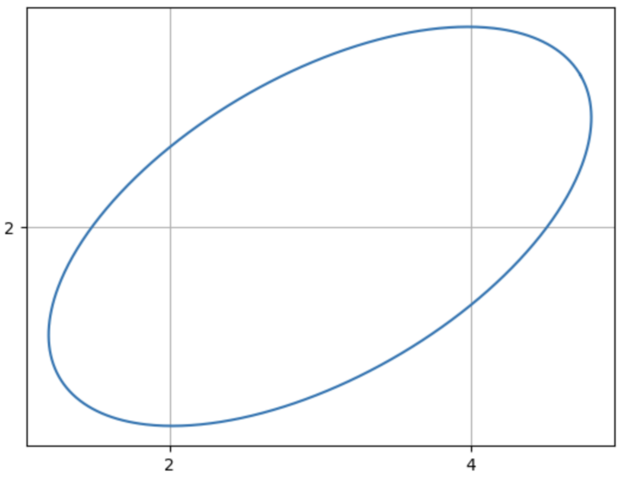
where a and b are the semi-major and semi-minor axes respectively, and $\theta = \frac{\pi}{6}$ radians (30 degrees). The negative sign in the x-component ensures counterclockwise rotation.

For the initial conditions, the starting point on the ellipse must also be rotated. If the unrotated starting point is $\mathbf{x}_0$, the rotated initial position $\mathbf{x}$ is obtained by:

$$\mathbf{x} = \text{rotate}(\mathbf{x}_0 - \text{center}, \theta) + \text{center} \tag{10}$$

where *center* is the center of the ellipse.

The simulation proceeds by updating the robot's position using the rotated control inputs over the time interval $[0, 2\pi]$ with a time step $\Delta t = 0.01$. This interval corresponds to one complete cycle of the elliptical path. The trajectory logged in **x_log** is expected to outline the rotated ellipse when plotted.



III. Exercise 3

The third exercise involves developing a control policy for a robot to follow the Lemniscate of Gerono, which is characterized by its figure-eight shape. The Lemniscate of Gerono can be described by parametric equations as follows:

$$x(t) = a\cos(t), \tag{11}$$
$$y(t) = b\sin(2t), \tag{12}$$

where a and b are constants that scale the figure along the x and y axes respectively. To formulate the control inputs, the derivatives of these parametric equations are computed:

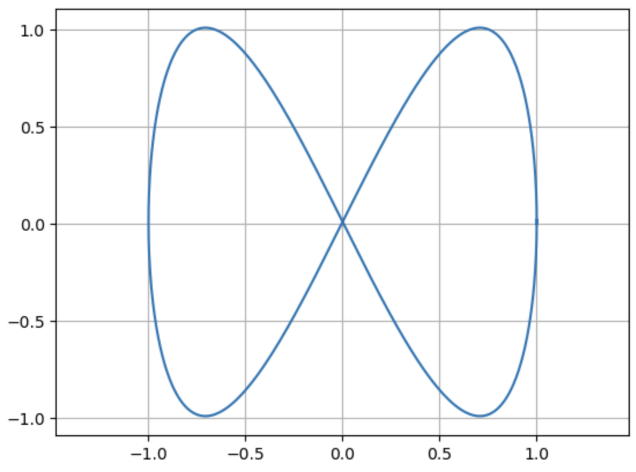
$$\frac{dx}{dt} = -a\sin(t), \tag{13}$$
$$\frac{dy}{dt} = 2b\cos(2t). \tag{14}$$

The control inputs for the robot are the velocity components derived from these equations. The control policy function *control_3* is thus defined to return the velocity vector $\mathbf{u}$:

$$\mathbf{u}(t) = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} -a\sin(t) \\ 2b\cos(2t) \end{bmatrix}. \tag{15}$$

A simulation loop is implemented to update the robot's position at discrete time intervals. The initial condition is set by evaluating the parametric equations at $t = 0$, thus starting the robot at one end of the figure-eight path. As time progresses, the control inputs guide the robot along the Lemniscate, and the resulting trajectory is logged in **x_log**.

The complete trajectory is simulated over a time period $T_f = 2\pi$ with a time step $\Delta t = 0.01$, ensuring that the robot completes the figure-eight pattern within the duration. The trajectory can be visualized by plotting **x_log**, confirming the efficacy of the control policy in achieving the desired path.



IV. Exercise 4

The fourth exercise extends the simulation environment to include external disturbances in the form of wind while the robot is executing a circular trajectory. To accurately reflect these conditions, the `simulate` function is enhanced to integrate the effects of wind into the robot's dynamics.

Control Policy Function

A control policy for circular motion, denoted as `control_4`, is defined to produce control inputs u_x and u_y that adhere to the following equations:

$$u_x(t) = -a\sin(t), \tag{16}$$
$$u_y(t) = a\cos(t), \tag{17}$$

where a represents the radius of the circular path.

Simulation Function with Wind

The simulation function, named `simulate_4`, accepts additional parameters for wind influence:

- `wind_constant` is a vector representing a constant wind force applied to the robot.
- `wind_random_std` is the standard deviation of a zero-mean Gaussian distribution from which random wind perturbations are sampled.

The state update equation, incorporating the wind components, is expressed as:

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta t(\mathbf{u} + \mathbf{w}_{\text{constant}} + \mathbf{w}_{\text{random}}), \tag{18}$$

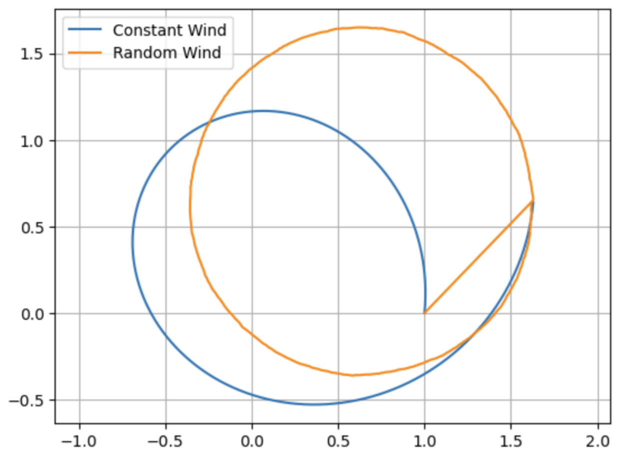
where $\mathbf{w}_{\text{random}}$ is the random wind vector generated at each time step.

Simulation Execution

Two simulations are conducted:

1) The first simulation introduces a constant wind vector $\mathbf{w}_{\text{constant}} = [0.1, 0.1]^\top$ throughout the robot's motion.
2) The second simulation models wind as a random process with a standard deviation of $\sigma_{\text{wind}} = 0.1$, affecting the robot in both x and y directions independently.

The robot's trajectory is recorded in two separate logs, `x_log_constant_wind` and `x_log_random_wind`, representing the path under constant and random wind conditions, respectively.


Constant Wind
Random Wind