# Optimized Flight Path Planning for Urban Air Mobility

Sutton Yazzolino

March 10, 2025

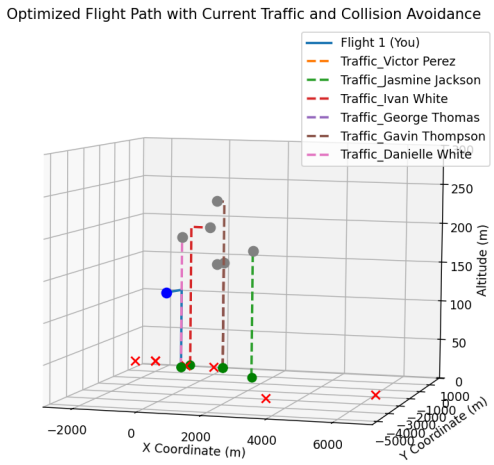


Figure 1

## Introduction

I want to start this off with a quick question: wouldn't it be cool to be able to do away with the limits of bumpy asphalt roads, often indirect routes, and wildly varying transport times which depend not only on your access to a road but your distance from a highway? Flying from Stanford to Skyline Ridge would be a stunningly short trip with huge benefits(i.e. The Fantastic View) without the detriments of the horrendously windy road that gets you there(and possible road sickness). However, such a fantastic technology comes with many setbacks in its deployment. One of the issues with flying cars, which is less often talked about, is Airspace Management. Flying cars would need to operate within a complex airspace system, requiring sophisticated air traffic management and communication protocols, which are not yet in place. With this draft of a simulator, route planner, as well as risk management, I hope to woo you into believing I should get a few extra points of credit.

# Background

Flying cars have long been envisioned as the future of urban transportation. Despite decades of speculation, widespread deployment has been hampered by challenges in feasibility, safety, and regulatory constraints. Recent trends suggest that while mass-market adoption may be far off, early deployments for upper-class consumers and specialized services (such as emergency air taxis) are becoming more likely.

A key challenge in the development of urban air mobility is the centralized route planning necessary to manage the complex airspace. Instead of each manufacturer developing proprietary navigation systems, a unified platform can aggregate data from multiple sources to optimize flight paths. This concept allows for efficient management of air traffic and real-time risk assessment. My project embodies this vision by integrating probabilistic models to simulate, optimize, and dynamically update flight paths, including emergency rerouting and collision avoidance, based on simulated weather and traffic data.

In this write-up, I'll describe the code implementation of my system, which uses probabilistic methods—such as bootstrapping, Bayesian updating, and Monte Carlo simulation—to manage uncertainties in weather, traffic density, and downwash interference. The system not only estimates expected travel times and risks but also provides emergency features to re-route other flights if necessary.

# Overview of the Project

This Python-based simulation optimizes flight paths by integrating advanced probabilistic methods. The core components include:

- **Expected travel time and variance calculation**, based on geographic coordinates and vehicle speed.
- **Probabilistic risk assessments** of weather and traffic using Monte Carlo sampling, Bayesian updating, and heuristic modeling.
- **Collision avoidance**, including real-time adjustments to flight paths and emergency rerouting capabilities.
- **Visualization** of flight trajectories using matplotlib.

# Probabilistic Implementations

## Weather Data via Boot Strapping

Historical weather data (particularly wind speed) is analyzed through bootstrapping to assess the probability of adverse weather conditions during a future trip. Wind speeds exceeding 7 m/s indicate higher risk scenarios. The bootstrap method samples historical

data repeatedly to provide an empirical estimate of weather risk probabilities. Bootstrapping allows for the creation of realistic datasets by capturing natural variability and uncertainty inherent in weather conditions, providing a reliable estimation even in the absence of extensive real-time data.

### Traffic Data via Monte Carlo Sampling

Traffic risk is computed by analyzing active flights around the user's planned start time. Monte Carlo sampling identifies flight interactions and calculates probabilities based on traffic density within defined temporal windows, simulating realistic urban traffic congestion patterns.

### Bayesian Updating for Dynamic Risk Assessment

Real-time data updates, such as newly available wind speed measurements, trigger Bayesian updates to refine previous risk estimations. The Bayesian method merges prior risk estimates with new observed data, adjusting predictions dynamically throughout the flight duration.

### Normal Distribution for Arrival Time Probability

Arrival times are probabilistically modeled using a normal distribution characterized by a mean (expected travel time) and variance. The variance considers both inherent travel uncertainty and additional risk-induced variability(windspeed), computed through a combination of weather and traffic risk factors. This implementation provides probabilistic arrival time windows for users.

### Collision Avoidance and Downwash Interference

Monte Carlo simulations are also applied to calculate intersection probabilities between flights, considering variable travel times and temporal overlaps. Additionally, the probability of downwash interference—air disturbance from overhead flights—is computed using heuristic probability estimates, based on spatial and altitude differences between flight paths(See Appendix C Figure 2).

## Code Architecture and Implementation

The Python-based simulation is structured into several clear functional modules:

- **Flight Path Generation**: Paths are created in three segments (vertical ascent, horizontal cruising, and descent), factoring in cruising altitude optimizations based on probabilistic risk.
- **Optimization**: Iterative exploration of cruising altitudes and start times utilizes probabilistic risk estimations to minimize a combined cost of travel time and risks.

- **Dynamic Updates**: Bayesian updates recalibrate risk assessments when new data becomes available, ensuring adaptive and resilient path planning.
- **Emergency Rerouting**: In emergency scenarios, flights dynamically adjust paths, prioritizing emergency flights through altitude-based separation. This intends to provide a minimum travel time estimate for yourself to your destination.

