Finding the Trajectory of a Drone Flying with a Constant Wind Velocity

Abinav Chari, Pranav Koushik, Sathvik Vangavolu

Georiga Tech MATH 2552

Contents

Introduction

1.1 Abstract

The purpose of this project is to utilize the Euler's Method to optimize a differential equation that represents the optimal trajectory for a drone, which is assumed to not have a GPS, to get home under the condition of a constant wind velocity. In a real world application of this project, in the event that the GPS of a drone fails, a wind velocity sensor attached to the drone could detect the wind speed and direction; with just this information, the drone could navigate back to its home position if this differential equation methodology is implemented into its system.

Modeling the Flight Path

2.1 Deriving the Differential Equation

We assume the drone will always fly at the same height, and so we can create a 2D graph to represent the situation, with our "home" location set to the origin. The diagram below depicts the initial value problem at hand.

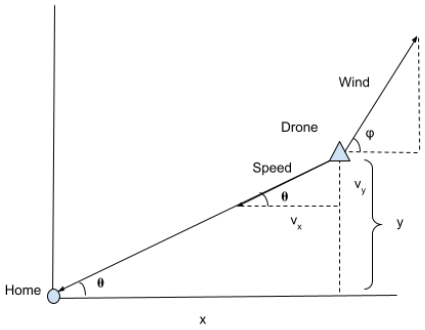

Figure 2.1: Diagram of the Drone

Here, we see the necessary quantities to consider when creating the differential equation. From the position, we must consider the drone's starting position or the horizontal and vertical distance (x and y) from the home position, the angle between the x and y, the drone's velocity and its components (v_x, v_y, θ), and the wind speed's magnitude and its angle relative to the horizontal φ)).

Based on this picture, we can create a system of parametric differential equations for x and y in terms of t. The two factors we will consider that will impact the drone's speed are the drone itself and the wind speed. Furthermore, the drone's speed will stay constant and is always directed toward the origin.

Assume that the drone is in the first quadrant. First let us consider this system without the wind speed. We know that the horizontal component of speed is $-v\cos\theta$ and the vertical component is $-v\sin\theta$ (the values are negative because the origin is to the right of the drone in this diagram). We can also notice that trigonometric functions of θ can be represented in terms of x and y via Pythagoras's Theorem. Thus, we can generate the following system of parametric equations:

$$\frac{dx}{dt} = -v\cos\theta = \frac{-vx}{\sqrt{x^2+y^2}}$$

$$\frac{dy}{dt} = -v\sin\theta = \frac{-vy}{\sqrt{x^2+y^2}}$$

Now, to factor in the wind condition, we simply need to add the respective components of the wind velocity vector, which is pointing to the right. Thus, we will have the following parametrics where v is drone speed, w is the wind speed, and φ is the direction of the wind speed in radians north of east:

$$\frac{dx}{dt} = -v\cos\theta + w\cos\phi = \frac{-vx}{\sqrt{x^2+y^2}} + w\cos\varphi \tag{2.1}$$

$$\frac{dy}{dt} = -v\sin\theta + w\sin\phi = \frac{-vy}{\sqrt{x^2+y^2}} + w\sin\varphi \tag{2.2}$$

We wish to obtain $\frac{dy}{dx}$, and for parametric equations,

$$\frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}.$$

Dividing Equation 2.2 by Equation 2.1, we will therefore get the IVP:

$$\frac{dy}{dx} = \frac{w\sin\varphi\sqrt{x^2+y^2} - vy}{w\cos\varphi\sqrt{x^2+y^2} - vx} \tag{2.3}$$

Euler's Method

The differential equation we obtained in Equation 2.3 cannot be solved via ordinary methods and even by a computer under standard computation time, so the next best approach is to utilize the Euler's Method to approximate the solution.

