# THIS BLOWS?: EVALUATION OF ADDITIVE MANUFACTURED FIVE HOLE PROBES FOR UNMANNED SYSTEM BASED WIND MEASUREMENTS

By

### KYLE HICKMAN

Bachelor of Science in Mechanical Engineering Oklahoma State University Stillwater, Oklahoma 2019

Bachelor of Science in Aerospace Engineering Oklahoma State University Stillwater, Oklahoma 2019

> Submitted to the Faculty of the Graduate College of the Oklahoma State University in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE July, 2021

# THIS BLOWS?: EVALUATION OF ADDITIVE MANUFACTURED FIVE HOLE PROBES FOR UNMANNED SYSTEM BASED WIND MEASUREMENTS

Thesis Approved:

Dr. Jamey D. Jacob

Thesis Advisor

Dr. He Bai

Dr. Imraan Faruque

#### ACKNOWLEDGMENTS

There are many people to acknowledge, I'm sure I will miss a few. To start I want to thank my friends and family for always being supportive, not sure how I lucked into having such quality people around me. My advisor Dr. Jacob for his time and guidance. I appreciated the freedom to explore and make mistakes. Not that I always enjoyed the mistakes part. Literally crashing my thesis into the ground was not so fun. I also want to thank the rest of my committee Drs. Faruque and Bai. I have been able to learn much from both. All my professors at OSU have taught me much. The lessons I have taken from them have been as valuable for my thesis as I'm sure they will be down the road. I also would like to thank the entire staff at USRI. They have always been eager to help answer questions and teach what they know. The opportunities and knowledge I have gained from my experience there have been tremendous. I especially want to thank Levi Ross. His knowledge and willingness to help saved me an untold number of headaches and hours. From MATLAB to sensor DAQs I have learned much from him. Finally, my main research partner for the past two years James Brenner. He has done much on this project from designing and printing hardware to helping running calibrations on the probes. Being able to work with him has been invaluable and made the hours spent in a the wind tunnel a bit better.

This work is supported in part by the National Science Foundation under Grant No. 1539070, Collaboration Leading Operational UAS Development for Meteorology and Atmospheric Physics

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

(CLOUD-MAP), and Grant No. 1925147, NRI: Safe Wind-Aware Navigation for Collaborative Autonomous Aircraft in Low Altitude Airspace, and by NASA under the University Leadership Initiative. Additional support provided by the OSU Unmanned Systems Research Institute. I appreciate the assistance of many USRI staff and students.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: KYLE HICKMAN

Date of Degree: JULY, 2021

# Title of Study: THIS BLOWS?: EVALUATION OF ADDITIVE MANUFACTURED FIVE HOLE PROBES FOR UNMANNED SYSTEM BASED WIND MEASUREMENTS

# Major Field: MECHANICAL AND AEROSPACE ENGINEERING

Abstract: A deep understanding of the wind in the atmosphere is becoming valuable as unmanned aerial systems (UAS) are integrated into the National Airspace System. Unmanned vehicles are also becoming a key tool in weather observation and research. One of the most effective ways to measure wind from a fixed wing aircraft is a multi-hole probe (MHP). Commercial probes can be prohibitively expensive for wide scale use. Custom probes machined from metal can require extensive manufacture time and expensive tools for machining. Additive manufactured probes have been proven to be accurate in a wind tunnel and offer a good option for a cheap and easy to manufacture MHP. Limited flight testing and evaluation of the probes in flight has been done. This thesis investigates the use of a fully additive manufactured probe for UAS based wind measurements. A variety of different evaluation techniques from literature are used and compared. The paper covers the design, calibration, and preliminary validation of the probes. The probe is demonstrated for use in a meteorological application where the probes are used with a full meteorological sensor suite to produce atmospheric boundary layer(ABL) observations.

# TABLE OF CONTENTS

Chapter			Page
I.	INTRO	DUCTION	1
1.1	Motiv	vation	1
1.2	Background		
1.3	Goals and Objectives		
1.4	Thesis Outline		
II.	LITER	ATURE REVIEW	5
2.1	Wind	l Measurement Instruments	5
	2.1.1	Cup Anemometer	5
	2.1.2	Propeller Anemometer	6
	2.1.3	Hot Wire Anemometer	7
	2.1.4	Ultrasonic Anemometers	8
	2.1.5	Pitot Tube	9
	2.1.6	5 Hole Probes	10
2.2	Syste	ems for Atmospheric Observation	12
	2.2.1	Weather Towers	12
	2.2.2	Weather Balloons	13
	2.2.3	Manned Aircraft	16
	2.2.4	Unmanned Aircraft Weather and Wind Measurements	17
2.3	Algoi	rithms For Fixed Wing Wind Estimation	18
	2.3.1	Instrument Free Wind Estimation	19

Chapte	er		Page
	2.3.2	Multi-Hole Probe Based wind Estimation	. 20
	2.3.3	Unmanned Aircraft Atmospheric Measurements	. 23
III.	FIVE	HOLE PROBE DESIGN AND GROUND CALIBRATION	N
	AND 7	ΓΕSTING	. 28
3.1	Five	Hole Probe Development	. 28
	3.1.1	Development - Hybrid Probe	. 33
	3.1.2	Quantization Error	. 47
3.2	Ultra	asonic Anemometers	. 49
IV.	INSTF	RUMENT INTEGRATION AND FLIGHT TESTING ANI	D
	CALIE	BRATION	. 53
	4.0.1	Data Acquisition System Architecture	. 53
	4.0.2	Aircraft Description	. 55
	4.0.3	Sensor Mounting	. 56
	4.0.4	3D Print Durability Issues	. 59
	4.0.5	Wind Vector Realization	. 60
4.1	SKB	3 1000 - Trisonica Calibration	. 69
V.	RESU	LTS	. 71
5.1	Valie	lation and Error Analysis	. 71
	5.1.1	Noise Analysis	. 71
	5.1.2	Gaussian Error Propagation	. 72
	5.1.3	Kolmogorov's -5/3 Law	. 74
	5.1.4	Mesonet Station	. 76
	5.1.5	Comparison Flights	. 79
	5.1.6	Transect Flights	. 90

Chapter		Р	age
5.2	Use For a Holistic Meteorological Application		93
VI. C	ONCLUSION		96
6.1	Five Hole Probe Comparison to other sensors		96
6.2	Probe Challenges		97
6.3	Future Work		98
6.4	Conclusion		99
REFERI	ENCES		101
APPEN	DICES		106

# LIST OF TABLES

Table	P	age
1	Desired Sensor Specifications for Meteorological Observation $[23]$	4
$\frac{2}{3}$	University of Kentucky LAPSE-RATE Correction Factors [11] UAS MHP System Accuracies	26 26
4 5	Anemometer Table Part 1.Anemommeter Table Part 2.	50 50
6	Middle 24in Calm Day	70
$7\\8\\9\\10\\11\\12\\13\\14\\15\\16$	Nano Talon Error Propagation	73 74 79 82 83 83 84 84 85 85
17 18 19 20 21 22 23	Flight 2 Sensor Data Compare5-28 Trisonic MHP Wind Square Data Analysis5-28-21 Trisonica MHP Wind Square Comparison5-28-21 Trisonica MHP Wind Square Steady Level Analysis5-28-21 Trisonica MHP Wind Square Steady Level Sensor Compare5-28-21 Trisonica MHP Wind Square Steady Level Sensor CompareChoctaw Transects Steady Level Upwind AnalysisChoctaw Transects Steady Level Upwind Sensor Compare	<ul> <li>86</li> <li>88</li> <li>89</li> <li>90</li> <li>90</li> <li>92</li> <li>92</li> </ul>
$\begin{array}{c} 24 \\ 25 \end{array}$	System Comparison       Preliminary System Characteristics	96 99

# LIST OF FIGURES

Figure		Page
1	Diurnal Cycle of the ABL with Proposed UAS Operations $[23]$	1
2	NASA Safe Autonomous Flight Environment SAFE50 Navigational Haz-	
	ards in Urban Environments $[2]$	2
3	Cup Anemometer	5
4	Propeller Anemometer	6
5	Hot Wire Anemometer	7
6	Young 81000 3D Ultrasonic Anemometer	8
7	Pitot Probe Layout	9
8	Pitot Probe Layout	10
9	5HP Tip Geometry Types[19]	11
10	Oklahoma Mesonet Tower Schematic <sup>[29]</sup>	12
11	Weather Balloon Launch[28]	13
12	Skew-T Weather Sounding [10]	14
13	Skew-T Weather Sounding with Different Derived Parameters $[10]$	15
14	NCAR C130Q Hercules Equipped for Horizontal Wind Measurements[27]	16
15	Early RC Atmospheric Flights [24]	17
16	Aircraft Frames Visualization	18
17	Visualization of algorithm wind with derived wind triangle[28]	19
18	Comparison of LAPSE-RATE Wind Measurements $[15]$	24
19	Probe Dimensions: A. Side View, B. Back View, C. Front View, D. Bottom	
	View, E. Isometric View	30
20	Probe in Mid Print on Formlabs Printer	31
21	Probe Tubes Being Cleared with Alcohol	32
22	Metal Probe	33
23	Metal Probe Tip	34
24	Wind Tunnel Arrangement	36
25	Digital Sensor Package	36
26	MHP Mounted in Wind Tunnel	36
27	Calibration Sweep Orientations	37
28	Notional pressure coefficient curves for pitch and yaw depicting a 5HP	
	traversing on the pitching axis while the yaw axis nulled $\ldots$	39
29	Illustration depicting pressure coefficient linear range $\ldots \ldots \ldots \ldots$	39

Figure

# Page

30	Orientation of Vector pitch and yaw angle	40
31	Ideal Velocity Magnitude Curve	41
32	Probe2 (a) $C_p$ Alpha Vs. Angle and (b) $C_p$ Beta Vs Angle	42
33	Probe3 (a) $C_p$ Alpha Vs. Angle and (b) $C_p$ Beta Vs Angle $\ldots \ldots \ldots$	43
34	Magnitude Pitot Velocity Vs Angle	43
35	$C_{p_pitot}$ Change with Velocity and Angle	44
36	Probe3 Interpolated Comparison to Actual Angle	45
37	Percent Error of Interpolated Angles	46
38	Velocity Comparison for 10m/s Alpha Sweep	46
39	Percent Error of Interpolated Angles	47
40	Pitot Step Response	48
41	Alpha Step Response	49
42	Anemomet Trisonica Mini	51
43	R.M. Young 92000 on Mount for Tower Operations	51
44	R.M. Young 5103	52
. ~		- 1
45	Wire Diagram of Logger System	54
46	Integrated Vehicles	55
47	DAQ in Nano Talon	57
48	Nano Talon Probe Mount	57
49	Nimbus Probe Mount	58
50	Probe Wear	59
51	Broken Probe Pitot Attachment	60
52	Orbit Flight Path From $6/2/2021$	61
53	Raw Velocity MHP Data From $6/2/2021$ (Orbit)	61
54	NED MHP Data From $6/2/2021$	62
55	NED MHP Data From $6/2/2021$ with Angular Rates Removed	63
56	Single Sided Amplitude Spectral Analysis Data From 6/2/2021 (Orbits)	64
57	Averaged NED MHP Data From 6/2/2021	64
58	Wind Square Flight Path 6/2/2021	65
59	Raw Velocity MHP Data From 6/2/2021 (Squares)	66
60	Single Sided Amplitude Spectral Analysis Data From 6/2/2021 (Squares)	66
61	NED MHP Data From 6/2/2021(Square)	67
62	Averaged NED MHP Data From 6/2/2021(Square)	67
63	Wind speed and Direction Graphs	68
64	SKB-1000 with Trisonica Mount	69
65	Spectral Power Density with -5/3 Law (Flight Data)	75
66 66	Spectral Power Density with 5/3 Law (Pright Data)	76
67	Wind Speed Comparison to Marshall Mesonet Site 4/2/2021	10 77
69	Wind Speed Comparison to Marshall Mesonet Site $4/2/2021$	11 70
00 60	Wind Direction Comparison to Marshall Mesonet Site $4/2/2021$	10 70
09	which Direction Comparison to Marshall Mesonet Site $4/2/2021$	18
70	1500 Surface Map	80

Figure

# Page

71	1800 Surface Map	80
72	Waypoint Parameters for Nimbus Flight $5/3/2021$	80
73	Flight Profile For Nimbus and Skateboard $5/3/2021$	81
74	Young, Trisonica, and MHP Wind Speed Plot $5/3/2021$ - Flight 1	82
75	Trisonica, and MHP Wind Angle Plot $5/3/2021$ -Flight 1	83
76	Young, Trisonica, and MHP Wind Speed Plot $5/3/2021$ -Flight 2	84
77	Trisonica, and MHP Wind Direction Plot $5/3/2021$ -Flight 2	85
78	Way-points For Wind Square Calibration Flights Revision For 5-28-2021	86
79	Wind Square Flight Path 5-28-2021	87
80	Wind Speed Comparisons 5-28-2021	88
81	Straight Section Wind Speed Comparisons 5-28-2021	89
82	Choctaw Flight Path	91
83	Choctaw Wind Plots	92
84	Wind Speed Sounding	94
85	Sounding	95
86	ABL Wind Sounding	100

# CHAPTER I

### INTRODUCTION

#### 1.1 Motivation

The use of unmanned aerial systems as meteorological tools has been developing rapidly over the years. Unmanned systems have demonstrated the ability to support meteorological observation and research efforts. These efforts have promise to be able to supplement other atmospheric observation methods such as balloons and mesonets.



Figure 1: Diurnal Cycle of the ABL with Proposed UAS Operations[23]

Figure 1 shows the Atmospheric Boundary Layer (ABL) over the diurnal cycle. The diurnal cycle is how the boundary layer transition in a 24 hour night/day cycle. Unmanned systems have enabled the study of this cycle with increased detail[15]. This would lead to better data

for meteorologist enabling better forecasting and more accurate data reporting.



Figure 2: NASA Safe Autonomous Flight Environment SAFE50 Navigational Hazards in Urban Environments [2]

Beyond meteorological research for the integration of UAS into our cities and urban areas wind research is needed[2]. Figure 2 are challenges NASA has identified as barriers to UAS integration into urban areas. One of these areas is near real time robust wind information. Multi-hole probes are not the full solution to this issue. However, development of a cost effective, mass producible probe could enable the data needed to support simulation and modeling that will help solve the problem.

#### 1.2 Background

Multi-hole probes have been used for aerial based atmospheric measurements since the late 1960s[13]. They have been used for everything from measurement of wind on a ship, characterization of flows through turbo machinery, and measurement of wind from aircraft. The use of MHPs for atmospheric research from fixed wings has been explored for a long time. The probes used on many of these existing systems are incredibly expensive or difficult to manufacture. A cheap easy to manufacture probe

## 1.3 Goals and Objectives

The over arching goal of this thesis will be to continue the design, evaluation, and integration of an additive manufactured multi hole probe. With a final objective of demonstrating the use of the probe in a meteorological application. The specific objective are as follows:

- Design 5 hole probe for UAS integration
- Calibrate 5 hole probe evaluate baseline errors
- Integrate a five hole probe to unmanned aircraft
- Implement algorithm to address biases and isolate wind vector from probe measurements
- Calibrate and validate 5 hole probe system by comparison with weather towers and other UAS wind systems
- Compare accuracy and errors to other other wind sensing UAS systems
- Demonstrate the use of the 5 hole probe system in a holistic meteorological application

The design of the probe builds upon previous work done on 3D manufactured probes[14]. This worked demonstrated the accuracy and use of additive manufactured in a wind tunnel setting. With the objective of using the probe in a meteorological application meteorological the data quality desired for meteorological observation should be discussed.

Table 1 are the desired accuracies as reported through the CLOUD-MAP project. These are not realistic objectives for this project. This will become apparent as the accuracies of other systems are explored in the next chapter.

Meteorological Variables and Accuracy		Sensor Response Time	
Temperature	$\pm 0.2$ °C	Time	<5s ( $<1s$ Preferred)
Relative Humidity	$\pm 5.0\%$	<b>Operational Envir</b>	conmental Conditions
Pressure	$\pm 1.0 hPa$	Temperature	-30 to 40 $^{\circ}\mathrm{C}$
Wind Speed	$\pm 0.5 \mathrm{m/s}$	Relative Humidity	0 to $100\%$
Wind Direction	$\pm 5$ Degrees Azimuth	Wind Speed	0 to 45 m/s $$

Table 1: Desired Sensor Specifications for Meteorological Observation [23]

# 1.4 Thesis Outline

The discussion of the probe will begin with a review of previous work on multi-hole probes and other systems and sensors that are used for wind measurements. The wind tunnel calibration of the probe will then be discussed in Chapter 3. All ground based instruments will be included here as well. Chapter 4 will involve the basics of wind vector realization from a multi-hole probe. Different flight methods and data processing methods are described. An evaluation of an anemometer equipped quad used for calibration is also found here. The results of comparison flights are found in Chapter 6. The final section will include the conclusion and discussion of results leaving off with future work.

# CHAPTER II

# LITERATURE REVIEW

# 2.1 Wind Measurement Instruments

Modern meteorologist have a range of instruments to measure wind speed from the flow. These systems have a wide range of size, accuracy, cost, etc. While many novel wind sensors exist this section of the literature review will be systems that are often mounted to unmanned systems or are relevant validation of wind sensors in presented in this thesis.

# 2.1.1 Cup Anemometer



Figure 3: Cup Anemometer

Cup anemometers, as pictured above, are one of the most simple and widely used anemometers for basic wind speed. Cup anemometers are widely used because of three main benefits. First, as long as the cups are horizontal the anemometer is omni-directional. Second, modern cup anemometers have linear calibration for wind speed and direction. Cup anemometers are also very robust and easy to operate. The cup anemometer does have a variety of disadvantages that make reduce its effectiveness for some applications. Cup anemometers suffer from a slow step response since it takes time for the wind to impart momentum on the cups. This is more sever for decreases than increases in wind speed. This leads to "Over speeding" or the anemometer measuring something higher than the actual averaged wind speed [25].

#### 2.1.2 Propeller Anemometer



Figure 4: Propeller Anemometer

Propeller anemometers are typically mounted on a wind vane for atmospheric measurements. Although slightly more complex mechanically than the cup anemometer it is still a simple system. The propeller vane anemometer is able to measure the wind angle more simply than a cup anemometer. Due to the interaction of the vane and the propeller the system has a second order response and calibration. The propeller vane does have a few systematic errors like the cup anemometer. It has a similar issue with a slow step response to wind speed, high response systems can minimize this. A direction lag is inherent due to the vane always lagging the wind. The rotation of the vane in the absence of wind can also cause false propeller speed readings[26].

#### 2.1.3 Hot Wire Anemometer



Figure 5: Hot Wire Anemometer

Hot wire anemometers have two different methods of measuring flow speed. Constant temperature hot wire anemometer and constant current hot wire anemometer. Both types of hot wire work by relating the heat transfer from a wire to the velocity of a flow field. In the case of a constant temperature the relation is the current required to pass through a wire to maintain temperature for a flow velocity. The constant current hot wire does the inverse, keeping the current constant and relating the temperature change to flow velocity. Hot wires if calibrated properly are able to measure difficult to observe phenomena such as turbulence. This is because hot wires are very sensitive to flow changes. This high sensitivity makes measurements from hot wires very susceptible to error from mounting and alignment with the flow[18].

#### 2.1.4 Ultrasonic Anemometers



Figure 6: Young 81000 3D Ultrasonic Anemometer

Ultrasonic anemometers come in both 2D and 3D versions. The anemometer operate using an inverse time transit difference[12]. This is done by generating high frequency acoustic pulses. These pulses are then measured upstream and downstream of the source. The baseline equation for this is

$$\frac{1}{t_{12}} - \frac{1}{t_{21}} = \frac{(c_0 + v\cos\theta)}{L} - \frac{c_0 - v\cos\theta}{L}$$
(2.1.1)

where  $c_0$  is the speed of sound in air,  $t_1^2$  and  $t_2^1$  are the times it take to traverse between sensors, v is the wind speed, L is the distance between sensors. Ultrasonic anemometers are beneficial because they have no moving parts and can be compact in form. However, they are one of the more expensive anemometer types.

#### 2.1.5 Pitot Tube



Figure 7: Pitot Probe Layout

Pitot tube anemometers calculate airspeed based on the difference between stagnation pressure  $(P_{total})$  and static pressures  $(P_{static})$ . Figure 7 is an illustration of how a pitot probe works. The relationship between these pressures can be equated to airspeed using Bernoulli's equation:

$$p_{static} = p_{total} + \frac{\rho u_{\infty}^2}{2} + \rho gh \qquad (2.1.2)$$

Equation 2.1.2 is the simplified version of the Bernoulli equation. The  $\rho gh$  term in the Bernoulli is a potential pressure term, where  $\rho$  is density, g is gravity, and h is a vertical height. Since the pitot probe does not have a large vertical distance between measurements this term can be ignored.  $u_{\infty}$  is the free stream airflow speed. Rearranging Equation 2.1.2 to isolate the free stream velocity and removing the potential term results in the following:

$$u_{\infty} = \sqrt{\frac{2\Delta p}{\rho}} \tag{2.1.3}$$

Equation 2.1.3 is used to calculate velocity from a pitot probe. The  $\Delta p$  term is the difference

between the stagnation pressure and the total pressure  $(p_{static} - p_{total})$ . These calculation carry over for multi-hole probes where air speed calculations follow Bernoulli equation.

#### 2.1.6 5 Hole Probes

The 5 hole probe (5HP) was developed by Admiral Taylor in 1915 for the measurement of 3D wind from ships[37]. Multi-hole probes are similar to pitot probes in that they use a static and total pressure to calculate the speed of the flow. Multi-hole probe are able to combine those measurements with extra ports in perpendicular planes to the static port. For a 5 hole probe the layout of ports is shown in Figure 8.



Figure 8: Pitot Probe Layout

The tip geometries of the probe largely determines its operating characteristics. The most common tip geometries used are hemispherical, conical, and pyramid, as shown in Figure 2 from [19]. Numerous studies have demonstrated that differences in conical and pyramid tips is primarily based on performance of how flow separation behaves around the probe. One study shows that with smooth surfaces (see hemispherical or conical) separation occurs more slowly [36]. This is generally a good trait to have for MHPs, but this also comes at a cost of being sensitive to changes in Reynolds Number due to abrupt changes in the free-streamvelocity. From [35] most MHPs usually have an angular operating range of -25 degrees to 25 degrees. From previous literature studies, for the flow regions that a small UAV will be experiencing a hemispherical tip head has been shown to be the best geometry [39].



Figure 9: 5HP Tip Geometry Types[19]

Due to imperfections in manufacturing multi-hole probe must be extensively calibrated. There are two methods used to calibrate multi-hole probes nulling and non-nulling mode[37]. In a nulling calibration method, a probe is inserted into a flow and rotated until the error signal across two opposing ports (e.g alpha low and alpha high) is nulled[31]. Then, by taking the inclination angle at that position, it is related to the flow direction, and by measuring the pressure at that position, the velocity is obtained. This method is highly accurate, but it is a very involved process that requires large amounts of space for traversing and very long data acquisition sessions. The non-nulling method calibration, while tedious, is easier to perform in a standard wind-tunnel test section. This method involves having the probe fixed in the center of the wind-tunnel and slowly rotated from one of its angular operating limits to the other, while each pressure reading of the probe is measured and correlated to the angular position of the probe. This data is then compared to a Pitot-static probe upstream in the wind-tunnel, which measures the free-stream tunnel velocity. This method was first introduced by Treastor and Yocum[37].

### 2.2 Systems for Atmospheric Observation

The following section will cover different systems used for atmospheric observations. Rather than just wind these systems take a multitude of atmospheric measurements including wind.

#### 2.2.1 Weather Towers

Weather towers are a popular method for continuous weather monitoring. The Oklahoma Mesonet is a system of 121 weather towers scattered across Oklahoma, at least one for each county. These towers are outfitted a wide range of sensors.



Figure 10: Oklahoma Mesonet Tower Schematic<sup>[29]</sup>

The schematic above shows the general layout of an Oklahoma Mesonet station. The Oklahoma Mesonet reports data every five minutes. For the 10m wind speed a R.M. Young 5103 anemometer is equipped at 2m a R.M. Young 3101 is equipped. Along with wind speed and direction the Mesonet sites take temperature, pressure, humidity, rainfall, solar radiation,

and a multitude of soil parameters[29].

# 2.2.2 Weather Balloons



Figure 11: Weather Balloon Launch[28]

Weather balloons have been used since the 1930s for weather observations[38]. Weather balloons are filled with hydrogen and can take data up to 115,000ft. Radiosondes are attached to the weather balloons for pressure, temperature, and humidity readings. Rawinsondes are GPS enabled or tracked by radar, this allows for the wind speed to be estimated from the track of the balloons[8]. One issue with weather balloons is cost since many radiosondes are not recovered. Balloons are used to create soundings.



Figure 12: Skew-T Weather Sounding [10]

Figure 12 is an example of how the data from a weather balloon is presented. The figure is referred to as an atmospheric sounding. From the sounding important atmospheric measures can be derived.



Figure 13: Skew-T Weather Sounding with Different Derived Parameters [10]

Figure 13 shows important parameters that can be derived from the Skew-T sounding. Theses parameters are used in forecasting. Going from bottom to top on the figure with CIN or the Convective Inhibition shows how stable the atmosphere is. The Level of Free Convection (LFC) this is a boundary where above particles will begin to accelerate upwards. The Lifted Index the temperature difference between the parcel and environmental temperature at 500mb. The Convective Available Potential Energy (CAPE) the positive area between the red line and the dotted blue line (parcel and environmental temperature) this indicates the instability in the atmosphere, correlating to likelihood and severity of storms. The Equilibrium Level (EL) is the pressure value at the top of the CAPE. The Lifted Condensation Level (LCL) is near where the CIN is labeled on this figure, it signifies where clouds will form (the peak of the blue triangle on the x axis at 20).

# 2.2.3 Manned Aircraft

The implementation of Inertial navigation systems (INS) on aircraft allowed for accounting for the motion of aircraft in atmospheric measurements. This implementation began being taken advantage of in the late 1960s. One of the first major implementation was sounding flight flown with a Canberra PR 3 aircraft done by Axford in 1968[13]. The aircraft was outfit with a multi-hole probe and gimbal based INS system. The INS system is the key to deriving wind component from aircraft based multi hole probe measurements. The INS system allows for the removal of aircraft dynamics from the measurement as will be described in the Multihole probe algorithm section. While the flight campaigns were successful the lack of a data acquisition system made the processing of the data prohibitively time consuming.



Figure 14: NCAR C130Q Hercules Equipped for Horizontal Wind Measurements[27]

A flight campaign performed by NCAR has been able to use the methods developed by Axford[13] with modern INS and data acquisition. The program uses C130Q Hercules with a 5 hole probe system as pictured in Figure 14. The studies done with this aircraft establish calibration and error investigation procedures. The limitations found of the system are a frequency response of 10Hz and a short term wind accuracy of  $\pm 1 \text{m/s}[27]$ 

# 2.2.4 Unmanned Aircraft Weather and Wind Measurements

The use of Unmanned aircraft for weather observation was first explored in 1970 by Konrad et al [24]. The flight tests conducted demonstrated the feasibility of using unmanned systems for atmospheric research. By today's standards the system used was rudimentary having no autopilot and limited endurance.





(a) Atmospheric RC Aircraft in Flight(b) Aircraft Spotting SystemFigure 15: Early RC Atmospheric Flights [24]

The aircraft used in the Konrad study had sensors to measure temperature, pressure, and, humidity. The aircraft was also equipped with a pitot probe and thermistor mass flow sensor to track the aircraft's airspeed. Wind speed was not an objective of the flight campaign.

#### 2.3 Algorithms For Fixed Wing Wind Estimation

One of the major challenges of getting accurate wind measurements from unmanned aircraft is removing the aircraft state from the raw measurements. For discussion of these algorithms a definition of aircraft frames is required.



Figure 16: Aircraft Frames Visualization

Figure 16 is a visualization of the different frames of reference that will be needed for discussion in this paper. In green,  $x_g$  and  $y_g$  are ground or Earth based coordinates. These relate to North and East on a map respectively and are independent of the aircraft. In blue,  $x_b$  and  $y_b$  are aircraft or body based coordinates. For these coordinates  $x_b$  relates to the heading of the aircraft and is positive out of the nose,  $y_b$  is positive out of the starboard(right) wing these rotate with the aircraft as it maneuvers.  $\psi$  is the angle that  $x_b$  is offset from North. In orange,  $x_a$  and  $y_a$  are air or wind frame coordinates.  $x_a$  points in the direction that the airflow over the aircraft is coming from. This airflow is a function of the wind and the aircraft ground speed.  $\beta$  is the angle offset of the airflow from  $x_b$ . For all discussion of coordinate frames in this paper the subscript g will refer to earth frame, subscript b will refer to body frame, and subscript a will refer to the wind frame.

### 2.3.1 Instrument Free Wind Estimation

One method of instrument free wind estimation is developed by Mayer et al.[28]. The algorithm uses the assumption that an aircraft with a constant throttle and pitch should maintain a constant airspeed.



Figure 17: Visualization of algorithm wind with derived wind triangle [28]

Figure 17 illustrates the wind triangle the algorithm is based around. In the wind triangle GS is the ground speed of the aircraft, TAS is the true airspeed of the aircraft, and u is a wind vector. For an aircraft without an airspeed sensor only ground speed is know from GPS data. Based off the wind triangle:

$$TAS + u = GS \tag{2.3.1}$$

True air speed (TAS) is calculated as

$$\overline{TAS} = \frac{1}{N} \sum_{i}^{N} \|GS_i - u\|$$
(2.3.2)

With TAS and GS variance is calculated

$$\sigma_{TAS}^2 = \frac{1}{N} \sum_{i}^{N} (\|GS_i - u\| - \overline{TAS})^2$$
(2.3.3)

Using this algorithm the variance is minimized to find the wind speed(u).

This method does have significant drawbacks. One of the major issues is that the algorithm breaks if the aircraft does not maintain the assumed constant throttle and pitch[28]. While the algorithm can be used for any constant flight profile (orbits, racetracks, etc.) it does require two full cycles to average over to be accurate. The method also contains large uncertainties[33].

#### 2.3.2 Multi-Hole Probe Based wind Estimation

Multi hole probes allow for algorithms to be simplified and reduced to mostly frame rotations. This is because the  $\beta$  and  $\alpha$  angles can be known from the 5HP data reducing the need for extra calculations to estimate  $\beta$  or  $\alpha$ . A few different multi hole probe algorithms have been used. The following is a reduced version of the base algorithm used in a variety of multi hole probe wind studies[13][27][33]. With many other study using a augmented versions of the algorithm[30][11]. Referring back to the reference frames established in Figure 16 the wind vector in ground frame can be described as

$$\overrightarrow{w}_g = \overrightarrow{v}_g + \mathbf{T}_{gb} \mathbf{T}_{ba} \overrightarrow{u}_a \tag{2.3.4}$$

In Equation 2.3.4  $\overrightarrow{v}_g$  is the ground speed of the aircraft.  $\mathbf{T}_{gb}$  and  $\mathbf{T}_{ba}$  are rotation matrices operating on  $\overrightarrow{u}_a$  the total airspeed of the vehicle. For this equation  $\overrightarrow{u}_a$  is being rotated from wind to body ( $\mathbf{T}_{ba}$ ) then from body to earth ( $\mathbf{T}_{gb}$ ). This allows the merger with the aircraft ground speed and results in a wind speed from the ground reference which is desired for meteorological wind analysis.

The algorithm calculates true airspeed  $\overrightarrow{u}_a$  from the norm,  $|\overrightarrow{u}_a|$ . Where,

$$|\vec{u}_{a}|^{2} = 2c_{p}T_{tot}[1 - (\frac{p}{p+q})^{\kappa}]$$
(2.3.5)

 $\kappa$ , the poisons ratio, is calculated as  $\kappa = Rc_p^{-1}$  where for standard dry air  $R = 287 J k g^{-1} K^{-1}$ and  $c_p = 1004 J k g^{-1} K^{-1}$ . From this point the  $u_a$  is rotate from wind frame to body frame using  $\mathbf{T}_{ba}$  based on *alpha* and *beta* as follows.

$$\overrightarrow{u}_{b} = -\frac{|\overrightarrow{u}_{a}|}{\sqrt{1 + \tan^{2}\alpha + \tan^{2}\beta}} \begin{pmatrix} 1\\ \tan\beta\\ \tan\beta \\ \tan\alpha \end{pmatrix}$$
(2.3.6)

Once rotated into body frame next is earth frame.  $\mathbf{T}_{gb}$  is done with three turns about each of the aircraft's axis  $T_1(\phi)$ ,  $\mathbf{T}_2(\theta)$  and  $\mathbf{T}_3(\psi)$  roll pitch and yaw respectively. So,

$$\mathbf{T}_{gb} = \mathbf{T}_{1}(\psi)\mathbf{T}_{2}(\theta)\mathbf{T}_{3}(\phi)$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.3.7)$$

The equations can then be combined to form the following where  $D = \sqrt{1 + tan^2\beta + tan^2\alpha}$ . For the sake of space in the following equation c is cosine, s is sine and , t is tangent.

$$\vec{w}_{g} = \vec{v}_{g} - \frac{|\vec{u}_{a}|}{D} \left( c\psi c\theta + t\alpha (s\phi s\psi + c\phi c\psi s\theta) + t\beta (c\psi s\phi s\theta - c\phi c\psi) \\ c\theta s\psi + t\alpha (c\phi s\psi s\theta - c\psi s\phi) + t\beta (c\phi c\psi + s\phi s\psi s\theta) \\ -s\theta + c\phi c\theta t\alpha + c\theta s\psi t\beta \right)$$
(2.3.8)

Equation 2.3.8 could be considered the baseline 5 hole probe algorithm. The complete version was originally developed for large manned aircraft by Axford[13]. The major difference between the presentation of the equation by Rautenberg for UAS and the full form is the removal of the angular velocities term. The addition of the angular rates term to Equation 2.3.4

$$\overrightarrow{w}_g = \overrightarrow{v}_g + \mathbf{T}_{gb}([\mathbf{T}_{ba}\overrightarrow{u}_a] + [\dot{\Omega}\times\overrightarrow{r}]_b)$$
(2.3.9)

In Equation 2.3.9  $\dot{\Omega}$  is the angular rates of the aircraft  $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ . The term  $\overrightarrow{r}$  relates to the distance of the probe from the IMU, which should be near the center of gravity of the aircraft. The angular rates are multiplied by the probe arm to get angular velocities. For UAS much discussion is had over whether this term is necessary since the distance from the center of gravity to the probe tip is small in UAS [33][30]. Nichols et al. also notes a  $v_{flex}$ term that accounts for unaccounted flexing from the aircraft. While noted this term is not able to be measured so this term is left out of calculations[30].

