Glider Flight Simulation

Lilo Heinrich and Jackie Zeng

Question

How does the initial velocity of a glider plane affect its trajectory, specifically the distance traveled before reaching the ground?

This question is both explanatory and predictive. Our model strives to explain the behavior of current gliders as well as predict a glider's flight length given its initial velocity. Both of us found this question interesting as we were interested in glider planes, Lilo, in particular, has flown over a dozen times in gliders. Glider plane pilots and enthusiasts will also find this topic important because understanding how velocity affects the aircraft will allow them to have a better flight experience. By asking this question, we will learn whether the starting velocity of a glider is important, and if so, what speed should one start at to fly the maximum distance?

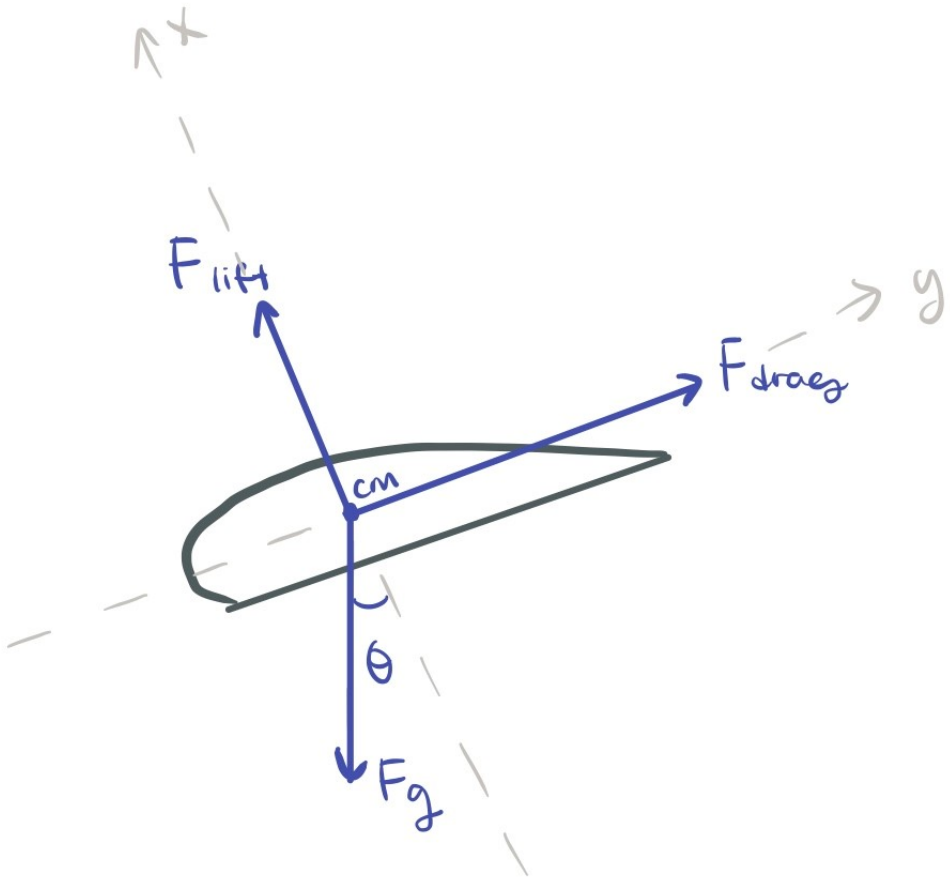
Methodology

To answer this question, we must first understand what a glider is and its related physics. A glider is a special kind of aircraft that does not have an engine. However, large piloted gliders, the type that we are modeling, are complete with everything else - standard aircraft parts, construction, and flight control systems, etc. During flight, since there is no engine for thrust on the aircraft, there are only three main forces on a glider - lift, drag, and gravity. These forces are vector quantities with magnitude and direction components. Lift and drag are aerodynamic forces. The lift is directed perpendicular to the direction of the glider and the drag is directed opposite of the direction of flight. The gravitational force acts through the center of gravity of the aircraft and is always directed towards the earth.

We plan to answer our proposed question by simulating a glider's flight starting from an altitude of 1000 meters. By simulating a glider's complete trajectory, we can find out a glider's total horizontal distance traveled. We based our project on the equations detailed in a paper on glider flight written by University of Connecticut Physics Students.