3.1 Python Code for Euler's Method

```
import math
import matplotlib.pyplot as plt

wind_speeds = [5, 7, 5, 5]
drone_speeds = [10, 6, 10, 10]
wind_angles = [2, 0.78, 3.14, math.pi/4] #lists of testable wind
    speeds, drone_speeds, wind angles
#input corresponding speeds and angle so that the indexes of
    respective lists match

dy_dx = lambda x, y:
  (w*math.sin(phi)*math.sqrt(x**2 + y**2) - v*y)/(w*math.cos(phi)*
    math.sqrt(x**2 + y**2) - v*x)

#generating graphs with euler's method
for i in range(len(drone_speeds)):
  w = wind_speeds[i]
  v = drone_speeds[i]
  phi = wind_angles[i] #selecting respective wind speeds and angles

  x0 = 0; xf = 10 # x-interval
  h = 0.01 #step size
  x, y = (3, 5) #initial condition
  n = (xf-x0)/h #number of complete calulations for euler's method

  x_vals = []
  y_vals = []

  for j in range(1, int(n)): #for loop implementing euler's method
    x_vals.append(x)
```

```
28      y_vals.append(y)
29
30      y = y + dy_dx(x, y) * h
31      x = x + h
32
33  plt.plot(x_vals, y_vals, linewidth=2.0, color = 'green')
34  plt.xlabel('x')
35  plt.ylabel('y')
36  plt.title(label = f'Flight Path {i+1}: v = {round(v, 2)} m/s, w =
        {round(w, 2)} m/s,     = {round(phi, 2)} rad', color = 'red')
37
38  plt.show()
```

The above code was created in Python to generate solution curves for the IVP. The user can use this code to generate multiple graphs at once for comparison purposes. On lines 4-6, one can modify this portion to simulate multiple test cases with different speeds and angles while keeping consistent initial condition across all the graphs for testing purposes. The step size and initial condition can be tweaked on lines 19 and 20.

3.2 Graphs Generated with the Euler's Method Code

Here are some graphs generated via the code. The initial condition for all of the following graphs is $y(3) = 5$, and the step size was $h = 0.01$. For more accurate results, one can tailor the step size.

