FOR INTERNAL USE ONLY

BalloonDrift2.m

Purdue Orbital, Purdue University

PURDUE ORBITAL

Revision History			
Version	**Date**	**Author/Reviser**	**Notes**
Rev. 0	[Date]	{Author}	Initial Release

Position	Signature	Date
Program Director	Paul Belingon	
Chief Engineer	Dakota Hamilton	
Chief Systems Engineer	Josh Fitch	
R&D Director	Jon Webb	
Operations Director	Adithye Menon	
Avionics Lead	Colin Miller	
Ground Station/Comms Lead	Ken Sodetz	
Flight Systems Lead	Justin Heskamp	
Commercial Rocket Lead	Nicholas Masso	
Experimental Rocket Lead	Harrison Parker	
Rocket Simulation Lead	Michael Bailey	
Payload Lead	Sarah Rasche	

Launch Structure Lead	Matthew Powell	
Mission Visualization Services Lead	Collin Czech	
Author		

Table of Contents

1.0 Purpose

In order for Purdue Orbital and other regulatory officials to understand and approve future launches and projects, they must be able to anticipate the flight path and trajectory of our balloon and rocket. Due to our uncommon and innovative methods, in-house simulations are the only possible tool in predicting our flight. Readily available high altitude balloon models exist and are abundantly available through several sources[1], but are heavily focused on consumer balloons and their uses. There are primarily two extensive reasons for the failure of currently accessible balloon models: balloon type and payload mass.

The prevailing high altitude balloon specification throughout industry and the commercial sphere is constructed from latex and gradually inflates until the maximum altitude is reached. Due to the criteria of the Orbital project, latex is simply not an option, and instead we were forced to turn to more infrequent alternative: a zero-pressure polyethylene balloon. Unfortunately, all available high altitude balloon drift models additionally calculate the rate of ascent of the craft, which is entirely reliant upon which balloon type is being used (latex is presumed). The BalloonDrift2.m simulation successfully separates the altitude calculations (dependant on balloon type) and the drift calculations (independent of balloon type); therefore, allowing it to be used for *any* type of high altitude balloon.

Not only is the type of high altitude balloon a prevailing factor for the requirement of the creation of new models, but payload weight is a major issue regarding current online simulations. Due to the amateur target audience of these models, they are not able to predict flights where the payload exceeds 6 kilograms (an FAA regulation). The eventual payload of the Purdue Orbital project will far outway the 6 kilograms that is offered online. Payload weight is a vital aspect of drift simulation in several distinct ways. First, it drastically affects the modeled rate of ascent, but this issue was solved by separating the altitude and drift calculations. More vitally, online drift simulations do not account for the mass of the balloon in its movement because the weight of a normal commercial balloon and its small payload are negligible to the forces of the wind. Purdue Orbital can not make these assumptions as the payload weight will dramatically scale far beyond commercial applications. The previously mentioned separation of calculations will greatly help the payload mass issues, but Zero_model.m additionally utilize wind force calculations to factor the mass of the balloon into its movement.

Overall, a requirement for independence from online drift simulations is clear, and it is vital that BalloonDrift2.m is able to account for a heavy payload, zero-pressure, polyethylene high altitude balloon.

[1] Online drift simulation referenced throughout the document is Astra High Altitude Balloon Planner. (www.astra-planner.soton.ac.uk) because its methods and equations were published.

2.0 Inputs

BalloonDrift2.m requires seven inputs for it to properly function. They are as follows:
Alt. This is an array of altitudes (in meters) calculated at a certain time interval (*timeInterval*). These altitudes will most likely be obtained by the zero_model.m code. The output of zero_model.m is compatible and should be used as the *alt* input[2]
area. This is the impacted (not total) surface area of the balloon. It should be the approximate surface area of one side of the balloon in m^2.
mass. This is the total mass of the balloon and its payload in grams.
phi. This is the starting latitude of the balloon's flight. Use as many significant digits as possible to increase accuracy of the simulation.
theta. This is the starting longitude of the balloon's flight. Use as many significant digits as possible to increase accuracy of the simulation
timeInterval. This is the time interval between the measurements of altitudes in seconds. For example, zero_model.m has a time interval of 1 second. Astra has a time interval of 3 seconds.

