Original Work Progress Assessment #1

Name: Ansh Pathak

Project: STRATOS

Project Description:

Date: 2/18/2026

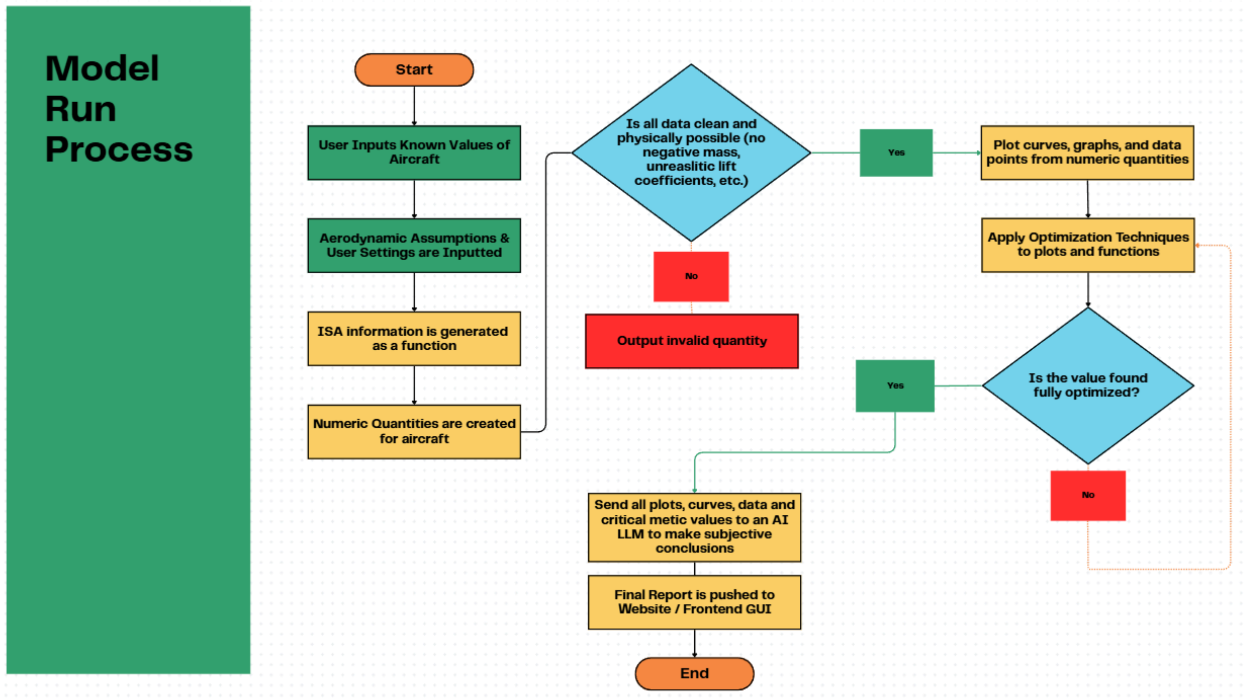
Introduction and Statement of Purpose

STRATOS (Simulation of Thrust, Rate-of-climb, Aerodynamics, and Total Operating States) is a High-Fidelity Aircraft Performance & Envelope Simulation Environment. It is a meaningful aerospace engineering tool that serves a real purpose, enabling specific calculations on aircraft. It serves as a physics-based aircraft flight performance simulator intended for preliminary aerospace engineering analysis. This is a custom-built computational tool that models the performance of a subsonic aircraft, utilizing aerodynamic and performance equations to establish a first-principles model. Real use cases for this could include, but are not limited to, engineers testing whether a proposed aircraft configuration can meet arbitrary predetermined mission requirements. Finally, this simulator will output plots and key metrics on set data quantities. To serve as a physical demonstration, data will be input into this simulator, analyzed, and interpreted to showcase an example use case in operation. STRATOS is comprised of four subsystems, all working in harmony with each other to deliver the final result.

The Atmospheric Condition & Standard Earth Systems subsystem is the foundation of the aircraft performance simulator; its primary purpose is to compute set atmospheric properties affecting the aircraft as a function of altitude using standardized, physics-based approaches. Without an accurate atmospheric model, the simulator would not produce meaningful results, making this subsystem critical to the overall practicality and usability of the project. This model will be based on the International Standard Atmosphere (ISA), the industry standard reference atmosphere. The ISA assumes a dry, hydrostatic atmosphere with no weather effects. For the scope of this project and constraints, the model will be implemented and cover the scope all the way from sea level through the troposphere, up to approximately 15 kilometers, which fully covers the operational gradient of most subsonic aircraft. This is an appropriate assumption and aligns with real aerospace engineering practices. The model will not account for real-world weather phenomena such as wind, turbulence, humidity, or temperature deviations from standard day conditions. Gravity will be treated as a constant value (9.81 m/s/s), and Earth will be assumed to be non-rotating and perfectly spherical. The Raw Logistic Modulator serves as a liaison between the input aircraft values and the core physics-based calculations of the simulator. The primary function is to take raw input data, such as geometry, weight, propulsion information, and aerodynamic coefficients, and convert them into standardized, numeric variables specific to each aircraft. The result is that the simulator now operates on clean, validated data, preventing physically impossible scenarios (e.g., negative mass or unrealistic lift coefficients). The Optimal Statistics Calculator has the duty of extracting meaningful performance metrics from the raw numerical outputs generated by the previous modules. The main deliverables are performance curves, contour plots, and critical metric values (minimum thrust required, maximum level flight speed, etc.). The report summarizer will be the final part of the framework; its main purpose is to synthesize all the raw data and statistics and turn that into a written/visual format for end users to easily understand and draw conclusions from. Powered by an AI Learning-Language Model, users will get almost instantaneous reports once the simulation is finished. An API key will be used to link the interface to a backend model available to public access. A template will be created to ensure similarity between all reports and to prevent AI model deviation.