## 2.3.3 Unmanned Aircraft Atmospheric Measurements

Wind measurement techniques have been tested for both fixed wing platform and multirotors. Flight campaigns such as the Lower Atmospheric Process Studies at Elevation - a Remotely piloted Aircraft Team Experiment (LAPSE-RATE)[15] and the Collaboration Leading Operational UAS Development for Meteorology and Atmospheric Physics (CLOUDMAP)[23] have demonstrated the potential for UAS based meteorological efforts.



Figure 18: Comparison of LAPSE-RATE Wind Measurements[15]

The LAPSE-RATE flight campaign was a multi-university co-operative flight campaign that took place in July of 2018. A total of 8 universities and 2 private companies. 13 different aircraft were equipped to measure wind speed. Figure 18 is a comparison between the wind speed measurements of all 13 of these vehicles and a ground sensor station the Mobile UAS Research Collaboratory (MURC). Figure 18 shows a comparison of all the wind measurements of the aircraft flown. Overall all the mean value differences between all the aircraft wind speeds was  $.22m/s \pm .59[15]$ . One of the major issues noted with comparing these systems is managing different data operations between the teams. For the comparisons no control was placed on the post processing of the data.

The post processing method implement by the University of Kentucky is a good example of how much post processing can change the accuracy of the probe[11]. To account for biases in probe measurement a optimization process is implemented on an objective function( $\delta$ ). The process is applied to a section of the flight where the aircraft does not have significant
acceleration or deceleration. Orbits after take off are recommended. The objective function is as follows

$$\delta_U = \langle U \rangle_{U_{ac} > 0} - \langle U \rangle_{U_{ac} < 0} \tag{2.3.10}$$

$$\delta_V = \langle V \rangle_{V_{ac} > 0} - \langle V \rangle_{V_{ac} < 0} \tag{2.3.11}$$

$$\delta = \delta_U^2 + \delta_v^2 \tag{2.3.12}$$

In equation 2.3.10 and 2.3.11  $\langle \rangle$  signifies an average over the total data set. The equations are taking the difference in velocity over positive and negative aircraft inertial speeds,  $U_{ac}$ . Then equation 2.3.12 is minimized for each source of bias. The sources of bias that are assumed are as follows

$$\theta(t) = \theta_m(t) + \Delta\theta \tag{2.3.13}$$

$$\phi(t) = \phi_m(t) + \Delta\phi \tag{2.3.14}$$

$$\psi(t) = \psi_m(t) + \Delta\psi \tag{2.3.15}$$

$$Q(t) = \zeta Q(t) \tag{2.3.16}$$

$$U_m(t) = U_m(t_m + \Delta t) \tag{2.3.17}$$

These biases account for heading errors:  $pitch(\theta)$ ,  $roll(\phi)$ , and  $yaw(\psi)$ . Dynamic pressure error Q. The final error is a time shift error. The source for this is a time shift between the sensor reading and the INS readings.

Table 2: University of Kentucky LAPSE-RATE Correction Factors [11]

sUAS	Flight	$\Delta \theta$	$\Delta \phi$	$\Delta \gamma$	ζ	$\Delta t$
1	08:30 MDT	$-4.9^{\circ}$	0.17°	2.6°	1.1	0.98 s
1	09:30 MDT	$-5.8^{\circ}$	2.7°	$2.1^{\circ}$	1.1	0.01 s
2	09:10 MDT	$-3.9^{\circ}$	$0.85^{\circ}$	1.4°	1.15	2.9 s
2	10:10 MDT	$-3.9^{\circ}$	$0.78^{\circ}$	1.3°	1.15	2.6 s

Applying these calculations led to the correction factors shown in Table 2. The corrections were most significant in the pitch axis. The corrections on heading and dynamic pressure are large enough to be significant in accuracy of the the probe.

Table 3: UAS MHP System Accuracies

Vehicle	Probe Type	Speed Accuracy (m/s)	Accuracy Method
M2AV[1]	5HP	0.5	$1\sigma$
Blackswift S0[9]	5HP	0.48	N/A
ALADINA[16]	5HP	0.5	Gaussian error
wind RPA $[17]$	5HP	1.1	Gaussian error
Manta[34]	9HP	0.045	$1\sigma$
Scan Eagle $[34]$	5HP	0.045	$1\sigma$

Table 3 is a comparison of a few different MHP equipped UAS aircraft. Some aircraft have different uncertainties for horizontal and vertical wind, the table lists only horizontal uncer-

tainties. The Manta and the Scan Eagle both have significantly better accuracies due to the higher quality autopilots that are used to resolve the aircraft state. The Manta is designed by BAE systems and the Scan Eagle is designed by BOEING. Both autopilots have extremely low uncertainties. For example the Blackswift S0 maintains a 0.02m/s accelerometer uncertainty and a 1° attitude uncertainty[9]. The Manta and Scan Eagle both have accelerometer accuracies 0.01 m/s and attitude uncertainties ranging from 0.005 to 0.011 degrees[34].

# CHAPTER III

# FIVE HOLE PROBE DESIGN AND GROUND CALIBRATION AND TESTING

#### 3.1 Five Hole Probe Development

Multi hole probes make an ideal instrument for wind speed measurements from fixed wing aircraft. Multi hole probes can provide an accurate 3D wind measurement given proper calibration and integration. The details and methodology of the probe design, development, and wind tunnel calibration is presented in chapter three. Details on probe integration to the vehicle and validation of vehicle system measurement is discussed in chapter 4.

#### **Development - 3D Printed Probe**

As with regular manufactured MHPs, with 3D manufactured probes there are three main components of consideration for designing the probe: size, shape, and length. Taking the knowledge from studies over 5HP's, the design for the 3D-printed 5HP has a maximum outer 0.5inch diameter and hemispherical head[14]. The maximum diameter was chosen because the diameter provided enough room for all five of the internal hole lines to be printed with the available resolution of the SLA FormLabs printer while still small enough to fit inside a small UAV and provide the performance needed based off of previous studies [14]. The inner hole diameter for the 5HP was chosen to be 0.06 inches. This was verified to be the smallest hole diameter the SLA printer could reliably print while maintaining the integrity of the hole and keeping clear of resin debris. This diameter was also the same diameter as the pressure transducer ports on the sensor board. Each of the five holes were evenly spaced apart from each other across the 0.36 in diameters of the probe body. The static line was placed at 1.5 inches away from the tip of the probe. The static ports themselves consist of a static ring of 0.2 inches in diameter. This ring is connected by 4 perpendicular holes that extend to the outside of the probe. Then a 0.06 inch hole is connected to the static ring and runs down to the end of the probe at 2 degree offset to provide additional room at the end probe for mounting. The total length of the probe is seven inches ending in a chamfered end. The length of the probe was chosen such that it provided enough room to be safely mounted to a fixed-wing UAV or wind tunnel for testing. As the intention for this probe was to be fully 3D-printed, the pressure lines had to be able to connect to the probe itself. At the end of the probe six ports extends 0.15 inches out from the chamfered end. The end was chamfered to put each of the connecting ports at different distance to allow for easier accessibility when connecting the Tygon pressure lines to pressure ports. The probe's isometric dimensions are shown in Figure 3.



Figure 19: Probe Dimensions: A. Side View, B. Back View, C. Front View, D. Bottom View, E. Isometric View

## **3D** printing Process

The 3D prints of the probes were prone to failure. This was typically caused by clogged ports. Through much trial and error a semi reliable printing process was found. One of the major trade-offs found was between print time and print quality.



Figure 20: Probe in Mid Print on Formlabs Printer

Figure 20 is a photo of on one of the probes nearing the end of the print. This probe is at around a 45 degree angle to the bed. This was found to be a good angle to print the probe at. Any lesser angle, although it printed faster, resulted in a clogged probe. It was found that for best print results the probe needs to be cleared with alcohol, preferably within an hour of print finish. This prevents the resin from air drying.



Figure 21: Probe Tubes Being Cleared with Alcohol

Figure 21 shows how the probe is cleaned post print. A small bottle of isopropol alcohol is squirted into each of the holes. This is cycled with blowing with an air hose until the alcohol goes through and there is no resin clogging the probe. After this process the probe is sent through the Form Wash and Cure. This process submerges the probe in an isopropol alcohol solution that is then agitated to clean the probe further. After that the probe is inserted into a cure chamber where the temperature is raised to 80C and exposed to LED light. This cures the and hardens the resign improving strength and quality.

## 3.1.1 Development - Hybrid Probe



Figure 22: Metal Probe

While the work of Solmoz proved the feasibility of a 3D printed probe[14]. As previously discussed, when the 3D printed probe was initially lengthened finding proper print procedures keep the holes clear enough for the probe to function was a challenge. While a printing process was eventually established that led to successful probes, two different "hybrid" probes were made. The idea was developed from the RPP probe designed by RMIT University. This probe was designed with only a 3D printed tip, and end cap this was to be able to change the tip if the probe took damage during flight operations[32]. With this as a driver a probe was made using the same dimensions as the fully 3D printed, however only the tip was 3D printed. The body of the probe was a quarter inch aluminum tube with a .36 inch inner

diameter. 4 holes were drilled at right angles in the tube for static ports. The tip has 5 .10 inch holes for .06 inch inner diameter brass tubes. These act as the ports for the 5HP.



Figure 23: Metal Probe Tip

Seen in Figure 23 a notch was placed on the 3D printed tip and the aluminum tube. This prevented the tip from rotating. On the back end of the tube there are 6 tube pass through. One for each of the 5 holes and one for the static pressure. These probes have similar performance as the 3D manufactured probes, operationally these probes were found to be more reliable as will be discussed in the flight testing section.

### Calibration

Because of the inherent flaws in manufacture of MHPs. To measure 3-dimensional flow, each probe requires its own calibration[22]. This is compounded by the fact that these probes are 3d-printed and will have slightly different flow characteristics even when printed on the same 3d-printer. The goal is to determine experimentally the pressure data sets that show probe's behavior in a known flow field. Using the total and static pressures, pitch and yaw angles, and the velocity magnitudes, the data can be expressed as non-dimensional pressure coefficient values. This known flow field is taken over a range of known Reynolds numbers, Mach numbers, and velocities as the probes are traversed across a range of angles incrementally.

Due to the set up of the lab experiment available, the non-nulling method was used to calibrate all of the 3d-printed probes for this research. The wind-tunnel used for non-nulling calibration of the 3D manufactured probes in this experiment is a Flotech 1440 produced by GDJ Inc. with a 12X12X36 inch test section as shown in Figure 4. The wind-tunnel is equipped with a 2hp motor and capable of producing flow speeds from 0 – 25 m/s. The test is best suited for low Reynolds number testing and is controlled by an analog wheel that changes the RPM of the motor. The upstream Pitot-static probe that is connected to an Omega Analog sensor that outputs the pressure and velocity data through a Labview DAQ. The stepper motor used to sweep the probe back forth is a DMX-UMD-23 stepper motor produced by ARCUS Technology. The sensor that is being used to record the pressure from the MHP is a custom made sensor board that is using a Teensy 3.6 and three Honeywell SSCDRRN001PD2A5 digital pressure transducers seen in Figure 5. This sensor package allows for a data capture of 200 hertz. Both pitot-static probe and the MHP were connecting to their respective sensors via Tygon tubing. The sensor system will be discussed further in Chapter 3 where the DAQ can be discussed on an aircraft level.



Figure 24: Wind Tunnel Arrangement



Figure 25: Digital Sensor Package

The procedure for the experiment started by mounting the MHP to the stepper mount via a steel rod and placing it in the center of the wind tunnel at 0 degrees as shown Figure 6. To keep the tip of the probe away from the boundary layer effects from the edges of the test chamber, the probe will pivot around its tip instead of its end. This was done by 3D-printing an extension mount that connects the probe to the steel rod.



Figure 26: MHP Mounted in Wind Tunnel

Because of the nature of the non-nulling calibration method (yawing sweeps) and 5 hole probes having pitch and yaw angular observations, only the one set of pressure holes will receive pressures differential in a given sweep. Therefore the probes will needed to be mounted and swept in both an alpha and a beta orientation independently to properly calibrate them, as seen in Figure 7.





(a) Alpha Orientation

(b) Beta Orientation



After the probe is mounted securely to the wind tunnel, the Tygon tubes are run through the bottom of the wind tunnel and connected to the digital sensor board. The board is then turned on by connecting the micro-USB port to a 5V power supply. Then, the windtunnel is turned on to a fixed speed and allowed to run continuously to reach steady-state (30 seconds). Once this is done, the stepper motor program is started and the probe moves from -45 degrees to +45 degrees in 5 degree increments. The range was chosen because the operating range for similar MHPs has been proven to be around -30 to +30 degrees. Thus the entire operating range of the probe is captured. The probe is held at each step for 30 seconds, allowing the tip flow to reach steady state and gather enough data to acquire an accurate average at that point. It is worth mentioning that the response time for the probe is much shorter than than 30 seconds ( $\sim 10$ ms), but this allows for a more accurate calibration. The sweep is then repeated for each test cycle, and all three sweeps are averaged together for that specific speed. There are three total speeds the probes were tested at: 10, 15, and 20 m/s for both alpha orientation and beta orientation, totaling nine test per orientation. Once the experiment is performed, the data is offloaded via micro SD card, parsed in Matlab, and the graphs are generated.

Calibration of the multi-hole probes in known flow-field and the curves associated derived from the calibrations is critical for accurate in flight measurements. The key to calibrating MHPs are the coefficient of pressures. The pressure coefficient is a non-dimensional feature that is defined by the pressure difference over the dynamic pressure. The dynamic pressure is defined as,

$$q = p_a - p_{\infty} = \frac{1}{2} * \rho_{\infty} * U_{\infty}^2$$
(3.1.1)

For the pitch angle, this is

$$C_{p\theta} = \frac{P_2 - P_4}{P_1 - P_a} \tag{3.1.2}$$

and for the yaw angle, the equation is

$$C_{p\phi} = \frac{P_3 - P_5}{P_1 - P_a} \tag{3.1.3}$$

and for pitot coefficient of pressure

$$C_{pPitot} = \frac{U_{pitot}}{U_{\infty}} \tag{3.1.4}$$

As shown in the Figure 8 the axis being tested  $C_{p\theta}$  has a slope of some gradient, whereas,  $C_{p\phi}$  is nulled and has zero or nearly zero value. Another important parameter for MHP's is the angular operating range. As Figure 9. depicts, the further along the curve within the linear range gives the angular range. As mentioned before, different MHP's tip geometries affect the MHP's operating angle range. Outside of the nonlinear curve of the probe the angle sensitive is less, therefor, maintaining values of  $C_p$  that correspond to the associated angle is harder to correlate.



Figure 28: Notional pressure coefficient curves for pitch and yaw depicting a 5HP traversing on the pitching axis while the yaw axis nulled



Figure 29: Illustration depicting pressure coefficient linear range

Figure 10. shows the decomposition of velocity, U, where the rotational axis for pitch and yaw  $\theta$  and  $\phi$  are determined by from pressure differences on their respective axis using Equations 2 and 3.



Figure 30: Orientation of Vector pitch and yaw angle

Using the Bernoulli's equation the magnitude of the velocity vector can be resolved.

$$U = \sqrt{\frac{2}{\rho_{\infty}}q} = \sqrt{\frac{2}{\rho_{\infty}}(p_o - p_{\infty})}$$
(3.1.5)

Fig. 11 shows a ideal velocity magnitude curve with symmetry about about  $\theta$  and  $\phi$ . As the probe is symmetric, the curve should ideally be symmetric about  $\theta$  and  $\phi$ .



Figure 31: Ideal Velocity Magnitude Curve

After calculating the magnitude and obtaining the direction of the velocity, the three velocity components can be be calculated by using equations 3.1.5, 3.1.8, and 3.1.7.

$$Re = \rho * u * L/u \tag{3.1.6}$$

$$w = Usin(\phi) \tag{3.1.7}$$

$$v = Usin(\theta)cos(\phi) \tag{3.1.8}$$

The main measurement gain from using MHP is velocity, the effect of Reynolds number should be considered for calibration [20]. Increasing the know flow-field velocity of the wind tunnel will increase the Reynolds regime as well to its relationship to velocity.

$$Re = \frac{\rho * U_{\infty} * D}{u} \tag{3.1.9}$$

During the calibrations a range of Reynolds number should be set. The range of the Reynolds number is dependent on the expected operating range of the MHP. For these MHP's that operating range is the flight regime of the UAVs it being is mounted to (10 - 25 m/s). With that being stated multiple speeds of the flight regime should be tested to ensure the consistent of the MHP calibration. For these probes speeds of 10, 15, and 20 m/s were chosen.

Calibration results for the 3D-manufactured probes are presented below for both the pressure coefficient and velocity magnitude curves.



Figure 32: Probe2 (a)  $C_p$  Alpha Vs. Angle and (b)  $C_p$  Beta Vs Angle



Note the linear range of the Probe2 falls offs significantly at the outer ranges of the MHP (45 and -45). This confirms past research in that it shows the probes performance is not adequate at these angle ranges and therefor should only be operated in angles within its linear range of (30 to -30) degrees. Figure 12 also shows the similarity of  $C_p$  values across all three speeds tested. The linear relationship for both probes has a  $R^2$  value of 0.95 (almost 1) indicating there is a good overall data fit. The Probe3 results are shown in Figure 13. The comparison between the two probes demonstrates the repeat-ability of the probe design.



Figure 14 show the magnitude velocity curve for the 3D-manufactured probes. The figures show symmetry of the maximum velocities throughout the traverse from -45 to  $+45^{\circ}$ 

demonstrating the repeat-ability of the 3D-manufactured probes

#### Validation

After proper calibration of the probes the calibration data can be tabled and used to interpolate velocities and angles from unknown measurements. Development of the 3Dmanufactured probe is done with the intended use on UAS fixed winged vehicles in mind. The intended vehicles mostly stay under 40 m/s as the flight speed. This allows for the interpolation process for probe to be simplified since the probe will stay in a laminar flow for the duration of measurements. This is significant because the three major quantities used for interpolation,  $C_{p\alpha}$ ,  $C_{p\beta}$ , and  $C_{ppitot}$  will not have significant changes over the expected air speeds.



Figure 35:  $C_{p_pitot}$  Change with Velocity and Angle

Figure 15 illustrates how  $C_{ppitot}$  changes from 10 to 20 m/s. The  $C_{ppitot}$  trends for each or the three speeds are nearly identical. For interpolation purposes the assumption that no significant change between speed was made. This same assumption was made for  $C_{p\alpha}$  and  $C_{p\beta}$  these trends are graphed in Figure 32b. These assumptions allow for the interpolation process to be broken down into five major steps.

- 1. Calculate  $C_{p\beta}$  and  $C_{p\alpha}$  from the pressure measurements.
- 2. Interpolate for Alpha and Beta from the coefficient of pressures.
- 3. Use Alpha and Beta to interpolate for  $C_{ppitot}$
- 4. Use  $C_{ppitot}$  to Calculate U using Equation 3.1.4
- 5. Using Equations 3.1.5, 3.1.8, and 3.1.7 with Alpha as  $\psi$  and Beta as  $\theta$  resolve the velocity vectors.

As a step to verify that the interpolation method works the code was used to resolve angles and velocities from calibration data. The calibration data provides a good known quantity for to test the interpolations against. The values that are compared are the values of Alpha and Beta, and the velocities(total and components).



Figure 36: Probe3 Interpolated Comparison to Actual Angle

Figure 36 is a comparison between the known Alpha and Beta angles and the interpolated values. One major outlier is present. The -45° angle has a significant error. This is constant across all three probes and all test. The cause for this error is currently unknown. More test will be done to attempt to isolate the cause of the error. However, the error should be

outside the conditions encounter by a fixed wing UAS and should not affect normal data operations.



Figure 37: Percent Error of Interpolated Angles

Figure 37 shows the percent error for the different angle interpolations across the three different speeds. Note the for the graphs, the -45° percent error was near 100 percent. For scaling purposes it was removed from the graph. There is a peak around the 0°. One of the causes of this could be asymmetry in the probe tip that is unexpected causing pressure differentials. The second interpolation point to verify is the velocities.



Figure 38: Velocity Comparison for 10m/s Alpha Sweep

Figure 38 shows a comparison between the known velocities and the interpolated velocities

for a 10 m/s alpha sweep. As previously noted there is still an anomaly at the -45°step. Other than that issue the interpolated data is relatively accurate.



(a) Percent Error Total Velocity Alpha Angle

(b) Percent Error Total Velocity Beta

Figure 39: Percent Error of Interpolated Angles

Figure 39 is the percent error of the Total velocity interpolations. The errors stay relatively low except for at the 45° extremes. The percent error also increases with velocity. The error increase near zero degrees in the Beta orientation exist the same as it did with the angles. Overall the Beta error has a is less linear. This could be due to print support defects left on the probe. Due to the print method small bumps of support material are left along the side of the probe. While these are sanded they may still cause pressure disturbances along the probe.

#### 3.1.2 Quantization Error

Since the MHP is operating using digital pressure transducers a quantization error is present. The equation for quantization error is as follows:

$$\epsilon_r = \frac{V_{fso}}{2^n} \tag{3.1.10}$$

The Honeywell SSCDRRN001PD2A5 is a 14 bit digital sensor that operates on a  $\pm$  5V range. In equation 3.1.10  $V_{fso}$  is the voltage range of the sensor in this case 10V. n is the the bit count of of the senor, 14. This lead to a quantization error of the senor of .0006V. This is the smallest voltage change detectable by the sensor. This voltage change converts to a .00012psi resolution. Using equation 3.1.5 at sea level standard atmosphere with a density of 1.225kg/m<sup>3</sup>. The quantization error of .00012psi will be used as the  $(p_0 - p_{\infty})$ . This leads to a velocity precision of  $\epsilon_r = 0.014m/s$ .

#### Probe Response

A good understanding of probe response is of interest to begin to understand how the probe will react to gusts. This will also give a good idea of the probes maximum response time and begin to set a standard for frequency response.



Figure 40: Pitot Step Response

Figure 40 is a picture of the experimental setup. A balloon was placed around the tip of the probe and filled to capacity. Care was taken to leave the static ports uncovered to create a pressure difference between the static and total pressure ports. The balloon was then popped which led to the total port returning to the same pressure as the static port. This should be

nearly and instantaneous change.



Figure 41: Alpha Step Response

Figure 41 is the response of the alpha pressure transducer to the step response. The responses of the Beta and Pitot sensors can be found in the appendix in Figure ?? and ??. The three sensors all had a settling of around 20 ms or .02 seconds. Using this response time the probes should have a maximum response rate of 50Hz.

## 3.2 Ultrasonic Anemometers

Ultrasonic anemometers offer a good solution for wind measurements from multi-rotor UAS. Multi-rotors outfitted for wind measurements have been found to be more accurate than fixed wing methods[15]. Ultrasonic anemometers have large range of size, weight, and accuracy. Many of the more accurate anemometers are prohibitive in weight or size for easy mounting on a multi-rotor. Below is a comparison of the anemometers used to assist calibration and validation of the 5 hole probe.

Anemometer	Use	Type	Range $(m/s)$	Resolution $(m/s)$
R. M. Young 92000[5]	Ground Station	2D Anemometer	0-70	0.01
R.M. Young 5103[6]	Mesonet Tower	2D Anemometer	0-100	N/A
Trisonica Mini[3]	Quad Mounted	2D Anemometer	0-50	0.01

Table 4: Anemometer Table Part 1

Table 5: Anemommeter Table Part 2

Anemometer	Wind Speed Accuracy	Wind Direction Accuracy
R. M. Young 92000	0 - 30 m/s $\pm 2\%$ or	$\pm 2^{\circ}$
	0.3  m/s, 30  -  70  m/s	
	$\pm 3\%$	
R.M. Young 5103	$\pm$ 0.3 m/s (0.6 mph)	N/A
	or $1\%$ of reading	
Trisonica Mini	$0 - 10 \text{ m/s} \pm 0.1 \text{ m/s},$	$\pm 1^{\circ}$
	11 - 30 m/s $\pm 1\%$ , 31-	
	$50 \text{ m/s} \pm 2\%$	

All of the wind sensors are assumed to have the manufacture defined accuracy and resolution. The two anemometers operated by Oklahoma State use a TEENSY based data logger. Both of those sensors communicate via a RS-232 protocol, RS-232 uses  $\pm 13$  volts for communication. The TEENSY is equipped for TTL communication which operates via a  $\pm 3.3$ volt signal. To make these compatible a SparkFun RS232 Shifter is used to convert the  $\pm 13$ volt signal to a  $\pm 3.3$  volt signal. This allows for communication between the sensors and the TEENSY without damaging the TEENSY.



Figure 42: Anemomet Trisonica Mini

The TriSonica Mini(Figure 42) is small light weight ultrasonic anemometer. The anemometers size and weight make it a good choice for use on a multi-rotors. The sensor weights 50 grams(2oz) and measures 9.1x9.1x5.2cm. The manufacturers posted range, resolution, and accuracy are shown in Table 4. The sensor outputs wind speed and direction at 4Hz using a RS-242 communication protocol. The Trisonica operates at a Baud rate of 9600.



Figure 43: R.M. Young 92000 on Mount for Tower Operations

The R.M. Young 92000 is used as a reference anemometer for calibration flights. The R.M. Young 92000 is a 2D ultrasonic anemometer. The range resolution and accuracy can be found in Table 4. The anemometer is design to be mounted to a post for continuous wind measurement. The sensor outputs 2D wind and direction at 4Hz using a RS-242 communication protocol. The Young 5103 operates on at a Baud rate of 38400.



Figure 44: R.M. Young 5103

The R.M. Young 5103 is relevant for flights against around Oklahoma Mesonet site. The Oklahoma mesonet reports averaged wind speed and direction from the Young every 5 minutes. The Young 5103 output an anolog signal at 90Hz. A 1.0m/s wind is required for the sensor to register an accurate speed. A 1.4m/s wind is require for direction measurements. As discussed in the Chapter 2, since the Young 5103 is a propeller vane anemometer it has inherit lag for wind speed and direction. For the wind speed the 5103 has a distance constant of 2.7 meters with a 63% recovery. The direction has a 1.3m distance constant with a 50% recovery.

# CHAPTER IV

# INSTRUMENT INTEGRATION AND FLIGHT TESTING AND CALIBRATION

## 4.0.1 Data Acquisition System Architecture

The data acquisition system used for 5-hole Probe and anemometer sensors is built around a Arduino Teensy micro controller. The system functions for Arduino Teensy 3.6 or 4.1. The Teensy is programmable through the Arduino IDE with the Teensyduino software package. The Teensy has a good balance between function and size. The Teensy 4.1 has the dimensions of 2.4 in x .7 in. The Teensy has pins for 8 serial connections, 3 SPI connections, and 3 I2C.



Figure 45: Wire Diagram of Logger System

Figure 45 is the wiring diagram for how the Teensy interfaces with the Pixhawk and the sensors. The temperature and humidity sensors on the left side of the schematic are optional and are not installed on the Nano Talon. The temperature and humidity segment of the DAQ system are also still in early stages of development, they are not be calibrated or validated. Because of this when temperature and humidity are needed I-met sensors are used. The Nano Talon does have slightly different layout due to the different autopilot. Instead of the Pixhawk Cube with a carrier board the Nano Talon uses the mRo AUAV 2.1.

One of the major issues of MHP systems is time syncing the data of all of the sources[11]. The DAQ system is designed to minimize this time issue. This time error minimization is done by sharing the Pixhawk board time to the Teensy. This is done over the RX TX UART lines between the Pixhawk and Teensy using the MavLINK protocol. Both devices keep time via there clock cycles. Since this is shared between the and logged on the Teensy, the hypothetical time lag between data sets would be just the time it takes for the two systems to communicate. Although there is also an inherit lag in the physical interaction of the MHP

with the surrounding air. This time comparison is the foundation of the data processing method.

## 4.0.2 Aircraft Description

The multi hole probes are integrated with 2 different vehicles the VTOL Nimbus and the Nano Talon. The two vehicles are pictured below.



(a) VTOL Nimbus



(b) Nano Talon



The Nano Talon and the VTOL Nimbus were selected for weather operations due to the ability for the aircraft to be quickly deployed. The Nimbus is able to be taken off and landed in a VTOL configuration. This allows quick and simple deployment that does not require an special launch methods or a runway. The Nano Talon is a small hand launched foam aircraft allowing for quick and easy deployment from nearly anywhere. The aircraft is belly landed this is possible on a variety of surfaces, although grass is preferred.

Both aircraft are controlled with Pixhawk flight controllers. The Pixhawk operates using Ardupilot control algorithms. The Ardupilot built in Kalman Filters are used to remove the aircraft states from the probe measurements. As a result the accuracy and errors in the wind estimation of the system are significantly influenced by the quality of the Pixhawk sensors and Kalman Filter.

#### 4.0.3 Sensor Mounting

The process to integrate and use a multi-hole probe on a UAS can be broken down into 4 different steps.

- 1. Wind Tunnel calibration of probe
- 2. Physical Integration on Aircraft
- 3. System Error and Offset Calculation
- 4. System Validation

Aligning the sensor with the aircraft axis is important to avoid bias errors. The mounting methods of the Nimbus and Nano Talon are both designed to keep the sensor central. However, one challenge with MHP measurements from small UAS is consistent mounting. Much of this bias has been able to be addressed in post processing[11].

## Nano Talon

The Nano Talon mounting presented a challenge due to the small payload volume of the aircraft. The mounting method for the Nano Talon went through three different iterations. For all of the mount configurations the data acquisition board was left in a payload area in the belly of the Nano Talon.



Figure 47: DAQ in Nano Talon

Figure 47 shows how the DAQ is integrated into the Nano Talon. The board sits on top of a RFD 900 Telemetry unit. The RFD 900 is used for telemetry link to a ground station. Attached to the pressure sensors on the DAQ is 6 15.5 in lengths of pitot tubing. This tubing runs up front to the probe at the nose of the Nano Talon.



Figure 48: Nano Talon Probe Mount

Figure 48 shows the location of the final probe mount. This location allowed the pitot tubing to be passed through a hole in the nose back to the DAQ. It also avoided conflict with the battery since the probe comes off with the hatch. This location is centrally located on the aircraft and the probe is place carefully to minimize any issues with offsets from the aircraft axis.

# Nimbus

The Nimbus has a large payload area in the nose in front of the battery this is where the the DAQ is stored. The Nimbus has 21 inches of pitot tubing between the DAQ and the probe. The probe is mounted to a wood plate integrated into the nose.



Figure 49: Nimbus Probe Mount

The probe is then bolted to the wooden plate. Figure 49 shows a top view of the nose section. The black section is a airflow duct for the integration of temperature, pressure, and humidity sensors.

## 4.0.4 3D Print Durability Issues

Fully additive manufactured multi-hole probes have been proven to be accurate in a wind tunnel setting[14]. Significant problems with the probes became apparent with repeated field use. The probes printed with the Formlabs Form 2 printer. Two different resin types were tested for in the field. The first resin used is Formlabs Tough resin, unfortunately Formlabs does not have material properties posted for this material. The other used is Formlabs Durable resin. The technical data can be found on Formlabs website[7].









Figure 50: Probe Wear

Figure 50 shows damage to the probe that occurred from regular use. Figure 50a show warping that occurred to the probe this was observed even when the probe was in storage, although transportation increased the chance of the warping. This warping got severe enough to be an issue and make the calibrations invalid. The significant scrapes on the probe tip shown in Figure 50b presents a similar issue.



Figure 51: Broken Probe Pitot Attachment

The most significant issue with the fully 3D printed probe was the nubs the pitot tubing attach to breaking off. These nubs are very brittle and multiple flight tests were cut short due to them breaking. Due to the small nature of both aircraft the probe was always in an area where flight operations such as replacing batteries risked breakage. These issues with the 3D print material led to a switch to the hybrid probes for flight testing. These probes proved very durable and reliable.

#### 4.0.5 Wind Vector Realization

This section will walk through the process of taking raw 5 hole probe and aircraft data and deriving wind vectors from it. The flight data used is from flight tests on 6/2/2021. Two different flight were flown, a flight in a wind box pattern and a flight doing orbits. The weather for the flights was very low winds, between 0-4 m/s with the Stillwater airport recording a max wind speed of 8mph or 3.5m/s at the times of the flights. The wind was blowing SSW shifting to NNW. The temperature for the flights was 74°F with a relative
humidity of 60%. We will start the wind vector realization process on the orbit flight first. This flight was flown between 1:50 and 2:05 CST.



# **Orbit Processing**

Figure 52: Orbit Flight Path From 6/2/2021

Figure 52 shows the flight path flown for the orbit flights. These orbits were at constant altitude with minimal changes in airspeed.



Figure 53: Raw Velocity MHP Data From 6/2/2021 (Orbit)

To acquire a wind vector the aircraft states must be removed from the 5 hole probe measurements. To remove the states from the probe the algorithm discussed in chapter 2 was implemented. Before the algorithm can be implemented the probe data and the aircraft state must be fused. The 5HP runs at 60 to 100Hz. Speeds have been maintained at 180Hz, however not reliably. The Pixhawk Kalman filter runs at 25Hz. To match these data sets the MHP was averaged down to 25Hz for all state removal.

Recalling equation 2.3.8 the aircraft heading  $(\phi, \theta, \psi)$  come from aircraft Attitude data. This data comes mostly from compass and IMU calculations. The ground velocity  $\overrightarrow{v}_g$ (VN,VE,VD) is derived from the GPS, Barometer, Pitot Probe, and IMU. Alpha, Beta, and,  $\overrightarrow{u}_a$  come from the MHP. The wind frame states are the values displayed in Figure 53.



Figure 54: NED MHP Data From 6/2/2021

Figure 54 is the resulting wind vector from removing the aircraft states (other than angular velocities) from the MHP data. The data appears to be very noisy at the beginning this is due to the aircraft being in takeoff conditions not level flight. This kind of data is discarded for wind measurements. For comparison equation 2.3.9 is used to remover the angular rates.



Figure 55: NED MHP Data From 6/2/2021 with Angular Rates Removed

Removing the angular rates let to insignificant differences. The overall wind speed estimation saw a percent difference of .04%. This is because the small lever arm of the aircraft combined with a minimal rotation rate leads to insignificant changes in probe measurements. For example the maximum pitch rate for the aircraft on this flight is 0.0341rad/s with a lever are of .4m that lead to an observed air speed on the probe of .013m/s. For general wind data this is an minimal change. The rates for the rest of the calculations will be assumed to be irrelevant.

Beyond that there is significant oscillations in the wind vectors. This is one of the issues noted by Rautenberg about taking data with orbit patterns[33]. However, the data can be averaged over the orbit period to get an accurate wind vector from an orbit. Since the orbit period is difficult to measure in the field a single-sided amplitude spectral analysis is used. To begin this process first the sample frequency and sample length must be calculated. For this analysis a Fast Fourier Transform (FFT) can be run on the raw direction or airspeed data. For the FFT on this data Matlab's fft function is used[4].