Below is the diagram we used to create our equations.

forces on a cambered airfoil



Equations

$$m\frac{dV}{dt} = -mg\sin\theta - \frac{1}{2}\rho V^2 C_D S$$
$$mV\frac{d\theta}{dt} = -mg\cos\theta + \frac{1}{2}\rho V^2 C_L S$$
$$\frac{dX}{dt} = V\cos theta$$
$$\frac{dY}{dt} = V\sin theta$$

Parameters

h_init = initial height of glider
v_init = initial velocity of glider
theta_init = initial angle of glider
g = acceleration due to gravity
rho = density of air
mass = mass of glider plane
Cl = coefficient of lift
Cd = coefficient of drag
S = frontal area of plane wing

State

R: position vector
V: velocity vector
T: angle (theta) vector

Assumptions

Since glider flight is a complicated physics topic, we had to make several assumptions in order to simplify into a model that we would be able to execute.

We assumed firstly that the air density remains constant with altitude, which is a fair assumption as long as the glider's maximum altitude isn't very high. Given that we are starting our glider at only 1km high, which is relatively high for a glider but not high enough to see a big change in air density, this assumption should have a minimal impact on our model.

Next, we assumed that the air is not moving. Since we are interested in how the initial velocity affects the distance travelled by the glider, we didn't want to complicate the model by adding in moving air, such as thermals and areas of sink that would move the sailplane by exerting additional forces on it. This results in relatively short flight times in our simulation, since usually glider pilots will spiral on the thermals to gain altitude as necessary to stay in the air while in our model the air is unmoving.

We also haven't written any code preventing the glider from flying too slow, below stall speed. This is because we could not find the stall speed of a given glider on Wikipedia. But to make sure that this isn't a problem we have made sure that our velocity values never go below about 20 m/s, a fairly typical minimum flight speed for

gliders.

Lastly, our biggest assumption is that the wing profile of a glider is a standard cambered airfoil, and that therefore the coefficient of lift should be set to the values shown on the graphs from the wikipedia pages below at an angle of attack of 0 degrees. We were unable to find information on the coefficients of lift and drag for specific gliders so we had to instead resort to this more general assumption in order to create our model.

In [1]:
```
# Configure Jupyter so figures appear in the notebook
%matplotlib inline

# Configure Jupyter to display the assigned value after an assignment
%config InteractiveShell.ast_node_interactivity='last_expr_or_assign'

# import functions from the modsim.py module
from modsim import *

# for trigonometric functions
import math

# for graphing results
import matplotlib.pyplot as plt
```

In [2]:
```
# Units:

s = UNITS.second
deg = UNITS.degree
N = UNITS.newton
kg = UNITS.kilogram
m = UNITS.meter
km = UNITS.kilometer
h = UNITS.hour
```

Out[2]: hour

In [3]:
```
class Glider:
    def __init__(self, name, area, mass_min, mass_max, Cl, Cd):
        self.name = name
        self.area = area
        # According to Wikipedia, gliders function better with heavier weight
s. We chose to use
        # the median rate to simulate average conditions.
        self.mass = (mass_min + mass_max) / 2
        self.Cl = Cl
        self.Cd = Cd
```

In [4]:
```
glider = Glider("Schleicher_ASK_13", 17.5 *m**2, 295 *kg, 480 *kg, 0.5, 0.125)
```

Out[4]:
```
<__main__.Glider at 0x7fbdce7ba9b0>
```

In [5]:
```
params = Params(h_init = 1000.0 *m,
                v_init = 30.0 *m/s,
                theta_init=0.0,
                g = 9.8 *m/s**2,
                rho = 1.2 *kg/m**3,
                Cl = glider.Cl,
                Cd = glider.Cd,
                mass = glider.mass,
                S = glider.area)
```

Out[5]:

	values
h_init	1000.0 meter
v_init	30.0 meter / second
theta_init	0
g	9.8 meter / second ** 2
rho	1.2 kilogram / meter ** 3
Cl	0.5
Cd	0.125
mass	387.5 kilogram
S	17.5 meter ** 2

In [6]:
```
def make_system(params):
    """
    Makes a System object for the given conditions.

    params: Params object with glider info and constants

    returns: System object
    """
    h_init, v_init, theta_init = params.h_init, params.v_init, params.theta_in
it
    init = State(R=Vector(0.0, h_init), V=v_init, T=theta_init)

    t_end = 3600 * s # 1 hour
    dt = 1 * s # 1 second

    return System(params, init=init, t_end=t_end, dt=dt)
```