Current State of Progression

Firstly, I have produced an algorithm and flowchart for how the simulation will run, from beginning to end. This allows me to follow a step-by-step plan as I develop the rest of this model.


Model Run Process
Start
User Inputs Known Values of Aircraft
Aerodynamic Assumptions & User Settings are Inputted
ISA information is generated as a function
Numeric Quantities are created for aircraft
Is all data clean and physically possible (no negative mass, unreaslitic lift coefficients, etc.)
Yes
No
Output invalid quantity
Plot curves, graphs, and data points from numeric quantities
Apply Optimization Techniques to plots and functions
Is the value found fully optimized?
Yes
No
Send all plots, curves, data and critical metic values to an AI LLM to make subjective conclusions
Final Report is pushed to Website / Frontend GUI
End

Next, in terms of developing the model, I have made significant progress. The atmospheric model is fully complete and features full outputs between 0 and 15km. I have an aircraft object file, defining the characteristics of the aircraft, storing all the parameters. The performance file is part of the Raw Logistic Modulator and accurately computes all the variables using equations outlined in my Original Work Proposal. The main.py file calls the aircraft object, which I used for testing the simulator while developing it. Finally, I have made a UI application, which runs locally on the user's computer and offers a streamlined UI to input the aircraft and observe the results. Currently, the simulator takes in all inputs, and outputs a drag vs velocity plot and power vs velocity plot along with the stall speed and air density for the aircraft at that altitude.

I have embodied myself in a very rigorous process in the past two weeks, developing this, putting in the hours to land myself weeks ahead of schedule, for which I plan on using the additional time at the end to add additional features/write more informational reports on this model.

So far, this project has been fully modeled in Python. I attempted to use SCILAB, but after determining that the learning curve was too steep rather than using Python, I opted for the latter, as I already had experience with it.

Finally, to test the validity of the work I have done, I researched the key input values (mass, coefficients, wing area, etc.) of a standard, fully-fueled Boeing 737-800, and input those values into my simulator. After that, it calculated the stall speed for the aircraft, and after comparing that with the official stall speed of the Boeing 737-800, my simulator produced a number within a 1.5% accuracy, proving that I have been accurate in my development of this model thus far.

