AAE 497: UAV Simulator
Spring 2020 Simulation Guide

Ajay Chandra, Alan Gelman, Youssef Noureddine

Overview:
The focus of this AAE 497 project was to make upgrades to a given UAV Simulator. The team has chosen to add more aircraft to the available fleet and update the aircraft's weight in flight to account for fuel consumption. This report discusses how to run the simulator code, the upgrades performed by the team, and further possible improvements that can be made. It is recommended to run the simulator using Python 3.8, as the upgrades were made using this version. Prior versions may not contain the necessary toolboxes for ideal simulation.

How to use:
This code was primarily run using command prompt. After having all appropriate files saved, the directory in the command prompt can be changed to the correct file location using the cd command. Once the file location is set, the code can be executed by running the *inputs.py* code. While the code is running, a progress bar will appear, and it will either reach 100%, or it will stop once the aircraft reaches an altitude of 0 or if the aircraft has run out of fuel. If the altitude reaches zero, then the program will output the percent completion, and a statement saying "Simulation stopped due to aircraft altitude reaching 0". After this, the data is output to a CSV file. To view this file, the directory can be changed to *post_processing*, and by entering *output.csv*. In addition to this, the output can be more clearly viewed by running the *plot.py* command in the *post_processing* directory. This command will output several plots including time histories of trajectory, lift, weight, thrust, and drag. All input values can be changed by editing the *inputs.py* file in IDLE in between the sections labeled as "begin input fields" and "end input fields". Input variables that can be changed include start time, end time, time step size, airplane type, initial position vector, initial velocity vector, and angle of attack (in radians).

Improvements:

1. **Aircraft fleet**
   The team added two aircraft, namely the Boeing 747 and the B-17 Bomber. This task involved adding the relevant aircraft's name as an additional line in the *inputs.py* file, creating a separate class for the aircraft (containing the aircraft's name, mass, wing area, wingspan, maximum available thrust, maximum speed, mass of fuel and coefficients of drag), and lookup tables (LUTs) containing data concerning the relevant angle of attack vs lift coefficients.

   The lookup tables for each aircraft were found on github. All values that the code takes as input and produces as output are in SI units.

2. **Weight calculations**
   A new addition to the simulator, and the most significant, is the weight variation as a function of time. This is to account for the fuel burn. This however added an extra input requirement from the user, now the user will be expected to know and input how much fuel the aircraft will be starting at in the 'inputs' file. We first tried to find the Thrust

Specific Fuel Consumption (TSFC) of engines of all the planes but were unable to find most of them online and so we had to find a different method to calculate fuel burn. A value we were able to find is the fuel burn at the aircraft's maximum speed and by making the assumption that fuel consumption varies linearly with speed (which it does not) we were able to approximate a fuel burn rate.

We noticed when the airplane runs out of fuel it plummets to the ground with an acceleration of about 9.8. Since the code does not yet calculate drag accurately at transonic/supersonic speeds we saw that the plane can reach Mach numbers exceeding 10 right before crashing, so we decided to stop the program as soon as the aircraft ran out of fuel to avoid plotting the trajectory it takes to drop to earth.

3. **Weight variations as a function of altitude**
   This addition, although not very significant, was made after noticing g was assumed to be a constant 9.81. We manually calculated by hand that high altitude planes can experience gs as low as 9.8 during their flight. We therefore decided that since it was based on the inverse square of the height from the center of earth that it was significant enough to include in the code. That along with the fact that the rounded value of 9.81 would produce a rounded weight which would be cycled through the loop approximately 1000+ times (depending on the simulation delta t the user has selected) would make a significant enough difference at the end of the plane's trajectory

Samples Values
Although the code will accept all values, only few of those values will work, if you put a -0.1 radian (-6 degrees) angle of attack, most planes will crash. For that reason, we have attached a sample of data you can use for the code that would produce a stable flight and from there you can alter the values as you like.

Design	End time (sec)	Altitude (m)	Velocity (m/s)	AOA (rad)	Fuel Mass (kg)	Scenario
B17	2000	3000	72	0.1	6000	-
B737	2000	9000	240	0.08	20000	-
B737	20000	9000	240	0.08	10000	Out of fuel
B747	2000	9400	240	0.07	150000	-
C172	2000	4000	50	0.02	100	-
T38	2000	6000	150	0.2	2000	-
T38	2000	6000	150	0.05	2000	Zero Altitude

Observing the improvement

Although the gravitational constant varying improvement is not noticeable on the plots, by inserting a print command to output the value of g at every interval shows that it does indeed slightly vary the result by up to 0.2%. The main way the improvement can be noticed is that as the flight time progresses, the cruising altitude slowly increases and weight (and consequently thrust) decrease. This is to be expected, as the weight of the plane decreases due to the fuel burn, the altitude of the plane increases due to the unbalanced force but as the altitude increases the air density decreases and so less lift is generated (since lift is directly proportional to air density)

Possible Modifications

1. Calculate drag properties at transonic, supersonic, hypersonic speeds
This would help us calculate the trajectory after a plane runs out of fuel along with allowing for faster flying planes in our model

2. Updating fuel burn according to TSFC
This allows for a much more accurate burn rate that would better simulate the trajectory of the plane

3. Factor in initial location (as latitude and longitude)
So that we can factor in varying air densities and temperatures in different regions around the world and so that we are able to plot the trajectory of the plane on a map

4. Allow user to request for cruising conditions as opposed to a fixed angle of attack
Adding different features to the program would make the program useful to a larger variety of settings.