In [7]:
```
system = make_system(params)
```

Out[7]:

	values
h_init	1000.0 meter
v_init	30.0 meter / second
theta_init	0
g	9.8 meter / second ** 2
rho	1.2 kilogram / meter ** 3
Cl	0.5
Cd	0.125
mass	387.5 kilogram
S	17.5 meter ** 2
init	R [0.0 meter, 1000.0 meter] V 30.0...
t_end	3600 second
dt	1 second

In [8]:
```
def slope_func(state, t, system):
    """
    Compute derivatives of the state.

    state: position, velocity, angle
    t: time
    system: System object

    returns: derivatives of position, velocity, and angle
    """
    R, V, T = state
    mass, g, rho, Cd, Cl, S = system.mass, system.g, system.rho, system.Cd, sy
stem.Cl, system.S

    drdt = Vector(V*math.cos(T), V*math.sin(T))
    dvdt = -g*math.sin(T) - rho*V**2*Cd*S/(2*mass)
    dtdt = -g*math.cos(T)/V + rho*V*Cl*S/(2*mass)

    return drdt, dvdt, dtdt
```

In [9]:
```
def event_func(state, t, system):
    """
    Returns whether the glider has reached the ground.

    state: position, velocity, angle
    t: time
    system: system object

    returns: vertical component of position
    """
    R, V, T = state
    return R.y
```

In [10]:
```
results, details = run_ode_solver(system, slope_func, events=event_func)
details
```

Out[10]:

	values
success	True
message	A termination event occurred.

In [11]:
```
results
```

Out[11]:

	R	V	T
0.000000	[0.0 meter, 1000.0 meter]	30.0 meter / second	0
1.000000	[28.446141361417077 meter, 1001.1151763820393 ...	26.860114728730117 meter / second	0.04170348338202773 dimensionless
2.000000	[53.85834056587474 meter, 1002.1677647419053 m...	24.26122990768919 meter / second	0.000922476847128581 dimensionless
3.000000	[77.09737666577317 meter, 1001.3271303837507 m...	22.78707437308742 meter / second	-0.10513480672456976 dimensionless
4.000000	[99.20414231446522 meter, 997.6618936441829 me...	22.68919258348414 meter / second	-0.2322806242375137 dimensionless
...	...	...	...
154.000000	[3949.1270116190826 meter, 24.898284843221482 ...	26.49030782156189 meter / second	-0.24497866306808846 dimensionless
155.000000	[3974.8263850788016 meter, 18.473441480122393 ...	26.49030782084312 meter / second	-0.24497866306182076 dimensionless
156.000000	[4000.525758537833 meter, 12.0485981170928 meter]	26.49030782028278 meter / second	-0.24497866307375246 dimensionless
157.000000	[4026.2251319963025 meter, 5.623754753685175 m...	26.49030781998432 meter / second	-0.24497866309626473 dimensionless
157.875314	[4048.7201510128366 meter, 0.0 meter]	26.49030781996153 meter / second	-0.24497866311786917 dimensionless

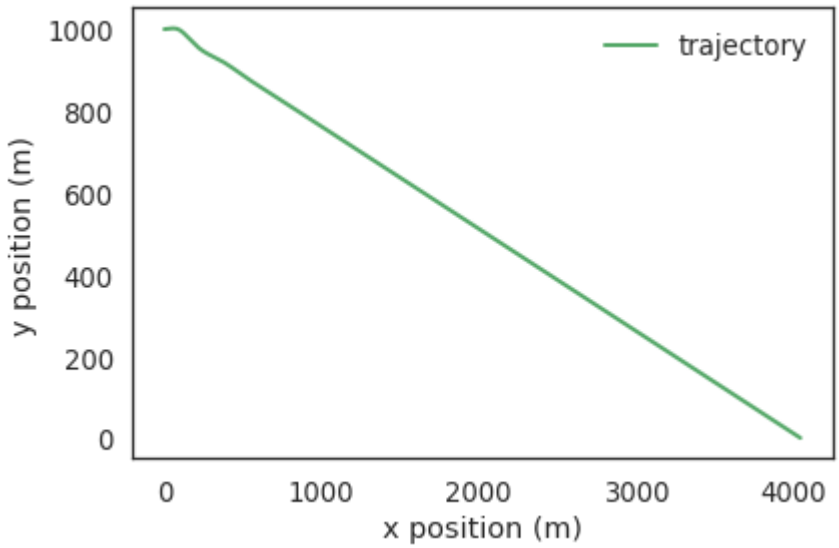
159 rows × 3 columns

Results

What results did your model generate? What metrics did you record?

