Spencer Drach
Intro to Robotics
Dr. McGough

HW#3

Chapter 5 problem 4:

For this problem I created the code in appendix 1. Using that code I generated the path the robot would take with the given wheel velocities and plotted it with matplotlib, I added arrows on top to show the direction the robot was pointing at each time step. Below is the plot generated with the arrows on top of the path of the robot. And below that is a close-up of the arrows.

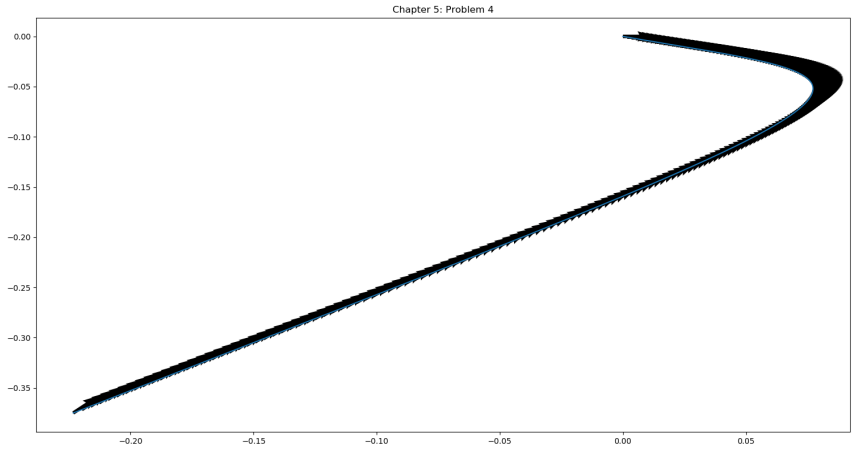


Figure 1: Path of Robot

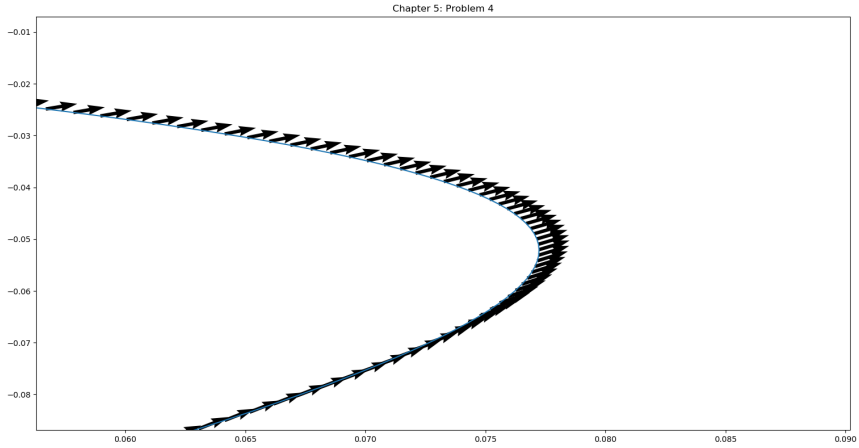


Figure 2: Close-up of directional arrows

Chapter 5 problem 5:

For this problem I was unable to get close to the [50,60] mark that the problem states that I should get close to and I have no idea why. But I'm going to press on as if I got the right numbers to form the correct path. I plotted the path of the robot

