Procedures

The classes: *Airplane.m, Paraset.m, Conditions.m, recovery.m, Trajectory.m, atmos.m, BestDifference.m, Simulation.m, ExamplePlots.m, AreaFinder.m, AreaFinder2.m* (located above, indicated by tab name) have been designed to collectively run the program such that the optimal parachute diameter will be indexed given certain specifications such as the plane type the simulation was run for and the properties of the (preliminary researched) relative strongest parachute applicable to this scenario. The size (diameter) of the parachute is the dependent variable such that the plane type can be the independent variable.

The *Airplane.m* class declares the properties of the airplane which can later be initialized by the specific airplane. The properties include mass (of the airplane), bottomReferenceArea, frontalReferenceArea, name (of the airplane), numPassengers, initialConditions, Parasets (the parachute sets that will be used), XDragCoefficient and YDragCoefficient (the drags in the horizontal and vertical planes respectively. Two functions were written in this class to declare XDrag (the horizontal drag) as

$$XDrag = \frac{density \cdot speed^2 \cdot frontalReferenceArea \cdot XDragCoefficient + dragOfRespectiveParachutes}{2}$$

The equation for the YDrag (Horizontal Drag) was inputted into the Airplane.m class as

$$ZDrag = \frac{density \cdot speed^2 \cdot frontalReferenceArea \cdot ZDragCoefficient + dragOfRespectiveParachutes}{2}$$

The *Paraset.m* class declares the properties of the parachutes which can later be initialized with the specifications of the most effective drag coefficient (determined by prior research). The properties include numParachutes (number of Parachutes), parachuteMass, inflatedDiameter, dragCoefficient, tetherMass, nominalDiameter, shape and material. Two functions were written to represent the mass of the parachute set and drag if the parachute set respectively.

$$mass = numParachutes(tetherMass + parachuteMass)$$

Given the art of the inflatedDiameter = $\frac{inflatedDiameter^2}{2} \cdot \pi$, the

$$Drag = \frac{density \cdot speed^2 \cdot area \cdot DragCoefficient \cdot numParachutes}{2}.$$

The *Conditions.m* class initializes constant properties such as the earthMass, G (universal gravitational constant and earthRadius. Other properties include location, stationCode, tOffset (temperature offset), date, and windArray; these variables are used to access random weather conditions to test the parachute system against. Functions to access rather conditions follow

Aspects such as density, speed of sound, temperature, pressure, kinematic viscosity, altitude and density ratios. Density can be measured using temperature offset of the *atmos.m* class
$[rho, a, T, P, nu, z, sigma] = atmos$

Gravity can be represented by $= \dfrac{G \cdot earthMass}{heightOfPlane + earthRadius^2}$

Wind inputs for the simulation can be entered from a windArray that represents westerly wind (2nd column) and northerly wind (3rd column) as a matter of altitude (1st column) (data stored in MATLab Simulation).

The *recovery.m* class declares time steps as to output values in the simulation with the first time step being 0.1 and the timespan being from 0 to 5000. The ODE45 (ordinary differential equations solver) was used collectively in this class. State derivatives represented as a function of descent as a matter of time and states with positions in $x, y,$ and z being declared as states 1 through 3 and respective momenta being declared as states 4 through 6. Variables such as gravity, mass and density are called; and vectors for momenta and velocity relative to the earth are declared. Expressions to represent the forces in $x, y,$ and z directions are initialized based on airplane drag, density and earthRelVelocity. Velocity being equal to the change in position, stateDerivatives (1 through 3) are equated to earthRelVelocities (1 through 3) and force being equal to the change in momenta, stateDerivatives (4 through 6) are equated to the forces in $x, y,$ and z directions respectively.

The *trajectory.m* class declares properties *trajectoryVals,* and *time*. Functions are declared for the trajectory as a matter of prior conditions. A function is written to allow the plotting of the trajectory in 3D based upon their respective column vectors.

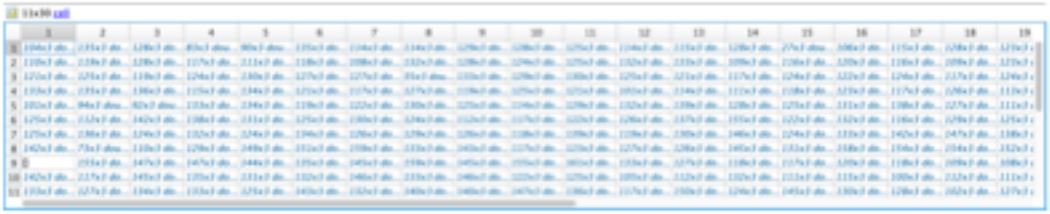
The file *BestDifference.m* is used to generally declare certain variables. In this *for* loop in which values are repeatedly solved, diameterArray lengths are inputed, landingVelocity is equated to variables such as the trajectory as a matter of a coordinate in the trajectoryVals array and divided by the airplaneMass. A difference is equated to the difference of the absolute value of the landingVelocity and targetVelocity. An *if* statement such that the absolute value of the difference must be less than the bestDifference.