Figure 56: Single Sided Amplitude Spectral Analysis Data From 6/2/2021 (Orbits)

Figure 56 is the result of running the data through an FFT. There is a significant and easily observable spike around a frequency of .05Hz. This relates to a 20 second period which aligns with the typical orbit period for a Nano Talon. The peak is then selected in the Matlab program.



Figure 57: Averaged NED MHP Data From 6/2/2021

Figure 57 is the averaged data. For this average the data has had a moving average with the length of one orbit period applied. The requirement for this average is one thing that makes

orbit patterns undesirable. Much of the wind resolution is lost having to average out the orbit period. For wind speed and direction data, which is what is desired for meteorological purposes, the orbits remain useful. For data that does not have to be averaged different flight patterns can be flown. The most popular of these is a wind box.



# Wind Square Processing

Figure 58: Wind Square Flight Path 6/2/2021

Figure 80 shows the flight path flown for the wind square flights. These squares are at constant altitude with minimal changes in airspeed.



Figure 59: Raw Velocity MHP Data From 6/2/2021 (Squares)

Figure 59 is the raw wind frame data from the MHP from the wind square flights on 6/2/2021. Compared to figure 53 the Beta angle is far less oscillatory. The data has more noise than the orbit data.



Figure 60: Single Sided Amplitude Spectral Analysis Data From 6/2/2021 (Squares)

Figure 60 is the single sided amplitude spectral analysis data from the wind square flights. While there are some low frequency spikes, a significant one such as the one observed in the orbit data (Figure 56) is not seen. This is because the wind square path does not contaminate the data with an maneuver as badly as the orbits do. This reduces the need to average the data after the aircraft states are removed.



Figure 61: NED MHP Data From 6/2/2021(Square)

Figure 61 is the wind squared data with the aircraft states removed. No a significant oscillations are seen in the data. For baseline meteorological purposes a moving average over the duration of the square is still valuable. This square took around 1 minute for each complete rotation. Applying a 1 minute moving average.



Figure 62: Averaged NED MHP Data From 6/2/2021(Square)

Figure 62 is the wind square data with a moving average of a minute applied. While higher frequency events will be lost in this average it will be valuable to compare to average wind speeds from other sensors.

### Wind Vector Conversions



While the Earth based NED is the preferred method of display for meteorology converting the vector to a wind angle and a wind speed can make comparisons between data sources easier. Figure 63 shows the wind ploted as a speed and direction. To plot this the 3D wind vector is converted to a 2D horizontal speed and angle. This is helpful for many of the data comparisons. This is done with the following two equations.

$$Wind_{speed} = \sqrt{(VWN)^2 + (VWE)^2} \tag{4.0.1}$$

$$wind_{direction} = \arctan(\frac{WVN}{WVE})$$
 (4.0.2)

In equation 4.0.1 and 4.0.2 WVN and WVE are the North wind vector and East wind vector respectively.



### 4.1 SKB 1000 - Trisonica Calibration

Figure 64: SKB-1000 with Trisonica Mount

For in flight validation of the probe comparison to a Trisonica mounted to the SKB-1000 is done. The SKB-1000 is a H-quad designed by USRI. The quad has a 6 lb payload capacity with a 20 minute endurance. The Trisonica is mounted to a gimbal system to reduce the effect of the quads movements. The calibration of the quad was done by James Brenner and is fully documented in his thesis. A brief overview will be provided here.

The quad system was calibrated and assessed in a series of flights next to a tower with a Young 92000. It was found that the most accurate location for the Trisonica mount was at

the center of the quad. The Trisonica is mounted to a 24in carbon fiber tube. The data logger system for the Trisonica is the same as the one designed for the MHP.

	v
Statistics	Result
Avg. Vel. Young (m/s)	1.75
Avg. Vel. Trisonica (m/s)	2.14
Avg. Difference	0.39
RMSE (m/s)	0.57
Bias (m/s)	0.402
Correlation Coeff.	0.71
Variance Young	0.15
Variance Trisonica	0.33
Skewness Toung	0.30
Kurtosis Voung	0.54
Kurtosis Trisonica	0.01
ixuruosis irisoinea	0.10

Table 6: Middle 24in Calm Day

Table 6 is the results of the calibration flights. The root mean square error of the system was found to be around 0.6 m/s. It was noted that higher winds seemed to increase the error of the system.

## CHAPTER V

#### RESULTS

#### 5.1 Validation and Error Analysis

In flight validation of MHPs tends to be a challenge. Literature takes a few different approaches to MHP validation. The three prevailing methods are comparisons to ground stations, comparisons to other aircraft, and comparison to simulations/atmospheric theory. Many research projects have done a mixture or all three. The validation for this probe will involve flying against more validated sensors and an Gaussian error propagation.

#### 5.1.1 Noise Analysis

Before moving onto the validation of the probe the noise in the wind signal will be addressed. The noise is quantifiable through a signal to noise ratio(SNR). SNR can be calculated by taking the mean of a measurement(signal) dividing by the standard deviation(noise) as follows.

$$SNR = \frac{Mean}{\sigma} \tag{5.1.1}$$

Equation 5.1.1 is used to calculate the Signal to Noise ratio. The mean and standard deviation ( $\sigma$ ) are calculated from a steady level flight section. In this section the mean was around 18m/s with the standard deviation of 0.25m/s. This resulted in a SNR of 70. Any value greater than one shows a signal that is stronger than the noise. The SNR can be improved by the implementation of a filter.

#### 5.1.2 Gaussian Error Propagation

The first method of analysis of the probe is a Gaussian error analysis. This analysis was done following the methods laid out in Garman et al.[21]. The Gaussian error propagation is done by calculating the error of all of the major quantifiable factors that will contribute to the wind estimation error. These values can then be placed into a Gaussian propagation equation that incorporates these errors.

The probe error is a combination of all the sensor and parameters required for a wind vector as discussed in Chapter 4. These error sources for the wind vector come from two sources the MHP DAQ system and the aircraft state estimator. The sources of error for the MHP system are from two different sources the probe and the pressure transducers. For horizontal wind this will be the angle accuracy of the probe found via the calibration of the probe( $\alpha, \beta$ ). The other measurement from the probe is airspeed ( $U_a$ ), the precision of this is driven by the pressure transducers. The state estimations of the aircraft aircraft are what is removed from the probe measurement to get the wind vector. This makes the state precision very influential to the error of the system. The roll pitch and yaw precision are used to remove the aircraft angle from the probe measurements. The precision of the attitude states are listed as  $\sigma_{\phi}, \sigma_{\theta}$ , and  $\sigma_{\psi}$ . The aircraft ground speed are the other state taken from the aircraft estimator. These are removed from the probe speed measurement for the calculation of the wind speed. The precision of the these are listed as  $\sigma_{VN}, \sigma_{VE}$ , and  $\sigma_{VD}$ . These relate to the North, East, and down velocities respectively. For integration into the equation the angles  $(\sigma_{\alpha}, \sigma_{\beta}, \sigma_{\phi}, \sigma_{\theta}, \sigma_{\psi})$  must be converted to a velocity(m/s). This is done by converting to radians and multiplying by the airspeed of the aircraft. This returns the velocity change that the probe will encounter from the errors. Finally the terms are combined to develope the following Gaussian estimation equation.

$$\sigma_{wind} = \sqrt{(\sigma_{\beta}U_a)^2 + (\sigma_{U_a})^2 + (\sigma_{VN})^2 + (\sigma_{VE})^2 + (\sigma_{\psi}U_a)^2 + (\sigma_{\theta}U_a)^2}$$
(5.1.2)

. The alpha and beta uncertainties are taken from the probe calibrations and wind tunnel validations shown in chapter 3. This a 4% precision was found assuming a maximum angle change in flight of  $\pm$  10 degrees, from this a precision of .4 degrees can be calculated. The precision of the parameters from the aircraft state estimator are taken from steady level flight data on calm wind days. A 1 $\sigma$  precision is used by taking the standard deviation of the logged states over the steady level flight section.

MRO 2.1x (Nano Talon)			
Variable	Symbol	Precision	
Roll Angle	$\sigma_{\phi}$	$1.4 \deg$	
Pitch Angle	$\sigma_{ heta}$	$1.1 \deg$	
Heading Angle	$\sigma_\psi$	$1.8 \deg$	
Velocity East	$\sigma_{VE}$	$0.25 \mathrm{m/s}$	
Velocity North	$\sigma_{VN}$	$0.25 \mathrm{m/s}$	
Vertical Velocity	$\sigma_{VD}$	$0.25 \mathrm{m/s}$	
True Air Speed	$U_a$	$0.26\mathrm{m/s}$	
Angle of Attack	$\alpha$	$0.4 \deg$	
Sideslip Angle	b	$0.4 \deg$	
$\sigma_{wind}$		.85m/s	

Table 7: Nano Talon Error Propagation

Pixhawk Orange Cube (Nimbus)			
Variable	Symbol	Precision	
Roll Angle	$\sigma_{\phi}$	$1.5 \deg$	
Pitch Angle	$\sigma_{ heta}$	$1.2 \deg$	
Heading Angle	$\sigma_{ heta}$	$2.3 \deg$	
Velocity East	$\sigma_{VE}$	$0.41 \mathrm{m/s}$	
Velocity North	$\sigma_{VN}$	$0.41 \mathrm{m/s}$	
Vertical Velocity	$\sigma_{VD}$	$0.41 \mathrm{m/s}$	
True Air Speed	$U_a$	$0.41 \mathrm{m/s}$	
Angle of Attack	$\alpha$	$0.4 \deg$	
Sideslip Angle	eta	$0.4 \deg$	
$\sigma_{wind}$ 1.26m/s			

 Table 8: Nimbus Error Propagation

Table 7 is the propagation for the Nano Talon. It has a final uncertainty estimation of 0.85 m/s. The Nimbus is shown in table 8 it has a larger uncertainty estimation because of its higher flights speed which is apparent in the angle precision's affect on the propagation. Overall the Nimbus has higher errors in the compass and GPS. This may be caused by the large motors around the autopilot unit. Further study should be done to minimize this noise.

## 5.1.3 Kolmogorov's -5/3 Law

One of the initial methods for checking the quality of a MHPs velocity measurements is to check the adherence to Kolmogorov's -5/3 power law. The law is based around atmospheric turbulence having a frequency decay of -5/3.



Figure 65: Spectral Power Density with -5/3 Law (Flight Data)

Figure 65 is the power spectral density of a wind square flight flown on 5/28/2021. The blue is the the power spectral density of the raw MHP velocity data. The frequency decay of the data does follow the -5/3 law.

For validation that the -5/3 decay is due to atmospheric measurements and not probe noise a power spectral density was ran on 5HP data when the probe was covered. This will remove any atmospheric measurements from the probe and isolate the data to just probe noise.



Figure 66: Spectral Power Density with -5/3 Law (Covered Probe)

Figure 66 is the result of running the covered probe through a power spectral density. The slope of the data is mostly flat near zero. This enforces that Figure 65 is representative of atmospheric turbulence.

#### 5.1.4 Mesonet Station

On 4/2/21 the Nano Talon was flown at the Marshall Mesonet Station. This flight campaign was the reason for the switch to a hybrid probe due to probes breaking in the field. However, the flight data shown here is taken from a full 3D manufactured probe. This is the only flight presented in this paper that is done with a fully 3D manufactured probe. The performance was not significantly different between the probe though. The winds for the test were very high around 10m/s the flight speed of the aircraft. This allowed for a unique flight where the aircraft was able to maintain position and point into the wind.



Figure 67: Wind Speed Comparison to Marshall Mesonet Site 4/2/2021

Figure 67 is the first orbit of the Nano Talon. The random lines on the right side are from the Nano Talon sitting in place and moving side to side. This allowed for a good comparison to the Mesonet station.



Figure 68: Wind Speed Comparison to Marshall Mesonet Site 4/2/2021



Figure 69: Wind Direction Comparison to Marshall Mesonet Site 4/2/2021

Figure 68 and 69 are the airspeed and direction graphs respectively. The direction graph

starts very far from the Mesonet tower. As the flight moves on the value gets closer. This may be a factor of the estimator struggling while stationary. Near the end of the flight the aircraft begins maneuvering more which would help the estimator become more accurate.

Table 9: Mesonet Probe Data Compare

	Mesonet	Probe
Wind Avg (m/s)	8.775	9.64
Angle Avg (deg)	186.5	112.08

The speed comparison is close. The Nano Talon showing a slightly higher speed could be due to the slight difference in altitude. The Nano Talon was flown at 50m the Mesonet tower only has readings up to 10m.

#### 5.1.5 Comparison Flights

The following sections will cover flights where the MHP was flown against other sensors. The sensors flown against are the Young 92000, the SKB-1000 mounted Trisonica, and Young 5103. The details on these instruments can be found in table ??. The analysis of the Trisonica on the quad-rotor is found in table 6.

#### All 3 Sensors (Orbit Flights 5/3/2021)

On May third 2021 the SKB-1000 with the Trisonica with an iMET-XQ2 and the Nimbus with a 5-HP and the TPH sensor suite were flown. The test provided a good test for the system performance in dynamic atmosphere. The flights were performed at the Oklahoma State University Unmanned Aircraft Flight Station, East of Stillwater Oklahoma. On the date of the flight a small cold front moved through the area between 1500CST and 1800CST.



Figure 70: 1500 Surface Map



Figure 71: 1800 Surface Map

The Nimbus was flown in a completely autonomous fashion. The SKB-1000 was flown in QLoiter, this mode holds altitude and position. The two flights were flown at round 15:00 CST and 16:00CST aligning with the passing of the cold front and shown in surface charts shown above.



Figure 72: Waypoint Parameters for Nimbus Flight 5/3/2021

The waypoint set up for the Nimbus is shown in Figure 72. The flight was done with 4 personnel two pilots (James Brenner and Kyle Hickman) and two ground station monitors (Eric Abele and Andrew Walsh). The ground station monitors communicated the altitude of each vehicle to each other and the pilots. The ground station operator of the Nimbus would

command way point changes (changes in loiter altitude) after the SKB-1000 had reached the current altitude of the Nimbus and 2 full orbit of the Nimbus had been completed. Not ever altitude achieved a full loiters due to battery restraints. Many of the descending loiters were limited to 1 orbit.



Figure 73: Flight Profile For Nimbus and Skateboard 5/3/2021

Figure 73 shows the actual flight of the Nimbus (Red) and SKB-1000 (Green). This track is for flight 1, however flight 2 follows the same pattern. This pattern was designed to be able to average data at each altitude. Getting 2 full orbits allowed for a the 5HP to fully measure the wind at each altitude since the orbit period must be averaged out from the measurements. The level orbits also provide a time for the 2 vehicles to ensure they are at the same altitude.



Figure 74: Young, Trisonica, and MHP Wind Speed Plot 5/3/2021 - Flight 1

Figure 74 is a plot of the wind speed measurement of all 3 sensors together. The spike at the beginning of the MHP data is from take off condition and is not included in the analysis.

	Young	Trisonica	Multi-hole Probe
Average Speed (m/s)	5.319	7.596	6.629
Variance	0.897	2.279	0.219
Skewness	0.01	-1.01	0.156
Standard Deviation	0.947	1.51	0.468
Kurtosis	2.826	4.546	3.404

Table 10: 5-3-2021 Flight 1 Sensor Data Analysis

Table 10 displays some different data analysis parameters of each of the data sets. The average speed of the Young is significantly lower than the other 2 sensors because it is ground based and due to the atmospheric boundary layer the airspeed increases as altitude increases. The ground sensor is a good control to have reliable sensor that gives an idea what the atmospheric trends are. The variance are decent indicators of how gusty the wind is. On this first flight wind was had significant gust conditions. It is noting that the MHP variance and standard deviation low due to the significant averaging required in the data processing.

	Correlation Coefficient	RMSE $(m/s)$
Probe Young	0.116	1.714
Probe Trisonica	0.3884	1.676
Trisonica Young	0.0522	4.349

Table 11: 5-3-2021 Flight 1 Sensor Comparison

Table 11 compares the correlation and root mean square error for all three sensors. The Young shows nearly no relationship with either sensor. This is expected since it is measuring the conditions on the ground. The MHP and Trisonica have a moderate relationship. The MHP delay on gust is most likely what causes this lower correlation coefficient.



Figure 75: Trisonica, and MHP Wind Angle Plot 5/3/2021-Flight 1

Figure 75 is a plot of the angle through the whole flight of the vertical profile.

Table 12	: Flight	1 Angle	Data	Analysis
----------	----------	---------	------	----------

	Trisonica	Multi-hole Probe
Average Angle (deg)	312.7	278.9
Variance	512.6	932.6
Skewness	-10	-0.01
Standard Deviation	22.83	30.53
Kurtosis	137.6	1.79

Table 13: Flight 1 Sensor Angle Compare

	Correlation Coefficient	RMSE (deg)
Probe Trisonica	0.06	50.32

Table 12 and 13 is the data from the MHP comparison to the Trisonica. There is significant differences between the two measurements. Not many conclusions can be drawn from this data.



Figure 76: Young, Trisonica, and MHP Wind Speed Plot 5/3/2021-Flight 2

	Young	Trisonica	Multi-hole Probe
Average Speed (m/s)	4.017	6.084	5.857
Variance	0.508	0.781	0.129
Skewness	0.929	-0.756	-0.066
Standard Deviation	0.713	0.884	0.36
Kurtosis	4.69	5.217	2.101

Table 14: 5-3-2021 Flight 2 Data Analysis

Flight 2 had much calmer conditions that flight one. This is illustrated by the variance and the standard deviation for all of the sensors being much lower. This is indicative of more stable and less gusty conditions. As such the MHP performs much better in comparison to the Trisonica.

	Correlation Coefficient	RMSE $(m/s)$
Probe Young	0.19	2.061
Probe Trisonica	0.36	0.85
Trisonica Young	0.028	3.86

Table 15: 5-3-2021 Flight 2 Sensor Comparison

The correlation coefficients stayed relatively constant with the first flight. The root mean square between the MHP was reduced to  $\pm .85$ m/s this is most likely due to the more constant wind.



Figure 77: Trisonica, and MHP Wind Direction Plot 5/3/2021-Flight 2

Figure 77 is the wind angle derived from the MHP plotted against the Trisonica angle. There are sections where the probe and Trisonica align well.

	Trisonica	Multi-hole Probe
Average Angle (deg)	323.4	307.7
Variance	137	322
Skewness	0.24	-1.04
Standard Deviation	11.71	17.94
Kurtosis	2.62	3.74

Table 16: Flight 2 Data Analysis

	Correlation Coefficient	RMSE $(m/s)$
Probe Trisonica	0.41	23.15

Table 17: Flight 2 Sensor Data Compare

Table 16 and 17 are the data from the MHP comparison to the Trisonica. In general they get the same cardinal direction for the wind. A specific angle would not be able to be confirmed from this data.

# Wind Square Flights

Winds squares are flown with the Nano Talon. These flight test were performed with the Trisonica on the western side of the square near the midpoint. This allows for two different tests. The first is a wind square average. The second is pulling data from the straight next to the Trisonica this will allow for a comparison of non-averaged data.



Figure 78: Way-points For Wind Square Calibration Flights Revision For 5-28-2021

Figure 78 are the way-points for the flight. The optimal way point radius for the Nano Talon was around 125ft. This gave the Nano Talon time to full turn and prevented significant

### overshoots.



Figure 79: Wind Square Flight Path 5-28-2021

Figure 80 is the flight path of the Nano Talon. The red dot on the west side is the location of the SKB-1000 with the Trisonica. Both vehicles were positioned at the same height. The SKB-1000 was left to hover while the Nano Talon flow the square.



Figure 80: Wind Speed Comparisons 5-28-2021

The conditions for the flights were very low. The winds were variable ranging between 0-6 m/s. This provided a good low wind test opportunity. Due to the wind speed being significantly lower than the aircraft speed any system errors are magnified. This makes the aircraft state removal more significant and will highlight any errors.

	Trisonica	Multi-hole Probe
Average Speed (m/s)	2.69	3.77
Variance	2.2	0.26
Skewness	0.31	1.71
Standard Deviation	1.48	0.51
Kurtosis	0.18	0.24

Table 18: 5-28 Trisonic MHP Wind Square Data Analysis

The Trisonica does show high standard deviation and variances indicating gusts. The MHP does read a much higher speed than the Trisonica. Across multiple flight test the when the wind drops below 3m/s the MHP measurements stall and stop lowering. The cause of this

has not been isolated and will need further investigation.

	Correlation Coefficient	RMSE $(m/s)$
Probe Trisonica	0.69	1.62

Table 19: 5-28-21 Trisonica MHP Wind Square Comparison

The wind square shows very strong correlation to the Trisonica data. This may be attributed to the reduced averaging of the data in the processing of wind squares.



Figure 81: Straight Section Wind Speed Comparisons 5-28-2021

The wind square straights allow for comparison of non averaged MHP wind data. Since the probe is traveling in steady level flight there is no turns that need to be averaged out. Figure 81 is one of these straight sections.

	Trisonica	Multi-hole Probe
Average Speed (m/s)	4.13	4.38
Variance	1.19	1.93
Skewness	-0.26	-0.44
Standard Deviation	1.09	1.93
Kurtosis	2.49	1.98

Table 20: 5-28-21 Trisonica MHP Wind Square Steady Level Analysis

The analysis of this data provides a good insight to the probes ability to measure micro scale atmospheric phenomena. Since the gust are not averaged out in this data it has the closest comparisons to the Trisonica measurements.

Table 21: 5-28-21 Trisonica MHP Wind Square Steady Level Sensor Compare

	Correlation Coefficient	RMSE $(m/s)$
Probe Trisonica	0.70	1.47

The steady level comparison shows a very strong correlation between data sets. The root mean square error still remains large.

#### 5.1.6 Transect Flights

On June 10th 2021 the Nimbus was used to fly transects at the Choctaw Flight Field near Daisy Oklahoma. The terrain in the area is hilly with dense forest. The flights were flown at 200ft AGL. With a 2 mile straight line flight path. The Young 92000 was posted on a 40ft tower mid flight path.



Figure 82: Choctaw Flight Path

Transects, similar to the wind square can provide a good opportunity for for direct comparison between probes since no averaging is required. Figure 82 is the flight path of the nimbus. The Nimbus was able to hold a straight line for a long period. Holding quality steady level flight can be key to good data.



Figure 83: Choctaw Wind Plots

Figure 83 shows the comparison between MHP and the Young. This data snippet is taken from when the Nimbus is flying over the tower during steady level flight.

	Young	Multi-hole Probe
Average Speed (m/s)	5.72	10.49
Variance	0.47	1.91
Skewness	0.037	-0.35
Standard Deviation	0.69	1.38
Kurtosis	2.1	2.06

Table 22: Choctaw Transects Steady Level Upwind Analysis

As expected the average wind speed observed by the probe is higher than the Young due to the altitude change.

Table 23: Choctaw Transects Steady Level Upwind Sensor Compare

	Correlation Coefficient	RMSE $(m/s)$
Probe Young	0.312	5.862

The correlation of the two data sets does show moderate correlation.

### 5.2 Use For a Holistic Meteorological Application

One of the major motivations for a cost effective MHP system that can be integrated with a UAS is for weather monitoring and research. A valuable tool as discussed in the Chapter 2, the literature review, are atmospheric soundings. UAS can assist in low level soundings to supplement data in areas that are under sampled by balloons.

#### Weather Front Flights (5/27/21)

The flights used for the weather soundings are the 5/27/21 flights at the Oklahoma state Unmanned Aircraft Flight Station. The flights were flown on a day with a high severe weather potential. In this section the multi-hole probe measurements are combined with temperature, pressure, and humidity data from I-met XQ2 sensors. These flights were flown with the Nimbus, Nano Talon, and SKB-1000. The SKB-1000 flew vertical profiles while the Nano Talon did orbits around it. The Nimbus performed transects down range.



Figure 84: Wind Speed Sounding

Figure 86 is the MHP and Trisonica wind speeds plotted against altitude. This is a good visualization of the atmospheric boundary layer.



Figure 85 is a sounding created from the MHP and I-met data. This data can be shared with meteorologist to help forecast. This holistic application is a demonstration of what the probe can be used for. The blue and black lines are the temperature profiles with the blue barbs on the right side corresponding to wind speed and direction.

# CHAPTER VI

## CONCLUSION

#### 6.1 Five Hole Probe Comparison to other sensors

The results from the calibrations can now be compared to other known systems. The RMSE and the calculated Gaussian error as calculated in Chapter 5 are compared to the aircraft from Table 3.

Vehicle	Probe Type	Speed Accuracy (m/s)	Accuracy Method
M2AV	5HP	0.5	$1\sigma$
Blackswift S0	5HP	0.48	N/A
ALADINA	5HP	0.5	Gaussian error
wind RPA	5HP	1.1	Gaussian error
Manta	5HP	0.045	$1\sigma$
Scan Eagle	5HP	0.045	$1\sigma$
Nano Talon	5HP	0.85	Gaussian error
Nimbus	5HP	1.26	Gaussian error
Nimbus/Nano Talon	5HP	1.12	RMSE

Table 24: System Comparison

Table 24 is a comparison between the two different probe accuracy methods and other multihole probe systems. The Gaussian error method and the RMSE from flight tests show similar accuracies. While the accuracy is not on the level desired for atmospheric measurements it has reached a level that is on par with other systems. The root mean square error is an
average of the root mean square over all of the comparison flights between the Trisonica and probe. The final error of the system is around 1.12m/s though this value can vary based on conditions and flight profile. It is important to note that Gaussian Error propagation gives a good baseline for the uncertainty of the system and is not an accuracy. Averaging the two systems together gives an uncertainty of  $\pm 1.05$ .

#### 6.2 Probe Challenges

One of the main goals of this project was to use a fully additive manufactured probe for UAS measurements. As the 3D printed probe was implemented it was discovered that it was not suited for the rigors of flight operations. The issues with the additive manufactured probes was not an accuracy or design issue. The main issue was a material issue. Finding a material and printer combination that can reach the desired strength and durability while maintaining the required accuracy will be a challenge.

Multiple different materials have been attempted for this project. The two Formlabs materials were most reliable for have clear passage of air through the tubes, as discussed this took extra work to ensure. Specially ordered Nylon prints all had clogged holes, although these prints showed potential for able to be cleared as it was not a total blockage.

A hybrid probe is a good solution until a reliable printer/material selection can be found. The hybrid probe still achieves the goal of keeping costs down. While it does take more man hours to manufacture that cost would most likely be less than having to re-calibrate after breaking a 3D printed probe.

The single most important influence to a probe's accuracy is the quality of the sensors used for vehicle state estimation. This is illustrated well by the Scan Eagle and Manta in table 3. These two systems are able to achieve significantly better accuracies than other systems because of the more accurate state estimation. If cost is the primary objective Pix-hawks are suitable for accurate average wind measurements. If more detailed measurement is needed using a more accurate system to account for aircraft dynamics will be required.

### 6.3 Future Work

This project can serve as the foundation for different endeavors. One of the major issues that still linger is finding the best material to print the probe out of. Based off the findings of field tests a hybrid probe is still recommended. While the body of the probe and tip will work well 3D printed with the right material, finding a material strong enough to make the tube attachment points will be difficult.

Re-evaluation of the pressure sensors and data acquisition system should be conducted. The flight speed of the vehicles does not encompass much of the  $\pm 1$ psi range of the pressure sensors. A sensor with reduced range would reduce the quantization error and be better suited for this application. The Teensy based data acquisition system has some inherit limitations. A system with a better ADC converter and higher clock rates could be used to increase the resolution and frequency of the data. A properly designed noise filter paired with the system would also help improve results.

One of the major shortcomings of this project has been wind angle accuracy. The wind angle measurements should be further evaluated and refined. The probe should be flown against more known data sources to evaluate this such as tall weather towers. For soundings and ABL research, a higher altitude limit would allow for a more complete data set. A higher altitude waiver for UAS weather operations should be pursued.

A more holistic system design should also be pursued. Improvements like integration of Pixhawk error and variance logs to flag bad data should be investigated. A method to make the identification of steady level flight a non manual process would also improve the ease of data processing. As the system moves beyond the prototype phase better integration with the aircraft should be done. Many of the hardware issues could be avoided if the board and pitot tube had better integration.

#### 6.4 Conclusion

With preliminary flight testing and calibration of the probe done some performance measures can be presented. While these should not be taken as true for every flight due to the variability mounting and atmospheric conditions the parameters can serve as a good baseline.

Table 25: Preliminary System Characteristics

Accuracy $(m/s)$	1.12
Uncertainty $(m/s)$	1.05
Response Time (ms)	20
Max Frequency (Hz)	50

The MHP system developed is a good starting point for a reliable and accurate system for atmospheric research. Table 25 shows the preliminary characteristics of the 5HP system. While further analysis of the probe will need to be performed the results from this paper demonstrates a cheap fixed wing wind sensing platform. While challenges with a fully 3D manufactured probe were found the use of one is presented to be feasible. An alternative hybrid probe design that followed the same sizing of the 3D manufactured one was also developed. The probes tend to be of similar accuracy of other probes developed. The probes are also demonstrated in taking data that can support meteorological objectives such as ABL research.



Figure 86: ABL Wind Sounding

#### REFERENCES

- Measuring the wind vector using the autonomous mini aerial vehicle M2 AV, Journal of Atmospheric and Oceanic Technology 25 (2008), no. 11, 1969–1982.
- [2] Safe autonomous flight environment (SAFE50) for the notional last "50 ft" of operation of "55 lb" class of UAS, AIAA Information Systems-AIAA Infotech at Aerospace, 2017 (2017), no. January, 1–7.
- [3] Features: 3d ultrasonic anemometer: Trisonica mini: Wind sensor, https:// anemoment.com/features/, May 2021.
- [4] *fft*, https://www.mathworks.com/help/matlab/ref/fft.html, May 2021.
- [5] Instructions: Response model 92000 weather transmitter, https://www.fondriest. com/pdf/rm\_young\_92000\_manual.pdf, May 2021.
- [6] Instructions: Wind monitor model 05103, https://www.youngusa.com/wp-content/ uploads/2021/02/05103-90N.pdf, May 2021.
- [7] Material data sheet: Durable, https://formlabs-media.formlabs.com/datasheets/
   1801084-TDS-ENUS-OP.pdf, May 2021.
- [8] Rawinsondes, https://www.sciencedirect.com/topics/ earth-and-planetary-sciences/rawinsondes, May 2021.
- [9] S0 data sheet\_v2, https://www.https:bst.aero/wp-content/uploads/2021/06/
   VTOL-S0-Data-Sheet.pdf, May 2021.

- [10] Skew-t parameters, https://www.weather.gov/source/zhu/ZHU\_Training\_Page/ convective\_parameters/skewt/skewtinfo.html, May 2021.
- [11] Loiy Al-Ghussain and Sean Bailey, An Approach to Minimize Aircraft Motion Bias in Multi-Hole Probe Wind Measurements made by Small Unmanned Aerial Systems, Atmospheric Measurement Techniques Discussions (2020), 1–19.
- [12] A. Alberigi Quaranta, G. C. Aprilesi, G. De Cicco, and A. Taroni, A microprocessor based, three axes, ultrasonic anemometer, Journal of Physics E: Scientific Instruments 18 (1984), no. 5, 384–387.
- [13] D. N. Axford, On the accuracy of wind measurements using an inertial platform in an aircraft, and an example of a measurement of the vertical mesostructure of the atmosphere, Journal of Applied Meteorology 7 (1968), no. 4, 645–666.
- [14] Solmoz K Azertash-Namin, Evaluation of low-cost multi-hole probes for atmospheric boundary layer investigation, Masters Thesis, Oklahoma State University (2017).
- [15] Lindsay Barbieri, Stephan T. Kral, Sean C.C. Bailey, Amy E. Frazier, Jamey D. Jacob, Joachim Reuder, David Brus, Phillip B. Chilson, Christopher Crick, Carrick Detweiler, Abhiram Doddi, Jack Elston, Hosein Foroutan, Javier González-Rocha, Brian R. Greene, Marcelo I. Guzman, Adam L. Houston, Ashraful Islam, Osku Kemppinen, Dale Lawrence, Elizabeth A. Pillar-Little, Shane D. Ross, Michael P. Sama, David G. Schmale, Travis J. Schuyler, Ajay Shankar, Suzanne W. Smith, Sean Waugh, Cory Dixon, Steve Borenstein, and Gijs De Boer, Intercomparison of small unmanned aircraft system (sUAS) measurements for atmospheric science during the LAPSE-RATE campaign, Sensors (Switzerland) 19 (2019), no. 9.

- [16] Konrad Bärfuss, Falk Pätzold, Barbara Altstädter, Endres Kathe, Stefan Nowak, Lutz Bretschneider, Ulf Bestmann, and Astrid Lampert, New setup of the UAS ALADINA for measuring boundary layer properties, atmospheric particles and solar radiation, Atmosphere 9 (2018), no. 1.
- [17] Radiance Calmer, Gregory C. Roberts, Jana Preissler, Kevin J. Sanchez, Solène Derrien, and Colin O'Dowd, Vertical wind velocity measurements using a five-hole probe with remotely piloted aircraft to study aerosol-cloud interactions, Atmospheric Measurement Techniques 11 (2018), no. 5, 2583–2599.
- [18] Genevieve Comte-bellot, *Hot-wire anemometry*, (1976), no. Thomas 1920, 209–231.
- [19] Geoffrey W Donnell, Jordan A Feight, Nate Lannan, and Jamey D Jacob, Wind characterization using onboard imu of suas, 2018 Atmospheric Flight Mechanics Conference, 2018, p. 2986.
- [20] Thomas J Dudzinski and Lloyd N Krause, Flow-direction measurement with fixedposition probes, National Aeronautics and Space Administration, 1969.
- [21] K. E. Garman, K. A. Hill, P. Wyss, M. Carlsen, J. R. Zimmerman, B. H. Stirm, T. Q. Carney, R. Santini, and P. B. Shepson, An airborne and wind tunnel evaluation of a wind turbulence measurement system for aircraft-based flux measurements, Journal of Atmospheric and Oceanic Technology 23 (2006), no. 12, 1696–1708.
- [22] G David Huffman, DC Rabe, and ND Poti, Flow direction probes from a theoretical and experimental point of view, Journal of Physics E: Scientific Instruments 13 (1980), no. 7, 751.
- [23] Jamey D. Jacob, Phillip B. Chilson, Adam L. Houston, and Suzanne Weaver Smith, Considerations for atmospheric measurements with small unmanned aircraft systems, Atmosphere 9 (2018), no. 7, 1–16.