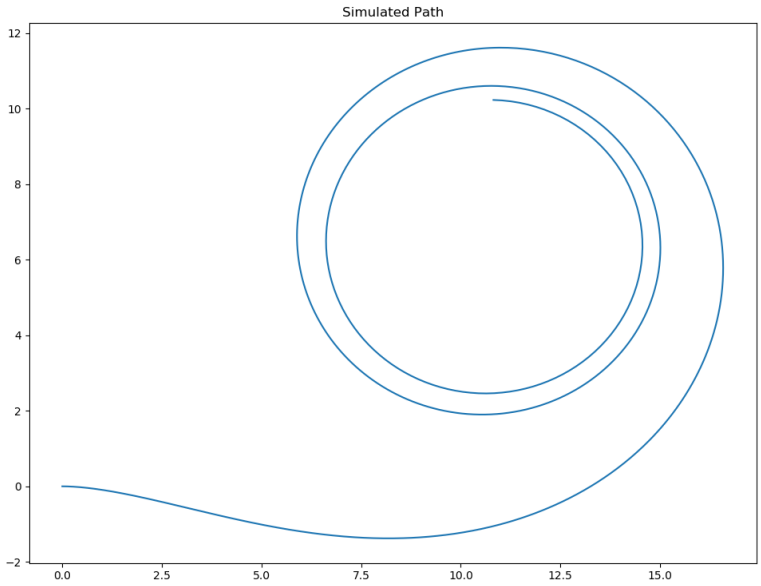


Figure 3: Path of the robot for 0<t<5

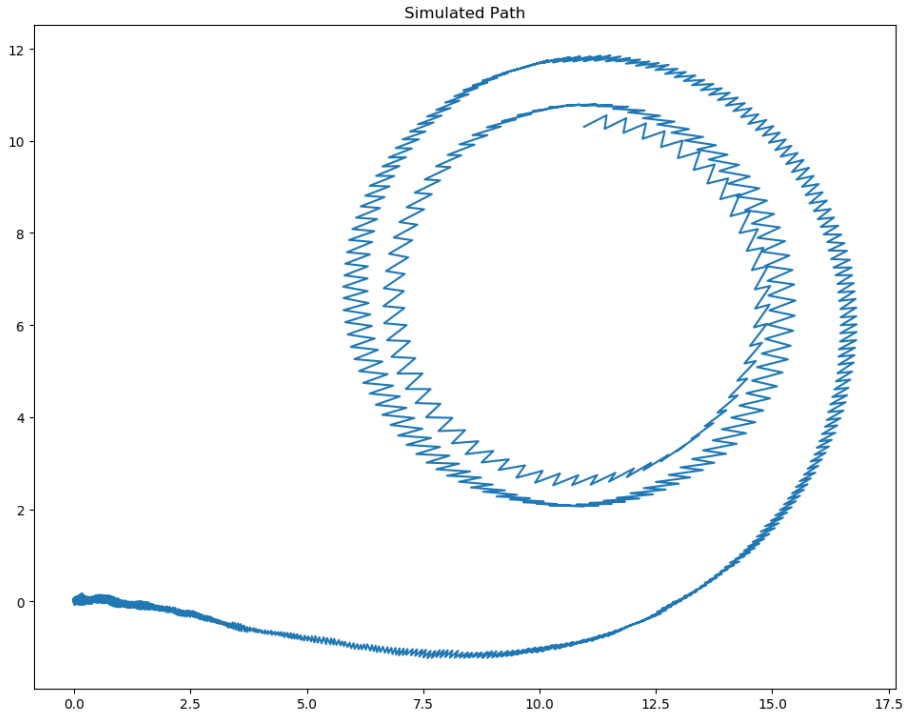


Figure 4: Path of the robot with noise

Chapter 7 problem 5:

PWM stands for pulse width modulated it is a way of supplying a variable amount of power with only turning on and off the current. You can vary the amount of power going to the device by changing how much of the time the switch is on. This is called the duty cycle and it is measured in percentages from 0% to 100%. Below is a picture illustrating the duty cycle in effect. It also shows the average voltage which can be used to calculate the average power going to the device. This is a really nice way of powering non-linear devices that can respond unpredictably when given input voltages that are not 0V or 5V.

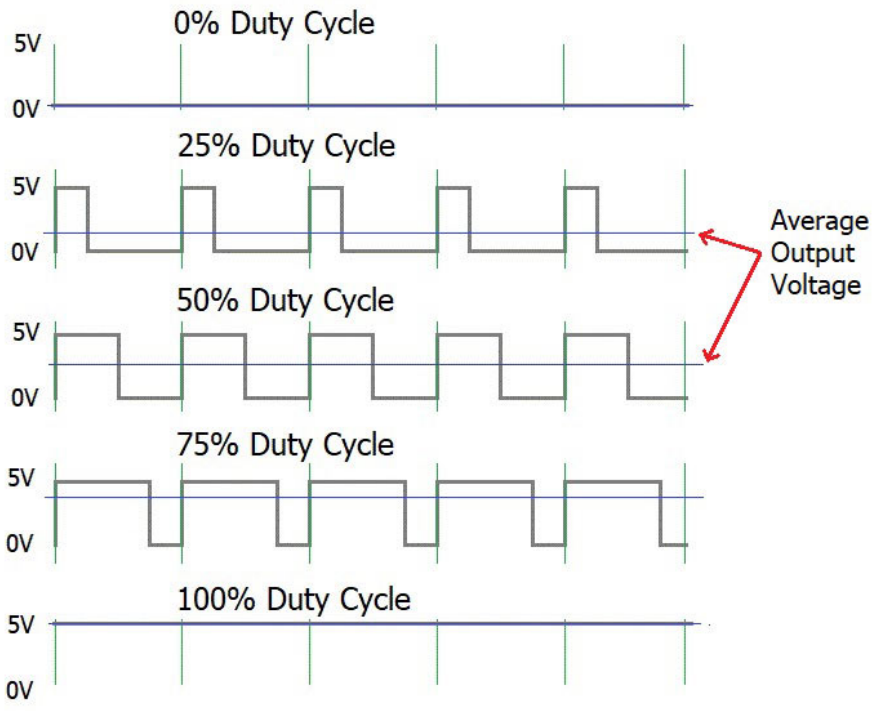


Figure 4: PWM Duty Cycle

Chapter 7 problem 6:

Designing a circuit to accept positive or negative DC voltages is actually pretty simple. Below is a picture of a circuit I designed to accomplish this. It uses three diodes to ensure that the voltage the dummy load(resistor) sees is always positive. The diodes will decrease the voltage a little bit (~0.7V per diode) but that's much better than any negative voltage in most cases

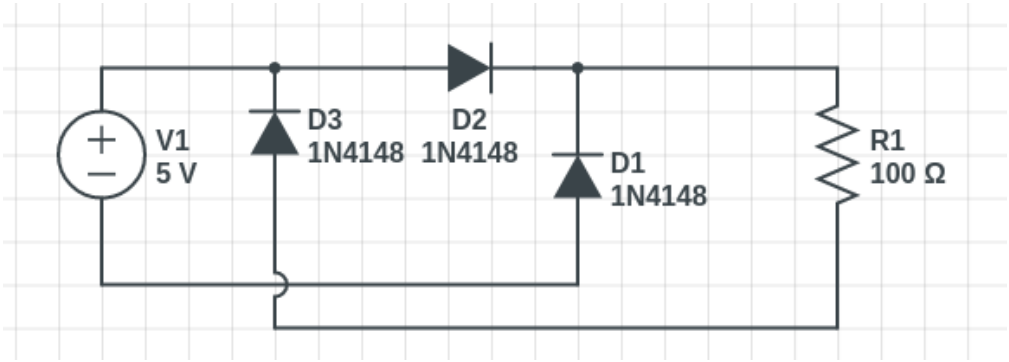


Figure 5: Reverse Voltage Protection Circuit

Appendix 1: problem4.py

```
def problem4():

    import numpy as np
    import scipy as sp
    from scipy import integrate
    import matplotlib as mpl
    import matplotlib.pyplot as plt

    t = np.arange(0, 20.05, 0.05)

    # Initial Conditions about robot
    L1 = 0.30
    L2 = 0.20
    r = 0.08

    # Wheel velocities
    FLW = 0.75*np.cos(t/3.0)
    FRW = 1.50*np.cos(t/3.0)
    BLW = -1.0
    BRW = 0.5

    # initialize x,y, and theta
    theta = [0]
    x = [0]
    y = [0]

    # space saving code
    short = (r*0.05)/4
    A = FLW+FRW+BLW+BRW
    B = -FLW+FRW+BLW-BRW
    C = -FLW+FRW-BLW+BRW

    for i in range(1, 201):
        theta.append(theta[i-1] + short*(1/(L1+L2)*C[i]))
        x.append(x[i-1] + short * (A[i] * np.cos(theta[i-1]) + B[i] * np.sin(theta[i-1])))
        y.append(y[i-1] + short * (A[i] * np.sin(theta[i-1]) + B[i] * np.cos(theta[i-1])))


    u = np.cos(theta)
    v = np.sin(theta)

    plt.quiver(x,y,u,v)
    plt.title("Chapter 5: Problem 4")

    plt.plot(x,y)
    plt.show()

problem4()
```

Appendix 2: Problem5.py

```
def Problem5():
    import numpy as np
    import scipy as sp
    from scipy import integrate
    import matplotlib as mpl
    import matplotlib.pyplot as plt

    timepoints = 1000
    t = np.linspace(0,5,timepoints)
    dt = 5/timepoints

    r = 20
    L = 12
    w1 = 0.25*(t*t)
    w2 = 0.5*t

    print(w1)

    #noise generation
    mu, sigma = 0.0, 0.01
    xerr = np.random.normal(mu,sigma, timepoints)
    yerr = np.random.normal(mu,sigma, timepoints)

    # comment out to include error
    # xerr = np.zeros(timepoints)
    # yerr = np.zeros(timepoints)

    x = np.array([0.0])
    y = np.array([0.0])
    theta0 = np.array(0)

    theta = r/(2*L) * (w1 - w2) * t + theta0
    for i in range(len(t)):
        xnew = (x[i-1] + xerr[i] + (r*dt/2.0) * (w1[i]+w2[i]) * np.cos(theta[i]))
        ynew = (y[i-1] + yerr[i] + (r*dt/2.0) * (w1[i]+w2[i]) * np.sin(theta[i]))
        x = np.append(x, xnew)
        y = np.append(y, ynew)

    plt.title('Simulated Path')
    plt.plot( x , y )
    plt.show()


Problem5()
```