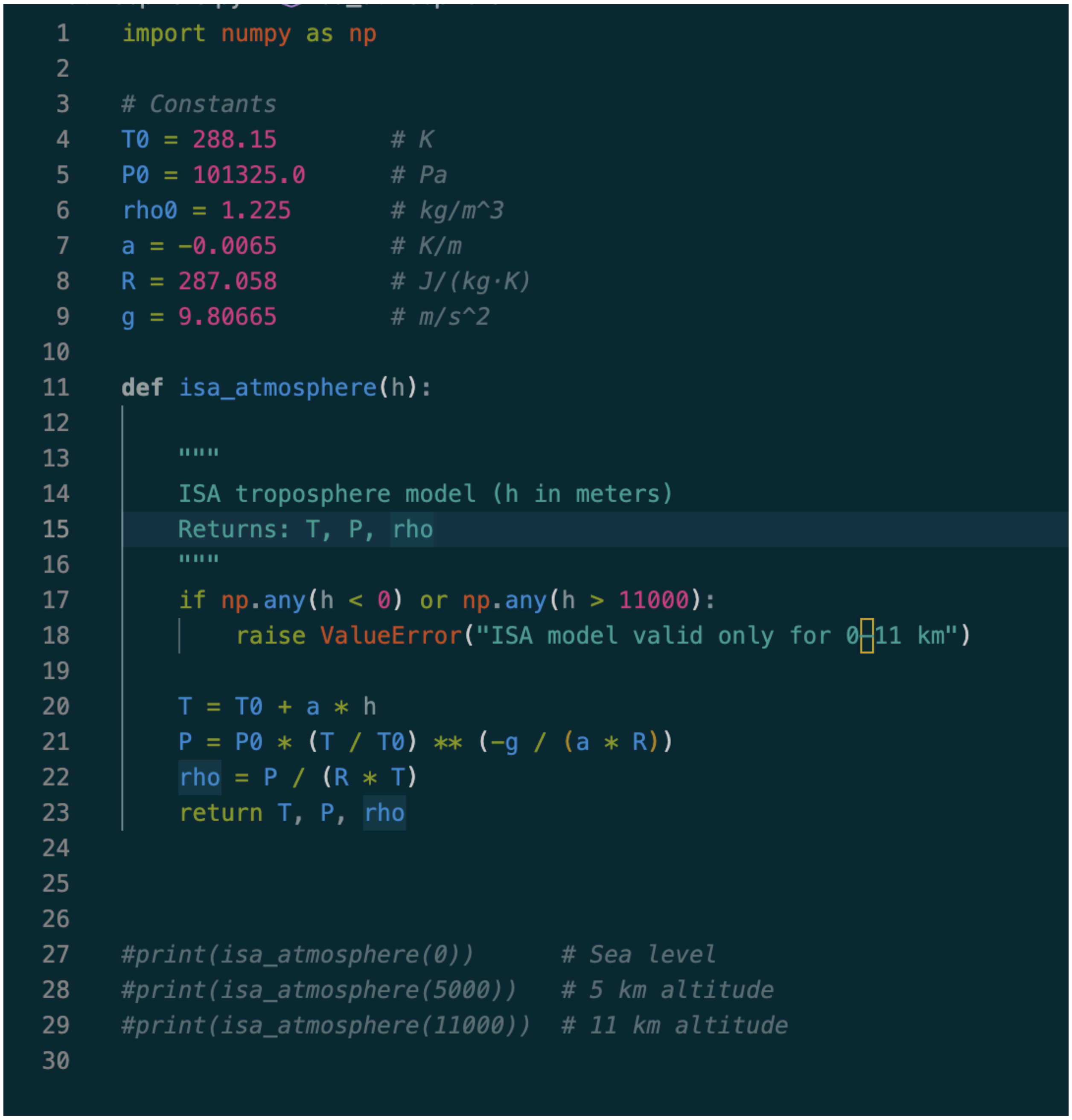
I recently showed my mentor, Dr. Cataldo, the progress I had made. He told me that the atmospheric module, aircraft object, and aerodynamics module were excellently built, and if I wanted to take it a step further, I could adopt a higher-precision drag model as well as account for more variable atmospheric conditions (turbulence, weather, etc). For the app overlay, he suggested that I bring the key metrics to the top of the page, followed by the graphs and plots I had (for which I had drag vs airspeed and power-required & power-available vs airspeed). Overall, he was very impressed with the cohesiveness and the complete design.

Moving forward, to address these comments, I plan to separate this page into multiple tabs. For example, I will have one for atmospheric conditions, one for lift, one for drag, etc. This would allow my entire application to be streamlined and organized whilst maximizing the data available to the end-user. In addition to this, we discussed increasing the maximum altitude of the simulator from 11km to possibly up to 15km, to cover the full operating altitudes of all commercial aircraft (considering that newer aircraft fly even higher due to efficiency and design improvements to reduce drag and increase fuel efficiency). Similar to this and my stall speed metric, I will add an optimal altitude metric for an engineer to determine the optimal altitude before the cost-advantage of flying higher is detrimental.

So far, my main challenges have been with ensuring the code has failsafes, and there are no technical issues that may skew the results (buggy code that creates negative mass/altitude). I have had to brush up on my Python skills for this. For my app overlay, I did not have any experience creating an app for a project similar to this, so I utilized AI to create a temporary, visually friendly interface that I am using for development while I move it onto a more long-term, sustainable platform.