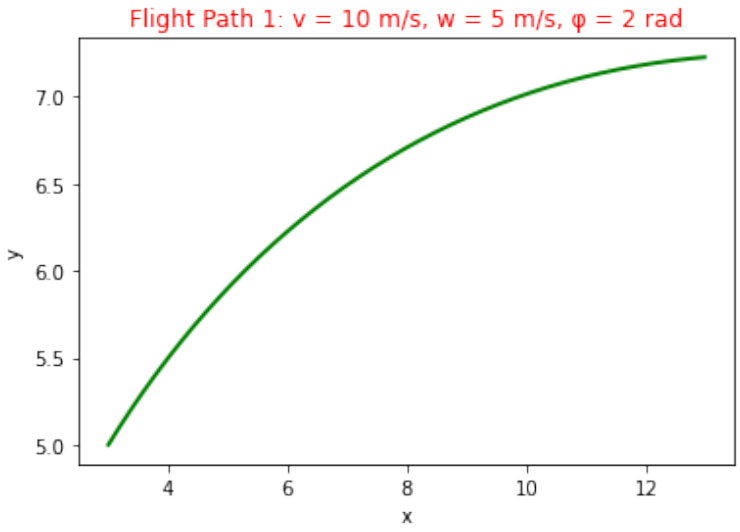

Figure 3.1: Test Case 1

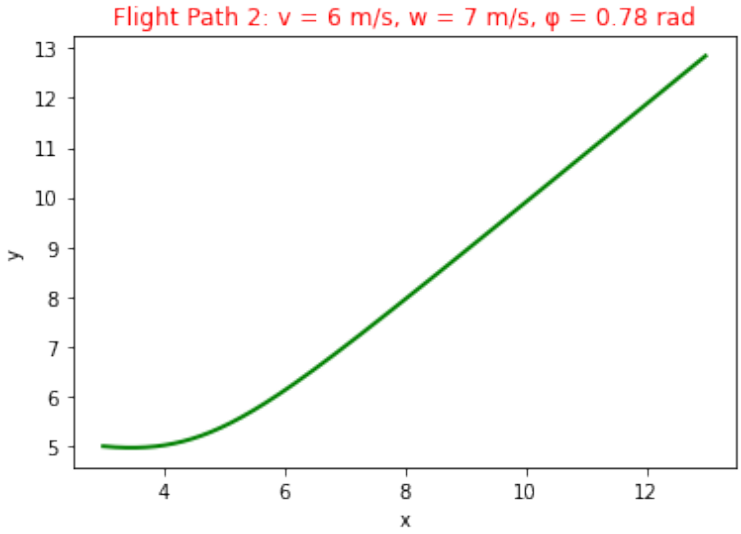

Figure 3.2: Test Case 2

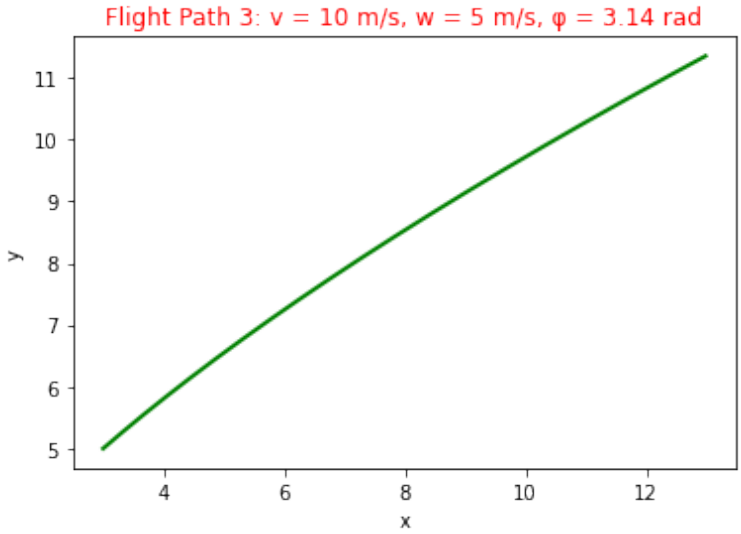

Figure 3.3: Test Case 3

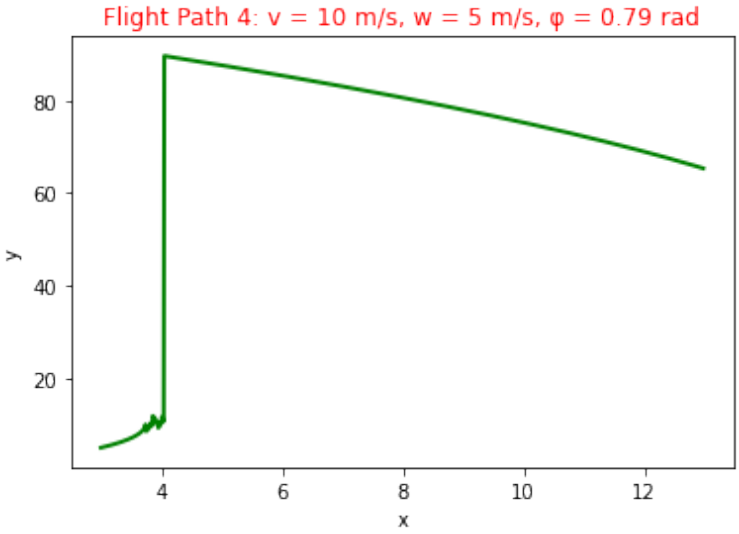

Figure 3.4: Test Case 4 (possible error with step size)

Conclusion

4.1 Conclusion

This project would be especially helpful in the event that a drone's GPS system fails, or if the drone does not have a GPS, it will still be able to navigate back home. The graphs that have been generated thus far depict trajectory for the drone to take under the given conditions for it to return home. There are limitations to this method, however. The step size of 0.01 used in the previous examples may not be small enough for the scenario. There were cases in which the drone's flight path made a considerable jump in the y direction without much movement in the x direction (such as in Figure 3.4), and this problem was most likely due to the step size. Decreasing step size would increase the computational power required to generate the graphs, so other methods about optimizing this particular problem's solution could include the use of the improved Euler's method or the Range-Kutta for more accurate results.

4.2 Extensions

For future versions of this project, we can extend it to a third dimension by adding in a parametric equation for the z axis. That way, we don't need to assume that the drone does not change altitude. Also, at varying altitudes, wind speeds are likely to be different. Therefore, we could implement a non-constant wind speed in which the wind speed is a function of time or altitude and use that function in our set of parametric equations. Lastly, we can incorporate other factors that are at play in real life, such as a limited battery life to the drone (which could be modeled as a function of time), maneuvering around obstacles in the way of the path, and drag/gravitational forces.

Bibliography

[1] Brannan J. R., Boyce W. E. (2015). Differential Equations: An Introduction to Modern Methods and Applications. [Yuzu]

[2] Desmos Graphing Calculator. (n.d.). Desmos. https://www.desmos.com/calculator Krueger, R., Stachura, E. (2018). "6-024-S-DronePackageDelivery."

[3] https://www.simiode.org/resources/5422 Wolfram Research, Inc. (2022). Wolfram—Alpha. Wolfram—Alpha. Retrieved April 18, 2022, from https://www.wolframalpha.com/