- [24] T. G. Konrad, M. L. Hill, J. R. Rowland, and J. H. Meyer, A Small, Radio-Controlled Aircraft As A Platform for Meteorlogical Sensors, APL Technical Digest (1970), no. December, 11–19.
- [25] L. Kristensen, The cup anemometer and other exciting instruments, J. Appl. Meteorol.
   J. Atmos. Oceanic Technol. J. Appl. Meteorol. J. Atmos. Oceanic Technol 615 (1993),
   no. April, 1–83.
- [26] L Kristensen, Cups, props and vanes, **766** (1994), no. August, 1–37.
- [27] D H Lenschow, Bulletin No . 23 MEASUREMENT TECHNIQUES: AIR MOTION SENSING, (1989), no. 23, 1–29.
- [28] Stephanie Mayer, Gautier Hattenberger, Pascal Brisset, Marius Jonassen, and Joachim Reuder, A 'no-flow-sensor' wind estimation algorithm for unmanned aerial systems, International Journal of Micro Air Vehicles 4 (2012), no. 1, 15–29.
- [29] Renee A. McPherson, Christopher A. Fiebrich, Kenneth C. Crawford, Ronald L. Elliott, James R. Kilby, David L. Grimsley, Janet E. Martinez, Jeffrey B. Basara, Bradley G. Illston, Dale A. Morris, Kevin A. Kloesel, Stephen J. Stadler, Andrea D. Melvin, Albert J. Sutherland, Himanshu Shrivastava, J. D. Carlson, J. Michael Wolfinbarger, Jared P. Bostic, and David B. Demko, *Statewide monitoring of the mesoscale environment: A technical update on the Oklahoma Mesonet*, Journal of Atmospheric and Oceanic Technology **24** (2007), no. 3, 301–321.
- [30] Tevis Nichols, Brian Argrow, and Derek Kingston, Error sensitivity analysis of small UAS wind-sensing systems, AIAA Information Systems-AIAA Infotech at Aerospace, 2017 (2017), no. January, 1–15.
- [31] C. F.R. Nowack, Improved calibration method for a five-hole spherical Pitot probe, Journal of Physics E: Scientific Instruments 3 (1970), no. 1, 21–26.

- [32] S. Prudden, A. Fisher, M. Marino, A. Mohamed, S. Watkins, and G Wild, Measuring wind with Small Unmanned Aircraft Systems, Journal of Wind Engineering and Industrial Aerodynamics 176 (2018), no. December 2017, 197–210.
- [33] Alexander Rautenberg, Martin S. Graf, Norman Wildmann, Andreas Platis, and Jens Bange, *Reviewing wind measurement approaches for fixed-wing unmanned aircraft*, Atmosphere 9 (2018), no. 11, 1–24.
- [34] Benjamin D. Reineman, Luc Lenain, Nicholas M. Statom, and Kendall K. Melville, Development and testing of instrumentation for UAV-based flux measurements within terrestrial and marine atmospheric boundary layers, Journal of Atmospheric and Oceanic Technology 30 (2013), no. 7, 1295–1319.
- [35] N. Sitaram and K. Srikanth, Effect of chamfer angle on the calibration curves of the five hole probes, International Journal of Rotating Machinery (2014).
- [36] Y. Telionis, D. Yang and O. Rediniotis, Recent developments in multi-hole probe (mhp) technology, International Congress of Mechanical Engineering (2009).
- [37] A. L. Treaster and A. M. Yocum, The calibration and application of five-hole probes, ISA transactions (1979).
- [38] NOAA US Department of Commerce, Weather balloon / upper air observations, https: //www.weather.gov/chs/upperair, Oct 2018.
- [39] M Zeiger, L Chalmeta, and D Telionis, Tip geometry effects on calibration and performance of seven-hole probes, 29th AIAA, Fluid Dynamics Conference, 1998, p. 2810.

# APPENDICES

All codes used can be found on Github: https://github.com/LeviRoss2/USRI\_ Sensor\_Suite

## Codes

1	00	Pixhawk DFL Analyzer V4.8.10
2	00	Created by: Levi Ross   levi.ross@okstate.edu
3	00	Edited by: Kyle Hickman   kthickm@okstate.edu
4	00	Unmanned Systems Research Institute
5	00	Creation Date: 11/17/2019
6	00	Last Modified: 06/20/2021 - Kyle
7	00	
8	00	New features:
9	00	* Replay _Parsed files
10	00	$\star$ No longer need to set Ardupilot type, auto-selected from MSG1 log
11	00	$\star$ Added Wind Rose to view estimated wind speed data form Pixhawk
12	00	$\star$ Added PlotMergedData codeset, allowing for simultaneous or series
13	00	data plots
14	00	$\star$ Added custom .bin -> .mat converter, eliminating the need for
15	00	Mission Planner
16	00	* Complete CopterSonde support
17	00	$\star$ Added met data averaging for vertical profiles or stationary hovers
18	00	$\star$ Added Animation of side-by-side T, P, and H graphs against GPS data
19	00	* 3D-isometric viewing of Multi-Aircraft animation

<ul> <li>a &amp; Added altitude parsing to allow for multi-aircraft 3D plots</li> <li>a &amp; Brought in NKF1 parameters for GPS NED velocities and aircraft</li> <li>a attitude states</li> <li>b SHP data gets averaged down to Kalman Filter Rates</li> <li>c SHP data has aircraft states removed for 3D wind vector</li> <li>b Added plots for Corrected SHP data</li> <li>c Added an FTT function and Option for SHP data analysis</li> <li>c Added ability to average SHP data for windspeed (period based off</li> <li>c FTT peak, must be selected)</li> <li>c FTT peak, must be selected)</li> <li>c General Updates:</li> <li>c Consolidated DFL.New and DFL.Old SHP plots and output file saving</li> <li>c S = Fright referencing Temp and Humidty values in Sensor Suite data</li> <li>c Fixed Figure 5 plot (changed from CTUN(:,2) to 3)</li> <li>c = Added aniWid initialization for multi vehicle animation (Fixed</li> <li>t = Yixed NKF2 issues</li> <li>c = Added aniWid initialization for multi vehicle animation (Fixed</li> <li>c = Added the Suite Size</li> <li>c = Added the Suite Size</li> <li>c = Added aniWid initialization for multi vehicle animation (Fixed</li> <li>f = Cude Asks for SHP when just TPH is "Yes"</li> <li>s = Imet Parser date time column call is wrong</li> <li>s = Nemoving IMU states from Shp data</li> <li>s = Nemoving IMU states from Shp data</li> <li>s = Nemoving IMU states from Shp data</li> <li>s = Neing in Mesonet Data and/or other ground sources</li> </ul>			
<pre>a % * Brought in NKF1 parameters for GFS NED velocities and aircraft 2 % attitude states 2 % * 5HP data gets averaged down to Kalman Filter Rates 2 % * 5HP data has aircraft states removed for 3D wind vector 2 % * Added plots for Corrected 5HP data 2 % * Added an FFT function and Option for 5HP data analysis 2 % * Added ability to average 5HP data for windspeed (period based off 2 % PFT peak, must be selected) 2 % * Fully implemented 5hp algorithm 3 % 3 % General Updates: 3 % * Consolidated DFL.New and DFL.Old 5HP plots and output file saving 3 % 4 % Bug Fixes: 3 % * Consolidated DFL.New and DFL.Old 5HP plots and output file saving 3 % 4 % Bug Fixes: 3 % * Properly referencing Temp and Humidty values in Sensor Suite data 3 % * Explicitly defined all figures so there's no chance to overwrite 3 % * Temperature 1 is Plotting time v time - Fixed 3 % * Added animVid initialization for multi vehicle animation (Fixed 4 % * Orde Asks for 5HP when just TPH is "Yes" 4 % Code Asks for 5HP when just TPH is "Yes" 4 % * Code Asks for 5HP when just TPH is "Yes" 4 % * Projects In Progress/To Re Done: 4 % * Removing IMU states from 5hp data 5 % * Include Anemometer Parser Options 5 % * Bring in Mesonet Data and/or other ground sources 5 % * Code Sug Parser Cate and/or other ground sources 5 % * Code Asks for 5HP when just PH is "Yes" 5 % * Removing IMU states from 5hp data 5 % * Projects In Progress/To Re Done: 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU states from 5hp data 5 % * Removing IMU stat</pre>	20	00	* Added altitude parsing to allow for multi-aircraft 3D plots
<pre>s attitude states s % . 5HP data gets averaged down to Kalman Filter Rates 4 % . 5HP data gets averaged down to Kalman Filter Rates 4 % . 5HP data has aircraft states removed for 3D wind vector 5 % . Added plots for Corrected 5HP data 5 % . Added an FFT function and Option for 5HP data analysis 5 % . Added ability to average 5HP data for windspeed (period based off 5 % . Fully implemented 5hp algorithm 5 % 5 % Ceneral Updates: 5 % . Consolidated DFL.New and DFL.Old 5HP plots and output file saving 5 % 5 % Consolidated DFL.New and DFL.Old 5HP plots and output file saving 5 % 5 % Properly referencing Temp and Humidty values in Sensor Suite data 5 % Explicitly defined all figures so there's no chance to overwrite 5 % Added animVid initialization for multi vehicle animation (Fixed 5 % Added animVid initialization for multi vehicle animation (Fixed 5 % Added animVid initialization for multi vehicle animation (Fixed 5 % Code Asks for 5HP when just TPH is "Yes" 6 % Immet Parser date time column call is wrong 7 % 5 % Removing IMU states from 5hp data 7 % Removing IMU states from 5hp data 7 % Ramoving IMU states from 5hp data 7 % Pring in Mesonet Data and/or other ground sources 7 % Pring in Mesonet Data and/or other ground sources 7 % Pring Parser date to the form form form form form form for the form form for the form form for the form form form form form form form form</pre>	21	00	$\star$ Brought in NKF1 parameters for GPS NED velocities and aircraft
<pre>123 % · SHP data gets averaged down to Kalman Filter Rates 124 % · SHP data has aircraft states removed for 3D wind vector 125 % · Added plots for Corrected 5HP data 126 % · Added an FFT function and Option for 5HP data analysis 127 % · Added ability to average 5HP data for windspeed (period based off 128 % FFT peak, must be selected) 129 % · Fully implemented 5hp algorithm 120 % 13 % General Updates: 13 % · Consolidated DFL New and DFL.Old 5HP plots and output file saving 13 % 14 % Bug Fixes: 15 * · Consolidated DFL New and DFL.Old 5HP plots and output file saving 13 % 14 % Bug Fixes: 15 * Properly referencing Temp and Humidty values in Sensor Suite data 14 % · Explicitly defined all figures so there's no chance to overwrite 15 * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 15 % · Temperature 1 is Plotting time v time - Fixed 19 % · Added animVid initialization for multi vehicle animation (Fixed 10 % Triple animate) 11 % · Fixed NKF2 issues 12 % 13 % General Notes/Unresolved Bugs: 14 % Ouadcopter NKF1 imports untested 15 % · Code Asks for 5HP when just TPH is "Yes" 16 % · Imet Parser date time column call is wrong 17 % 18 % Projects In Progress/To Be Done: 19 % · Removing IMU states from Shp data 10 % · Include Anemometer Parser Options 11 % · Fring in Mesonet Data and/or other ground sources 12 % · Data for the sources 13 % · Details and/or other ground sources 14 % · Data for the sources 14 % · Data for the source of the sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Data for the source of the sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground sources 14 % · Details A and/or other ground source</pre>	22	00	attitude states
<pre>24 % · SHP data has aircraft states removed for 3D wind vector 25 % · Added plots for Corrected SHP data 26 % · Added an FFT function and Option for SHP data analysis 27 % · Added ability to average SHP data for windspeed (period based off 28 % FFT peak, must be selected) 29 % · Fully implemented 5hp algorithm 30 % 31 % General Updates: 32 % · Consolidated DFL.New and DFL.Old SHP plots and output file saving 33 % 34 % Bug Fixes: 35 * Properly referencing Temp and Humidty values in Sensor Suite data 36 * Explicitly defined all figures so there's no chance to overwrite 37 % · Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % · Temperature 1 is Plotting time v time - Fixed 39 % · Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % · Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % Quadcopter NKF1 imports untested 45 % · Code Asks for SHP when just TPH is "Yes" 46 % · Inet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % · Removing IMU states from Shp data 50 % · Include Anemometer Parser Options 51 % · Bring in Mesonet Data and/or other ground sources</pre>	23	00	$\star$ 5HP data gets averaged down to Kalman Filter Rates
<pre>s \$ Added plots for Corrected 5HP data s Added an FFT function and Option for 5HP data analysis s Added ability to average 5HP data for windspeed (period based off s FFT peak, must be selected) s Consolidated DFL.New and DFL.Old 5HP plots and output file saving s FFT peak, must be selected by algorithm s Fixed Figure 5 plot (changed from CTUN(:,2) to 3) s Fixed Figure 5 plot (changed from CTUN(:,2) to 3) s Fixed Figure 5 plot (changed from CTUN(:,2) to 3) s Added animVid initialization for multi vehicle animation (Fixed s Added animVid initialization for multi vehicle animation (Fixed s Fixed NKF2 issues s Conscilent NKF2 issues s Code Asks for 5HP when just TFH is "Yes" s Immet Parser date time column call is wrong s Removing IMU states from 5hp data s Fingi in Mesonet Data and/or other ground sources s Fingi in Mesonet Data and/or other ground sources</pre>	24	00	$\star$ 5HP data has aircraft states removed for 3D wind vector
<pre>29 % * Added an FFT function and Option for 5HP data analysis 27 % * Added ability to average 5HP data for windspeed (period based off 28 % FFT peak, must be selected) 29 % * Fully implemented 5hp algorithm 30 % 31 % General Updates: 32 % * Consolidated DFL.New and DFL.Old 5HP plots and output file saving 33 % 34 % Bug Fixes: 35 % * Properly referencing Temp and Humidty values in Sensor Suite data 36 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Added animVid initialization for multi vehicle animation (Fixed 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % Quadcopter NKF1 imports untested 45 % Code Asks for 5HP when just TPH is "Yes" 46 % Imet Parser date time column Call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 40 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources 53 % * Details of the surface of the surfa</pre>	25	00	* Added plots for Corrected 5HP data
<pre>127 % * Added ability to average 5HP data for windspeed (period based off 128 % FFT peak, must be selected) 129 % * Fully implemented 5hp algorithm 130 % 131 % General Updates: 132 % * Consolidated DFL.New and DFL.Old 5HP plots and output file saving 133 % 134 % Bug Fixes: 138 % * Properly referencing Temp and Humidty values in Sensor Suite data 136 % * Explicitly defined all figures so there's no chance to overwrite 137 % * Added animVid initialization for multi vehicle animation (Fixed 138 % * Temperature 1 is Plotting time v time - Fixed 139 % * Added animVid initialization for multi vehicle animation (Fixed 14 % Fixed NKF2 issues 14 % Quadcopter NKF1 imports untested 15 % Code Asks for 5HP when just TPH is "Yes" 16 % Imet Parser date time column call is wrong 17 % 18 % Removing IMU states from 5hp data 19 % * Bring in Mesonet Data and/or other ground sources</pre>	26	00	$\star$ Added an FFT function and Option for 5HP data analysis
<pre>22 % FFT peak, must be selected) 22 % * Fully implemented 5hp algorithm 30 % 31 % General Updates: 32 % * Consolidated DFL_New and DFL_Old 5HP plots and output file saving 33 % 34 % Bug Fixes: 38 % * Properly referencing Temp and Humidty values in Sensor Suite data 38 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % Code Asks for 5HP when just TPH is "Yes" 46 % Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % Removing IMU states from 5hp data 50 % * Bring in Mesonet Data and/or other ground sources</pre>	27	00	$\star$ Added ability to average 5HP data for windspeed (period based off
<pre>20 % * Fully implemented 5hp algorithm 30 % 31 % General Updates: 32 % * Consolidated DFL.New and DFL.Old 5HP plots and output file saving 33 % 34 % Bug Fixes: 35 % * Properly referencing Temp and Humidty values in Sensor Suite data 36 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	28	%	FFT peak, must be selected)
30       %         31       % General Updates:         32       % Consolidated DFLNew and DFLOId 5HP plots and output file saving         33       %         34       % Bug Fixes:         35       % Properly referencing Temp and Humidty values in Sensor Suite data         36       % Properly referencing Temp and Humidty values in Sensor Suite data         36       % Properly referencing Temp and Humidty values in Sensor Suite data         36       % Properly referencing Temp and Humidty values in Sensor Suite data         37       % Explicitly defined all figures so there's no chance to overwrite         37       % Fixed Figure 5 plot (changed from CTUN(:,2) to 3)         38       % Temperature 1 is Plotting time v time - Fixed         39       % Added animVid initialization for multi vehicle animation (Fixed         40       % Triple animate)         41       % Fixed NKF2 issues         42       %         43       % General Notes/Unresolved Bugs:         44       % Quadcopter NKF1 imports untested         45       % Code Asks for 5HP when just TPH is "Yes"         46       % Imet Parser date time column call is wrong         47       %         48       % Projects In Progress/To Be Done:         49       % Removing IM	29	0 0	* Fully implemented 5hp algorithm
<pre>s General Updates: s &amp; Consolidated DFL.New and DFL.Old 5HP plots and output file saving s &amp; s &amp; Properly referencing Temp and Humidty values in Sensor Suite data s &amp; Properly referencing Temp and Humidty values in Sensor Suite data s &amp; Properly referencing Temp and Humidty values in Sensor Suite data s &amp; Explicitly defined all figures so there's no chance to overwrite s &amp; Explicitly defined all figures so there's no chance to overwrite s &amp; Temperature 1 is Plotting time v time - Fixed s &amp; Added animVid initialization for multi vehicle animation (Fixed s &amp; Added animVid initialization for multi vehicle animation (Fixed s &amp; Triple animate) s &amp; Fixed NKF2 issues s &amp; General Notes/Unresolved Bugs: s &amp; Quadcopter NKF1 imports untested s &amp; Code Asks for 5HP when just TPH is "Yes" s &amp; Timet Parser date time column call is wrong s &amp; Removing IMU states from 5hp data s &amp; Removing IMU states from 5hp data s &amp; Include Anemometer Parser Options s &amp; Bring in Mesonet Data and/or other ground sources</pre>	30	0 0	
* * Consolidated DFL_New and DFL_Old SHP plots and output file saving * * Consolidated DFL_New and DFL_Old SHP plots and output file saving * * Bug Fixes: * * Properly referencing Temp and Humidty values in Sensor Suite data * * Explicitly defined all figures so there's no chance to overwrite * * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) * * Temperature 1 is Plotting time v time - Fixed * * Added animVid initialization for multi vehicle animation (Fixed * * Added animVid initialization for multi vehicle animation (Fixed * * Triple animate) * * Fixed NKF2 issues * * General Notes/Unresolved Bugs: * * Quadcopter NKF1 imports untested * * Code Asks for 5HP when just TPH is "Yes" * * Code Asks for 5HP when just TPH is wrong * * Imet Parser date time column call is wrong * * * * Removing IMU states from Shp data * * Include Anemometer Parser Options * * Bring in Mesonet Data and/or other ground sources	31	0 0	General Updates:
<pre>33 % 34 % Bug Fixes: 35 % * Properly referencing Temp and Humidty values in Sensor Suite data 36 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	32	%	$\star$ Consolidated DFL_New and DFL_Old 5HP plots and output file saving
<pre>34 % Bug Fixes: 35 % * Properly referencing Temp and Humidty values in Sensor Suite data 36 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	33	00	
35 % * Properly referencing Temp and Humidty values in Sensor Suite data 36 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources	34	0 0	Bug Fixes:
<pre>36 % * Explicitly defined all figures so there's no chance to overwrite 37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	35	%	* Properly referencing Temp and Humidty values in Sensor Suite data
<pre>37 % * Fixed Figure 5 plot (changed from CTUN(:,2) to 3) 38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	36	0 0	* Explicitly defined all figures so there's no chance to overwrite
<pre>38 % * Temperature 1 is Plotting time v time - Fixed 39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	37	0 0	* Fixed Figure 5 plot (changed from CTUN(:,2) to 3)
<pre>39 % * Added animVid initialization for multi vehicle animation (Fixed 40 % Triple animate) 41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	38	0 0	* Temperature 1 is Plotting time v time - Fixed
<ul> <li>40 % Triple animate)</li> <li>41 % * Fixed NKF2 issues</li> <li>42 %</li> <li>43 % General Notes/Unresolved Bugs:</li> <li>44 % * Quadcopter NKF1 imports untested</li> <li>45 % * Code Asks for 5HP when just TPH is "Yes"</li> <li>46 % * Imet Parser date time column call is wrong</li> <li>47 %</li> <li>48 % Projects In Progress/To Be Done:</li> <li>49 % * Removing IMU states from 5hp data</li> <li>50 % * Include Anemometer Parser Options</li> <li>51 % * Bring in Mesonet Data and/or other ground sources</li> </ul>	39	%	$\star$ Added animVid initialization for multi vehicle animation (Fixed
<pre>41 % * Fixed NKF2 issues 42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	40	%	Triple animate)
<pre>42 % 43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	41	0 0	* Fixed NKF2 issues
43 % General Notes/Unresolved Bugs: 44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources	42	0 0	
<pre>44 % * Quadcopter NKF1 imports untested 45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	43	0/0	General Notes/Unresolved Bugs:
<pre>45 % * Code Asks for 5HP when just TPH is "Yes" 46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	44	0 0	* Quadcopter NKF1 imports untested
<pre>46 % * Imet Parser date time column call is wrong 47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	45	0 0	* Code Asks for 5HP when just TPH is "Yes"
<pre>47 % 48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	46	0 0	* Imet Parser date time column call is wrong
<pre>48 % Projects In Progress/To Be Done: 49 % * Removing IMU states from 5hp data 50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	47	0/0	
<ul> <li>49 % * Removing IMU states from 5hp data</li> <li>50 % * Include Anemometer Parser Options</li> <li>51 % * Bring in Mesonet Data and/or other ground sources</li> </ul>	48	0 0	Projects In Progress/To Be Done:
<pre>50 % * Include Anemometer Parser Options 51 % * Bring in Mesonet Data and/or other ground sources</pre>	49	0 0	* Removing IMU states from 5hp data
51 % * Bring in Mesonet Data and/or other ground sources	50	0/0	* Include Anemometer Parser Options
	51	0 0	* Bring in Mesonet Data and/or other ground sources

```
* Comparison between multiple data sources
52 %
53
  8
54 % Current Fig Count:
  8
       * Single: 16
55
       * Multi: 1
  8
56
57
58 %% Clear All Data
59 close all
60 clear all
61 clc
62
  %% Initialize User-Defined Content
63
64 display('Initializing user-defined variable space.');
65
66 thrMinPWM = 1100;
                                % Default, scales THR% plots
_{67} thrMaxPWM = 1900;
                                 % Default, scales THR% plots
68
  Single_Multi = 'Single';
69
70
  % Single Controls
71
r2 tripleAnimateTPH = 'No'; % Side-by-side T, P, and H plots against ...
      GPS position
73 graphToggle = 'No';
                                % Show plots of parsed data
74 animateToggle = 'No';
                                % Flight animation based on GPS and Alt(AGL)
75 recordAnimation = 'No';
                                % Record animation for future playback
76 hoverProfile = 'None';
                                % Profile (vertical) or Hover ...
      (stationary) averaging
77 stateSpace = 'No';
                       % Pixhawk recorded SS variable output ...
      (new file)
78 iMetValue = 'No';
                                % Load iMet data (XQ1 - small | XQ2 - large)
79 mhpValue = 'Yes';
                                % Load 5HP/TPH data (use 5HP)
```

```
80 mhpAvg = 'Yes';
                               % Lets user set averaging bounds for ...
     wind speed averages (not implemented)
s1 tphValue = 'No';
                              % Load 5HP/TPH data (use TPH)
82 overlay = 'No';
                  % Show iMet & 5HP/TPH alongside Pixhawk ...
     data plots
                               % Output GPS data as individual file ...
83 gpsOut = 'No';
     (lat, long, alt)
84 sensorOut = 'No';
                               % Output parsed sensor data (iMet ...
     seperate from 5HP/TPH)
ss attitudeOut = 'No';
                              % Output parsed attitude data
86 sensorCompare = 'No'; % Show iMet, 5HP/TPH, and Pixhawk ...
      atmoshperic sensors on same plots
87 dflNewOld = 'Unknown';
ss arduPilotType = 'Default';
89 pitchToggle = 'No';
                               % TIA-Specific
90 throttleToggle = 'No';
                               % TIA-Specific
91 aircraft = 'N/A';
                               % TIA-Specific
92 tvToggle = 'No';
                              % TIA-Specific
93 indvToggle = 'No'; % TIA-Specific
94
95 % Only used for Single Animation
96 animateSpeed = 2;
                              % Overall speed
97 animateHeadSize = 3;
                               % Icon size
98 animateTailWidth = 2;
                               % Width of tail
99 animateTailLength = 10000; % Length of tail
100 animateFPS = 30;
101 animateHeadSize = animateHeadSize + 5;
102 plotTitle = '';
103
104 % Only used for Multi Animation
105 animation = 'Yes'; % Default: Yes | Yes: turn on animation | No: ...
     turn Off
```

```
106 simultaneous = 'Yes'; % Default: Yes | Yes: Files in same time ...
      frame | No: Files in concurrent time frames
                            % Default: Yes | Yes: view in 3D space | No: ...
107 isometric = 'No';
      view in 2D with Satellite map overlay
108 azimuthAngle = 15;
                             % Default: 45 | Rotation angle in x-y plane ...
      to view the 3D plot (positive values -> CCW rotation, negative CW)
109 elevationAngle = 45;
                            % Default: 15 | Elevation angle above the X-Y ...
      plane (90 is top down X-Y plot, 0 is looking from the X-Y plane ...
      depending on azimuthAngle)
                             % Default: 5 | High numbers: increase ...
110 iterationSkip = 5;
      animation speed, reduce smoothness. If increasing, decrease ...
      animFrameRate to have useful video time
111 animFrameRate = 30; % Default: 30 | High numbers: increase ...
      animation speed, increase smoothness. If increasing, decrease ...
      iteration_skip to have useful video time
112
113 %% Main Code
114 display('Executing user-defined operations.');
   if(strcmpi(Single_Multi, 'Single'))
115
       %% Gather All Files For Parsing
116
117
       % Have user browse for a file, from a specified "starting folder."
118
       % For convenience in browsing, set a starting folder from which to ...
119
          browse.
       % Start in the current folder.
120
       startingFolderDFL = pwd;
121
       % Get the name of the file that the user wants to use.
122
       defaultFileNameDFL = fullfile(startingFolderDFL, {'*.bin;*.mat'});
123
       [baseFileNameDFL, folderDFL] = uigetfile(defaultFileNameDFL, ...
124
           'Select a Pixhawk DFL file (.bin or .mat only)');
125
       startingFolderDFL = folderDFL;
126
127
       if baseFileNameDFL == 0
```

```
% User clicked the Cancel button.
128
129
            return;
       end
130
131
       fullInputMatFileNameDFL = fullfile(folderDFL, baseFileNameDFL);
132
       %% Pixhawk bin->mat Converter
133
       if(strcmpi(baseFileNameDFL((end-2):end), 'bin'))
134
            DFL_NewOld = 'New';
135
136
            rawDFL = Ardupilog(fullInputMatFileNameDFL);
137
            %% Check if CopterSonde or Standard DFL file
138
            Rover = 0;
139
            Copter = 0;
140
            QuadPlane = 0;
141
            Plane = 0;
142
143
            %% ArduPilotType Parser (MSG1)
144
            for i=1:length(rawDFL.MSG.LineNo);
145
                MSG1{1,i}{1,1} = 'MSG';
146
                MSG1{1,i}{2,1} = rawDFL.MSG.TimeUS(i);
147
                MSG1{1,i}{3,1} = rawDFL.MSG.Message(...
148
                     i,1:length(rawDFL.MSG.Message(1,:)));
149
                if(contains(rawDFL.MSG.Message(i,1:length(...
150
                         rawDFL.MSG.Message(1,:))), 'ArduCopter'))
151
                     Copter = 1;
152
                elseif(contains(rawDFL.MSG.Message(i,1:length(...
153
                         rawDFL.MSG.Message(1,:)), 'QuadPlane'))
154
                     QuadPlane = 1;
155
                elseif(contains(rawDFL.MSG.Message(i,1:length(...
156
                         rawDFL.MSG.Message(1,:)), 'ArduRover'))
157
                    Rover = 1;
158
159
                elseif(contains(rawDFL.MSG.Message(i,1:length(...
```

```
rawDFL.MSG.Message(1,:)), 'ArduPlane'))
160
                     Plane = 1;
161
                end
162
            end
163
164
            if(Rover == 1)
165
                arduPilotType = 'ArduRover';
166
            elseif(Copter == 1)
167
                arduPilotType = 'ArduCopter';
168
            elseif(QuadPlane == 1)
169
                arduPilotType = 'QuadPlane';
170
            elseif(Plane == 1)
171
                arduPilotType = 'FixedWing';
172
173
            end
174
            if(strcmpi(arduPilotType, 'FixedWing') |...
175
                     strcmpi(arduPilotType, 'QuadPlane'))
176
177
                %% GPS
178
                GPS(:,1) = rawDFL.GPS.LineNo;
179
                GPS(:, 2) = ...
180
                    fix(rawDFL.GPS.TimeS/rawDFL.GPS.fieldMultipliers.TimeUS);
                GPS(:,3) = rawDFL.GPS.Status;
181
                GPS(:,4) = rawDFL.GPS.GMS;
182
                GPS(:,5) = rawDFL.GPS.GWk;
183
                GPS(:,6) = rawDFL.GPS.NSats;
184
                GPS(:,7) = rawDFL.GPS.HDop;
185
                GPS(:,8) = rawDFL.GPS.Lat;
186
                GPS(:,9) = rawDFL.GPS.Lng;
187
                GPS(:,10) = rawDFL.GPS.Alt;
188
                GPS(:,11) = rawDFL.GPS.Spd;
189
190
                GPS(:,12) = rawDFL.GPS.GCrs;
```

i l	
191	<pre>GPS(:,13) = rawDFL.GPS.VZ;</pre>
192	<pre>GPS(:,14) = rawDFL.GPS.Yaw;</pre>
193	<pre>GPS(:,15) = rawDFL.GPS.U;</pre>
194	<pre>GPS(:,16) = rawDFL.GPS.DatenumUTC;</pre>
195	<pre>GPS_label = {'LineNo';'TimeUS';'Status';'GMS';'GWk';</pre>
196	'NSats';'HDop';'Lat';'Lng';'Alt';'Spd';'GCrs';
197	<pre>'VZ';'Yaw';'U';'DatenumUTC'};</pre>
198	%% ATT
199	<pre>ATT(:,1) = rawDFL.ATT.LineNo;</pre>
200	ATT(:, 2) = fix(
201	<pre>rawDFL.ATT.TimeS/rawDFL.ATT.fieldMultipliers.TimeUS);</pre>
202	<pre>ATT(:,3) = rawDFL.ATT.DesRoll;</pre>
203	<pre>ATT(:,4) = rawDFL.ATT.Roll;</pre>
204	<pre>ATT(:,5) = rawDFL.ATT.DesPitch;</pre>
205	<pre>ATT(:,6) = rawDFL.ATT.Pitch;</pre>
206	<pre>ATT(:,7) = rawDFL.ATT.DesYaw;</pre>
207	<pre>ATT(:,8) = rawDFL.ATT.Yaw;</pre>
208	<pre>ATT(:,9) = rawDFL.ATT.ErrRP;</pre>
209	<pre>ATT(:,10) = rawDFL.ATT.ErrYaw;</pre>
210	<pre>ATT(:,11) = rawDFL.ATT.DatenumUTC;</pre>
211	<pre>ATT_label = {'LineNo';'TimeUS';'DesRoll';'Roll';'DesPitch';</pre>
212	'Pitch';'DesYaw';'Yaw';'ErrRP';'ErrYaw';'DatenumUTC'};
213	%% BARO
214	<pre>BARO(:,1) = rawDFL.BARO.LineNo;</pre>
215	BARO(:, 2) = fix(
216	rawDFL.BARO.TimeS/rawDFL.BARO.fieldMultipliers.TimeUS);
217	<pre>BARO(:,3) = rawDFL.BARO.Alt;</pre>
218	<pre>BARO(:,4) = rawDFL.BARO.Press;</pre>
219	<pre>BARO(:,5) = rawDFL.BARO.Temp;</pre>
220	<pre>BARO(:,6) = rawDFL.BARO.CRt;</pre>
221	<pre>BARO(:,7) = rawDFL.BARO.SMS;</pre>
222	<pre>BARO(:,8) = rawDFL.BARO.Offset;</pre>

1	
223	<pre>BARO(:,9) = rawDFL.BARO.GndTemp;</pre>
224	<pre>BARO(:,10) = rawDFL.BARO.Health;</pre>
225	<pre>BARO(:,11) = rawDFL.BARO.DatenumUTC;</pre>
226	BARO_label = {'LineNo';'TimeUS';'Alt';'Press';'Temp';'CRt';
227	'SMS';'Offset';'GndTemp';'Health';'DatenumUTC'};
228	%% CTUN
229	<pre>CTUN(:,1) = rawDFL.CTUN.LineNo;</pre>
230	CTUN(:, 2) = fix(
231	<pre>rawDFL.CTUN.TimeS/rawDFL.CTUN.fieldMultipliers.TimeUS);</pre>
232	<pre>CTUN(:,3) = rawDFL.CTUN.NavRoll;</pre>
233	CTUN(:,4) = rawDFL.CTUN.Roll;
234	CTUN(:,5) = rawDFL.CTUN.NavPitch;
235	CTUN(:,6) = rawDFL.CTUN.Pitch;
236	CTUN(:,7) = rawDFL.CTUN.ThrOut;
237	CTUN(:,8) = rawDFL.CTUN.RdrOut;
238	CTUN(:,9) = rawDFL.CTUN.ThrDem;
239	CTUN(:,10) = rawDFL.CTUN.Aspd;
240	<pre>CTUN(:,11) = rawDFL.CTUN.DatenumUTC;</pre>
241	CTUN_label = {'LineNo';'TimeUS';'NavRoll';'Roll';
242	'NavPitch';'Pitch';'ThrOut';'RdrOut';'ThrDem';
243	'Aspd'; 'DatenumUTC'};
244	%% NKF2
245	try
246	<pre>NKF2(:,1) = rawDFL.NKF2.LineNo;</pre>
247	NKF2(:,2) = fix(
248	<pre>rawDFL.NKF2.TimeS/rawDFL.NKF2.fieldMultipliers.TimeUS);</pre>
249	<pre>NKF2(:,3) = rawDFL.NKF2.AZbias;</pre>
250	<pre>NKF2(:,4) = rawDFL.NKF2.GSX;</pre>
251	<pre>NKF2(:,5) = rawDFL.NKF2.GSY;</pre>
252	<pre>NKF2(:,6) = rawDFL.NKF2.GSZ;</pre>
253	NKF2(:,7) = rawDFL.NKF2.VWN;
254	<pre>NKF2(:,8) = rawDFL.NKF2.VWE;</pre>