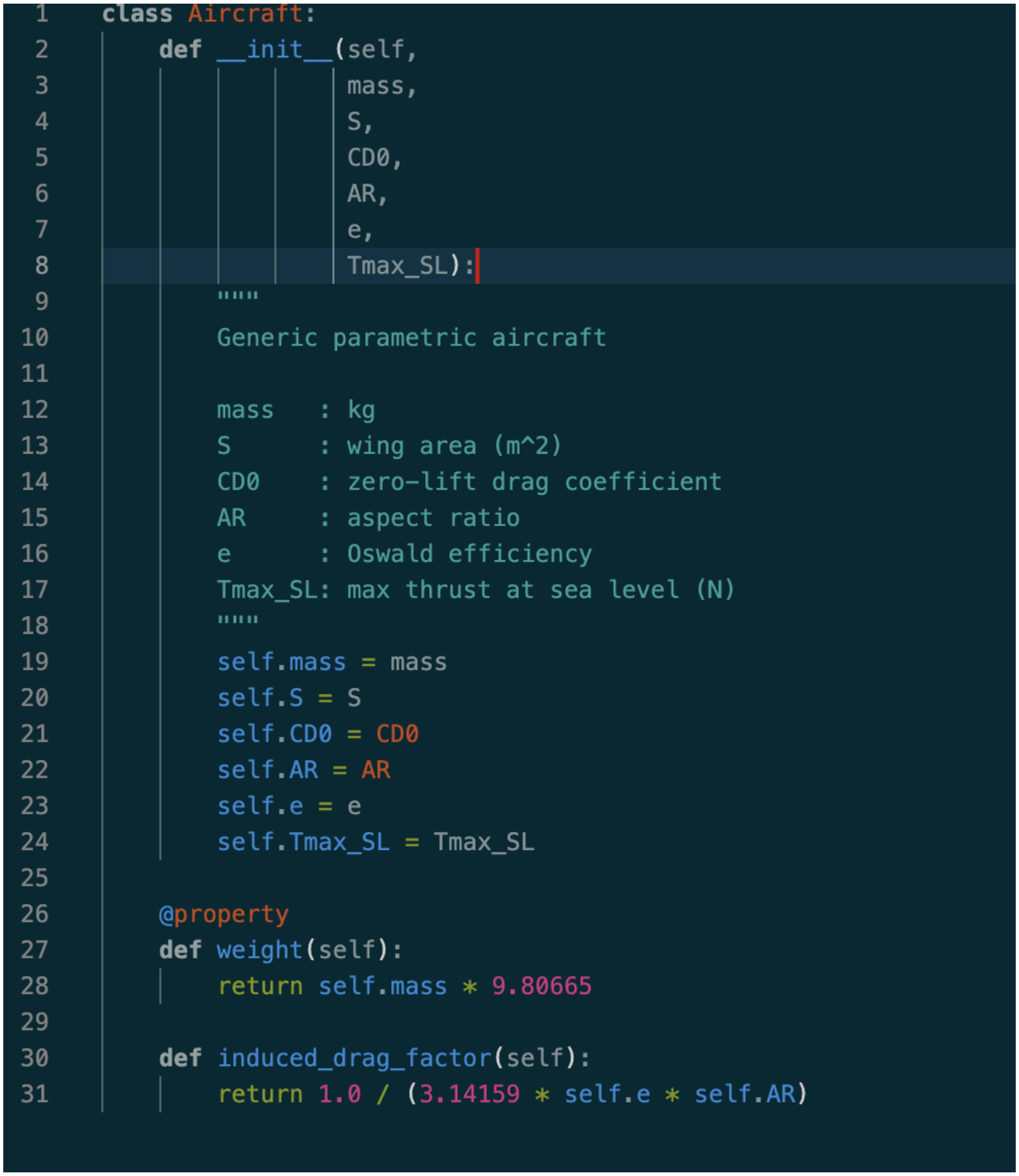
Below, I am attaching screenshots of both the code and what the current interface looks like.

atmosphere.py



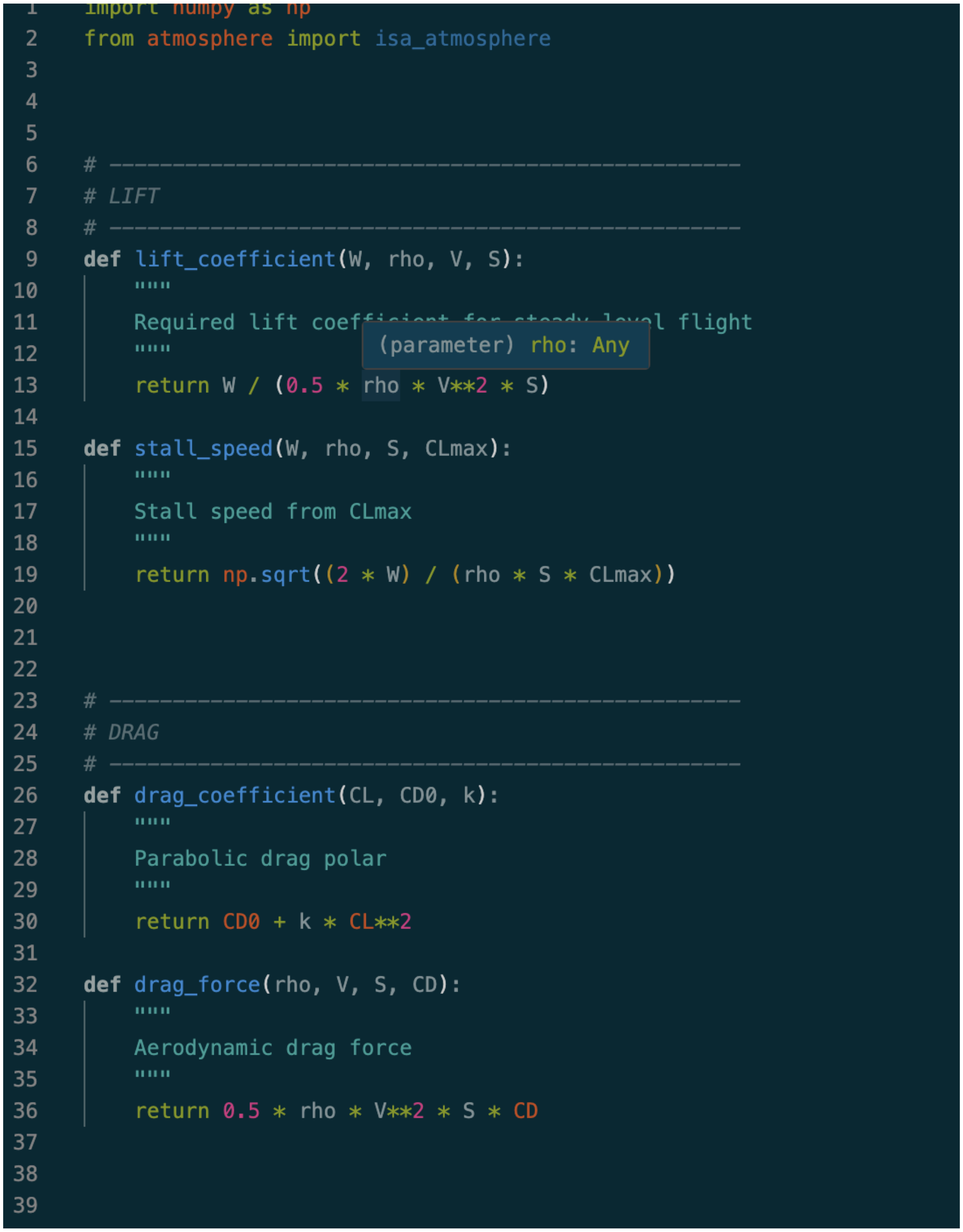
```python
import numpy as np

# Constants
T0 = 288.15          # K
P0 = 101325.0        # Pa
rho0 = 1.225         # kg/m^3
a = -0.0065          # K/m
R = 287.058          # J/(kg·K)
g = 9.80665          # m/s^2

def isa_atmosphere(h):

    """
    ISA troposphere model (h in meters)
    Returns: T, P, rho
    """
    if np.any(h < 0) or np.any(h > 11000):
        raise ValueError("ISA model valid only for 0–11 km")

    T = T0 + a * h
    P = P0 * (T / T0) ** (-g / (a * R))
    rho = P / (R * T)
    return T, P, rho



#print(isa_atmosphere(0))       # Sea level
#print(isa_atmosphere(5000))    # 5 km altitude
#print(isa_atmosphere(11000))   # 11 km altitude