In [12]:
```
def plot_trajectory(results):
    """
    Plots a particular timeseries on a 2D grid.

    results: timeseries resulting from the run_ode_solver function

    returns: plot of the glider flight
    """
    xs = results.R.extract('x')
    ys = results.R.extract('y')
    plot(xs, ys, color='C2', label='trajectory')

    decorate(xlabel='x position (m)',
             ylabel='y position (m)')

plot_trajectory(results)
```

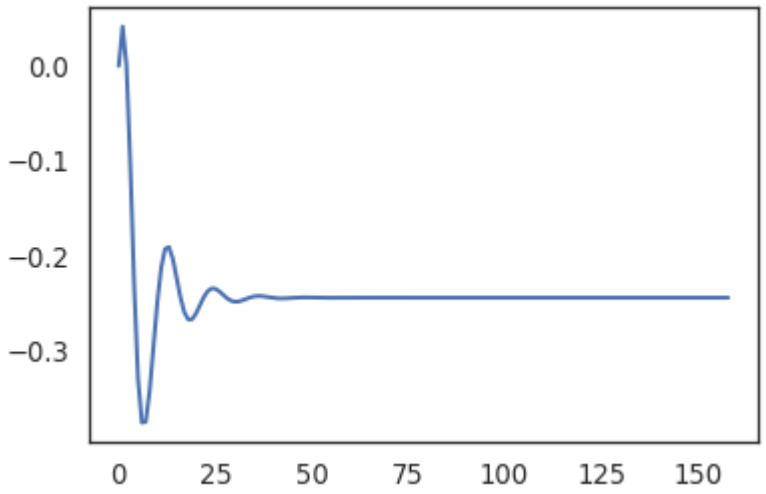
Out[12]:



In [13]:
```
plot(results.T)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x7fbdcb556e10>]
```

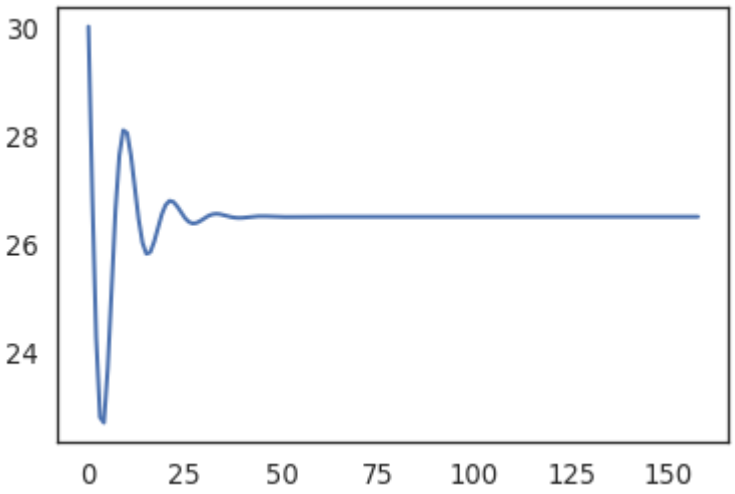
Out[13]:



In [14]:
```
plot(results.V)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x7fbdcb4a2da0>]
```

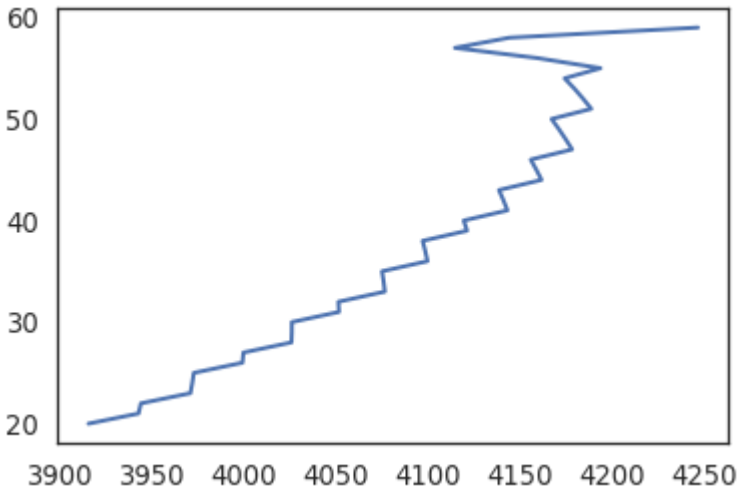
Out[14]:



In [15]:
```
sweepResults = SweepSeries()
for i in range(20, 60):
    params = Params(h_init = 1000.0 *m, v_init = i *m/s, g = 9.8 *m/s**2, rho
= 1.2 *kg/m**3, mass = glider.mass,
                    Cl = glider.Cl, Cd = glider.Cd, S = glider.area, theta_ini
t=0.0)
    system = make_system(params)
    results, details = run_ode_solver(system, slope_func, events=event_func)
    xs = results.R.extract('x')
    index = len(xs)-2
    sweepResults[xs[index]] = i
#     print("angle:", i, " distance flown:", xs[index], " time flown:", index,
"seconds")

plot(sweepResults)
```

Out[15]: [<matplotlib.lines.Line2D at 0x7fbdcb4856a0>]

Out[15]:



Interpretation

How do the results answer the question? Is your answer valid? What are the limitations of your model?

When we simulated a glider flying with an initial angle of 0 degrees and inital speed of 30 m/s, the trajectory showed an almost linear descent. Looking at the velocity and angle graphs however, they show more oscillation. Since we are letting the glider fly on its' own without taking control and setting a fixed pitch angle, this oscillation makes a lot of sense. After a few cycles, the oscillations settle and it reaches its' equilibrium at approximately -0.24 radians or 13 degrees pitch. As the NASA webpage showed, gliders often/always fly with the nose down, usually at a shallow angle to avoid stalling. This shows that 13 degrees is a fairly correct value.

Next, we swept over different starting velocities, starting with a minimum of 20m/s and a maximum of 60m/s. Gliders typically fly between 20 and 40 m/s and the Scheichler ASK 13 that we used in our model has a maximum of about 55 m/s, but we expanded the range of velocities to be higher because we wanted to see if there were interesting behaviors at this extreme. As can be seen in the graph above, above around 55 m/s the line breaks and becomes squiggly. This shows where the glider isn't behaving as consistently and starts becoming more difficult to handle. This actually is validated by the fact that the maximum recommended flight speed for this glider is 124 mph, or 55.1 m/s, according to its' Wikipedia entry.

Citations

NASA Glider flight forces explanation: (https://www.grc.nasa.gov/WWW/k-12/airplane/glidvec.html (https://www.grc.nasa.gov/WWW/k-12/airplane/glidvec.html))

Wikipedia pages on Lift and Drag coefficient parameter values:

- https://en.wikipedia.org/wiki/Lift_coefficient (https://en.wikipedia.org/wiki/Lift_coefficient)
- https://en.wikipedia.org/wiki/Airfoil (https://en.wikipedia.org/wiki/Airfoil)

Book source on Scheichler ASK 13 glider coefficient of drag:

- https://books.google.com/books?id=ldIBDgAAQBAJ&pg=SA3-PA6&lpg=SA3-PA6&dq=Schleicher+ASK+13+coefficient+of+lift&source=bl&ots=k5k6fgyzb-&sig=ACfU3U2jq269G7DHtKLsfvVSJnL_8BqALQ&hl=en&sa=X&ved=2ahUKEwi_j8blj5vmAhUvWN8KHX7PC (https://books.google.com/books?id=ldIBDgAAQBAJ&pg=SA3-PA6&lpg=SA3-PA6&dq=Schleicher+ASK+13+coefficient+of+lift&source=bl&ots=k5k6fgyzb-&sig=ACfU3U2jq269G7DHtKLsfvVSJnL_8BqALQ&hl=en&sa=X&ved=2ahUKEwi_j8blj5vmAhUvWN8KHX7PC

Wikipedia pages that were not using:

- https://de.wikipedia.org/wiki/Geschwindigkeitspolare (https://de.wikipedia.org/wiki/Geschwindigkeitspolare)
- https://en.wikipedia.org/wiki/Polar_curve_(aerodynamics (https://en.wikipedia.org/wiki/Polar_curve_(aerodynamics))

Glider paper: (https://www.phys.uconn.edu/~rozman/Courses/P2200_14F/downloads/glider/glider-2014-11-05.pdf (https://www.phys.uconn.edu/~rozman/Courses/P2200_14F/downloads/glider/glider-2014-11-05.pdf))