CMPE 8 Lab 4: Modeling the Heading of the Scribbler 2

Lab Objectives

By the end of this lab you should be able to:

1. **Identify** how the heading of your robot is defined in the form of a discrete-time dynamic equation.
2. **Identify** the turn rate of your robot through experimentation.
3. **Create** a function in MATLAB to simulate the turn rate of your robot.
4. **Verify** your simulations match with the actual performance of your robot.

Heading and Turn Rate

When we program our robot to move around, it is useful to be able to define its position or orientation for the purposes of controlling its motion. One of the most natural ways to define its position is using coordinates in the x-y plane. Although this a good way to define its position, creating a dynamic model for control purposes becomes tricky since we have to keep track of motion in two directions. Instead of defining the robot's position, we can define its orientation by modeling its heading. The heading of our robot can tell us how many degrees the robot has turned from its initial starting position. The symbol we will use for our heading is $\theta$ (theta). Below is a figure that illustrates the concept of the robot’s heading relative to its initial starting position.

Modeling the Robot’s Heading with a Dynamic Equation

If we want to control the motion of our robot we need to know what input is required to achieve a desired result (e.g., how long should we run the motors to turn the robot 90°?). We can learn this information by recording the response of our robot to specific commands and creating a mathematical model based on that data. In order to create a model for the robot's heading, we first need to calculate the robot’s turn rate, $r$ (i.e., how fast our robot turns). Then we can use the turn rate to write a discrete-time dynamic equation. You can read more about discrete-time dynamic equations in your class reader on pages 24-28. The discrete-time dynamic equation
for the robot's heading can be defined as:

$$\Theta_{n+1} = \Delta r + \Theta_n$$

where $\Theta$ is the heading in degrees, $r$ is the turn rate in degrees/sec and $\Delta$ (delta) is the sampling period in seconds. Remember that we are working in a discrete-time space, which means we take “snapshots” of the robot’s heading. The time between snapshots is called our sample period. For the Scribbler 2 robot, the sampling period is 1 ms.

To calculate the robot's turn rate, we need to conduct several experiments where we turn the robot for a specific amount of time and record its heading. The command that we will use to make the robot turn is:

```plaintext
s2.wheels_now(leftMotor,rightMotor,time)
```

The variables `leftMotor` and `rightMotor` set the velocity of the robot's wheels and can take on any integer values between -255 and +255. For this lab, we will set the wheels to full speed. The time variable sets the length of the duration of the command in milliseconds. To make the robot spin, we need to have one wheel drive forward and one wheel drive in reverse. We can use positive velocities to drive the wheels forward and negative velocities to drive the wheels in reverse.

**Making Your Robot Move When You Want it To**

When we program the Scribbler 2 to perform a motion, it will begin to move as soon as the program has been uploaded to its memory. In most cases, we would rather have the robot wait until we are ready before it starts moving. To keep the our robot from moving immediately we can use the command:

```plaintext
s2.reset_button_count
```

This command will return the number of times the blue reset button on the front of the robot has been pressed since the robot executed its previous program. We can use this command as a condition inside an `if` statement to prevent the robot from executing its program until we push the reset button. For example, the code:

```plaintext
if(s2.reset_button_count == 1)
    some action
```

will make the robot wait until the reset button has been pressed once before performing the action inside the `if` statement.

We can use the `s2.reset_button_count` command with up to eight different `if` statements to have the robot perform eight different actions based on the number of reset button presses. This means that you can write a single program with six different `if` statements to perform the actions required in exercise 1.

**Exercises**

1. Program your robot to turn counterclockwise for each of the times in the time vector $t = 0.5:0.5:3$ where time is in seconds. Using the provided sheet and MATLAB,
calculate your robot's heading for each time step and plot its heading versus time starting from \( t = 0 \).

2. Calculate your robot's turning rate using the data collected in exercise 1. Are the turn rates the same for each time step?

3. Using MATLAB, create a function called `RobotTurn(r,d,tf)` that takes the turn rate \( r \), sample period \( d \) and finish time \( tf \) as arguments and returns a plot of your robot's predicted heading versus time.

4. Using the function from exercise 3, plot the predicted heading of your robot with \( r \) equal to the average of your turn rates from exercise 2, \( d \) equal to 1 ms and \( tf \) equal to 3 seconds. Compare this plot with the plot from exercise 1. Comment on why you think they either did or didn't match up.

**Note:** For this lab you should comment every line of your code. It shows that you know what you are doing and makes it easier for someone else (i.e., the grader) to understand your code.