```

aircraft.py



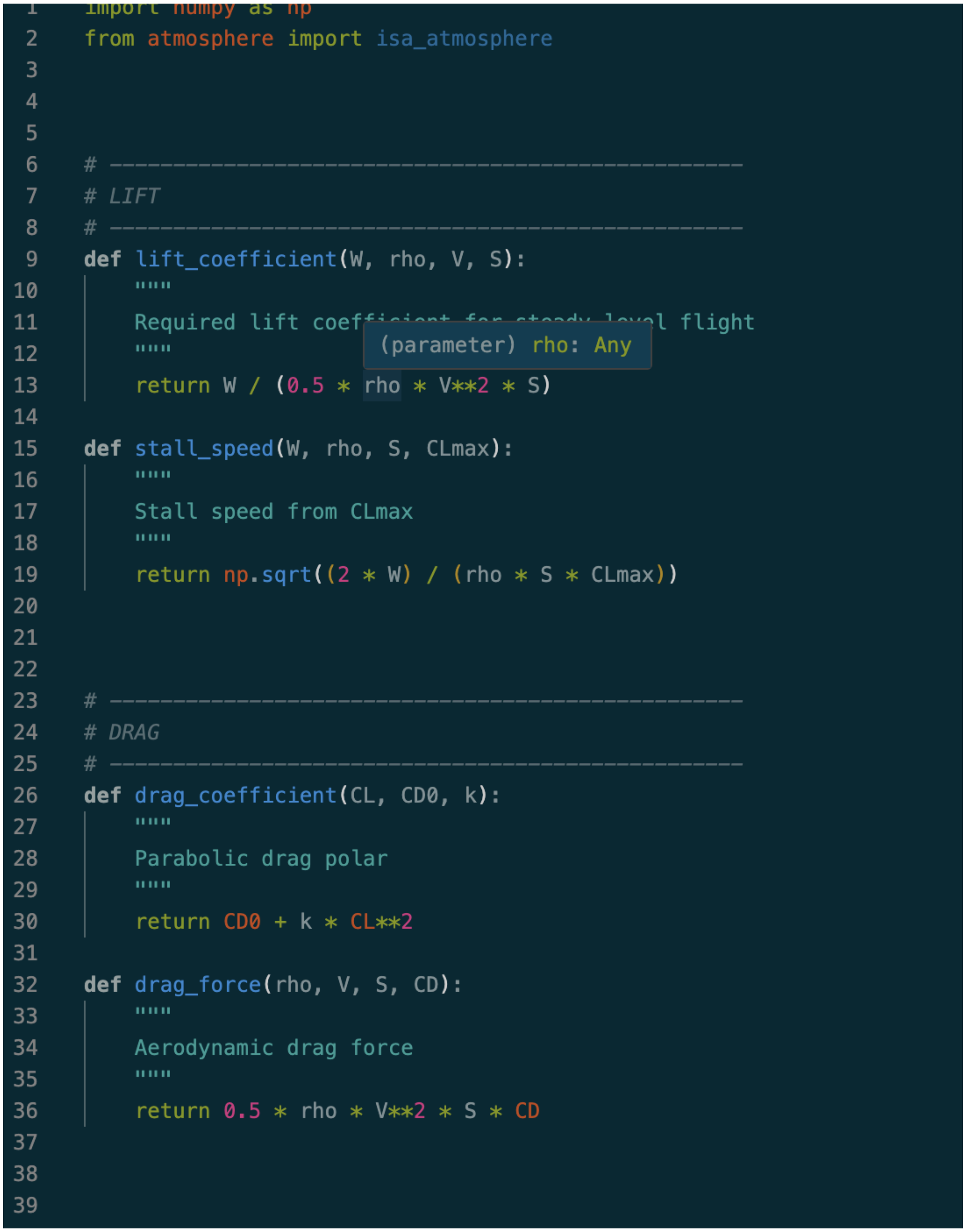
```python
class Aircraft:
    def __init__(self,
                 mass,
                 S,
                 CD0,
                 AR,
                 e,
                 Tmax_SL):
        """
        Generic parametric aircraft

        mass    : kg
        S       : wing area (m^2)
        CD0     : zero-lift drag coefficient
        AR      : aspect ratio
        e       : Oswald efficiency
        Tmax_SL: max thrust at sea level (N)
        """
        self.mass = mass
        self.S = S
        self.CD0 = CD0
        self.AR = AR
        self.e = e
        self.Tmax_SL = Tmax_SL

    @property
    def weight(self):
        return self.mass * 9.80665

    def induced_drag_factor(self):
        return 1.0 / (3.14159 * self.e * self.AR)
```