255       NKF2(:,9) = rawDFL.NKF2.MN;         266       NKF2(:,10) = rawDFL.NKF2.ME;         277       NKF2(:,11) = rawDFL.NKF2.MD;         288       NKF2(:,12) = rawDFL.NKF2.MX;         299       NKF2(:,13) = rawDFL.NKF2.MX;         200       NKF2(:,14) = rawDFL.NKF2.MX;         201       NKF2(:,15) = rawDFL.NKF2.MZ;         202       NKF2(:,16) = rawDFL.NKF2.MI;         203       NKF2(:,16) = rawDFL.NKF2.MI;         204       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';         205       'MI';'DatenumUTC'};         206       'MI';'DatenumUTC'};         207       catch         208       NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';         205       'MI';'DatenumUTC'};         206       'MI';'DatenumUTC'};         207       catch         208       NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';         209       'GS2';'WN';'WE';'MN';'ME';'MD';'MX';'MY';         209       'GS2';'WN';'WE';'MN';'ME';'MD';'MX';'MY';         200       'KKF2(:,10) = rawDFL.XKF2.LineNo;         211       end         212       % XKF2(:,2) = fix (         213       try         214       XKF2(:,2) = fix (	1	
256       NKF2(:,10) = rawDFL.NKF2.ME;         257       NKF2(:,11) = rawDFL.NKF2.MC;         258       NKF2(:,12) = rawDFL.NKF2.MC;         259       NKF2(:,13) = rawDFL.NKF2.MC;         260       NKF2(:,14) = rawDFL.NKF2.MC;         261       NKF2(:,15) = rawDFL.NKF2.MC;         262       NKF2(:,16) = rawDFL.NKF2.MC;         263       NKF2(:,16) = rawDFL.NKF2.DatenumUTC;         264       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';         265       'MI';'DatenumUTC'};         266       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';         266       'MI';'DatenumUTC'};         267       catch         268       NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';         269       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         271       end         272       % XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AX;         279       XKF2	255	NKF2(:,9) = rawDFL.NKF2.MN;
237       NKF2(:,11) = rawDFL.NKF2.MD;         238       NKF2(:,12) = rawDFL.NKF2.MX;         259       NKF2(:,13) = rawDFL.NKF2.MY;         260       NKF2(:,14) = rawDFL.NKF2.MZ;         261       NKF2(:,15) = rawDFL.NKF2.MI;         262       NKF2(:,16) = rawDFL.NKF2.DatenumUTC;         263       NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';         264       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';         265       'MI';'DatenumUTC'};         266       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'GSX';'GSY';         266       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GS2';'VWN';'VWE';'MN';'ME';'MD';'MX';'M';         271       end         272       %% XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AZ;	256	<pre>NKF2(:,10) = rawDFL.NKF2.ME;</pre>
238       NKF2(:,12) = rawDFL.NKF2.MX;         259       NKF2(:,13) = rawDFL.NKF2.MY;         260       NKF2(:,14) = rawDFL.NKF2.MZ;         261       NKF2(:,15) = rawDFL.NKF2.MI;         262       NKF2(:,16) = rawDFL.NKF2.DatenumUTC;         263       NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';         264       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';         265       'MI';'DatenumUTC'};         266       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         266       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         266       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'MZ';'MI';'DatenumUTC'};         271       end         272       %* XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AX;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	257	<pre>NKF2(:,11) = rawDFL.NKF2.MD;</pre>
<pre>239 NKF2(:,13) = rawDFL.NKF2.MY; 800 NKF2(:,14) = rawDFL.NKF2.MZ; 811 NKF2(:,15) = rawDFL.NKF2.MI; 822 NKF2(:,16) = rawDFL.NKF2.DatenumUTC; 823 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 826 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ'; 826 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'NZ'; 827 catch 828 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 829 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 820 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 820 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 820 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'VWE';'MN';'ME';'MD';'MX';'MY'; 829 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 820 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 821 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 822 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 823 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 824 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 825 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 826 'GSZ';'WN';'WE';'MN';'ME';'MD';'MX';'MY'; 827 'GSZ';'WN';'WE';'MN';'ME';'MD';'MZ';'MY';'NZ';'M'';'MZ';'MX';'MY';'NZ';'M'';'MZ';'NZ';'MZ';'MZ';'MZ';'MZ</pre>	258	<pre>NKF2(:,12) = rawDFL.NKF2.MX;</pre>
200       NKF2(:,14) = rawDFL.NKF2.MZ;         201       NKF2(:,15) = rawDFL.NKF2.MI;         202       NKF2(:,16) = rawDFL.NKF2.DatenumUTC;         203       NKF2.label = {'LineNo'; 'TimeUS'; 'AZbias'; 'GSX'; 'GSY';         204       'GSZ'; 'VWN'; 'VWE'; 'MN'; 'ME'; 'MD'; 'MX'; 'MY'; 'MZ';         205       'MI'; 'DatenumUTC'};         206       267         268       NKF2.label = {'LineNo'; 'TimeUS'; 'AZbias'; 'GSX'; 'GSY';         269       'GSZ'; 'VWN'; 'VWE'; 'MN'; 'ME'; 'MD'; 'MX'; 'MY';         270       'GSZ'; 'VWN'; 'VWE'; 'MN'; 'ME'; 'MD'; 'MX'; 'MY';         270       'GSZ'; 'VWN'; 'DatenumUTC'};         271       end         272       %% XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AY;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	259	<pre>NKF2(:,13) = rawDFL.NKF2.MY;</pre>
<pre>201 NKF2(:,15) = rawDFL.NKF2.MI; 202 NKF2(:,16) = rawDFL.NKF2.DatenumUTC; 203 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSY'; 204 'GSZ';'VWN';'VWE';'MN';'MZ';'MY';'MZ'; 205 'MI';'DatenumUTC'}; 206 catch 207 catch 208 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSY'; 200 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 210 'MZ';'MI';'DatenumUTC'}; 221 end 222 %% XKF2 223 try 224 XKF2(:,1) = rawDFL.XKF2.LineNo; 225 XKF2(:,2) = fix( 226 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 227 XKF2(:,3) = rawDFL.XKF2.AX; 228 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	260	<pre>NKF2(:,14) = rawDFL.NKF2.MZ;</pre>
<pre>262 NKF2(:,16) = rawDFL.NKF2.DatenumUTC; 263 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 264 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ'; 265 'MI';'DatenumUTC'}; 266 267 catch 268 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 269 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	261	<pre>NKF2(:,15) = rawDFL.NKF2.MI;</pre>
<pre>263 NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 264 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ'; 265 'MI';'DatenumUTC'}; 266  267 catch 268 NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 269 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	262	<pre>NKF2(:,16) = rawDFL.NKF2.DatenumUTC;</pre>
264       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';         265       'MI';'DatenumUTC'};         266       267         268       NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';         269       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';         270       'MZ';'MI';'DatenumUTC'};         271       end         272       %% XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AZ;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	263	NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';
<pre>265 'MI';'DatenumUTC'}; 266 267 catch 268 NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSY'; 269 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	264	'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';
<pre>206 267 catch 268 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 269 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	265	<pre>'MI'; 'DatenumUTC'};</pre>
<pre>267 catch 268 NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 269 'GSZ';'VWN';'WE';'MN';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	266	
<pre>268 NKF2.label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY'; 269 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	267	catch
<pre>269 'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY'; 270 'MZ';'MI';'DatenumUTC'}; 271 end 272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AZ;</pre>	268	<pre>NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';</pre>
270       'MZ';'MI';'DatenumUTC'};         271       end         272       %% XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AZ;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	269	'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';
271       end         272       %% XKF2         273       try         274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AZ;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	270	<pre>'MZ';'MI';'DatenumUTC'};</pre>
<pre>272 %% XKF2 273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AY; 279 XKF2(:,5) = rawDFL.XKF2.AZ;</pre>	271	end
<pre>273 try 274 XKF2(:,1) = rawDFL.XKF2.LineNo; 275 XKF2(:,2) = fix( 276 rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS); 277 XKF2(:,3) = rawDFL.XKF2.AX; 278 XKF2(:,4) = rawDFL.XKF2.AY; 279 XKF2(:,5) = rawDFL.XKF2.AZ;</pre>	272	%% XKF2
274       XKF2(:,1) = rawDFL.XKF2.LineNo;         275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AY;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	273	try
275       XKF2(:,2) = fix(         276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AY;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	274	<pre>XKF2(:,1) = rawDFL.XKF2.LineNo;</pre>
276       rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);         277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AY;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	275	XKF2(:,2) = fix(
277       XKF2(:,3) = rawDFL.XKF2.AX;         278       XKF2(:,4) = rawDFL.XKF2.AY;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	276	<pre>rawDFL.NKF2.TimeS/rawDFL.XKF2.fieldMultipliers.TimeUS);</pre>
278       XKF2(:,4) = rawDFL.XKF2.AY;         279       XKF2(:,5) = rawDFL.XKF2.AZ;	277	<pre>XKF2(:,3) = rawDFL.XKF2.AX;</pre>
279 XKF2(:,5) = rawDFL.XKF2.AZ;	278	<pre>XKF2(:,4) = rawDFL.XKF2.AY;</pre>
	279	<pre>XKF2(:,5) = rawDFL.XKF2.AZ;</pre>
280 XKF2(:,6) = rawDFL.XKF2.VWN;	280	<pre>XKF2(:,6) = rawDFL.XKF2.VWN;</pre>
$XKF2(\cdot, 7) = rawder. XKF2 VWE \cdot$	281	<pre>XKF2(:,7) = rawDFL.XKF2.VWE;</pre>
	282	<pre>XKF2(:,8) = rawDFL.XKF2.MN;</pre>
282  XKF2(:,8) = rawDFL.XKF2.MN;	283	<pre>XKF2(:,9) = rawDFL.XKF2.ME;</pre>
281       XKF2(:,9)       FdwDFL.XKF2.WN;         282       XKF2(:,9)       = rawDFL.XKF2.ME;         283       XKF2(:,9)       = rawDFL.XKF2.ME;	284	<pre>XKF2(:,10) = rawDFL.XKF2.MD;</pre>
281       XKF2(:,7)       FdwDFL.XKF2.WD;         282       XKF2(:,8)       = rawDFL.XKF2.MN;         283       XKF2(:,9)       = rawDFL.XKF2.ME;         284       XKF2(:,10)       = rawDFL.XKF2.MD;	285	<pre>XKF2(:,11) = rawDFL.XKF2.MX;</pre>
281       XKF2(:,7)       FdwDFL.XKF2.WD;         282       XKF2(:,8)       = rawDFL.XKF2.MN;         283       XKF2(:,9)       = rawDFL.XKF2.ME;         284       XKF2(:,10)       = rawDFL.XKF2.MD;         285       XKF2(:,11)       = rawDFL.XKF2.MX;	286	<pre>XKF2(:,12) = rawDFL.XKF2.MY;</pre>
XKF2(:,6) = rawDFL, XKF2, VWN:	279	XKF2(:,5) = rawDFL.XKF2.AZ; XKF2(:,6) = rawDFL.XKF2.VWN:
	279	XKF2(:,5) = rawDFL.XKF2.AZ;
	279	XKF2(:,5) = rawDFL.XKF2.AZ;
	279	<pre>XKF2(:,5) = rawDFL.XKF2.AZ;</pre>
	279	<pre>XKF2(:,5) = rawDFL.XKF2.AZ;</pre>
	279	<pre>XKF2(:,5) = rawDFL.XKF2.AZ;</pre>
	279	XKF2(:,5) = rawDFL.XKF2.AZ;
	279	XKF2(:,5) = rawDFL.XKF2.AZ;
	279	<pre>XKF2(:,5) = rawDFL.XKF2.AZ;</pre>
	279	XKF2(:,5) = rawDFL.XKF2.AZ;
	279	XKF2(:,5) = rawDFL.XKF2.AZ;
$XKE2(\cdot, 6) = rawDEL XKE2 WWN.$	280	$XKE2(\cdot, 6) = rawdel XKE2 VWN.$
$XKE2(\cdot, 6) = rawDEL XKE2 WWN.$	219	$XKE2(.,5) = r_{2}WDE1 XKE2 VWN.$
280 XKF2(:,6) = rawDFL.XKF2.VWN;	280	<pre>XKF2(:,6) = rawDFL.XKF2.VWN;</pre>
280 XKF2(:,6) = rawDFL.XKF2.VWN;	280	<pre>XKF2(:,6) = rawDFL.XKF2.VWN;</pre>
280 XKF2(:,6) = rawDFL.XKF2.VWN;	280	<pre>XKF2(:,6) = rawDFL.XKF2.VWN;</pre>
$280 \qquad \qquad \text{AKF2}(:,0) = 1 \text{ awdrL}.\text{AKF2}.\text{VWN};$	280	$\operatorname{ARF2}(:, 0) = \operatorname{IdwDFL}.\operatorname{ARF2}.\operatorname{VWN};$
$XKF2(\cdot, 7) = rawder. XKF2 VWE \cdot$	281	<pre>XKF2(:,7) = rawDFL.XKF2.VWE;</pre>
(991) $(1)$ $(1$	281	XKF2(:,/) = rawDFL.XKF2.VWE;
	282	<pre>XKF2(:,8) = rawDFL.XKF2.MN;</pre>
XKF2(:,8) = rawDFL.XKF2.MN;	202	$XKE3(\cdot \theta) = x = m DEI XKE3 ME \cdot$
282  XKF2(:,8) = rawDFL.XKF2.MN;	283	XKF2(:,9) = rawDFL.XKF2.ME;
281       XKF2(:,9)       FawDFL.XKF2.MN;         283       XKF2(:,9)       = rawDFL.XKF2.ME;	284	<pre>XKF2(:,10) = rawDFL.XKF2.MD;</pre>
281       XKF2(:,7)       FdwDFL.XKF2.WD;         282       XKF2(:,8)       = rawDFL.XKF2.MN;         283       XKF2(:,9)       = rawDFL.XKF2.ME;         284       XKF2(:,10)       = rawDFL.XKF2.MD;	285	<pre>XKF2(:,11) = rawDFL.XKF2.MX;</pre>
281       XKF2(:,7)       FdwDFL.XKF2.WE;         282       XKF2(:,8)       = rawDFL.XKF2.ME;         283       XKF2(:,9)       = rawDFL.XKF2.ME;         284       XKF2(:,10)       = rawDFL.XKF2.MD;         285       XKF2(:,11)       = rawDFL.XKF2.MX;	286	<pre>XKF2(:,12) = rawDFL.XKF2.MY;</pre>
281       XKF2(:,7)       FdwDFL.XKF2.WE;         282       XKF2(:,8)       = rawDFL.XKF2.ME;         283       XKF2(:,9)       = rawDFL.XKF2.ME;         284       XKF2(:,10)       = rawDFL.XKF2.MD;         285       XKF2(:,11)       = rawDFL.XKF2.MX;         286       XKF2(:,12)       = rawDFL.XKF2.MY;		

1	
287	<pre>XKF2(:,13) = rawDFL.XKF2.MZ;</pre>
288	<pre>XKF2(:,14) = rawDFL.XKF2.MI;</pre>
289	<pre>XKF2(:,15) = rawDFL.XKF2.DatenumUTC;</pre>
290	<pre>XKF2_label = {'LineNo';'TimeUS';'AX';'AY';'AZ';'VWN';</pre>
291	<pre>'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';'MI';'DatenumUTC'};</pre>
292	
293	catch
294	<pre>XKF2_label = {'LineNo';'TimeUS';'AX';'AY';'AZ';'VWN';</pre>
295	<pre>'VWE';'MN';'ME';'MD';'MX';'MY';'MZ';'MI';'DatenumUTC'};</pre>
296	end
297	%% RCOU
298	<pre>RCOU(:,1) = rawDFL.RCOU.LineNo;</pre>
299	RCOU(:, 2) = fix(
300	<pre>rawDFL.RCOU.TimeS/rawDFL.RCOU.fieldMultipliers.TimeUS);</pre>
301	<pre>RCOU(:,3) = rawDFL.RCOU.C1;</pre>
302	<pre>RCOU(:,4) = rawDFL.RCOU.C2;</pre>
303	<pre>RCOU(:,5) = rawDFL.RCOU.C3;</pre>
304	RCOU(:, 6) = rawDFL.RCOU.C4;
305	<pre>RCOU(:,7) = rawDFL.RCOU.C5;</pre>
306	<pre>RCOU(:,8) = rawDFL.RCOU.C6;</pre>
307	<pre>RCOU(:,9) = rawDFL.RCOU.C7;</pre>
308	RCOU(:, 10) = rawDFL.RCOU.C8;
309	<pre>RCOU(:,11) = rawDFL.RCOU.C9;</pre>
310	<pre>RCOU(:,12) = rawDFL.RCOU.C10;</pre>
311	<pre>RCOU(:,13) = rawDFL.RCOU.C11;</pre>
312	<pre>RCOU(:,14) = rawDFL.RCOU.C12;</pre>
313	<pre>RCOU(:,15) = rawDFL.RCOU.C13;</pre>
314	<pre>RCOU(:,16) = rawDFL.RCOU.C14;</pre>
315	<pre>RCOU(:,17) = rawDFL.RCOU.DatenumUTC;</pre>
316	RCOU_label =
	{'LineNo';'TimeUS';'C1';'C2';'C3';'C4';'C5';'C6';
317	'C7';'C8';'C9';'C10';'C11';'C12';'C13';'C14';'DatenumUTC'};

0	818	%% IMU
0	319	<pre>IMU(:,1) = rawDFL.IMU.LineNo;</pre>
¢,	320	IMU(:,2) = fix(
¢,	321	rawDFL.IMU.TimeS/rawDFL.IMU.fieldMultipliers.TimeUS);
e.	322	<pre>IMU(:,3) = rawDFL.IMU.GyrX;</pre>
ę	323	<pre>IMU(:,4) = rawDFL.IMU.GyrY;</pre>
~	324	<pre>IMU(:,5) = rawDFL.IMU.GyrZ;</pre>
ç	325	<pre>IMU(:,6) = rawDFL.IMU.AccX;</pre>
ç	326	<pre>IMU(:,7) = rawDFL.IMU.AccY;</pre>
ç	327	<pre>IMU(:,8) = rawDFL.IMU.AccZ;</pre>
e.	328	<pre>IMU(:,9) = rawDFL.IMU.EG;</pre>
ç	329	<pre>IMU(:,10) = rawDFL.IMU.EA;</pre>
¢,	330	<pre>IMU(:,11) = rawDFL.IMU.T;</pre>
ę	331	<pre>IMU(:,12) = rawDFL.IMU.GH;</pre>
ę	332	<pre>IMU(:,13) = rawDFL.IMU.AH;</pre>
e,	333	<pre>IMU(:,14) = rawDFL.IMU.GHz;</pre>
e,	334	<pre>IMU(:,15) = rawDFL.IMU.AHz;</pre>
63	335	<pre>IMU(:,16) = rawDFL.IMU.DatenumUTC;</pre>
ę	336	<pre>IMU_label = {'LineNo';'TimeUS';'GyrX';'GyrY';'GyrZ';'AccX';</pre>
e,	337	'AccY';'AccZ';'EG';'EA';'T';'GH';'AH';'GHz';'AHz';
e,	338	<pre>'DatenumUTC'};</pre>
e,	339	
e,	340	% Save as new file for integration with normal code
0	341	<pre>[¬, baseNameNoExtDFL, ¬] = fileparts(baseFileNameDFL);</pre>
0	342	<pre>baseFileName = sprintf('%s.mat', baseNameNoExtDFL);</pre>
e,	343	<pre>fullParsedMatFileName = fullfile(folderDFL, baseFileName);</pre>
¢,	344	% Save file with parsed data as the original filename plus
		the added portion
ę	345	<pre>save(fullParsedMatFileName,'GPS','GPS_label','ATT',</pre>
60	346	'ATT_label','BARO','BARO_label','CTUN','CTUN_label',
ç	347	'NKF2', 'NKF2_label', 'XKF2', 'XKF2_label', 'IMU',
ç	348	'IMU_label','RCOU','RCOU_label','MSG1');

349	
350 full	InputMatFileNameDFL = fullParsedMatFileName;
351	
352 elseif(s	strcmpi(arduPilotType,'ArduCopter')
353	<pre>strcmpi(arduPilotType,'CopterSonde'))</pre>
354	
355 try	
356	rawDFL.IMET;
357	<pre>arduPilotType = 'CopterSonde';</pre>
358	%% WIND
359	<pre>WIND(:,1) = rawDFL.WIND.LineNo;</pre>
360	<pre>WIND(:,2) = rawDFL.WIND.Time;</pre>
361	<pre>WIND(:,3) = rawDFL.WIND.wdir;</pre>
362	WIND(:,4) = rawDFL.WIND.wspeed;
363	WIND(:,5) = rawDFL.WIND.R13;
364	WIND(:,6) = rawDFL.WIND.R23;
365	WIND(:,7) = rawDFL.WIND.R33;
366	WIND(:,8) = rawDFL.WIND.DatenumUTC;
367	WIND_label = {'LineNo';'TimeUS';'wdir';'wspeed';
368	'R13';'R23';'R33';'DatenumUTC'};
369	%% IMET
370	<pre>IMET(:,1) = rawDFL.IMET.LineNo;</pre>
371	<pre>IMET(:,2) = rawDFL.IMET.Time;</pre>
372	<pre>IMET(:,3) = rawDFL.IMET.R1./100;</pre>
373	<pre>IMET(:,4) = rawDFL.IMET.R2./100;</pre>
374	<pre>IMET(:,5) = rawDFL.IMET.R3./100;</pre>
375	<pre>IMET(:,6) = rawDFL.IMET.R4./100;</pre>
376	<pre>IMET(:,7) = rawDFL.IMET.T1;</pre>
377	<pre>IMET(:,8) = rawDFL.IMET.T2;</pre>
378	<pre>IMET(:,9) = rawDFL.IMET.T3;</pre>
379	<pre>IMET(:,10) = rawDFL.IMET.T4;</pre>
380	<pre>IMET(:,11) = rawDFL.IMET.Hth1;</pre>

381	<pre>IMET(:,12) = rawDFL.IMET.Hth2;</pre>
382	<pre>IMET(:,13) = rawDFL.IMET.Hth3;</pre>
383	<pre>IMET(:,14) = rawDFL.IMET.Hth4;</pre>
384	<pre>IMET(:,15) = rawDFL.IMET.Fan;</pre>
385	<pre>IMET(:,16) = rawDFL.IMET.DatenumUTC;</pre>
386	<pre>IMET_label = {'LineNo';'TimeUS';'RH1';'RH2';'RH3';'RH4';</pre>
387	'T1';'T2';'T3';'T4';'Health1';'Health2';'Health3';
388	<pre>'Health4';'Fan';'DatenumUTC'};</pre>
389	end
390	
391	%% GPS
392	<pre>GPS(:,1) = rawDFL.GPS.LineNo;</pre>
393	<pre>GPS(:,2) = rawDFL.GPS.TimeUS;</pre>
394	<pre>GPS(:,3) = rawDFL.GPS.Status;</pre>
395	<pre>GPS(:,4) = rawDFL.GPS.GMS;</pre>
396	<pre>GPS(:,5) = rawDFL.GPS.GWk;</pre>
397	<pre>GPS(:,6) = rawDFL.GPS.NSats;</pre>
398	GPS(:,7) = rawDFL.GPS.HDop;
399	<pre>GPS(:,8) = rawDFL.GPS.Lat;</pre>
400	<pre>GPS(:,9) = rawDFL.GPS.Lng;</pre>
401	<pre>GPS(:,10) = rawDFL.GPS.Alt;</pre>
402	<pre>GPS(:,11) = rawDFL.GPS.Spd;</pre>
403	<pre>GPS(:,12) = rawDFL.GPS.GCrs;</pre>
404	<pre>GPS(:,13) = rawDFL.GPS.VZ;</pre>
405	<pre>GPS(:,14) = rawDFL.GPS.U;</pre>
406	<pre>GPS(:,15) = rawDFL.GPS.DatenumUTC;</pre>
407	<pre>GPS_label = {'LineNo';'TimeUS';'Status';'GMS';'GWk';'NSats';</pre>
408	'HDop';'Lat';'Lng';'Alt';'Spd';'GCrs';'VZ';'U';
409	<pre>'DatenumUTC'};</pre>
410	%% ATT
411	<pre>ATT(:,1) = rawDFL.ATT.LineNo;</pre>
412	<pre>ATT(:,2) = rawDFL.ATT.TimeUS;</pre>

413	<pre>ATT(:,3) = rawDFL.ATT.DesRoll;</pre>
414	<pre>ATT(:,4) = rawDFL.ATT.Roll;</pre>
415	<pre>ATT(:,5) = rawDFL.ATT.DesPitch;</pre>
416	<pre>ATT(:,6) = rawDFL.ATT.Pitch;</pre>
417	<pre>ATT(:,7) = rawDFL.ATT.DesYaw;</pre>
418	ATT(:,8) = rawDFL.ATT.Yaw;
419	<pre>ATT(:,9) = rawDFL.ATT.ErrRP;</pre>
420	<pre>ATT(:,10) = rawDFL.ATT.ErrYaw;</pre>
421	<pre>ATT(:,11) = rawDFL.ATT.DatenumUTC;</pre>
422	<pre>ATT_label = {'LineNo';'TimeUS';'DesRoll';'Roll';'DesPitch';</pre>
423	<pre>'Pitch';'DesYaw';'Yaw';'ErrRP';'ErrYaw';'DatenumUTC'};</pre>
424	%% BARO
425	<pre>BARO(:,1) = rawDFL.BARO.LineNo;</pre>
426	<pre>BARO(:,2) = rawDFL.BARO.TimeUS;</pre>
427	<pre>BARO(:,3) = rawDFL.BARO.Alt;</pre>
428	<pre>BARO(:,4) = rawDFL.BARO.Press;</pre>
429	<pre>BARO(:,5) = rawDFL.BARO.Temp;</pre>
430	<pre>BARO(:,6) = rawDFL.BARO.CRt;</pre>
431	<pre>BARO(:,7) = rawDFL.BARO.SMS;</pre>
432	<pre>BARO(:,8) = rawDFL.BARO.Offset;</pre>
433	<pre>BARO(:,9) = rawDFL.BARO.GndTemp;</pre>
434	<pre>BARO(:,10) = rawDFL.BARO.DatenumUTC;</pre>
435	BARO_label = {'LineNo';'TimeUS';'Alt';'Press';'Temp';
436	'CRt';'SMS';'Offset';'GndTemp';'DatenumUTC'};
437	%% RCOU
438	<pre>RCOU(:,1) = rawDFL.RCOU.LineNo;</pre>
439	<pre>RCOU(:,2) = rawDFL.RCOU.TimeUS;</pre>
440	<pre>RCOU(:,3) = rawDFL.RCOU.C1;</pre>
441	<pre>RCOU(:,4) = rawDFL.RCOU.C2;</pre>
442	<pre>RCOU(:,5) = rawDFL.RCOU.C3;</pre>
443	<pre>RCOU(:,6) = rawDFL.RCOU.C4;</pre>
444	<pre>RCOU(:,7) = rawDFL.RCOU.C5;</pre>

445	<pre>RCOU(:,8) = rawDFL.RCOU.C6;</pre>
446	<pre>RCOU(:,9) = rawDFL.RCOU.C7;</pre>
447	<pre>RCOU(:,10) = rawDFL.RCOU.C8;</pre>
448	<pre>RCOU(:,11) = rawDFL.RCOU.C9;</pre>
449	<pre>RCOU(:,12) = rawDFL.RCOU.C10;</pre>
450	<pre>RCOU(:,13) = rawDFL.RCOU.C11;</pre>
451	<pre>RCOU(:,14) = rawDFL.RCOU.C12;</pre>
452	<pre>RCOU(:,15) = rawDFL.RCOU.C13;</pre>
453	<pre>RCOU(:,16) = rawDFL.RCOU.C14;</pre>
454	<pre>RCOU(:,17) = rawDFL.RCOU.DatenumUTC;</pre>
455	RCOU_label = {'LineNo';'TimeUS';'C1';'C2';'C3';'C4';'C5';
456	'C6';'C7';'C8';'C9';'C10';'C11';'C12';'C13';'C14';
457	'DatenumUTC'};
458	%% IMU
459	<pre>IMU(:,1) = rawDFL.IMU.LineNo;</pre>
460	<pre>IMU(:,2) = rawDFL.IMU.TimeUS;</pre>
461	<pre>IMU(:,3) = rawDFL.IMU.GyrX;</pre>
462	<pre>IMU(:,4) = rawDFL.IMU.GyrY;</pre>
463	<pre>IMU(:,5) = rawDFL.IMU.GyrZ;</pre>
464	<pre>IMU(:,6) = rawDFL.IMU.AccX;</pre>
465	<pre>IMU(:,7) = rawDFL.IMU.AccY;</pre>
466	<pre>IMU(:,8) = rawDFL.IMU.AccZ;</pre>
467	<pre>IMU(:,9) = rawDFL.IMU.EG;</pre>
468	<pre>IMU(:,10) = rawDFL.IMU.EA;</pre>
469	<pre>IMU(:,11) = rawDFL.IMU.T;</pre>
470	<pre>IMU(:,12) = rawDFL.IMU.GH;</pre>
471	<pre>IMU(:,13) = rawDFL.IMU.AH;</pre>
472	<pre>IMU(:,14) = rawDFL.IMU.GHz;</pre>
473	<pre>IMU(:,15) = rawDFL.IMU.AHz;</pre>
474	<pre>IMU(:,16) = rawDFL.IMU.DatenumUTC;</pre>
475	<pre>IMU_label = {'LineNo';'TimeUS';'GyrX';'GyrY';'GyrZ';</pre>
476	'AccX';'AccY';'AccZ';'EG';'EA';'T';'GH';'AH';'GHz';

477	'AHz';'DatenumUTC'};
478	%% NKF2
479	try
480	<pre>NKF2(:,1) = rawDFL.NKF2.LineNo;</pre>
481	NKF2(:,2) = fix(
482	<pre>rawDFL.NKF2.TimeS/rawDFL.NKF2.fieldMultipliers.TimeUS);</pre>
483	<pre>NKF2(:,3) = rawDFL.NKF2.AZbias;</pre>
484	<pre>NKF2(:,4) = rawDFL.NKF2.GSX;</pre>
485	NKF2(:,5) = rawDFL.NKF2.GSY;
486	NKF2(:,6) = rawDFL.NKF2.GSZ;
487	NKF2(:,7) = rawDFL.NKF2.VWN;
488	<pre>NKF2(:,8) = rawDFL.NKF2.VWE;</pre>
489	<pre>NKF2(:,9) = rawDFL.NKF2.MN;</pre>
490	<pre>NKF2(:,10) = rawDFL.NKF2.ME;</pre>
491	<pre>NKF2(:,11) = rawDFL.NKF2.MD;</pre>
492	<pre>NKF2(:,12) = rawDFL.NKF2.MX;</pre>
493	<pre>NKF2(:,13) = rawDFL.NKF2.MY;</pre>
494	<pre>NKF2(:,14) = rawDFL.NKF2.MZ;</pre>
495	<pre>NKF2(:,15) = rawDFL.NKF2.MI;</pre>
496	<pre>NKF2(:,16) = rawDFL.NKF2.DatenumUTC;</pre>
497	<pre>NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';</pre>
498	'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';
499	<pre>'MZ';'MI';'DatenumUTC'};</pre>
500	
501	catch
502	<pre>NKF2_label = {'LineNo';'TimeUS';'AZbias';'GSX';'GSY';</pre>
503	'GSZ';'VWN';'VWE';'MN';'ME';'MD';'MX';'MY';
504	<pre>'MZ';'MI';'DatenumUTC'};</pre>
505	end
506	
507	
508	% Save asnew file for integration with normal code

```
[¬, baseNameNoExtDFL, ¬] = fileparts(baseFileNameDFL);
509
                baseFileName = sprintf('%s.mat', baseNameNoExtDFL);
510
                fullParsedMatFileName = fullfile(folderDFL, baseFileName);
511
                % Save file with parsed data as the original filename plus ...
512
                    the added portion
513
                if(strcmpi(arduPilotType, 'ArduCopter'))
514
                     save(fullParsedMatFileName,'GPS','GPS_label','ATT',...
515
                         'ATT_label', 'BARO', 'BARO_label', 'NKF2', 'NKF2_label',...
516
                         'IMU', 'IMU_label', 'RCOU', 'RCOU_label', 'MSG1');
517
                elseif(strcmpi(arduPilotType, 'CopterSonde'))
518
                     save(fullParsedMatFileName,'GPS','GPS_label','WIND',...
519
                         'WIND_label', 'IMET', 'IMET_label', 'ATT', 'ATT_label',...
520
                         'BARO', 'BARO_label', 'NKF2', 'NKF2_label', 'IMU',...
521
                         'IMU_label', 'RCOU', 'RCOU_label', 'MSG1');
522
                end
523
                fullInputMatFileNameDFL = fullParsedMatFileName;
524
            end
525
       end
526
       %% Gather Additional Files
527
       variableInfo = who('-file',fullInputMatFileNameDFL);
528
529
       if(length(variableInfo)<20 && strcmpi(DFL_NewOld, 'Unknown')...</pre>
530
                && ismember('GPS_table',variableInfo))
531
532
533
            load(fullInputMatFileNameDFL);
534
            DFL_NewOld = 'Old';
535
536
            if(ismember('MHP_table', variableInfo))
537
                mhpValue = 'Yes';
538
539
            else mhpValue = 'No';
```

```
end
540
541
             if(ismember('TPH_table',variableInfo))
542
                 tphValue = 'Yes';
543
             else tphValue = 'No';
544
             end
545
546
             if(ismember('iMet_table',variableInfo))
547
                 iMetValue = 'Yes';
548
             else iMetValue = 'No';
549
             end
550
551
             if(ismember('CSIMET_table',variableInfo))
552
                 arduPilotType = 'CopterSonde';
553
             end
554
555
        else
556
             DFL_NewOld = 'New';
557
558
             % Parser for ArduPilotType
559
             if(strcmpi(arduPilotType, 'Default'))
560
561
                 if(ismember('IMET', variableInfo))
562
                      arduPilotType = 'Coptersonde';
563
                 else
564
                      load(fullInputMatFileNameDFL, 'MSG1');
565
566
                      QuadPlane = 0;
567
                      ArduPlane = 0;
568
                      for (i=1:length (MSG1))
569
                          try
570
                               if(strcmpi(MSG1{1,i}{3,1}(1,1:9),{'QuadPlane'}))
571
```

```
QuadPlane = 1;
572
                              elseif(strcmpi(MSG1{1,i}{3,1}(1,1:9),{'ArduPlane'}))
573
                                  ArduPlane = 1;
574
                              elseif(strcmpi(MSG1{1,i}{3,1}(1,1:10), {'ArduCopter'})))
575
                                   arduPilotType = 'ArduCopter';
576
                              end
577
578
                          catch
579
                          end
580
581
                     end
582
                     if((QuadPlane + ArduPlane) == 2)
583
                          arduPilotType = 'Quad-Plane';
584
                     elseif((QuadPlane + ArduPlane) == 1)
585
                          arduPilotType = 'Fixed Wing';
586
                     end
587
588
                 end
            end
589
590
            % Execution of ArduPilotType
591
            if(strcmpi(arduPilotType, 'Fixed Wing'))
592
                 load(fullInputMatFileNameDFL,'ATT','ATT_label','BARO',...
593
                     'BARO_label', 'CTUN', 'CTUN_label', 'GPS', 'GPS_label', ...
594
                     'IMU', 'IMU_label', 'NKF1', 'NKF1_label', 'NKF2', 'NKF2_label', .....
595
                     'RCOU', 'RCOU_label');
596
597
                 % Pre-parse for only relevant data series
598
                ATT = [ATT(:,1),ATT(:,2),ATT(:,4),ATT(:,6),ATT(:,8)];
599
                 ATT_label = ...
600
                     [ATT_label(1), ATT_label(2), ATT_label(4), ATT_label(6), ATT_label(8)];
                BARO = [BARO(:, 1:4)];
601
602
                 BARO_label = [BARO_label(1:4)];
```

CTUN = ... 603 [CTUN(:,1),CTUN(:,2),CTUN(:,4),CTUN(:,6),CTUN(:,8),CTUN(:,10)]; CTUN\_label = [CTUN\_label(1), CTUN\_label(2), CTUN\_label(4),... 604CTUN\_label(6),CTUN\_label(8),CTUN\_label(10)]; 605NKF1 = [NKF1(:,1), NKF1(:,2), NKF1(:,3), NKF1(:,4), ... 606 NKF1(:,5),... NKF1(:,6), NKF1(:,7), NKF1(:,8), NKF1(:,9), NKF1(:,10), ... 607 NKF1(:,11)];%, NKF1(:,12), NKF1(:,13), NKF1(:,14), ... NKF1(:,15), NKF1(:,16)]; 608NKF1\_label = [NKF1\_label(1), NKF1\_label(2), NKF1\_label(3),... NKF1\_label(4), NKF1\_label(5), NKF1\_label(6), ... 609 NKF1\_label(7),... NKF1\_label(8), NKF1\_label(9), NKF1\_label(10), ... 610NKF1\_label(11)]; %, NKF1\_label(12), NKF1\_label(13), ... NKF1\_label(14), NKF1\_label(15), NKF1\_label(16)];  $GPS = \ldots$ 611[GPS(:,1),GPS(:,2),GPS(:,4),GPS(:,5),GPS(:,8),GPS(:,9),... GPS(:,10),GPS(:,11)]; 612GPS\_label = ... 613[GPS\_label(1),GPS\_label(2),GPS\_label(4),GPS\_label(5),... GPS\_label(8), GPS\_label(9), GPS\_label(10), GPS\_label(11)]; 614615IMU = [IMU(:, 1:8)];IMU\_label = [IMU\_label(1:8)]; 616NKF2 = [NKF2(:,1), NKF2(:,2), NKF2(:,7), NKF2(:,8)]; 617NKF2\_label = ... 618[NKF2\_label(1), NKF2\_label(2), NKF2\_label(7), NKF2\_label(8)]; RCOU = [RCOU(:, 1:6)];619620RCOU\_label = [RCOU\_label(1:6)]; end 621622if(strcmpi(arduPilotType, 'Quad-Plane')) 623624load(fullInputMatFileNameDFL,'ATT','ATT\_label','BARO',...