In the *Simulation.m* file, parachute sets 1 through 3 (paraset1, paraset2, paraset3) are created as objects to be simulated. The imputes to the parachute object are as follows and must be inputed with the most effective variables in drag. Research from NASA's designs on space capsule landing indicate parachute sets of three with these certain variables to generate the greatest drag.

$obj = Paraset(numParachutes, parachuteMass, inflatedDiameter, dragCoefficient, tetherMass, nominalDiameter, shape, material)$

$paraset(1:3) = Paraset($3,80.7394,24.384,0.65,44.9056,29.2608, quarter spherical, kevlar)

A number of condition objects are then created to be simulated. A *while* is declared to index EPZwinds.



EPZWinds{1, 1}

	1	2	3
1	5	1.5371	0.1345
2	24	2.0550	0.1077
3	73	3.0791	-0.2153
4	93	3.5814	-0.3764
5	154	4.4921	-1.1201
6	246	6.0612	-2.8264
7	277	6.0109	-2.9317
8	582	6.3211	-4.4261
9	887	4.1310	-5.8907
10	1191	1.7309	-6.4599
11	1428	-0.5024	-7.1847
12	1496	-1.2507	-7.0928
13	1924	-6.1840	-6.1840
14	2193	-9.3829	-6.3288
15	2411	-12.5886	-5.8702
16	2715	-11.5836	-6.6878
17	3020	-9.8522	-8.2670
18	3042	-9.5577	-8.6058
19	3630	0	-11.3178
20	3691	0.7895	-11.2902
21	3935	3.5190	-9.6684
22	4039	2.7074	-8.8554
23	4538	0.3587	-4.0999
24	4544	0.3587	-4.0999
25	4663	0.4031	-4.6124
26	4849	0	-5.6589

The image above represents the array of EPZWinds, whereas, the image on the right represent a single of of the coordinates in the array in which the column represent the altitude, westerly winds and northerly winds.
The *AreaFinder and AreaFinder2* class initialize variables related to aspects of the parachutes and airplanes: Airbus A380 and Airbus A320 respectively.
Paraset conditions, location conditions and plane conditions are all represented to test for the optimal diameters (note diameters for the Airbus A380 and Airbus A320 will be different).

In addition to the ODE45 (ordinary differential equations solver which is a 4th order numerical routine, the lasses *interp1.m* and *atmos.m* were adopted and effectuated from GITHub to process components of the program. The ODE45 solver follows a notion in which $\sum F = MA$ (sum of all forces equals mass times acceleration). Furthermore, $\sum F = M\ddot{X}$ (X represents the second derivative of position which equals acceleration) and finally $-mg+\frac{1}{2}k\dot{x}^2 = M\ddot{X}$ such that the sum of all downward forces in mg and the sum of all upward forces in $\frac{1}{2}k\dot{x}^2$ (with k representing constants. The *atmos.m* class has been adopted from the 1976 Atmospheric model to represent weather conditions and *interp1.m* class interpolates to find values of the function $V = F(x)$ at query points.

The experimentation process consisted of testing for the optimal diameter for each the Airbus A380 and Airbus A320 (Note the great difference in mass may affect respective parachute sizes).
For each, parachute sizes were iterated from a range of 50 meters to 400 meters. Plot functions indicated the optimal parachute for the 0 m/s in horizontal movement and the safe target velocity of 6 to 7 m/s.

The *for* loops established here are used to determine the landing terminal velocities based on the iterated diameters for each the Airbus A320 and Airbus A380.

+11 | BestDifference.m | Simulation.m | ExamplePlots.m | AreaFin

```
% Terminal Velocity vs Parachute Diameter 380
for i = 1:length(diameterArray380)
    landingVelocity380(i) = trajectories380(i).trajectoryVals(end,6)/airplaneMass380;
end
plot(diameterArray380,landingVelocity380)

% Terminal Velocity vs Parachute Diameter 320
for i = 1:length(diameterArray320)
    landingVelocity320(i) = trajectories320(i).trajectoryVals(end,6)/A320.getMass;
end
hold on
plot(diameterArray320),(landingVelocity320)
```