performance.py



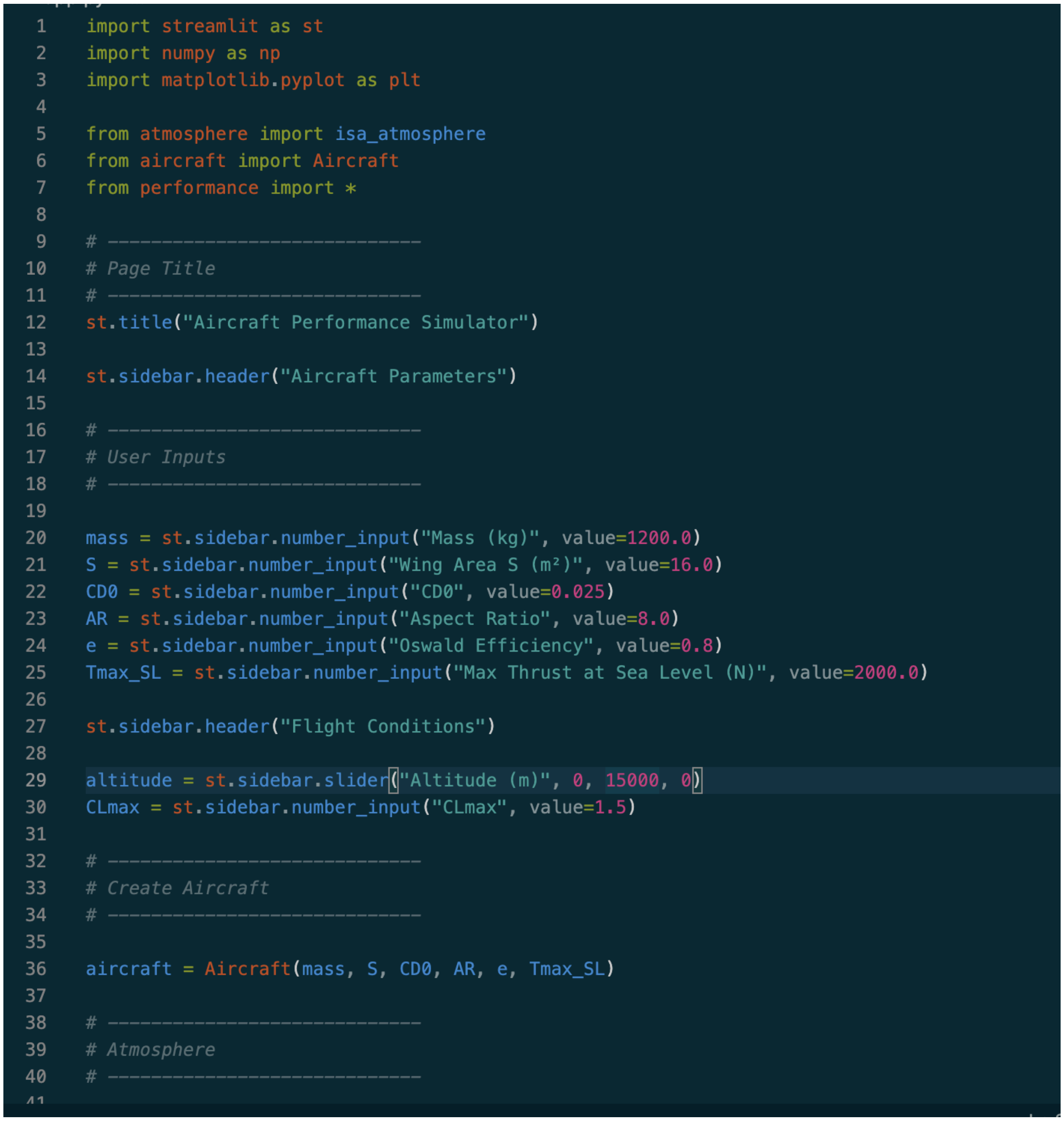
```python
import numpy as np
from atmosphere import isa_atmosphere



# ---------------------------------------------------
# LIFT
# ---------------------------------------------------
def lift_coefficient(W, rho, V, S):
    """
    Required lift coefficient for steady level flight
    """
    return W / (0.5 * rho * V**2 * S)

def stall_speed(W, rho, S, CLmax):
    """
    Stall speed from CLmax
    """
    return np.sqrt((2 * W) / (rho * S * CLmax))



# ---------------------------------------------------
# DRAG
# ---------------------------------------------------
def drag_coefficient(CL, CD0, k):
    """
    Parabolic drag polar
    """
    return CD0 + k * CL**2

def drag_force(rho, V, S, CD):
    """
    Aerodynamic drag force
    """
    return 0.5 * rho * V**2 * S * CD