'BARO\_label','CTUN','CTUN\_label','IMU','IMU\_label',... 625'GPS', 'GPS\_label', 'XKF1', 'XKF1\_label', 'XKF2', 'XKF2\_label',... 626'RCOU', 'RCOU\_label'); 627628% Pre-parse for only relevant data series 629ATT = [ATT(:, 1), ATT(:, 2), ATT(:, 4), ATT(:, 6), ATT(:, 8)];630 ATT\_label = ... 631[ATT\_label(1), ATT\_label(2), ATT\_label(4), ATT\_label(6), ATT\_label(8)]; BARO = [BARO(:, 1:4)];632633 BARO\_label = [BARO\_label(1:4)]; CTUN = ... 634[CTUN(:,1),CTUN(:,2),CTUN(:,4),CTUN(:,6),CTUN(:,8),CTUN(:,10)]; CTUN\_label = [CTUN\_label(1), CTUN\_label(2), CTUN\_label(4),... 635CTUN\_label(6),CTUN\_label(8),CTUN\_label(10)]; 636 NKF1 = [XKF1(:,1),XKF1(:,2), XKF1(:,3), XKF1(:,4), ... 637 XKF1(:,5),... XKF1(:,6), XKF1(:,7), XKF1(:,8), XKF1(:,9), XKF1(:,10), ... 638XKF1(:,11)]; %, XKF1(:,12), XKF1(:,13), XKF1(:,14), ... XKF1(:,15), XKF1(:,16)]; NKF1\_label = [XKF1\_label(1), XKF1\_label(2), XKF1\_label(3),... 639XKF1\_label(4), XKF1\_label(5), XKF1\_label(6), ... 640XKF1\_label(7),... XKF1\_label(8), XKF1\_label(9), XKF1\_label(10), ... 641XKF1\_label(11)]; %, XKF1\_label(12), XKF1\_label(13), ... XKF1\_label(14), XKF1\_label(15), XKF1\_label(16)]; GPS=[GPS(:,1),GPS(:,2),GPS(:,4),GPS(:,5),GPS(:,8),GPS(:,9),... 642GPS(:,10),GPS(:,11)]; 643GPS\_label = [GPS\_label(1), GPS\_label(2), GPS\_label(4), ... 644GPS\_label(5), GPS\_label(8), GPS\_label(9), GPS\_label(10), ... 645GPS\_label(11)]; 646IMU = [IMU(:, 1:8)];647IMU\_label = [IMU\_label(1:8)]; 648

```
NKF2 = [XKF2(:,1),XKF2(:,2),XKF2(:,6),XKF2(:,7)];
649
                NKF2_label = [XKF2_label(1), XKF2_label(2), XKF2_label(6), ...
650
                    XKF2_label(7);
651
                RCOU = [RCOU(:, 1:6)];
652
                RCOU_label = [RCOU_label(1:6)];
653
            end
654
655
            % Not configured OF NOTE: Have not implemented NKF1 in DFL load
656
            if(strcmpi(arduPilotType, 'ArduCopter'))
657
                load(fullInputMatFileNameDFL,'ATT','ATT_label','BARO',...
658
                     'BARO_label','IMU','IMU_label','GPS','GPS_label','NKF1',...
659
                     'NKF1_label', 'NKF2', 'NKF2_label', 'RCOU', 'RCOU_label');
660
661
                % Pre-parse for only relevant data series
662
                ATT = [ATT(:,1),ATT(:,2),ATT(:,4),ATT(:,6),ATT(:,8)];
663
                ATT_label = [ATT_label(1), ATT_label(2), ATT_label(4), ...
664
                    ATT_label(6),ATT_label(8)];
665
                BARO = [BARO(:, 1:4)];
666
                BARO_label = [BARO_label(1:4)];
667
                GPS=[GPS(:,1),GPS(:,2),GPS(:,4),GPS(:,5),GPS(:,8),GPS(:,9),...
668
                    GPS(:,10),GPS(:,11)];
669
670
                GPS_label = [GPS_label(1),GPS_label(2),GPS_label(4),...
                    GPS_label(5), GPS_label(8), GPS_label(9), GPS_label(10), ...
671
                    GPS_label(11)];
672
                IMU = [IMU(:, 1:8)];
673
                IMU_label = [IMU_label(1:8)];
674
                %new
675
676
                NKF1 = [NKF1(:,1), NKF1(:,2), NKF1(:,3), NKF1(:,4), ...
                   NKF1(:,5),...
                    NKF1(:,6), NKF1(:,7), NKF1(:,8)];%, NKF1(:,9), ...
677
                        NKF1(:,10), NKF1(:,11), NKF1(:,12), NKF1(:,13), ...
                        NKF1(:,14), NKF1(:,15), NKF1(:,16)];
```

	678	<pre>NKF1_label = [NKF1_label(1),NKF1_label(2), NKF1_label(3),</pre>
	679	NKF1_label(4), NKF1_label(5), NKF1_label(6),
		NKF1_label(7),
	680	NKF1_label(8)];%, NKF1_label(9), NKF1_label(10),
		NKF1_label(11), NKF1_label(12), NKF1_label(13),
		NKF1_label(14), NKF1_label(15), NKF1_label(16)];
	681	%new
	682	NKF2 = [NKF2(:,1), NKF2(:,2), NKF2(:,6), NKF2(:,7)];
	683	<pre>NKF2_label = [NKF2_label(1),NKF2_label(2),NKF2_label(6),</pre>
	684	NKF2_label(7)];
	685	RCOU = [RCOU(:, 1:6)];
	686	<pre>RCOU_label = [RCOU_label(1:6)];</pre>
	687 er	nd
	688	
	689 %	Not configured
	690 if	<pre>(strcmpi(arduPilotType,'CopterSonde'))</pre>
1	691	<pre>load(fullInputMatFileNameDFL,'ATT','ATT_label','WIND',</pre>
	692	'WIND_label','IMET','IMET_label','BARO','BARO_label',
	693	'IMU','IMU_label','GPS','GPS_label','RCOU','RCOU_label');
	694	
1	695	% Pre-parse for only relevant data series
	696	ATT = [ATT(:,1),ATT(:,2),ATT(:,4),ATT(:,6),ATT(:,8)];
	697	<pre>ATT_label = [ATT_label(1),ATT_label(2),ATT_label(4),</pre>
	698	<pre>ATT_label(6),ATT_label(8)];</pre>
(	699	BARO = [BARO(:,1:4)];
	700	BARO_label = [BARO_label(1:4)];
	701	GPS=[GPS(:,1),GPS(:,2),GPS(:,4),GPS(:,5),GPS(:,8),GPS(:,9),
	702	GPS(:,10),GPS(:,11)];
,	703	<pre>GPS_label = [GPS_label(1),GPS_label(2),GPS_label(4),</pre>
	704	<pre>GPS_label(5),GPS_label(8),GPS_label(9),GPS_label(10),</pre>
	705	GPS_label(11)];
	706	IMU = [IMU(:,1:8)];

```
IMU_label = [IMU_label(1:8)];
707
                RCOU = [RCOU(:, 1:6)];
708
                RCOU_label = [RCOU_label(1:6)];
709
            end
710
711
            % Not configured, placeholder only
712
            if(strcmpi(arduPilotType, 'ArduRover'))
713
                 load(fullInputMatFileNameDFL,'ATT','ATT_label','BARO',...
714
                     'BARO_label', 'CTUN', 'CTUN_label', 'GPS', 'GPS_label', 'IMU',...
715
                     'IMU_label', 'NKF2', 'NKF2_label', 'RCOU', 'RCOU_label',...
716
                     'STAT', 'STAT_label');
717
718
                % Pre-parse for only relevant data series
719
720
                ATT = [ATT(:, 1), ATT(:, 2), ATT(:, 4), ATT(:, 6), ATT(:, 8)];
                ATT_label = [ATT_label(1), ATT_label(2), ATT_label(4), ...
721
                     ATT_label(6),ATT_label(8)];
722
                BARO = [BARO(:, 1:4)];
723
                BARO_label = [BARO_label(1:4)];
724
                CTUN = [CTUN(:,1), CTUN(:,2), CTUN(:,4), CTUN(:,6), CTUN(:,8),...
725
                     CTUN(:,10)];
726
                CTUN_label = [CTUN_label(1), CTUN_label(2), CTUN_label(4),...
727
728
                     CTUN_label(6),CTUN_label(8),CTUN_label(10)];
                GPS = [GPS(:,1),GPS(:,2),GPS(:,4),GPS(:,5),GPS(:,8),...
729
                     GPS(:,9),GPS(:,10),GPS(:,11)];
730
                GPS_label = [GPS_label(1), GPS_label(2), GPS_label(4), ...
731
                     GPS_label(5),GPS_label(8),GPS_label(9),...
732
                     GPS_label(10), GPS_label(11)];
733
                 IMU = [IMU(:, 1:8)];
734
                IMU_label = [IMU_label(1:8)];
735
                NKF2 = [NKF2(:,1), NKF2(:,2), NKF2(:,7), NKF2(:,8)];
736
                NKF2_label = [NKF2_label(1),NKF2_label(2),NKF2_label(7),...
737
738
                     NKF2_label(8)];
```

```
RCOU = [RCOU(:, 1:6)];
739
                RCOU_label = [RCOU_label(1:6)];
740
                STAT = [STAT(:, 1:5)];
741
                STAT_label = [STAT_label(1:5)];
742
            end
743
744
       end
745
746
       % Get filename without the extension, used by Save Function
747
        [\neg, baseNameNoExtDFL, \neg] = fileparts(baseFileNameDFL);
748
749
       if (strcmpi(pitchToggle, 'Yes'))
750
751
            if(strcmpi(aircraft, 'TIA'))
752
                pitchMap=[1139,-7.9; 1150,-7.1; 1200,-6.3; 1250,-5.6;...
753
                    1300, -4.8; 1350, -4.0; 1400, -3.2; 1450, -1.6; ...
754
                        1500,0.0 ;...
                    1550,6.3 ; 1600,7.9 ; 1650,9.5 ; 1700,11.0 ; 1750,12.5 ...
755
                        ; 1800,14.0 ; 1850,15.5 ; 1900,17.0 ; 1939,17.0];
            else
756
                % Have user browse for a file, from a specified "starting ...
757
                    folder."
                % For convenience in browsing, set a starting folder from ...
758
                    which to browse.
                % Start in the current folder.
759
                startingFolder = pwd;
760
761
                % Get the name of the file that the user wants to use.
762
                defaultFileName = fullfile(startingFolder, '*.txt');
763
                [baseFileName, folder] = uigetfile(defaultFileName,...
764
                     'Select a Pitch PWM text file');
765
                if baseFileName == 0
766
```

% User clicked the Cancel button. 767return; 768end 769770 % Get the name of the input .mat file. 771fullInputMatFileName = fullfile(folder, baseFileName); 772pitchMap = load(fullInputMatFileName); 773end 774end 775776if (strcmpi(throttleToggle, 'Yes')) 777 778 if(strcmpi(aircraft, 'TIA')) 779throttleMap=[0,0.00 ; 10,1.02 ; 20,1.91 ; 30,2.85 ; 40,3.93 ... 780; 50,5.00 ; 60,6.31 ; 70,7.92 ; 80,9.57 ; 89,11.17 ; ... 100,13.37]; else 781% Have user browse for a file, from a specified "starting ... 782folder." % For convenience in browsing, set a starting folder from ... 783 which to browse. % Start in the current folder. 784startingFolder = pwd; 785786% Get the name of the file that the user wants to use. 787defaultFileName = fullfile(startingFolder, '\*.txt'); 788[baseFileName, folder] = uigetfile(defaultFileName, 'Select ... 789a Throttle Mapping text file'); if baseFileName == 0 790 % User clicked the Cancel button. 791return; 792793 end
```
794
                 % Get the name of the input .mat file.
795
                 fullInputMatFileName = fullfile(folder, baseFileName);
796
                 throttleMap = load(fullInputMatFileName);
797
            end
798
        end
799
        %% Set Parsing Bounds
800
        if(strcmpi(DFL_NewOld, 'New'))
801
802
            fiq1=figure(1);
803
            fig1.Name =...
804
    'Raw data from DFL. Click on graph for upper and lower bound for parsing.';
805
806
            if(strcmpi(arduPilotType, 'ArduCopter') ...
807
                     strcmpi(arduPilotType, 'CopterSonde'))
808
                plt1 = subplot(4, 1, 1);
809
                plot (GPS(:,2),GPS(:,8),'b')
810
                 title('Groundspeed vs Time')
811
                 ylabel({'Groundspeed (blue)';'(m/s)'})
812
            else
813
                 % Groundspeed plot
814
                 plt1 = subplot(4, 1, 1);
815
                 plot (GPS(:,2),GPS(:,8), 'b',CTUN(:,2),CTUN(:,6), 'r')
816
                 title('Groundspeed, Airspeed vs Time')
817
                 ylabel({'Groundspeed (blue)';'Airspeed (red)';'(m/s)'})
818
            end
819
820
            % Throttle Output
821
            plt2 = subplot(4, 1, 2);
822
            plot (RCOU(:,2), ((RCOU(:,5)-thrMinPWM)/(thrMaxPWM-thrMinPWM)*100), 'b')
823
            title('Throttle vs Time')
824
            ylabel({'Throttle';'(%)'})
825
```

```
ylim([0 100])
826
827
            % For the dotted line along x-axis of pitch plot
828
            zero=int8(zeros(length(ATT(:,2)),1));
829
830
            % Aircraft Pitch angle: Can change ylim to something more relevant.
831
            % TIV uses -20 to 50 to see high AoA landing
832
            plt3 = subplot(4, 1, 3);
833
            plot(ATT(:,2),ATT(:,4),'b',ATT(:,2),zero,'r:')
834
            title('Aircraft Pitch Angle vs Time')
835
            ylabel({'Aircraft Pitch';'Angle ( )'})
836
            ylim([-10 40])
837
838
            % Altitude plot
839
            plt4 = subplot(4, 1, 4);
840
            plot(GPS(:,2),GPS(:,7),'b')
841
            title('Altitude vs Time')
842
            ylabel({'Altitude MSL';'(m)'})
843
            xlabel('Time (microseconds)')
844
            linkaxes([plt1 plt2 plt3 plt4],'x')
845
            xlim([min(GPS(:,2)) max(GPS(:,2))])
846
847
            m=0;
848
            while true
849
                [horiz, vert, button] = ginput(1);
850
                if isempty(horiz) || button(1) == 3; break; end
851
                m = m+1;
852
                x_m(m) = horiz(1); % save all points you continue getting
853
                hold on
854
                if(strcmpi(arduPilotType, 'ArduCopter')...
855
                         strcmpi(arduPilotType, 'CopterSonde'))
856
```

857	<pre>else y_va(m) =CTUN(find(CTUN(:,2)≥x_m(m),1,'first'),6);</pre>
	% Airspeed
858	end
859	y_vg(m)=GPS(find(GPS(:,2)≥x_m(m),1,'first'),8); %
	Groundspeed
860	y_thr(m)=RCOU(find(RCOU(:,2)≥x_m(m),1,'first'),5); %
	Throttle Percent
861	$y_pitch(m) = ATT(find(ATT(:,2) \ge x_m(m), 1, 'first'), 4); %$
	Aircraft Pitch
862	$y_alt(m) = GPS(find(GPS(:,2) \ge x_m(m), 1, 'first'), 7);$ % Altitude
863	
864	<pre>if(strcmpi(arduPilotType,'ArduCopter')</pre>
865	<pre>strcmpi(arduPilotType,'CopterSonde'))</pre>
866	% Groundspeed plot
867	<pre>subplot(4,1,1)</pre>
868	<pre>plot(GPS(:,2),GPS(:,8),'b',x_m,y_vg,'kx')</pre>
869	<pre>title('Groundspeed vs Time')</pre>
870	<pre>ylabel({'Groundspeed (blue)';'(m/s)'})</pre>
871	else
872	% Groundspeed plot
873	<pre>subplot(4,1,1)</pre>
874	<pre>plot (GPS(:,2), GPS(:,8), 'b', CTUN(:,2),</pre>
875	CTUN(:,6), 'r', x_m, y_vg, 'kx', x_m, y_va, 'kx')
876	<pre>title('Groundspeed, Airspeed vs Time')</pre>
877	<pre>ylabel({'Groundspeed (blue)';'Airspeed (red)';'(m/s)'})</pre>
878	end
879	
880	% Throttle Output
881	<pre>subplot(4,1,2)</pre>
882	<pre>plot(RCOU(:,2),((RCOU(:,5)-thrMinPWM)/</pre>
883	(thrMaxPWM-thrMinPWM)*100),'b',x_m, ((y_thr-thrMinPWM)/
884	(thrMaxPWM-thrMinPWM) *100), 'kx')

```
title('Throttle vs Time')
885
                 ylabel({'Throttle';'(%)'})
886
                 ylim([0 100])
887
888
                 % For the dotted line along x-axis of pitch plot
889
                 zero=int8(zeros(length(ATT(:,2)),1));
890
891
                 % Aircraft Pitch angle: Can change ylim to something more ...
892
                     relevant.
                 % TIV uses -20 to 50 to see high AoA landing
893
                 subplot(4,1,3)
894
                 plot(ATT(:,2),ATT(:,4), 'b',ATT(:,2),zero,'r:',x_m,y_pitch,'kx')
895
                 title('Aircraft Pitch Angle vs Time')
896
                 ylabel({'Aircraft Pitch';'Angle ( )'})
897
                 ylim([-10 40])
898
899
                 % Altitude plot
900
                 subplot (4, 1, 4)
901
                 plot(GPS(:,2),GPS(:,7),'b',x_m,y_alt,'kx')
902
                 title('Altitude vs Time')
903
                 ylabel({'Altitude MSL';'(m)'})
904
                 xlabel('Time (microseconds)')
905
                 xlim([min(GPS(:,2)) max(GPS(:,2))])
906
907
                 drawnow
908
909
                 if (m \ge 2)
910
                     break;
911
                 end
912
913
            end
914
915
```

```
fiq2=figure(2);
916
917
            fig2.Name = 'Preview of user-parsed DFL data.';
918
            if(strcmpi(arduPilotType, 'ArduCopter') |...
919
                     strcmpi(arduPilotType, 'CopterSonde'))
920
                 % Groundspeed plot
921
                plt1 = subplot(4, 1, 1);
922
                 plot (GPS(:,2)/1000000,GPS(:,8),'b')
923
                 title('Groundspeed vs Time')
924
                 ylabel({'Groundspeed (blue)';'(m/s)'})
925
            else
926
                 % Groundspeed plot
927
                plt1 = subplot(4, 1, 1);
928
                 plot (GPS(:,2)/1000000, GPS(:,8), 'b', CTUN(:,2)/1000000, CTUN(:,6), 'r')
929
                 title('Groundspeed, Airspeed vs Time')
930
                 ylabel({'Groundspeed (blue)';'Airspeed (red)';'(m/s)'})
931
            end
932
933
            % Throttle Output
934
            plt2 = subplot(4, 1, 2);
935
            plot (RCOU(:,2)/1000000, ((RCOU(:,5)-thrMinPWM)/(thrMaxPWM-thrMinPWM)*100), 'b')
936
            title('Throttle vs Time')
937
            ylabel({'Throttle';'(%)'})
938
            ylim([0 100])
939
940
            % For the dotted line along x-axis of pitch plot
941
            zero=int8(zeros(length(ATT(:,2)),1));
942
943
            % Aircraft Pitch angle: Can change ylim to something more relevant.
944
            % TIV uses -20 to 50 to see high AoA landing
945
            plt3 = subplot(4, 1, 3);
946
            plot (ATT(:,2)/1000000,ATT(:,4), 'b',ATT(:,2)/1000000,zero, 'r:')
947
```

```
title('Aircraft Pitch Angle vs Time')
948
            ylabel({'Aircraft Pitch'; 'Angle ( )'})
949
            vlim([-10 40])
950
951
            % Altitude plot
952
            plt4 = subplot(4, 1, 4);
953
            plot(GPS(:,2)/1000000,GPS(:,7),'b')
954
            title('Altitude vs Time')
955
            ylabel({'Altitude MSL';'(m)'})
956
            xlabel('Time (seconds)')
957
            linkaxes([plt1 plt2 plt3 plt4],'x')
958
            xlim([x_m(1)/1000000 x_m(2)/1000000])
959
960
961
        end
        %% Configure Start/Stop Conditions
962
        tic
963
        if(strcmpi(DFL_NewOld, 'New'))
964
965
            % Finding the applicable data range based on minimum settings above
966
            % GPS line number of starting relavent data (used for parsing)
967
            TO = IMU(find(IMU(:,2) \ge x_m(1), 1, 'first'), 1);
968
            LND = IMU(find(IMU(:, 2) \ge x_m(2), 1, 'first'), 1);
969
970
            % Potision in each major dataset (GPS, CTUN, NKF1, NKF2, RCOU) ...
971
                for Takeoff (TO)
            % and Landing (LND)
972
            if(strcmpi(arduPilotType, 'ArduCopter'))
973
                 TO_NKF1 = find(NKF1(:,1)>TO,1,'first')-1;
974
                LND_NKF1 = find(NKF1(:,1)>LND,1,'first')-1;
975
                TO_NKF2 = find(NKF2(:,1)>TO,1,'first')-1;
976
                LND_NKF2 = find(NKF2(:,1)>LND,1,'first')-1;
977
978
            elseif(strcmpi(arduPilotType, 'CopterSonde'))
```

9	79	<pre>TO_WIND = find(WIND(:,1)&gt;TO,1,'first')-1;</pre>
9	80	<pre>LND_WIND = find(WIND(:,1)&gt;LND,1,'first')-1;</pre>
9	81	<pre>TO_IMET = find(IMET(:,1)&gt;TO,1,'first')-1;</pre>
9	82	<pre>LND_IMET = find(IMET(:,1)&gt;LND,1,'first')-1;</pre>
9	83	else
9	84	<pre>TO_CTUN = find(CTUN(:,1)&gt;TO,1,'first')-1;</pre>
9	85	<pre>LND_CTUN = find(CTUN(:,1)&gt;LND,1,'first')-1;</pre>
9	86	<pre>TO_NKF1 = find(NKF1(:,1)&gt;TO,1,'first')-1;</pre>
9	87	<pre>LND_NKF1 = find(NKF1(:,1)&gt;LND,1,'first')-1;</pre>
9	88	<pre>TO_NKF2 = find(NKF2(:,1)&gt;TO,1,'first')-1;</pre>
9	89	LND_NKF2 = find(NKF2(:,1)>LND,1,'first')-1;
9	90	end
9	91	<pre>TO_GPS = find(GPS(:,1)&gt;TO,1,'first')-1;</pre>
9	92	<pre>LND_GPS = find(GPS(:,1)&gt;LND,1,'first')-1;</pre>
9	93	TO_ATT = find(ATT(:,1)>TO,1,'first')-1;
9	94	<pre>LND_ATT = find(ATT(:,1)&gt;LND,1,'first')-1;</pre>
9	95	<pre>TO_RCOU = find(RCOU(:,1)&gt;TO,1,'first')-1;</pre>
9	96	<pre>LND_RCOU = find(RCOU(:,1)&gt;LND,1,'first')-1;</pre>
9	97	<pre>TO_IMU = find(IMU(:,1)&gt;TO,1,'first')-1;</pre>
9	98	<pre>LND_IMU = find(IMU(:,1)&gt;LND,1,'first')-1;</pre>
9	99	<pre>TO_BARO = find(BARO(:,1)&gt;TO,1,'first')-1;</pre>
10	00	<pre>LND_BARO = find(BARO(:,1)&gt;LND,1,'first')-1;</pre>
10	01	
10	02	%%%%%%%%% GPS Logs Parsing %%%%%%%%
10	03	% Latitude
10	04	x=GPS(TO_GPS:LND_GPS,5);
10	05	% Longitude
10	06	y=GPS(TO_GPS:LND_GPS,6);
10	07	% Altitude
10	08	z=GPS(TO_GPS:LND_GPS,7);
10	09	$z_AGL = z(:) - z(1);$
10	10	

1011	%%%%%%%% Airspeed and Windspeed Data %%%%%%%%
1012	% Ground speed
1013	v_g=GPS(TO_GPS:LND_GPS,8);
1014	% Max ground speed
1015	<pre>max_v_g=max(GPS(TO_GPS:LND_GPS,8));</pre>
1016	<pre>if(strcmpi(arduPilotType,'CopterSonde'))</pre>
1017	wind = WIND(TO_WIND:LND_WIND,4);
1018	<pre>VWN = cos(deg2rad(WIND(TO_WIND:LND_WIND,3))).*wind;</pre>
1019	<pre>VWE = sin(deg2rad(WIND(TO_WIND:LND_WIND,3))).*wind;</pre>
1020	<pre>elseif(strcmpi(arduPilotType, 'ArduCopter'))</pre>
1021	% North winds
1022	<pre>VWN=NKF2(TO_NKF2:LND_NKF2,3);</pre>
1023	% East winds
1024	<pre>VWE=NKF2(TO_NKF2:LND_NKF2,4);</pre>
1025	% Wind Speed
1026	wind=(VWN.^2+VWE.^2).^0.5;
1027	% Wind Angle
1028	<pre>windANG=atan2d(VWN,VWE);</pre>
1029	else
1030	% Airspeed
1031	<pre>v_a=CTUN(TO_CTUN:LND_CTUN,6);</pre>
1032	% North winds
1033	<pre>VWN=NKF2(TO_NKF2:LND_NKF2,3);</pre>
1034	% East winds
1035	<pre>VWE=NKF2(TO_NKF2:LND_NKF2,4);</pre>
1036	% Wind speed
1037	wind=(VWN.^2+VWE.^2).^0.5;
1038	% Wind Angle
1039	<pre>windANG=atan2d(VWN,VWE);</pre>
1040	end
1041	
1042	%%%%%%%% Aircraft Data %%%%%%%%

1043	% Pitch PWM signal	
1044	<pre>pitchPWM=RCOU(TO_RCOU:LND_RCOU,3);</pre>	
1045	% Aircraft pitch angle	
1046	<pre>pitchAC=ATT(TO_ATT:LND_ATT,4);</pre>	
1047	% Aircraft roll angle	
1048	<pre>rollAC = ATT(TO_ATT:LND_ATT,3);</pre>	
1049	% Aircraft yaw (Earth reference, degrees)	
1050	<pre>yawAC = ATT(TO_ATT:LND_ATT,5);</pre>	
1051	% Throttle output from Pixhawk	
1052	<pre>thr=(RCOU(TO_RCOU:LND_RCOU,5)-thrMinPWM)/(thrMaxPWM-thrMinPWM)*100;</pre>	
1053		
1054		
1055	%%%% EKF Aircraft Data %%%%	
1056	% EKF Roll	
1057	<pre>rollekf=NKF1(TO_NKF1:LND_NKF1,3);</pre>	
1058	% EKF Pitch	
1059	<pre>pitchEKF=NKF1(TO_NKF1:LND_NKF1,4);</pre>	
1060	% EKF Yaw	
1061	<pre>yawEKF=NKF1(TO_NKF1:LND_NKF1,5);</pre>	
1062	% North Aircraft Velocity	
1063	<pre>vnEKF=NKF1(TO_NKF1:LND_NKF1,6);</pre>	
1064	% East Aircraft Velocity	
1065	veEKF=NKF1(TO_NKF1:LND_NKF1,7);	
1066	% Down Aircraft Velocity	
1067	<pre>vdEKF=NKF1(TO_NKF1:LND_NKF1,8);</pre>	
1068	% Position North	
1069	<pre>pnEKF=NKF1(TO_NKF1:LND_NKF1,9);</pre>	
1070	% Position North	
1071	<pre>peEKF=NKF1(TO_NKF1:LND_NKF1,10);</pre>	
1072	% Position North	
1073	<pre>pdEKF=NKF1(TO_NKF1:LND_NKF1,11);</pre>	
1074	end	

```
%% Parse All Data and Save To Respective Tables
1075
1076
        if(strcmpi(DFL_NewOld, 'New'))
            % Parsed GPS data output
1077
            GPS_LN = GPS(TO_GPS:LND_GPS, 1);
1078
            GPS_time = GPS(TO_GPS:LND_GPS,2);
1079
            GPS_ms = GPS(TO_GPS:LND_GPS,3);
1080
            GPS_wk = GPS(TO_GPS:LND_GPS, 4);
1081
            GPS_time_out= (GPS_time-min(GPS_time))/1000000;
1082
            % Convert GPS timestamps to UTC time
1083
            leap_second_table = datenum...
1084
                 (['Jul 01 1981';'Jul 01 1982';'Jul 01 1983';'Jul 01 ...
1085
                    1985'; 'Jan 01 1988';...
                 'Jan 01 1990';'Jan 01 1991';'Jul 01 1992';'Jul 01 1993';...
1086
                'Jul 01 1994';'Jan 01 1996';'Jul 01 1997';'Jan 01 1999';...
1087
                 'Jan 01 2006';'Jan 01 2009';'Jul 01 2012';'Jul 01 ...
1088
                    2015';'Jan 01 2017']...
                 , 'mmm dd yyyy');
1089
            qps_zero_datenum = datenum('1980-01-06 ...
1090
                00:00:00.000', 'yyyy-mm-dd HH:MM:SS.FFF');
            days_since_gps_zero = GPS_wk*7 + GPS_ms/1e3/60/60/24;
1091
            recv_gps_datenum = gps_zero_datenum + days_since_gps_zero;
1092
            leapseconds = 18;
1093
            recv_utc_datenum = recv_gps_datenum - ((leapseconds)/60/60/24);
1094
            GPS_date=datestr(recv_utc_datenum, 'mmm-dd-yyyy');
1095
            GPS_full_time=datestr(recv_utc_datenum, 'HH:MM:SS.FFF');
1096
            tempGPS_date = datetime(GPS_date, 'InputFormat', 'MMM-dd-yyyy');
1097
            tempGPS_time = datetime(GPS_full_time, 'Format', 'HH:mm:ss.S');
1098
            tempGPS_comb = tempGPS_date + timeofday(tempGPS_time);
1099
1100
            GPS_final = datetime(tempGPS_comb, 'Format', 'MMM-dd-yyyy ...
                HH:mm:ss.S');
            % Output GPS table
1101
1102
            GPS = [GPS_LN, GPS_time, GPS_time_out, GPS_date, GPS_full_time,...
```

x, y, z, z\_AGL, v\_g]; 1103 GPS\_label = { 'Line No', 'Time since boot (us) ',... 1104 'Time from Arming (sec)', 'UTC Date', 'UTC Time', 'Lattitude',... 1105 'Longitude', 'Altitude (m, MSL)', 'Altitude (m, ... 1106 AGL)', 'Groundspeed (m/s)'}; GPS\_table = table(GPS\_LN, GPS\_time, GPS\_time\_out,tempGPS\_date,... 1107 tempGPS\_time, x, y, z, z\_AGL, v\_g, 'VariableNames',... 1108 {'Line Number', 'Time from boot (us)', 'Time from parse ... 1109 (sec)',... 'UTC Date', 'UTC Time', 'Lat', 'Long', 'Altitude (m, MSL)',... 1110 'Altitude (m, AGL)', 'Groundspeed (m/s)'}); 1111 1112 % Parsed Attidue data output 1113 ATT\_LN = ATT (TO\_ATT:LND\_ATT, 1); 1114 ATT\_time = ATT(TO\_ATT:LND\_ATT,2); 1115 ATT\_time\_out= (ATT\_time-min(ATT\_time))/1000000; 1116 ATT = [ATT\_LN, ATT\_time, ATT\_time\_out, rollAC, pitchAC, yawAC]; 1117 ATT\_label = { 'Line No', 'Time since boot (us) ',... 1118 'Time from Arming (sec)', 'Aircraft Roll (deg)',... 1119 'Aircraft Pitch (deg)', 'Aircraft Yaw (deg)'}; 1120 ATT\_table = table(ATT\_LN, ATT\_time, ATT\_time\_out, rollAC, ... 1121 pitchAC, yawAC, 'VariableNames', {'Line Number',... 1122 'Time from boot (us)', 'Time from parse (sec)',... 1123 'Aircraft Roll (deg)', 'Aircraft Pitch (deg)',... 1124 'Aircraft Yaw (deg, magnetic)'}); 1125 1126 if(strcmpi(arduPilotType, 'ArduCopter') | ... 1127 strcmpi(arduPilotType, 'CopterSonde')) 1128 else % Parsed CTUN data output 1129 CTUN\_LN = CTUN(TO\_CTUN:LND\_CTUN, 1); 1130 1131 CTUN\_time = CTUN (TO\_CTUN:LND\_CTUN, 2);