## Visualization and Simulation

Flight paths and their dynamic adjustments are visualized through 3D animations, clearly indicating the user's optimized flight alongside active traffic flights, highlighting collision avoidance maneuvers and probabilistic risk updates.

# Conclusion

This project implements an draft of an end-to-end simulation for optimizing flight paths in urban air mobility. By integrating advanced probabilistic methods—including Bayesian updating, bootstrapping, and risk-based optimization—with dynamic updates and emergency rerouting, the system demonstrates how centralized route planning for flying vehicles can be achieved. By dynamically integrating these approaches, the simulation provides robust risk management, accurate travel estimates, and flexible emergency rerouting, contributing to safer and more efficient airspace management. This project not only calculates and visualizes optimal flight routes but also provides essential risk metrics and safety guarantees, paving the way for future development and potential real-world applications. Future improvements to this simulation could include: more realistic flight pathing, more attention to air space zoning laws and requirements, delaying flights based on who signed up to fly first, making explicit pathways to navigate a populated area safely, and using more realistic data to use for the weather and traffic estimations.

# Appendix:

## A Monte Carlo Sampling Data Generation

Monte Carlo sampling in the data generation scripts is used to estimate the probability of adverse weather by drawing random samples from historical wind speed data. The process repeatedly samples the data using `np.random.choice` to mimic the natural variability of wind speeds. Each sampled wind speed is then compared to a threshold (e.g., 7 m/s) to determine if it represents an adverse condition. Finally, the fraction of samples exceeding the threshold is calculated to approximate the probability of encountering adverse weather.

## B Traffic Data Normal Distribution

Travel times are generated using a weighted random sampling of hours—favoring rush hour—and the duration, delay, and reliability values are computed using modulo arithmetic to produce cyclic variations. This approach mimics the natural variability in urban traffic patterns and outputs the data in a CSV format for use in simulation and analysis.

## C Downwash Visual Depiction

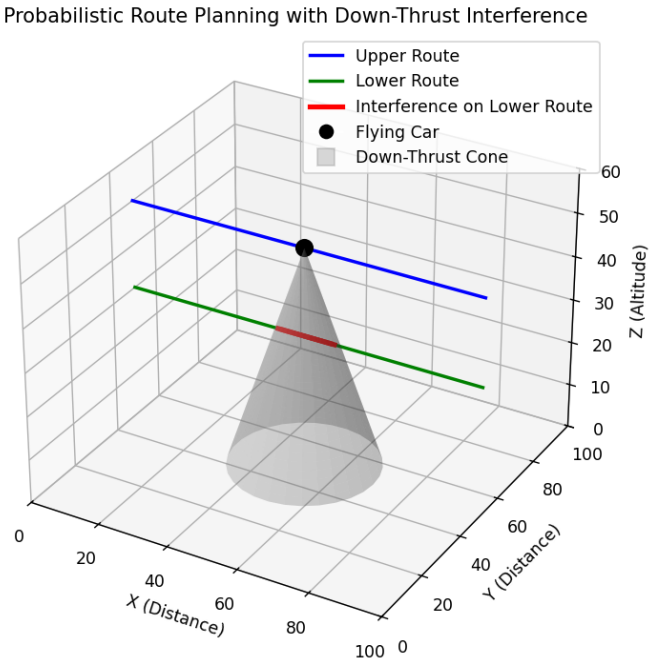


Figure 2

## D AI Usage Discussion

I utilized AI largely in generating a good looking write-up for the project after writing the main ideas in a google docs. I let the AI restructure my writing to make it more clear and add its own interpretations of the code to make sure there was a good description of the methods used in the project. Idea generation was entirely without AI, but code was AI assisted. I aimed to leave the probabilistic method implementations untouched by AI, while having AI assist in other areas like debugging, and anything pertaining to Matlab visualizations.

## E Other Notable Features

- **Preprocessing:** Converting predefined geographic coordinates of starting locations and destinations to a local Cartesian coordinate system.
  - Weather and traffic data are drawn from CSV files. In the absence of live data, simulated datasets are used. Predefined starting and destination locations (with latitude, longitude, and altitude) are converted to local x-y coordinates using standard geographic conversion formulas.
- **Flight Path Generation:** Using a `FlightPath` class, each flight path is generated as a piecewise-linear route (vertical ascent, horizontal cruise, vertical descent) based on a helicopter speed of approximately 95 mph (2548 m/min). Note, this is unrealistic for take-off and landing speeds, but is an accurate cruise speed for some VTOL vehicles.
- **Visualization and User Interaction:** A console-based interface allows user input, and an interactive 3D animation visualizes the flight paths. The user also controls how many paths to display. Users select their starting location, destination, and desired start time (input in hours and converted to minutes). They also specify the number of flight paths to plot. The simulation provides readouts of:
  - Expected travel time and variance.
  - Risk assessment probabilities (weather, traffic, and combined).
  - Probability of arriving by a specified target time.
  - Downwash interference probability.

  A matplotlib 3D animation visualizes the computed flight paths, with the user's flight highlighted and other active flights displayed as dashed lines.

# References


OpenAI. "ChatGPT." OpenAI, 2023, https://chat.openai.com/.

Matplotlib. "Matplotlib: Visualization with Python." Matplotlib, 2023, https://matplotlib.org/.

Piech, Chris. *Probability for Computer Science: Course Reader for Stanford CS109*. CS109, Department of Computer Science, Stanford University, Jan 2025, Version 1.0.