```

(performance.py has more lines of code not shown here) main.py



```python
import numpy as np
from atmosphere import isa_atmosphere



# --------------------------------------------------
# LIFT
# --------------------------------------------------
def lift_coefficient(W, rho, V, S):
    """
    Required lift coefficient for steady level flight
    """
    return W / (0.5 * rho * V**2 * S)

def stall_speed(W, rho, S, CLmax):
    """
    Stall speed from CLmax
    """
    return np.sqrt((2 * W) / (rho * S * CLmax))



# --------------------------------------------------
# DRAG
# --------------------------------------------------
def drag_coefficient(CL, CD0, k):
    """
    Parabolic drag polar
    """
    return CD0 + k * CL**2

def drag_force(rho, V, S, CD):
    """
    Aerodynamic drag force
    """
    return 0.5 * rho * V**2 * S * CD



```

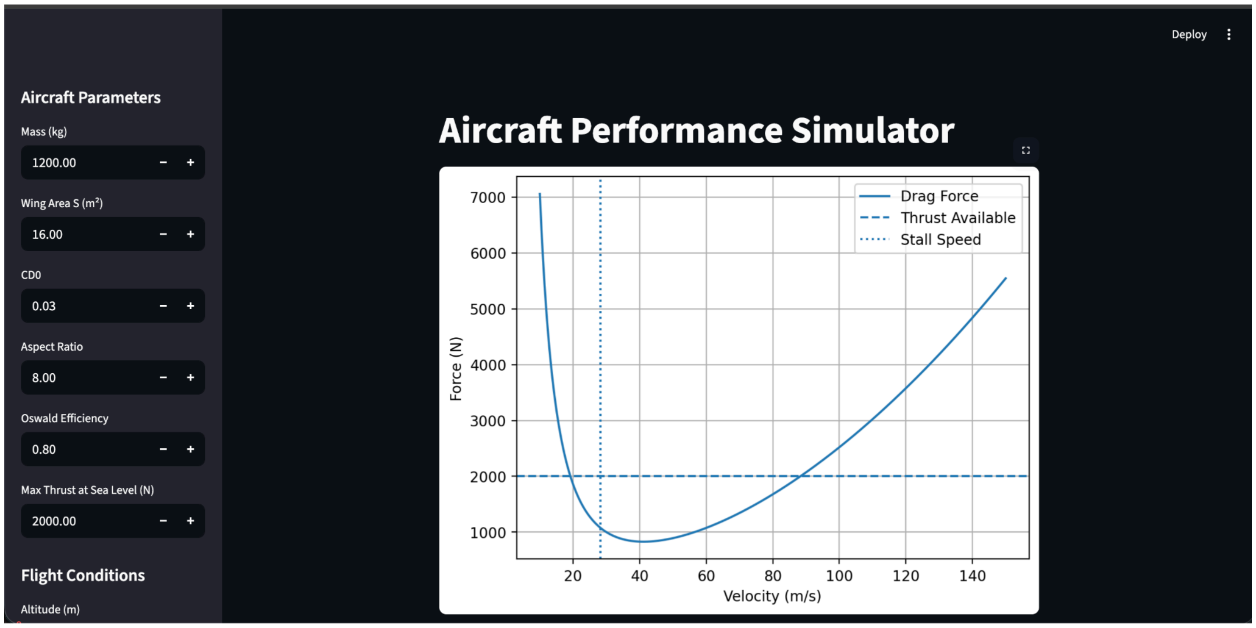
(main.py has more lines of code not shown here) app.py



```python
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt

from atmosphere import isa_atmosphere
from aircraft import Aircraft
from performance import *

# -----------------------------
# Page Title
# -----------------------------
st.title("Aircraft Performance Simulator")

st.sidebar.header("Aircraft Parameters")

# -----------------------------
# User Inputs
# -----------------------------

mass = st.sidebar.number_input("Mass (kg)", value=1200.0)
S = st.sidebar.number_input("Wing Area S (m²)", value=16.0)
CD0 = st.sidebar.number_input("CD0", value=0.025)
AR = st.sidebar.number_input("Aspect Ratio", value=8.0)
e = st.sidebar.number_input("Oswald Efficiency", value=0.8)
Tmax_SL = st.sidebar.number_input("Max Thrust at Sea Level (N)", value=2000.0)

st.sidebar.header("Flight Conditions")

altitude = st.sidebar.slider("Altitude (m)", 0, 15000, 0)
CLmax = st.sidebar.number_input("CLmax", value=1.5)

# -----------------------------
# Create Aircraft
# -----------------------------

aircraft = Aircraft(mass, S, CD0, AR, e, Tmax_SL)

# -----------------------------
# Atmosphere
# -----------------------------
```