3.0 Assumptions

This code assumes the standard atmosphere model as used in AAE 251 at Purdue (see Standard_Atmosphere.m)
This code also assumes that the wind speed and wind direction is the same as the sounding data recorded at the specified sounding station. This means that at all latitudinal and longitudinal locations utilize the same data from one sounding station.
This code assumes that all wind is *horizontal* when contacting the balloon. It negates all vertical aerodynamic forces.
This code negates frictional forces, wind force on the payload, drag, and rotational forces.
Also, this code assumes the conversion from miles to latitude/longitude is equal at all points on the earth

4.0 Method

BalloonDrift2.m is a code that will take in an array of altitude measurements, the starting coordinates of the launch site, and a time interval describing how often (in seconds) the altitude points in the array were calculated.. It achieves this simulation using simple calculations in a time step loop. It is referred to as Balloon Drift 2 because it is the second of 3 long term drift simulations where

[2] The original zero_model.m code does NOT output the array of alts. The zero_model.m code in the BalloonDrift2 folder DOES output the array of alts as its only output. Do not separate BalloonDrift2.m from its host folder.

each model will utilyze different wind data sources. BalloonDrift2.m obtains its wind data from the NASA's Earth Gram software. The wind data used is a vector wind speed at certain elevations, and it comes from a historical average for each month.

BalloonDrift2.m begins by taking in the inputs from the user (explained in section 2.0). Using these parameters, it initializes many variables in order to begin the looping process. The loop is now ready to run.

The loop runs from the beginning to the end of the inputted altitude array, and uses a time step given from the *timeInterval* variable. It then inputs all necessary parameters into the wind Direction function. This function connects the data gathered from Earthgram and outputs it into MATLAB as an east/west and north/south wind speed. The function initiates and inputs the given data into Earthgram, then reads the output generated on a .txt file within the fileFolder (?). The produced vector is converted into a mathematically correct cartesian vector. The code then converts the wind speed into a wind force felt on the balloon. The equation for wind load is $F_w = 1/2\, \rho\, v^2 A$. Where ρ is air density, v is wind velocity, and A is area of contact. Using Newton's second law, the force is then converted into acceleration of the craft due to the wind. The acceleration is then multiplied by the earlier time, and added to the initial velocity of the balloon. This will output a new *balloon* velocity, and when multiplied by time again, the code finds a distance vector.

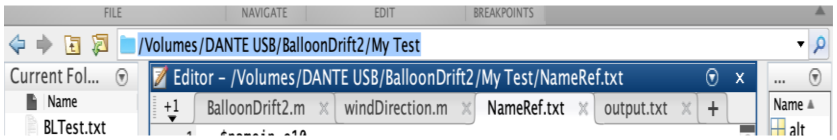
This distance vector shows how far horizontally the balloon traveled when it flew through the interval defined above. Converting this distance into latitude and longitude then adding to the previous coordinates, will allow us to find a new location for the balloon after it leaves the interval.

The last part of the code is simply to simplify the plotting process of the balloon's trajectory. It creates a .csv file with 3 columns in a specific order: height, latitude, and longitude.

HOW TO USE BalloonDrift2.m

1. Obtain payload mass, volume, balloon mass, helium mass, total mass, and impacted surface area of the balloon.
2. Determine starting location and find the coordinates (with as much significant figures as possible)
3. using zero_model.m and the parameters found in step 1, obtain an array of altitudes. Know what the time interval is between each measured altitude.
4. Open the BalloonDrift2 host folder.
    a. Open the "My Test" folder
    b. Open NameRef.txt. In lines 2, 3 and 9 you must change the location of the file from the default written. Do this by clicking the highlighted text in the picture below. Copy that and replace

the default location with it.



   c. Open Output.txt and repeat steps 4b on line 6.

5. Plug in all the required inputs in that you have gathered above into the function balloonDrift2.m and obtain the KLMFILE.xls output file
6. Go to http://www.gpsvisualizer.com/map_input?form=googleearth
   a. change the Output File Type to .kml
   b. change the altitude mode to “extruded”
   c. upload the KMLFILE.csv that is created when running the code
   d. download and open the new .kml file (must have google earth)

5.0 Outputs

There is one output to Balloon Drift.
xlmData. This is an array of data. The first column is every height where the wind data was used (NOT the same as the inputted altitude array). The second column is the corresponding latitude and the third column is the corresponding longitude. The array is set up in this very specific way so it can be read by the .klm converter on gpsvisualizer.com. This file’s only purpose is to be inputted to the online converter, as there is no use in seeing hundreds of waypoints in a .csv file form.