CTUN\_time\_out= (CTUN\_time-min(CTUN\_time))/1000000; 1132 CTUN = [CTUN\_LN, CTUN\_time, CTUN\_time\_out, v\_a]; 1133 CTUN\_label = { 'Line No', 'Time since boot (us) ',... 1134 'Time from Arming (sec)', 'Airspeed (m/s)'}; 1135 CTUN\_table = table(CTUN\_LN,CTUN\_time,CTUN\_time\_out,v\_a,... 1136 'VariableNames', {'Line Number', 'Time from boot (us)',... 1137 'Time from parse (sec)', 'Airspeed (m/s)'}); 1138 end 1139 1140 if(strcmpi(arduPilotType, 'CopterSonde')) 1141 % Parsed WIND (NKF2) data output 1142 NKF2\_LN = WIND(TO\_WIND:LND\_WIND,1); 1143 NKF2\_time = WIND(TO\_WIND:LND\_WIND,2); 1144 NKF2\_time\_out= (NKF2\_time\_min(NKF2\_time))/1000000; 1145 NKF2 = [NKF2\_LN, NKF2\_time, NKF2\_time\_out, VWN, VWE]; 1146 NKF2\_label = { 'Line No', 'Time since boot (us) ',... 1147 'Time from Arming (sec)', 'North Wind Vector ... 1148 (m/s)', 'East Wind Vector (m/s)'}; NKF2\_table = table(NKF2\_LN, NKF2\_time, NKF2\_time\_out,... 1149 VWN, VWE, wind, windANG, 'VariableNames', {'Line Number',... 1150 'Time from boot (us)', 'Time from parse (sec)',... 1151 'North Wind Vector (m/s)', 'East Wind Vector (m/s)',... 1152 'Wind Speed (m/s)', 'Wind Angle (deg)'}); 1153 1154 NKF1\_LN = NKF1 (TO\_NKF1:LND\_NKF1,1); 1155 NKF1\_time = NKF1(TO\_NKF1:LND\_NKF1,2); 1156 NKF1\_time\_out= (NKF1\_time\_min(NKF1\_time))/1000000; 1157 NKF1 = [NKF1\_LN, NKF1\_time, NKF1\_time\_out, Roll, Pitch,... 1158 Yaw, VN, VE, VD, PN, PE, PD]; 1159 NKF1\_label = { 'Line No', 'Time since boot (us) ', 'Time from ... 1160 Arming (sec)',... 'Roll', 'Pitch', 'Yaw', 'VN', 'VE', 'VD', 'PN', 'PE', 'PD'}; 1161

1162	NKF1_table = table(NKF1_LN, NKF1_time, NKF1_time_out, Roll,
1163	Pitch, Yaw, VN, VE, VD, PN, PE, PD, 'VariableNames',
1164	<pre>{'Line Number','Time from boot (us)','Time from parse</pre>
	(sec)',
1165	<pre>'Roll (deg)','Pitch (deg)','Yaw (deg)','Velocity North</pre>
	(m/s)',
1166	'Velocity East (m/s)', 'Velocity Down (m/s)',
1167	'Position North (m)', 'Position East (m)', 'Position Down
	(m) '});
1168	
1169	<pre>IMET_LN = IMET(TO_IMET:LND_IMET,1);</pre>
1170	<pre>IMET_time = IMET(TO_IMET:LND_IMET,2);</pre>
1171	<pre>IMET_time_out= (IMET_time-min(IMET_time))/1000000;</pre>
1172	CSIMET_table = table(IMET_LN, IMET_time, IMET_time_out,
1173	<pre>IMET(TO_IMET:LND_IMET,3), IMET(TO_IMET:LND_IMET,4),</pre>
1174	<pre>IMET(TO_IMET:LND_IMET,5), IMET(TO_IMET:LND_IMET,6),</pre>
1175	<pre>IMET(TO_IMET:LND_IMET,7), IMET(TO_IMET:LND_IMET,8),</pre>
1176	<pre>IMET(TO_IMET:LND_IMET,9), IMET(TO_IMET:LND_IMET,10),</pre>
1177	<pre>IMET(TO_IMET:LND_IMET,11), IMET(TO_IMET:LND_IMET,12),</pre>
	<pre>IMET(TO_IMET:LND_IMET,13),</pre>
	IMET(TO_IMET:LND_IMET,14),
	IMET(TO_IMET:LND_IMET,15),
	<pre>IMET(TO_IMET:LND_IMET,16),'VariableNames',{'LineNo','TimeUS','Time .</pre>
	from parse
	(sec)','RH1','RH2','RH3','RH4','T1','T2','T3','T4','Health1','Health
1178	
1179	else
1180	% Parsed NKF2 data output
1181	NKF2_LN = NKF2 (TO_NKF2:LND_NKF2,1);
1182	NKF2_time = NKF2(TO_NKF2:LND_NKF2,2);
1183	NKF2_time_out= (NKF2_time-min(NKF2_time))/1000000;
1184	NKF2 = [NKF2_LN, NKF2_time, NKF2_time_out, VWN, VWE];

1185	NKF2_label = {'Line No','Time since boot (us)',
1186	'Time from Arming (sec)', 'North Wind Vector
	<pre>(m/s)', 'East Wind Vector (m/s)'};</pre>
1187	NKF2_table = table(NKF2_LN, NKF2_time, NKF2_time_out,
1188	<pre>VWN, VWE, wind,windANG,'VariableNames',{'Line Number',</pre>
1189	'Time from boot (us)','Time from parse (sec)',
1190	'North Wind Vector (m/s)', 'East Wind Vector (m/s)',
1191	'Wind Speed (m/s)','Wind Angle (deg)'});
1192	
1193	NKF1_LN = NKF1(TO_NKF1:LND_NKF1,1);
1194	NKF1_time = NKF1(TO_NKF1:LND_NKF1,2);
1195	NKF1_time_out= (NKF1_time-min(NKF1_time))/1000000;
1196	NKF1 = [NKF1_LN, NKF1_time, NKF1_time_out, rollEKF,
1197	pitchEKF, yawEKF, vnEKF, veEKF, vdEKF, pnEKF, peEKF,
	pdEKF];
1198	NKF1_label = {'Line No','Time since boot (us)',
1199	'Time from Arming (sec)', 'Roll', 'Pitch', 'Yaw',
1200	'VN', 'VE', 'VD', 'PN', 'PE', 'PD'};
1201	NKF1_table = table(NKF1_LN, NKF1_time, NKF1_time_out,
1202	rollEKF, pitchEKF, yawEKF, vnEKF, veEKF, vdEKF,
1203	<pre>pnEKF, peEKF, pdEKF, 'VariableNames',{'Line Number',</pre>
1204	'Time from boot (us)','Time from parse (sec)','Roll',
1205	'Pitch','Yaw','VN', 'VE', 'VD','PN','PE','PD'});
1206	
1207	end
1208	
1209	% Parsed RCOU data output
1210	<pre>RCOU_LN = RCOU(TO_RCOU:LND_RCOU,1);</pre>
1211	<pre>RCOU_time = RCOU(TO_RCOU:LND_RCOU,2);</pre>
1212	<pre>RCOU_pitch = RCOU(TO_RCOU:LND_RCOU,3);</pre>
1213	<pre>RCOU_roll = RCOU(TO_RCOU:LND_RCOU, 4);</pre>
1214	<pre>RCOU_thr = RCOU(TO_RCOU:LND_RCOU,5);</pre>

1215	RCOU_yaw = RCOU(TO_RCOU:LND_RCOU, 6);
1216	RCOU_time_out = (RCOU_time-min(RCOU_time))/1000000;
1217	RCOU = [RCOU_LN, RCOU_time, RCOU_time_out, RCOU_pitch,
1218	RCOU_roll, RCOU_thr, RCOU_yaw];
1219	RCOU_label = {'Line No','Time since boot (us)',
1220	Time from Arming (sec)', 'C1 - Pitch PWM',
1221	<pre>' C2 - Roll PWM', 'C3 - Throttle PWM', 'C4 - Yaw PWM'};</pre>
1222	RCOU_table = table(RCOU_LN, RCOU_time, RCOU_time_out,
1223	RCOU_pitch, RCOU_roll, thr,RCOU_thr, RCOU_yaw,
1224	4 'VariableNames',{'Line Number','Time from boot (us)',
1225	Time from parse (sec)', 'Pitch PWM', 'Roll PWM', 'Throttle
	(%)',
1226	<pre>6 'Throttle PWM', 'Yaw PWM'});</pre>
1227	7
1228	8 % Parsed IMU data output
1229	<pre>iMU_LN = IMU(TO_IMU:LND_IMU,1);</pre>
1230	<pre>IMU_time = IMU(TO_IMU:LND_IMU,2);</pre>
1231	IMU_GyrX = IMU(TO_IMU:LND_IMU,3);
1232	<pre>IMU_GyrY = IMU(TO_IMU:LND_IMU,4);</pre>
1233	<pre>iMU_GyrZ = IMU(TO_IMU:LND_IMU,5);</pre>
1234	4 IMU_AccX = IMU(TO_IMU:LND_IMU,6);
1235	<pre>iMU_AccY = IMU(TO_IMU:LND_IMU,7);</pre>
1236	<pre>iMU_AccZ = IMU(TO_IMU:LND_IMU,8);</pre>
1237	<pre>TMU_time_out = (IMU_time-min(IMU_time))/1000000;</pre>
1238	s IMU = [IMU_LN, IMU_time, IMU_time_out, IMU_GyrX, IMU_GyrY,
1239	MU_GyrZ, IMU_AccX, IMU_AccY, IMU_AccZ];
1240	IMU_label = {'Line No','Time since boot (us)','Time from parse
	(sec)',
1241	X Gyro rotation ( /sec)','Y Gyro rotation ( /sec)','Z
	Gyro rotation ( /sec)',
1242	2 'X Acceleration ( /sec/sec)', 'Y Acceleration
	<pre>( /sec/sec)','Z Acceleration ( /sec/sec)'};</pre>

IMU\_table = table(IMU\_LN, IMU\_time, IMU\_time\_out, IMU\_GyrX, ... 1243 IMU\_GyrY,... IMU\_GyrZ, IMU\_AccX, IMU\_AccY, IMU\_AccZ, 'VariableNames',... 1244 {'Line Number','Time from boot (us)','Time from parse ... 1245 (sec)',... 'X Gyro rotation ( /sec)', 'Y Gyro rotation ( /sec)',... 1246 'Z Gyro rotation ( /sec)', 'X Acceleration ( /sec/sec)',... 1247 'Y Acceleration ( /sec/sec)', 'Z Acceleration ( /sec/sec)'}); 1248 1249 % Parsed Pixhawk barometric data 1250 BARO\_LN = BARO(TO\_BARO:LND\_BARO,1); 12511252BARO\_time = BARO(TO\_BARO:LND\_BARO,2); BARO\_time\_out = (BARO\_time\_min(BARO\_time))/1000000; 1253 BARO\_alt = BARO(TO\_BARO:LND\_BARO,3); 1254 BARO\_press = BARO (TO\_BARO:LND\_BARO, 4) /100; 1255 BARO = [BARO\_LN, BARO\_time, BARO\_time\_out, BARO\_press, BARO\_alt]; 1256BARO\_label = { 'Line No', 'Time since boot (us) ',... 1257 'Time from parse (sec)', 'Barometric pressure (mbar)',... 1258 'Barometric Altitude (m, AGL)'}; 1259BARO\_table = table (BARO\_LN, BARO\_time, BARO\_time\_out, BARO\_press, ... 1260 BARO\_alt, 'VariableNames', {'Line Number', 'Time from boot ... 1261(us)',... 'Time from parse (sec)', 'Barometric pressure (mbar)',... 1262 'Barometric Altitude (m, AGL)'}); 1263 1264 % Get the name of the intput.mat file and save as input\_parsed.mat 1265 baseFileName = sprintf('%s\_Parsed.mat', baseNameNoExtDFL); 1266 fullParsedMatFileName = fullfile(folderDFL, baseFileName); 1267app.LocationofOutputFilesEditField.Value = fullParsedMatFileName; 1268 % Save file with parsed data as the original filename plus the ... 1269 added portion 1270

```
if(strcmpi(arduPilotType, 'ArduCopter'))
1271
                 save(fullParsedMatFileName,'ATT_table','GPS_table','NKF2_table',...
1272
                     'RCOU_table', 'BARO_table', 'IMU_table');
1273
            elseif(strcmpi(arduPilotType, 'CopterSonde'))
1274
                 save(fullParsedMatFileName, 'ATT_table', 'GPS_table', 'NKF2_table', ...
1275
                     'RCOU_table', 'BARO_table', 'IMU_table', 'CSIMET_table');
1276
            else
1277
                 save(fullParsedMatFileName,'ATT_table','GPS_table','CTUN_table',...
1278
                     'NKF2_table', 'RCOU_table', 'BARO_table', 'IMU_table');
1279
1280
            end
        end
1281
        %% iMet Data Parsing and Output
1282
        if(strcmpi(DFL_NewOld, 'New'))
1283
            if (strcmpi(iMetValue, 'Yes'))
1284
1285
                 % Get the name of the file that the user wants to use.
1286
                 defaultFileNameiMet = fullfile(startingFolderDFL, '*.csv');
1287
                 [baseFileNameiMet, folderiMet] = ...
1288
                    uigetfile(defaultFileNameiMet,...
                     'Select an iMet .CSV file');
1289
                 if baseFileNameiMet == 0
1290
                     % User clicked the Cancel button.
1291
                     return;
1292
                 end
1293
1294
                 % Get the name of the input .mat file.
1295
                 fullInputMatFileNameiMet = fullfile(folderiMet, ...
1296
                    baseFileNameiMet);
                 % Get filename without the extension, used by Save Function
1297
                 [¬, baseNameNoExtiMet, ¬] = fileparts(baseFileNameiMet);
1298
                 % Load file in
1299
                 iMetData_temp = readtable(fullInputMatFileNameiMet);
1300
```

```
iMetData = iMetData_temp(2:end,:);
1301
                 clear iMetData_temp
1302
1303
                 dataLen = width(iMetData);
1304
1305
                 if(dataLen == 46)
1306
                     % Convert CSV data to general table to datetime array
1307
                     iMet_date_ref = iMetData(:,40);
1308
                     iMet_date_conv = table2array(iMet_date_ref);
1309
1310
                     iMet_date = datetime(iMet_date_conv, 'InputFormat', ...
                          'yyyy/MM/dd', 'Format', 'MMM-dd-yyyy');
1311
                     iMet_time_ref = iMetData(:,41);
1312
                     iMet_time_conv = table2array(iMet_time_ref);
1313
1314
                     iMet_time = datetime(datevec(iMet_time_conv), 'Format',...
                          'HH:mm:ss');
1315
                     Pix_time = GPS(:,3);
1316
1317
                 elseif(dataLen == 47)
1318
                     % Convert CSV data to general table to datetime array
1319
                     iMet_date_ref = iMetData(:, 41);
1320
                     iMet_date_conv = table2array(iMet_date_ref);
1321
                     iMet_date = datetime(iMet_date_conv, 'InputFormat', ...
1322
                          'yyyy/MM/dd', 'Format', 'MMM-dd-yyyy');
1323
                     iMet_time_ref = iMetData(:,42);
1324
                     iMet_time_conv = table2array(iMet_time_ref);
1325
                     iMet_time = datetime(datevec(iMet_time_conv), 'Format', ...
1326
                          'HH:mm:ss');
1327
                     Pix_time = GPS(:, 3);
1328
                 end
1329
1330
                 % Convert datetime arrays into datetime vectors
1331
1332
                 [iM_y, iM_m, iM_d] = datevec(iMet_date(:,1));
```

```
[ no, no, iM_h, iM_M, iM_s] = datevec(iMet_time);
1333
1334
                 iM_vec = [iM_y iM_m iM_d iM_h iM_M iM_s];
1335
                 GPS_vec = datevec(GPS_final);
1336
1337
                 % Convert datetime vectors datetime serials
1338
                 GPS_serial = datenum(GPS_vec);
1339
                 iMet_serial = datenum(iM_vec);
1340
1341
                 % Find iMet data at start and end of Pixhawk parsing
1342
                 TO_iMet = find(iMet_serial(:)≥min(GPS_serial),1,'first');
1343
                 LND_iMet = find(iMet_serial(:) >max(GPS_serial),1,'first');
1344
1345
                 % Parse iMet data
1346
                 iM_red = iM_vec(TO_iMet:LND_iMet,:);
1347
                 iMet_serial = iMet_serial(TO_iMet:LND_iMet);
1348
1349
                 % Add Pixhawk time since arming variable
1350
                 iter = 1;
1351
                 for i=1:length(iMet_serial)
1352
                     for j=iter:length(GPS_serial)
1353
                          if(iMet_serial(i) == GPS_serial(j))
1354
                              iM_Pix(i,1)=GPS_time_out(j);
1355
                              iter = j;
1356
                          end
1357
                     end
1358
                 end
1359
1360
                 for row = 1:length(iM_Pix)
1361
                     if iM_Pix(row, 1) == 0
1362
                          iM_Pix(row,1) = (iM_Pix(row-1,1)+iM_Pix(row+1,1))/2;
1363
1364
                     end
```

end 1365 1366 if(dataLen == 46)1367 i=TO\_iMet:LND\_iMet; 1368 iM\_time\_temp(:,1) = iMet\_time(i,1); 1369 iM\_date\_temp(:,1) = iMet\_date(i,1); 1370 iM\_pres\_temp(:,1) = iMetData(i,37); 1371 iM\_temp\_temp(:,1) = iMetData(i,38); 1372 iM\_humid\_temp(:,1) = iMetData(i,39); 1373 iM\_lat\_temp(:,1) = iMetData(i,42); 1374 iM\_long\_temp(:,1) = iMetData(i,43); 1375  $iM_alt_temp(:, 1) = iMetData(i, 44);$ 1376 iM\_sat\_temp(:,1) = iMetData(i,45); 1377 elseif(dataLen == 47) 1378 i=TO\_iMet:LND\_iMet; 1379 iM\_time\_temp(:,1) = iMet\_time(i,1); 1380 iM\_date\_temp(:,1) = iMet\_date(i,1); 1381 iM\_pres\_temp(:,1) = iMetData(i,37); 1382 iM\_temp\_temp(:,1) = iMetData(i,38); 1383 iM\_humid\_temp(:,1) = iMetData(i,39); 1384 iM\_humid\_temp\_t(:,1) = iMetData(i,40); 1385  $iM_{lat_temp}(:, 1) = iMetData(i, 43);$ 1386 iM\_long\_temp(:,1) = iMetData(i,44); 1387 iM\_alt\_temp(:,1) = iMetData(i,45); 1388 iM\_sat\_temp(:,1) = iMetData(i,46); 1389 end 1390 1391  $len1 = size(iM_Pix, 1);$ 1392 1393 len2 = size(iM\_time\_temp, 1); 1394 if(dataLen == 46)1395 1396 if(len1>len2)

13	97	<pre>iM_Pix = iM_Pix(1:len2);</pre>
13	98	<pre>iM_time = datestr(iM_time_temp(1:len2,1),'HH:MM:ss');</pre>
13	99	iM_date =
		<pre>datestr(iM_date_temp(1:len2,1),'mmm-dd-yyyy');</pre>
14	00	<pre>iM_pres = table2array(iM_pres_temp(1:len2,1));</pre>
14	01	<pre>iM_temp = table2array(iM_temp_temp(1:len2,1));</pre>
14	02	<pre>iM_humid = table2array(iM_humid_temp(1:len2,1));</pre>
14	03	<pre>iM_lat = table2array(iM_lat_temp(1:len2,1));</pre>
14	04	<pre>iM_long = table2array(iM_long_temp(1:len2,1));</pre>
14	05	<pre>iM_alt = table2array(iM_alt_temp(1:len2,1));</pre>
14	06	<pre>iM_sat = table2array(iM_sat_temp(1:len2,1));</pre>
14	07 else	eif(len2>len1)
14	08	<pre>iM_Pix = iM_Pix(1:len1);</pre>
14	09	<pre>iM_time = datestr(iM_time_temp(1:len1,1),'HH:MM:ss');</pre>
14	10	iM_date =
		<pre>datestr(iM_date_temp(1:len1,1),'mmm-dd-yyyy');</pre>
14	.11	<pre>iM_pres = table2array(iM_pres_temp(1:len1,1));</pre>
14	12	<pre>iM_temp = table2array(iM_temp_temp(1:len1,1));</pre>
14	13	<pre>iM_humid = table2array(iM_humid_temp(1:len1,1));</pre>
14	.14	<pre>iM_lat = table2array(iM_lat_temp(1:len1,1));</pre>
14	15	<pre>iM_long = table2array(iM_long_temp(1:len1,1));</pre>
14	16	<pre>iM_alt = table2array(iM_alt_temp(1:len1,1));</pre>
14	.17	<pre>iM_sat = table2array(iM_sat_temp(1:len1,1));</pre>
14	18 else	2
14	19	<pre>iM_Pix = iM_Pix(1:len1);</pre>
14	20	<pre>iM_time = datestr(iM_time_temp(:,1),'HH:MM:ss');</pre>
14	21	<pre>iM_date = datestr(iM_date_temp(:,1),'mmm-dd-yyyy');</pre>
14	22	<pre>iM_pres = table2array(iM_pres_temp(:,1));</pre>
14	23	<pre>iM_temp = table2array(iM_temp_temp(:,1));</pre>
14	24	<pre>iM_humid = table2array(iM_humid_temp(:,1));</pre>
14	25	<pre>iM_lat = table2array(iM_lat_temp(:,1));</pre>
14	26	<pre>iM_long = table2array(iM_long_temp(:,1));</pre>

```
iM_alt = table2array(iM_alt_temp(:,1));
1427
                         iM_sat = table2array(iM_sat_temp(:,1));
1428
                     end
1429
1430
                     iMet_table = table(iM_Pix, iM_date, iM_time, iM_pres,...
1431
                         iM_temp, iM_humid, iM_lat, iM_long, iM_alt, iM_sat,...
1432
                          'VariableNames', {'Time from Arming (sec)', 'Date ...
1433
                             UTC',...
                          'Time UTC', 'Barometric Pressure (hPa)',...
1434
                          'Air Temp ( C )', 'Relative Humidity (%)',...
1435
                          'GPS Lat', 'GPS Long', 'GPS Alt (m)', 'Sat Count'});
1436
                     save(fullParsedMatFileName, 'iMet_table', '-append');
1437
1438
                 elseif(dataLen == 47)
1439
                     if(len1>len2)
1440
                         iM_Pix = iM_Pix(1:len2);
1441
                         iM_time = datestr(iM_time_temp(1:len2,1),'HH:MM:ss');
1442
                         iM_date = \dots
1443
                             datestr(iM_date_temp(1:len2,1), 'mmm-dd-yyyy');
                         iM_pres = table2array(iM_pres_temp(1:len2,1));
1444
                         iM_temp = table2array(iM_temp_temp(1:len2,1));
1445
                         iM_humid = table2array(iM_humid_temp(1:len2,1));
1446
                         iM_humid_t = table2array(iM_humid_temp_t(1:len2,1));
1447
                         iM_lat = table2array(iM_lat_temp(1:len2,1));
1448
                         iM_long = table2array(iM_long_temp(1:len2,1));
1449
                         iM_alt = table2array(iM_alt_temp(1:len2,1));
1450
                         iM_sat = table2array(iM_sat_temp(1:len2,1));
1451
                     elseif(len2>len1)
1452
1453
                         iM_Pix = iM_Pix(1:len1);
                         iM_time = datestr(iM_time_temp(1:len1,1), 'HH:MM:ss');
1454
                         iM_date = ...
1455
                             datestr(iM_date_temp(1:len1,1), 'mmm-dd-yyyy');
```

1456	<pre>im_pres = table2array(im_pres_temp(1:len1,1));</pre>
1457	<pre>iM_temp = table2array(iM_temp_temp(1:len1,1));</pre>
1458	<pre>iM_humid = table2array(iM_humid_temp(1:len1,1));</pre>
1459	<pre>iM_humid_t = table2array(iM_humid_temp_t(1:len1,1));</pre>
1460	<pre>iM_lat = table2array(iM_lat_temp(1:len1,1));</pre>
1461	<pre>iM_long = table2array(iM_long_temp(1:len1,1));</pre>
1462	<pre>iM_alt = table2array(iM_alt_temp(1:len1,1));</pre>
1463	<pre>iM_sat = table2array(iM_sat_temp(1:len1,1));</pre>
1464	else
1465	<pre>iM_Pix = iM_Pix(1:len1);</pre>
1466	<pre>iM_time = datestr(iM_time_temp(:,1),'HH:MM:ss');</pre>
1467	<pre>iM_date = datestr(iM_date_temp(:,1),'mmm-dd-yyyy');</pre>
1468	<pre>iM_pres = table2array(iM_pres_temp(:,1));</pre>
1469	<pre>iM_temp = table2array(iM_temp_temp(:,1));</pre>
1470	<pre>iM_humid = table2array(iM_humid_temp(:,1));</pre>
1471	<pre>iM_humid_t = table2array(iM_humid_temp_t(:,1));</pre>
1472	<pre>iM_lat = table2array(iM_lat_temp(:,1));</pre>
1473	<pre>iM_long = table2array(iM_long_temp(:,1));</pre>
1474	<pre>iM_alt = table2array(iM_alt_temp(:,1));</pre>
1475	<pre>iM_sat = table2array(iM_sat_temp(:,1));</pre>
1476	end
1477	
1478	<pre>iMet_table = table(iM_Pix, iM_date, iM_time, iM_pres,</pre>
1479	<pre>iM_temp, iM_humid, iM_humid_t, iM_lat, iM_long,</pre>
	iM_alt,
1480	<pre>iM_sat,'VariableNames',{'Time from Arming (sec)',</pre>
1481	'Date UTC', 'Time UTC', 'Barometric Pressure (hPa)',
1482	'Air Temp ( C )', 'Relative Humidity (%)',
1483	'Humidity Temperature ( C )','GPS Lat','GPS Long',
1484	'GPS Alt (m)', 'Sat Count'});
1485	<pre>save(fullParsedMatFileName,'iMet_table','-append');</pre>
1486	end

```
1487
                 fig3=figure(3);
1488
                 fig3.Name = 'Parsed iMet data.';
1489
                 set(fig3,'defaultLegendAutoUpdate','off');
1490
                 yyaxis left
1491
                 plt = plot(iM_Pix(:), iM_temp(:),'y-',iM_Pix(:),iM_humid,'c-');
1492
                 title('Temp, Humidity, and Pressure vs Time')
1493
                 xlabel('Time (ms)');
1494
                 ylabel('Temp ( C ) and Humidity (%)');
1495
1496
                 yyaxis right
1497
                 plt = plot(iM_Pix(:), iM_pres(:), 'k-');
1498
                 ylabel('Pressure (hPa)');
1499
                 legend({'iMet Temp','iMet Humid','iMet ...
1500
                     Pres'}, 'Location', 'southeast')
1501
                 if(strcmpi(sensorOut, 'Yes'))
1502
                     % Get the name of the input.mat file and save as ...
1503
                         input_Parsed_TPH.csv
                     baseFileName = sprintf('%s_iMet.csv', baseNameNoExtDFL);
1504
                     fullOutputMatFileName = fullfile(folderDFL, baseFileName);
1505
                     % Write data to text file
1506
                     writetable(iMet_table, fullOutputMatFileName);
1507
                 end
1508
             end
1509
1510
        elseif(strcmpi(DFL_NewOld, 'Old'))
1511
             if(exist('iMet_table', 'var'))
1512
1513
                 if (strcmpi(iMetValue, 'Yes'))
1514
                     fig3=figure(3);
1515
1516
                     fig3.Name = 'Parsed iMet data.';
```

```
set(fig3,'defaultLegendAutoUpdate','off');
1517
                     yyaxis left
1518
                     plt = plot(iMet_table{:,1}, iMet_table{:,5}, 'y-',...
1519
                          iMet_table{:,1}, iMet_table{:,6}, 'c-');
1520
                     title('Temp, Humidity, and Pressure vs Time')
1521
                     xlabel('Time (ms)');
1522
                     ylabel('Temp ( C ) and Humidity (%)');
1523
1524
                     yyaxis right
1525
                     plt = plot(iMet_table{:,1}, iMet_table{:,4}, 'k-');
1526
                     ylabel('Pressure (hPa)');
1527
                     legend({'iMet Temp','iMet Humid','iMet ...
1528
                         Pres'}, 'Location', 'southeast')
1529
                     if(strcmpi(sensorOut, 'Yes'))
1530
                          % Get the name of the input.mat file and save as ...
1531
                             input_Parsed_TPH.csv
                          baseFileName = sprintf('%s_iMet_NS.csv', ...
1532
                             baseNameNoExtDFL);
                          fullOutputMatFileName = fullfile(folderDFL, ...
1533
                             baseFileName);
                          % Write data to text file
1534
                          writetable(iMet_table, fullOutputMatFileName);
1535
                     end
1536
                 end
1537
            end
1538
        end
1539
        %% 5HP Data Parsing and Output
1540
1541
        if (strcmpi(mhpValue, 'Yes') | strcmpi(tphValue, 'Yes'))
            if(strcmpi(DFL_NewOld, 'New'))
1542
                 % Get the name of the file that the user wants to use.
1543
1544
                 defaultFileNameMHP = fullfile(startingFolderDFL, '*.csv');
```

```
[baseFileNameMHP, folderMHP] = uigetfile(defaultFileNameMHP,...
1545
                     'Select a 5HP/TPH .CSV file');
1546
                if baseFileNameMHP == 0
1547
                     % User clicked the Cancel button.
1548
                    return:
1549
                end
1550
1551
                % Get the name of the input .mat file.
1552
                fullInputMatFileNameMHP = fullfile(folderMHP, baseFileNameMHP);
1553
                % Get filename without the extension, used by Save Function
1554
                 [¬, baseNameNoExtMHP, ¬] = fileparts(baseFileNameMHP);
1555
1556
                data = readmatrix(fullInputMatFileNameMHP);
1557
1558
                data(isnan(data)) = -1;
1559
                list = {'Probe 1', 'Probe 2', 'Probe 3', 'Metal Probe'};
1560
                [Probe, tf] = listdlg('ListString', list, ...
1561
                    'SelectionMode', 'single');
1562
                if Probe==1
1563
                     응응
1564
                     Probe_matrix =reshape([-45 5.4503245929122812 -40 ...
1565
                        3.6344070134585587 ...
                         -35 2.6910078916419944 -30 2.0741227786203615 ...
1566
                         -25 1.6177891317119037 -20 1.3047139241276557 ...
1567
                         -15 0.99765981580647944 -10 0.72488341948754176 ...
1568
                         -5 0.42287858693961322 0 0.11269996848776841 5 ...
1569
                         -0.18844278909812986 10 -0.51552305940602694 15 ...
1570
                         -0.82527532291594985 20 -1.0827862740935248 25 ...
1571
                         -1.3747195966697383 30 -1.7399669009211871 35 ...
1572
                         -2.2261994229329716 40 -2.83647895638291 45 ...
1573
                            -3.7775455967008882 ...
```

1574	-45 5.6088602743478519 -40 3.8471216439542708
1575	-35 2.758014522232052 -30 2.1393612470974079 -25
1576	1.6846516032825758 -20 1.2812487641404717 -15
1577	0.99769242818437875 -10 0.7108466648491224 -5
1578	0.39522993924508154 0 0.055666210297281971 5
	-0.26040468531481231
1579	10 -0.60604077802968226 15 -0.91007141723872353
1580	20 -1.2115256139194885 25 -1.5914907297890892
1581	30 -2.0768170988494874 35 -2.7082025856213936
1582	40 -3.6672797432896682 45 -5.11308906747139 -45
1583	0.40153363672050385 -40 0.58252778411973083 -35
1584	0.76207165540900135 -30 0.91124757982052851 -25
1585	1.0297461281014839 -20 1.1162791106122567 -15
1586	1.1602402228108957 -10 1.1682637817844033 -5
	1.1717460580991859
1587	0 1.1705980442512705 5 1.1742807474113173 10
	1.1750681375621821
1588	15 1.1761267609636077 20 1.1541350590812145 25
1589	1.0909288620176854 30 0.99477056787113616 35
	0.8677013434826869
1590	40 0.71724821586753418 45 0.55890366509478218
1591	-45 0.45693864133494283 -40 0.63502634767094046
1592	-35 0.81900446405055538 -30 0.98505044428739341
1593	-25 1.0967311790193095 -20 1.1760591378284164
1594	-15 1.2105339434476543 -10 1.225378332775521 -5
1595	1.2208141810618345 0 1.2167930824763777 5
	1.213004218757056
1596	10 1.202293865820901 15 1.1664968024605715 20
1597	1.1003915504970527 25 0.99694948648152915 30
	0.86274285528505079
1598	35 0.70827366374370193 40 0.5425004640691633 45
1599	0.3714015525546091], 2, 19, 4);