(app.py has more lines of code not shown here)

Interface Design



Aircraft Parameters

Mass (kg)
1200.00

Wing Area S (m²)
16.00

CD0
0.03

Aspect Ratio
8.00

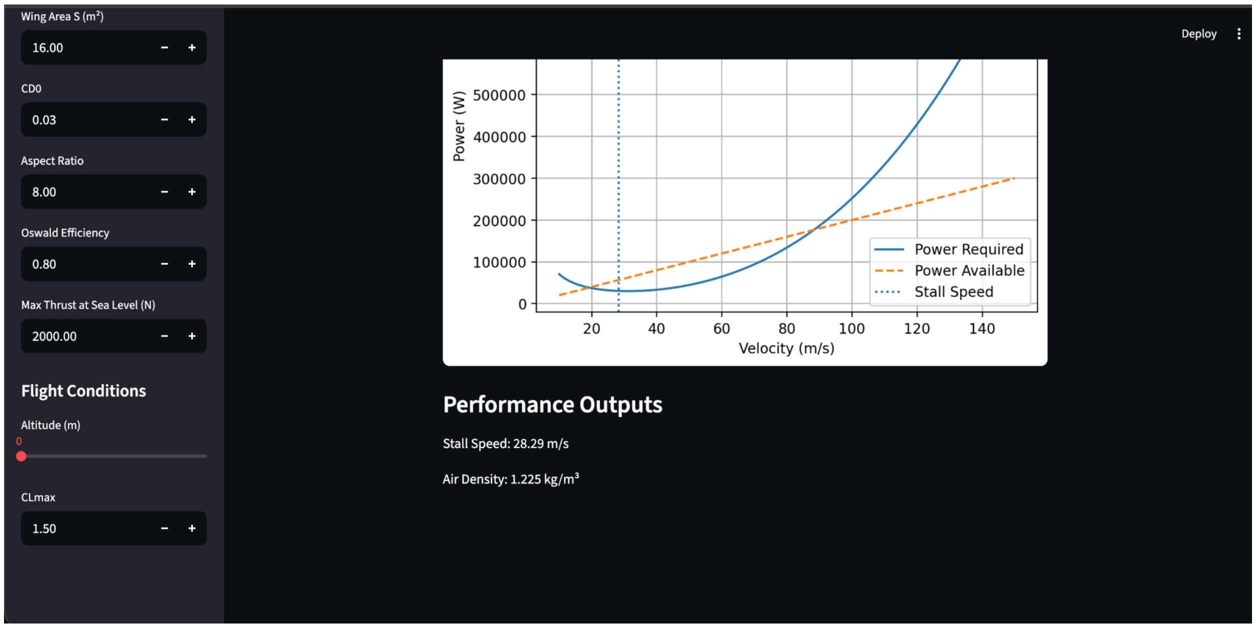
Oswald Efficiency
0.80

Max Thrust at Sea Level (N)
2000.00

Flight Conditions

Altitude (m)

Aircraft Performance Simulator



Wing Area S (m²)
16.00

CD0
0.03

Aspect Ratio
8.00

Oswald Efficiency
0.80

Max Thrust at Sea Level (N)
2000.00

Flight Conditions

Altitude (m)
0

CLmax
1.50

Performance Outputs

Stall Speed: 28.29 m/s

Air Density: 1.225 kg/m³

CALENDAR

4/29/2026 - Final Presentation Night
4/19/2026 - All work to be completed by

Week of 01/18/26

- Understanding of MATLAB, its capabilities, and the list of all software that is needed
- Research & Understand the meanings of the inputs/outputs (Range, Endurance, and Power metrics)

Week of 01/25/26

- Compile all necessary equations, formulas, coefficients, and constants needed

Week of 02/01/26

- Research and note-taking performance curves, and all other analyses that will be used in the model

Week of 02/08/26

- Module 1: Develop & Create (Find modules here: Information for Report)

Week of 02/15/26

- Module 2: Develop & Create

Week of 02/22/26

- Module 3: Develop & Create

Week of 03/01/26

- Module 4: Develop & Create

Week of 03/8/26

- REVIEW COMPLETED WORK AND PUBLISH FIRST REVISION (brainstorm if any functionality could be added or if anything is not practically possible). Complete tested model on predetermined aircraft (1st Time)

Week of 03/15/26

- Module 5: Determine specific curves and plots

Week of 03/22/26

- Module 5: Create

Week of 03/29/26

- Wrap Up / Add additional features

Week of 04/05/26

- Test model on another predetermined aircraft (2nd time)

Week of 04/12/26

- Finalize all data and findings, and publish the model
- List assumptions, metrics, and limitations
- Create a final report