Appendix A: Code

```
%% Purdue Orbital
%
% BALLOON DRIFT MODEL V2.0
%
% Written by Dante Sanaei and Michael Baily
% Written: October 22, 2018

%% Description of model
% This model uses average wind data from EARTHGRAM in
order to model a high
% altitude balloon
%
%
%
% ---ASSUMPTIONS---
%   standard atmosphere as used in AAE 251 at Purdue (see
%       Standard_Atmosphere.m)
%
%


%% Function
function [klmData] = BalloonDrift2(alt, area,
mass,phi,theta, timeInterval, month)
% inputs :
% alt:                  array of altitudes (m)
% soundingFile:         .txt file of wyoming sounding data.
(remeber to
%                       delete any columns or rows that
have empty spots)
% area:                 Surface area of balloon (m^2) 2
% mass:                 Total mass of balloon(g) 3500
% startLat:             Latitude of launch site (purdue:
40.41279385)
% startLon:             Longitude of launch site (Purdue: -
86.93937395889625)
% timeInterval:         the interval of time between each
altitude
%                       measurement(Matt Powell: 1, Astra:
3)

```

```
%% Initializations
clc
clear
alt = Zero_model(68.0389,2831.6847,27.2155,0.25,15.5,1);

alt = alt(1:20000) + 1191;
alt = alt(1:100:length(alt))
phi = 40.8500;
theta = -119.1238;
month = 3;
area = 10;
mass = 68.0389 * 1000;
timeInterval = 100;
day = 1;
year = 2018;
hour = 12;
minute = 0;
second = 0;
fileFolder = 'My Test';
latlon = [phi, theta]; %Starting coordniates (Purdue
Airport)
latlon1 = latlon;
heights = 0;
initV = [0 0];
totalD = 0;
dx =0;
dy = 0;
%% Time Step
for i = 1:length(alt)
    i
    time = timeInterval;
    height =  alt(i);
    [east_west,north_south] =
windDirection(height,phi,theta,month,day,year,hour,minute
,second,fileFolder);
    second = second + 1;
    if second == 60
        second = 0;
        minute = minute +1;
        if minute == 60
            minute = 0;
            hour = hour +1;
            if hour == 24
                hour = 0;
                day = day+1;
            end
        end
    end
    Xvelocity = east_west;                    % finding x component of wind velocity (m/s)
    yvelocity = north_south;                    % finding y component of wind velocity (m/s)
    windV = [Xvelocity, yvelocity];                % wind velocity vector (m/s)
    [GeopotentialAlt, DesiredPressure, rho, DesiredTemperature] = Standard_Atmosphere(height, 0);       %finds pressure using 1976 Standard Atmosphere
    vel = ((((.5)*(rho ).*([Xvelocity, yvelocity].^2).*area)/mass).* time + initV); %finds balloon velocity vector using wind load.
    initV = windV;
  distance = vel .* time;                              %finds distance (m)
    %distance = [Xvelocity, yvelocity] .* time  ;
   %if the previous method does not work comment it out and use this line only
    theta = theta + (distance(1) * 0.000621371) / 55.2428;             %convert x distance (m) to longitutde
    phi = phi + (distance(2) * 0.000621371) / 68.703 ;      %convert y distance (m) to latitude
    latlon = [phi theta];
    latlon1 = [latlon1; latlon];
    height = height;
    heights = [heights; height];
    totalD = totalD + distance;
end


%% Exporting Data to .CSV

klmData = [heights latlon1];
%dlmwrite('bestone.csv', xml, 'delimiter', ',', 'precision', 12,'roffset',1);
```

```
cHeader = {'Altitude' 'Latitude' 'Longitude'}; %dummy header
commaHeader = [cHeader;repmat({','},1,numel(cHeader))]; %insert commaas
commaHeader = commaHeader(:)';
textHeader = cell2mat(commaHeader); %cHeader in text with commas
%write header to file
fid = fopen('KLMFILE.csv','w');
fprintf(fid,'%s\n',textHeader);
fclose(fid);
%write data to end of file
dlmwrite('KLMFILE.csv', klmData, 'delimiter', ',', 'precision', 12,'-append');
latlon1;
totalD / 1000;
heights;


```