1600	88
1601	end
1602	if Probe==2
1603	88
1604	Probe_matrix= reshape([-45 8.0413125757286483 -40
	4.595840988208745 -35
1605	3.1960983613503395 -30 2.3438686451680391 -25
1606	1.7998009251294775 -20 1.3568843785915969 -15
1607	1.0417025439286454 -10 0.6719271337912911 -5
	0.31594859127806313
1608	0 -0.027775383920766113 5 -0.38525870149950231
1609	10 -0.76967960249393741 15 -1.0870073516266328
1610	20 -1.378333118288662 25 -1.7338076966226417 30
1611	-2.1416605956464423 35 -2.672348394280156 40
	-3.4957062360550872
1612	45 -4.9947263023335022 -45 5.4644672199708184
1613	-40 3.7051459239570832 -35 2.7802502582437754
1614	-30 2.1909163813094055 -25 1.745552975533075 -20
1615	1.3193857267226774 -15 0.989568212585923 -10
	0.68976185302006032
1616	-5 0.34537219692610427 0 0.00070104058184836159
1617	5 -0.35188215190568023 10 -0.70334097827774844
1618	15 -1.0151026074394103 20 -1.3372013554379576
1619	25 -1.7119408626720383 30 -2.1780237220699625
1620	35 -2.7495578563807559 40 -3.577709672348806 45
1621	-4.9039267737627332 -45 0.27126131923904334 -40
1622	0.438829235484768 -35 0.6069713454411777 -30
	0.76007586890568246
1623	-25 0.8757553537646996 -20 0.97230935336276991
1624	-15 1.0163124604875853 -10 1.0494565940075169
1625	-5 1.0440081543412782 0 1.0521140361945402 5
	1.0617346368847278

1626	10 1.0640740038270755 15 1.0530416652892758 20
1627	1.0150305427048212 25 0.93933046605058423 30
	0.84632064730305567
1628	35 0.71412230618293648 40 0.56257867097066894
1629	45 0.4088708445506693 -45 0.40039661393475073
1630	-40 0.56253456955569414 -35 0.71741173034906691
1631	-30 0.83895154863814991 -25 0.945508433447679
1632	-20 1.048989338380182 -15 1.1079676863677939 -10
1633	1.1217507631454158 -5 1.1213365893922569 0
	1.1102349448662681
1634	5 1.1103023687637184 10 1.0907471029595337 15
1635	1.0731456995653772 20 1.0326480971143717 25
	0.96008313343852325
1636	30 0.85664753210489064 35 0.74804723263594852
1637	40 0.60691269332640829 45 0.47048729685494273], 2,
	19, 4);
1638	응응 
1639	end
1640	if Probe==3
1641	88
1642	Probe_matrix= reshape([-45 8.0413125757286483 -40
1642	Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35
1642 1643	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25</pre>
1642 1643 1644	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15</pre>
1642 1643 1644 1645	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5</pre>
1642 1643 1644 1645	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5 0.31594859127806313</pre>
1642 1643 1644 1645	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5 0.31594859127806313 0 -0.027775383920766113 5 -0.38525870149950231</pre>
1642 1643 1644 1645 1646	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5 0.31594859127806313 0 -0.027775383920766113 5 -0.38525870149950231 10 -0.76967960249393741 15 -1.0870073516266328</pre>
1642 1643 1644 1645 1646 1647 1648	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5 0.31594859127806313 0 -0.027775383920766113 5 -0.38525870149950231 10 -0.76967960249393741 15 -1.0870073516266328 20 -1.378333118288662 25 -1.7338076966226417 30</pre>
1642 1643 1644 1645 1646 1647 1648 1649	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5 0.31594859127806313 0 -0.027775383920766113 5 -0.38525870149950231 10 -0.76967960249393741 15 -1.0870073516266328 20 -1.378333118288662 25 -1.7338076966226417 30 -2.1416605956464423 35 -2.672348394280156 40</pre>
1642 1643 1644 1645 1646 1647 1648 1649	<pre>Probe_matrix= reshape([-45 8.0413125757286483 -40 4.595840988208745 -35 3.1960983613503395 -30 2.3438686451680391 -25 1.7998009251294775 -20 1.3568843785915969 -15 1.0417025439286454 -10 0.6719271337912911 -5 0.31594859127806313 0 -0.027775383920766113 5 -0.38525870149950231 10 -0.76967960249393741 15 -1.0870073516266328 20 -1.378333118288662 25 -1.7338076966226417 30 -2.1416605956464423 35 -2.672348394280156 40 -3.4957062360550872</pre>

1	651	-40 3.7051459239570832 -35 2.7802502582437754
1	652	-30 2.1909163813094055 -25 1.745552975533075 -20
1	653	1.3193857267226774 -15 0.989568212585923 -10
		0.68976185302006032
1	654	-5 0.34537219692610427 0 0.00070104058184836159
1	655	5 -0.35188215190568023 10 -0.70334097827774844
1	656	15 -1.0151026074394103 20 -1.3372013554379576
1	657	25 -1.7119408626720383 30 -2.1780237220699625
1	658	35 -2.7495578563807559 40 -3.577709672348806 45
1	659	-4.9039267737627332 -45 0.27126131923904334 -40
1	660	0.438829235484768 -35 0.6069713454411777 -30
		0.76007586890568246
1	661	-25 0.8757553537646996 -20 0.97230935336276991
1	662	-15 1.0163124604875853 -10 1.0494565940075169
1	663	-5 1.0440081543412782 0 1.0521140361945402 5
		1.0617346368847278
1	664	10 1.0640740038270755 15 1.0530416652892758 20
1	665	1.0150305427048212 25 0.93933046605058423 30
		0.84632064730305567
1	666	35 0.71412230618293648 40 0.56257867097066894
1	667	45 0.4088708445506693 -45 0.40039661393475073
1	668	-40 0.56253456955569414 -35 0.71741173034906691
1	669	-30 0.83895154863814991 -25 0.945508433447679
1	670	-20 1.048989338380182 -15 1.1079676863677939 -10
1	671	1.1217507631454158 -5 1.1213365893922569 0
		1.1102349448662681
1	672	5 1.1103023687637184 10 1.0907471029595337 15
1	673	1.0731456995653772 20 1.0326480971143717 25
		0.96008313343852325
1	674	30 0.85664753210489064 35 0.74804723263594852
1	675	40 0.60691269332640829 45 0.47048729685494273], 2,
		19, 4);

1676	88
1677	end
1678	if Probe==4
1679	8 8 8
1680	Probe_matrix=reshape([-45 6.8270286994312324 -40
	3.9272264704959947
1681	-35 2.7098630146680343 -30 2.0311527134073506
1682	-25 1.5648399326122149 -20 1.2097555239893774
1683	-15 0.99401689018022032 -10 0.7109304970157786
1684	-5 0.39798725583059297 0 0.10610929666786524 5
1685	-0.16157327211291675 10 -0.45721492832552874
	15
1686	-0.73051006787347961 20 -1.0206346484460449 25
1687	-1.3758360236897476 30 -1.7831603701441343 35
1688	-2.377357311604094 40 -3.464604213809888 45
	-5.902768693651109
1689	-45 7.4133939397891409 -40 4.1198674730828735
1690	-35 2.7770897996831905 -30 2.0145359604997175
1691	-25 1.5078827091567195 -20 1.1318223410844281
1692	-15 0.86433128769217948 -10 0.5558516056739059
1693	-5 0.26359452098651637 0 -0.028656763433599269
1694	5 -0.34839374505470783 10 -0.65948481869867559
1695	15 -0.96825549560038959 20 -1.3025459837101847
1696	25 -1.7237901125815069 30 -2.2978387224876857
1697	35 -3.2329872283805456 40 -5.1776511809601873
1698	45 -13.031656173662968 -45 0.32254048615943415
1699	-40 0.52205954626286333 -35
	0.68966672023071152
1700	-30 0.831269693794436 -25 0.9490842976748094
	-20
1701	1.0427975635210041 -15 1.0930205988684738 -10

1	
1702	1.1249929311810012 -5 1.1393737524485081 0
	1.1365874970433751
1703	5 1.1398409797574214 10 1.1325220268485772 15
1704	1.1008008398450571 20 1.0435222498791885 25
	0.96164376880806623
1705	30 0.84143329041748371 35 0.69288464852455156
1706	40 0.52373434130229513 45 0.32906938976100375
1707	-45 0.290471428675032 -40 0.50325598712500985
1708	-35 0.68854268208577729 -30
	0.84765291448530922
1709	-25 0.97823914779086751 -20 1.0667329519213851
1710	-15 1.1137584193430363 -10 1.1390572418989386
1711	-5 1.1394729063509146 0 1.1372390682609228 5
	1.1429838322264543
1712	10 1.1285412771221324 15 1.0857994109712712 20
1713	1.0227196623145753 25 0.90281324987619327 30
	0.75359921806475
1714	35 0.5861378670636509 40 0.38887273357229107
	45
1715	0.15966154155110446], 2, 19, 4);
1716	<u>୧</u> ୧
1717	end
1718	<pre>nrows = length(data(:,1));</pre>
1719	<pre>ncols = length(data(1,:));</pre>
1720	
1721	rho=1.197; %kg/m3 Lets move this down (calc it)
1722	
1723	CTUNTemp = CTUN;
1724	
1725	Pixcount=0;
1726	SScount=0;
1727	

i=1:nrows; % vectorize instead of for loop (speed) 1728 1729 time = data(i, 1); 1730 PitotB1 = data(i,2);1731 PitotB2 = data(i,3);1732 AlphaB1 = data(i, 6);1733 AlphaB2 = data(i,7);1734BetaB1 = data(i, 10);1735 BetaB2 = data(i, 11);1736H1 = data(i, 15);1737 T1 = data(i, 16);1738 H2 = data(i, 19);1739 T2 = data(i, 20);1740 H3 = data(i, 23);1741 T3 = data(i, 24);1742 UnixT = data(i, 26);1743PixT = data(i, 27);17441745 PitotCount = ((PitotB1\*256)+PitotB2); 1746 AlphaCount = ((AlphaB1\*256)+AlphaB2); 1747 BetaCount = ((BetaB1\*256)+BetaB2); 1748 1749 Pitot\_psi=((PitotCount-1638)\*(1+1))/(14745-1638)-1; 1750 Alpha\_psi=((AlphaCount-1638) \* (1+1))/(14745-1638)-1; 1751 $Beta_psi=((BetaCount-1638) * (1+1)) / (14745-1638) -1;$ 1752 1753Pitot\_pa=Pitot\_psi\*6894.74; 1754 Alpha\_pa=Alpha\_psi\*6894.74; 1755 Beta\_pa=Beta\_psi\*6894.74; 1756 1757 %Cp Calc 1758 1759 CP\_a=Alpha\_pa./Pitot\_pa;

```
1760
                 CP_b=Beta_pa./Pitot_pa;
1761
                 % Calculate alpha and beta probe values
1762
                 Alpha=interp1(Probe_matrix(2,:,1), Probe_matrix(1,:,1),...
1763
                     CP_a(i), 'linear', 45); %just doing 1d interp for now ...
1764
                         until more speeds ran
                 Beta=interp1(Probe_matrix(2,:,2), Probe_matrix(1,:,2),...
1765
                 CP_b(i), 'linear', 45);
1766
1767
                 CP_pitot1 = interp1(Probe_matrix(1,:,4),...
1768
                     Probe_matrix(2,:,4), Beta(i), 'makima', .5);
1769
                 CP_pitot2 = interp1(Probe_matrix(1,:,3),...
1770
                     Probe_matrix(2,:,3), Alpha(i), 'makima', .5);
1771
1772
                 for i=1:nrows
1773
                     UnixTime = UnixT(i);
1774
                     Temp = T1(i);
1775
                     if (UnixTime≠-1)
1776
                         Pixcount=Pixcount+1;
1777
                         PixData(Pixcount,1)=time(i); % Sensor board time
1778
                         PixData(Pixcount,2)=UnixTime; % Unix time from Pix GPS
1779
                         PixData(Pixcount, 3) = PixT(i); % Pixhawk board time
1780
                     end
1781
1782
                     if (Temp\neq-1)
1783
                          SScount=SScount+1;
1784
                          THSense(SScount,1)=time(i); % Sensor board time
1785
                          THSense(SScount, 2) =-1; % Will become Pixhawk ...
1786
                             board time
                          THSense(SScount, 3)=T1(i);
1787
                          THSense(SScount, 4)=T2(i);
1788
1789
                          THSense(SScount, 5)=T3(i);
```

```
THSense(SScount, 6)=H1(i);
1790
1791
                          THSense (SScount, 7) = H2(i);
                          THSense(SScount, 8)=H3(i);
1792
                     end
1793
1794
                     if CP_pitot1(i) > CP_pitot2(i)
1795
                          CP_pitot(i) = CP_pitot2(i);
1796
                     else
1797
                          CP_pitot(i) = CP_pitot1(i);
1798
1799
                     end
1800
                     U(i) = ((2/rho) * (abs(Pitot_pa(i)/CP_pitot(i)))) .^{.5};
1801
1802
1803
                 end
1804
                 u = U'.*cosd(Alpha).*cosd(Beta);
1805
                 v = U' \cdot sind(Beta);
1806
                 w = U'.*sind(Alpha).*cosd(Beta);
1807
                 total = sqrt(abs(u).^2 + abs(v).^2 + abs(2).^2);
1808
1809
                 for i=1:nrows
1810
                 MHPData(i,1)=time(i);
                                                    % Sensor board time
1811
                                                 % Will become Pixhawk board time
                 MHPData(i,2)=-1;
1812
                 MHPData(i,3) = u(i);
                                                    % probe u velocity
1813
                                                   % probe v velocity
                 MHPData(i, 4) = v(i);
1814
                 MHPData(i, 5) = w(i);
                                                    % probe w velocity
1815
                 MHPData(i,6)=Alpha(i);
                                                  % alpha angle of the probe
1816
                 MHPData(i,7)=Beta(i); % beta angle of the probe
1817
                 MHPData(i, 8) = 0;
1818
                 MHPData(i, 9) = 0;
1819
                 MHPData(i,8)=Alpha_MA(i);
1821 %
                 MHPData(i,9)=Beta_MA(i);
```

```
MHPData(i,10)=total(i); % calculated total velocity ...
1822
                     of the probe (body frame)
                 MHPData(i, 11) = 0;
1823
                 MHPData(i, 12) = 0;
1824
                 MHPData(i, 13) = 0;
1825
                 MHPData(i, 14) = 0;
1826
                 end
1827
1828
                 % calculation of difference between teensy time and pixtime
1829
                 if (PixData(:, 3) == -1)
1830
                     serialGPS = posixtime(GPS_final);
1831
                      tempPixTime = GPS_table(:,2)/100000;
1832
1833
                      for i=length(GPS_table(:,1))
1834
                          offset(i) = serialGPS(i) - tempPixTime(i);
1835
                      end
1836
                     AvOffset = mean(offset);
1837
                      PixData(:,3) = PixData(:,2) - AvOffset(:);
1838
                 end
1839
1840
                 for i=1:length(PixData(:,1))
1841
                      PixOff(i) = PixData(i, 3) - PixData(i, 1);
1842
                      GPS_off(i)=PixData(i,2)-PixData(i,1)/1000;
1843
                 end
1844
1845
                 PixAv=mean(PixOff);
1846
                 GPS_av = mean(GPS_off);
1847
1848
                 leapseconds_unix = 28;
1849
1850
                 % Offsets between board time and Pix time
1851
1852
                 MHPData(:,2)=round((MHPData(:,1)+PixAv),0);
```
```
MHPData_Unix=round((MHPData(:,1)/1000)+GPS_av,1);
1853
1854
                 %% REST OF CODE
1855
1856
                 MHP_DateTime=datetime(MHPData_Unix,...
1857
                      'ConvertFrom', 'posixTime', 'Format', 'MMM-dd-yyyy ...
1858
                          HH:mm:ss.S');
                 MHP_Date=datestr(MHP_DateTime, 'mmm-dd-yyyy');
1859
                 MHP_Time=datestr(MHP_DateTime, 'HH:MM:SS.FFF');
1860
1861
                 if(exist('THSense', 'var'))
1862
                      THSense(:,2) = round((THSense(:,1) + PixAv),0);
1863
                      THSense_Unix=round((THSense(:,1)/1000)+GPS_av,1);
1864
1865
                      TH_DateTime=datetime(THSense_Unix, 'ConvertFrom', ...
1866
                           'posixTime', 'Format', 'MMM-dd-yyyy HH:mm:ss.S');
1867
                      TH_Date=datestr(TH_DateTime, 'mmm-dd-yyyy');
1868
                      TH_Time=datestr(TH_DateTime, 'HH:MM:SS.FFF');
1869
                 end
1870
1871
                 if (min(MHP_DateTime) < min(GPS_final))</pre>
1872
                      TO_MHP = find(MHP_DateTime(:) >min(GPS_final),1,'first');
1873
                 else
1874
                      TO_MHP = 1;
1875
                 end
1876
1877
                 if(exist('THSense', 'var'))
1878
                      % 5HP and TPH parsing
1879
                      if (min(TH_DateTime) < min(GPS_final))</pre>
1880
                           TO_TPH = find(TH_DateTime(:) >min(GPS_final),1,'first');
1881
                      else
1882
1883
                           TO_TPH = 1;
```

```
1884
                     end
1885
                 end
1886
                 if (max(MHP_DateTime) > max(GPS_final))
1887
                      LND_MHP = find(MHP_DateTime(:) >max(GPS_final),1,'first');
1888
                 else
1889
                     LND_MHP = length(MHP_DateTime);
1890
                 end
1891
1892
                 if(exist('THSense', 'var'))
1893
                      if (max(TH_DateTime) > max(GPS_final))
1894
                          LND_TPH = \ldots
1895
                              find(TH_DateTime(:) >max(GPS_final),1,'first');
1896
                      else
                          LND_TPH = length (TH_DateTime);
1897
                      end
1898
1899
                      TPH_entry = THSense(TO_TPH:LND_TPH,:);
1900
                      TH_Date = TH_Date(TO_TPH:LND_TPH,:);
1901
                      TH_Time = TH_Time(TO_TPH:LND_TPH,:);
1902
                      TPH_time_out = (TPH_entry(:,1)-min(TPH_entry(:,1)))/1000;
1903
1904
                 end
1905
                 MHP_entry = MHPData(TO_MHP:LND_MHP,:);
1906
                 MHP_Date = MHP_Date(TO_MHP:LND_MHP,:);
1907
                 MHP_Time = MHP_Time(TO_MHP:LND_MHP,:);
1908
                 MHP_time_out = (MHP_entry(:,1)-min(MHP_entry(:,1)))/1000;
1909
1910
1911
                 Pix_time_out = (PixData(:,1)-min(PixData(:,1)))/1000;
1912
                 % Create tables with the data and variable names
1913
```

1914MHP\_table = ... table(MHP\_entry(:,1),MHP\_entry(:,2),MHP\_time\_out,... MHP\_Date, MHP\_Time, MHP\_entry(:, 3), MHP\_entry(:, 4),... 1915 MHP\_entry(:,5), MHP\_entry(:,6), MHP\_entry(:,7), MHP\_entry(:,8), ... 1916 MHP\_entry(:, 9), MHP\_entry(:, 10), MHP\_entry(:, 11), ... 1917 MHP\_entry(:,12), MHP\_entry(:,13), MHP\_entry(:,14),... 1918 'VariableNames', {'Board Time from PowerUp (msec)',... 1919 'Pix Time from PowerUp (msec)', 'Pix time from parse',... 1920 'UTC Date', 'UTC Time', 'U (m/s)', 'V (m/s)', 'W (m/s)',... 1921 'Alpha(deg)', 'Beta(deg)', 'Alpha Mean Aver.',... 1922 'Beta Mean Aver.', 'Total Velocity (m/s)', 'TBD',... 1923 'TBD2', 'TBD3', 'TBD4'} ); 1924 save(fullParsedMatFileName, 'MHP\_table', '-append'); 1925 1926 PixData\_table = ... 1927 table(PixData(:,1),PixData(:,2),Pix\_time\_out,... PixData(:,3), 'VariableNames', {'Sensor board time (ms)',... 1928 'GPS Unix Time (sec)', 'Pix board time (sec)',... 1929 'Pix board time (ms)'}); 1930 save(fullParsedMatFileName, 'PixData\_table', '-append'); 1931 1932 if(exist('THSense', 'var')) 1933 TPH\_table = table(TPH\_entry(:,1), TPH\_entry(:,2),... 1934 TPH\_time\_out, TH\_Date, TH\_Time, TPH\_entry(:, 3),... 1935 TPH\_entry(:,4), TPH\_entry(:,5), TPH\_entry(:,6),... 1936 TPH\_entry(:,7),TPH\_entry(:,8) , 'VariableNames'... 1937 , {'Board Time from PowerUp (msec)',... 1938 'Pixhawk Time from PowerUp (msec)',... 1939 'Pix Time from parse', 'UTC Date', 'UTC Time',... 1940 'Temp 1 ( C )', 'Temp 2 ( C )', 'Temp 3 ( C )',... 1941 'Humidity 1 (%)', 'Humidity 2 (%)', 'Humidity 3 (%)'}); 1942 1943 save(fullParsedMatFileName, 'TPH\_table', '-append');

```
1944
                 end
1945
            end
1946
            if(exist('TPH_table', 'var'))
1947
                 TPH = [table2array(TPH_table(:,1:3)) ...
1948
                    table2array(TPH_table(:,6:end))];
            end
1949
            if(exist('MHP_table', 'var'))
1950
                 MHP = [table2array(MHP_table(:,1:3)) ...
1951
                    table2array(MHP_table(:,6:end))];
                 CTUN = table2array(CTUN_table);
1952
                 NKF1 = table2array(NKF1_table);
1953
            end
1954
1955
                         %% ADD CODE HERE
1956
1957
                            count = 0;
응
                            for i=2:(length(NKF2(:,1))-1)
1959
1960 %
                                if (NKF2(i, 4) ≠NKF2(i-1, 4) | NKF2(i, 5) ≠NKF2(i-1, 5))
                                    count = count+1;
1961 💡
                                    trueNKF2(count,:) = NKF2(i,:);
end
1964 %
                            end
1965 %
1966 %% Kolmogorov -5/3 law comparison
1967
   %fs_avgmhp = round(length(MHP(:,3))/MHP(end,3));
1968
1969
1970 %[freq, psdx] = Kolmogorov(MHP(:,11),fs_avgmhp);
1971
                              %% Removal of aircraft velocities from 5hp data ...
1972
                                  (no removal of rotational velocities)
```

```
% Variables to locate/Initialize
1973
1974
                bDATA = [NKF1(:,4), NKF1(:,5), NKF1(:,6)];
1975
1976
                % U_ac aircraft velocity [U_ac, V_ac, W_ac]
1977
                 % GPS/Filter data
1978
                 gDATA = [NKF1(:,7), NKF1(:,8), NKF1(:,9)]; % this is VN, ...
1979
                    VE, VD (will need to be rotated into body frame)
1980
                 % data fusion for 5hp data and pixhawk velocities
1981
                 % Pixhawk Kalman Filter Runs at ¬25hz (not formated to run yet)
1982
1983
                 for i = 1:length(NKF1(:,3))
1984
                     NKF1_5HP(i,1) = find(NKF1(i,3) > MHP(:,3),1,'last');
1985
                     NKF1_IMU(i,1) = find(NKF1(i,3) > IMU(:,3),1,'last');
1986
                  end
1987
1988
                  ProbeSpeed = MHP(:, 11);
1989
1990
                  %Windowing of data for fusion
1991
                  for i = 1:length(NKF1(:,3))-1
1992
1993
                     avgMHP(i, 1) = \dots
                        mean(ProbeSpeed(NKF1_5HP(i,1):NKF1_5HP(i+1,1))); % ...
                         average multi hole probe speed
                     avgMHP(i, 2) = \dots
1994
                        mean(Alpha(NKF1_5HP(i,1):NKF1_5HP(i+1,1))); % ...
                        Average probe alpha
                     avqMHP(i,3) = \dots
1995
                        mean(Beta(NKF1_5HP(i,1):NKF1_5HP(i+1,1))); % average ...
                        probe Beta
                     avgMHPBoardTime(i,1) = MHP(NKF1_5HP(i,1),3);
1996
1997
                     avgUnixT(i,1) = MHPData_Unix(NKF1_5HP(i,1),1);
```

1998	$avgIMU(i,1) = IMU(NKF1_IMU(i,1),3);$
1999	avgIMU(i,2) =
	<pre>mean(IMU((NKF1_IMU(i,1):NKF1_IMU(i+1,1)),4)); %</pre>
	Average IMU gyro x
2000	avgIMU(i,3) =
	<pre>mean(IMU((NKF1_IMU(i,1):NKF1_IMU(i+1,1)),5)); %</pre>
	average IMU gyro y
2001	avgIMU(i,4) =
	<pre>mean(IMU((NKF1_IMU(i,1):NKF1_IMU(i+1,1)),6)); %</pre>
	average IMU gyro z
2002	end
2003	
2004	<pre>avgMHP(length(NKF1(:,3)),1) = ProbeSpeed(length(NKF1(:,3)),1);</pre>
2005	<pre>avgMHPBoardTime(length(NKF1(:,3)),1) = avgMHPBoardTime(i,1);</pre>
2006	<pre>avgIMU(length(NKF1(:,3)),1) = avgIMU(i,1);</pre>
2007	<pre>avgUnixT(length(NKF1(:,3)),1) = MHPData_Unix(end,1);</pre>
2008	
2009	
2010	% Rate Data
2011	<pre>rDATA = [avgIMU(:,2),avgIMU(:,3), avgIMU(:,4)];</pre>
2012	
2013	%Function Calls
2014	
2015	%Hard Code Length of probe from center
2016	
2017	L = .40; % Nimbus
2018	%L = .23; % NT
2019	
2020	<pre>%Basic function just removes first order aircraft states</pre>
2021	[mhpVEL, mhpANG, mhpSPEED]=MHPA_Rautenberg2018(avgMHP,
	gDATA, bDATA);
2022	%Adds removal of second order acceleration terms

```
2023
                  [mhpVELr, mhpANGr, mhpSPEEDr]=MHPA_wRATES(avgMHP, gDATA, ...
                      bDATA, rDATA,L);
2024
                  for i = 1:length(GPS(:,8))
2025
                      GPS_5HP(i, 1) = ...
2026
                         find(GPS(i,3) ≥avgMHPBoardTime(:,1),1,'last');
                  end
2027
2028
                  %windowing for alt vs windspeed plots
2029
                  for i = 1:length(GPS(:, 8))-1
2030
                       avgWS4alt(i,1) = \dots
2031
                          mean(mhpSPEED(GPS_5HP(i,1):GPS_5HP(i+1,1)));
                  end
2032
2033
    8
                  avgWS4alt(length(GPS(:,8)),1) = mhpSPEED(length(GPS(:,8)),1);
2034
2035
                  %% All the spectral Analysis
2036
2037
                  fs_avg = round(length(NKF1_5HP)/MHP(end,3)); % post window ...
2038
                      sampling frequency
                  L_avg = length(NKF1_5HP); % sample length
2039
2040
                  [P1_avg, f_avg] = freq_this(mhpSPEED, fs_avg, L_avg);
2041
2042
                  figure(12)
2043
                  plot (f_avg, P1_avg)
2044
                  title('Single-Sided Amplitude Spectrum of X(t)')
2045
                  xlabel('f (Hz)')
2046
                  ylabel('|P1(f)|')
2047
                  xlim([0 .5])
2048
                  if(strcmpi(mhpAvg, 'Yes'))
2049
```

```
2050
                      [loop_freq, vert, button] = ginput(1); %grab the peak ...
                         of the fft graph (should be the frequency of the orbit)
2051
                     orbitTIME = 1/loop_freq;
2052
2053
                     n_samples = round(orbitTIME*fs_avg); % number of ...
2054
                         samples to average over
2055
                       % averaging of orbits
2056
2057
                      for i = 1:length(NKF1(:,3))
2058
                           if i < n_samples + 1</pre>
2059
                              windSPEEDavg(i,1) = mean(mhpSPEED(1:i+1));
2060
                              windANGLEavg(i,1) = mean(mhpANG(1:i+1));
2061
                              mhpVNavg(i, 1) = mean(mhpVEL((1:i+1), 1));
2062
                              mhpVEavg(i, 1) = mean(mhpVEL((1:i+1), 2));
2063
                              mhpVDavg(i,1) = mean(mhpVEL((1:i+1),3));
2064
                           else
2065
                              windSPEEDavg(i,1) = mean(mhpSPEED(i-n_samples:i));
2066
                              windANGLEavg(i,1) = mean(mhpANG(i-n_samples:i));
2067
                              mhpVNavg(i,1) = mean(mhpVEL((i-n_samples:i),1));
2068
                              mhpVEavg(i,1) = mean(mhpVEL((i-n_samples:i),2));
2069
                              mhpVDavg(i,1) = mean(mhpVEL((i-n_samples:i),3));
2070
                           end
2071
                       end
2072
2073
                       %windowing for alt vs windspeed plots
2074
                  for i = 1:length(GPS(:, 8))-1
2075
                       avgWS4alt2(i,1) = \dots
2076
                          mean(windSPEEDavg(GPS_5HP(i,1):GPS_5HP(i+1,1)));
                  end
2077
2078 %
```

```
avgWS4alt2(length(GPS(:,8)),1) = ...
2079
                      windSPEEDavg(length(GPS(:,8)),1);
2080
   응
                   figure(69)
2081
                   plot(avgWS4alt2(:,1),GPS_mat(:,7))
2082 %
                   xlabel('Wind Speed (m/s)')
   8
2083
                   ylabel('Altitude AGL (m)')
2084 %
2085
                  end
2086
2087
2088
2089
                 %% 5HP & TPH plotting
2090
2091
             if(strcmpi(mhpValue,'Yes') && strcmpi(tphValue,'Yes') && ...
2092
                exist('THSense', 'var'))
2093
                 fig4=figure(4);
2094
                 fig4.Name = 'Parsed 5HP and TPH Data';
2095
                 set(fig4, 'defaultLegendAutoUpdate', 'off');
2096
                 subplot(2,1,1);
2097
                 plt2 = ...
2098
                     plot (TPH(:,2)/1000, TPH(:,4), 'r-', TPH(:,2)/1000, TPH(:,5), 'b-', TPH(:,2)/10
                 title('Temp and Humidity vs Time')
2099
                 legend({'Temp 1', 'Temp 2', 'Temp 3', 'Humid 1', 'Humid 2', ...
2100
                     'Humid 3'}, 'Location', 'southeast')
                 xlabel('Time (sec)');
2101
                 ylabel('Temp ( C ) and Humidity (%)');
2102
2103
                 xlim([min(TPH(:,2)/1000) max(TPH(:,2)/1000)])
2104
                 subplot (2, 1, 2);
2105
```

```
2106
                 plt3 = ...
                     plot (MHP(:,3), MHP(:,4), 'r', MHP(:,3), MHP(:,5), 'b', MHP(:,3), MHP(:,6), 'g');
                 title('u, v, w (raw)')
2107
                 legend({'u', 'v', 'w'}, 'Location', 'northwest')
2108
                 ylabel('Velocity (m/s)');
2109
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2110
2111
                 fiq18=figure(18);
2112
                 fig18.Name='MHP Wind Frame';
2113
2114
                 plt3 = ...
                     plot (MHP(:,3), MHP(:,11), '-r', MHP(:,3), MHP(:,7), '-b', MHP(:,3), MHP(:,8), '-
                 ylabel('Airspeed (m/s), Alpha and Beta(deg)');
2115
                 ylim([-10,20])
2116
                 title('Airspeed, Alpha, and Beta (raw)')
2117
                 legend({'Airspeed', 'Alpha', 'Beta'}, 'Location', 'northwest')
2118
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2119
2120
                 fiq5=figure(5);
2121
                 fig5.Name = 'MHP vs Pix Airspeeds';
2122
                 set(fiq5, 'defaultLegendAutoUpdate', 'off');
2123
                 subplot(2,1,1);
2124
                 plt1 = plot(MHP(:,3),MHP(:,4),'r',CTUN(:,3),CTUN(:,4),'k');
2125
                 title('MHP Pitot and Pix Airspeeds with Time')
2126
                 legend({'MHP Pitot', 'Pix Arspd'},'Location','northwest')
2127
                 ylabel('Airspeed (m/s)');
2128
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2129
2130
                 subplot (2, 1, 2)
2131
2132
                 plt2 = ...
                     plot (MHP(:,3), MHP(:,4), 'r', MHP(:,3), MHP(:,5), 'b', MHP(:,3), MHP(:,6), 'q', C
                 title('MHP Pitot, Alpha, Beta, and Pix Airspeeds with Time')
2133
```

```
2134
                 legend({'MHP-Pitot', 'MHP-Alpha', 'MHP-Beta', 'Pix ...
                    Arspd'}, 'Location', 'northwest')
                 ylabel('Airspeed (m/s)');
2135
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2136
2137
                 fig11 = figure(11);
2138
                 fig11.Name = 'MHP vs. Pix Windspeeds';
2139
                 plt = plot(NKF1(:,2),mhpWIND,NKF1(:,2),wind);
2140
                 legend({'MHP Wind Estimation', 'PixWind Estimation'}, ...
2141
                    'Location', 'Best')
                 title('MHP vs. Pix Windspeeds (Body Removed)')
2142
                 xlabel('time (ms)')
2143
                 ylabel('wind speed (m/s)')
2144
2145
                 fig16 = figure(16);
2146
                 fig16.Name = 'MHP vs. Pix Wind Vector';
2147
                 plt = plot(NKF2(:,2), VWN, NKF2(:,2), VWE, ...
2148
                    NKF1(:,2), mhpVEL(:,1), NKF1(:,2), mhpVEL(:,2));
                 legend({'VWN Pix', 'VWE Pix', 'VWN MHP', 'VWE MHP'}, ...
2149
                     'Location', 'Best')
                 title('MHP vs. Pix Vector')
2150
                 xlabel('time(ms)')
2151
                 ylabel('wind speed (m/s)')
2152
2153
                 fig20 = figure(20);
2154
                 fig20.Name = 'MHP vs. Pix Wind Vector';
2155
                 plt = plot(NKF1(:,2),mhpVEL(:,1), NKF1(:,2), ...
2156
                    mhpVEL(:,2),NKF1(:,2),mhpVEL(:,3));
                 legend({'North Wind', 'East Wind', 'Down Wind'}, ...
2157
                    'Location', 'Best')
                 title('MHP Ground Based Wind Speed')
2158
2159
                 xlabel('time(ms)')
```

2160ylabel('wind speed (m/s)') 2161 if(strcmpi(mhpAvg, 'Yes')) 2162 fig13 = figure(13);2163 fig13.Name = 'MHP vs. Pix Windspeeds'; 2164 plt = plot(NKF1(:,2),windSPEEDavg,NKF2(:,2),wind); 2165legend({'MHP Wind Estimation', 'PixWind Estimation'}, ... 2166 'Location', 'Best') title('MHP vs. Pix Windspeeds (Orbit Averaged)') 2167xlabel('time (ms)') 2168 ylabel('wind speed (m/s)') 21692170 fig14 = figure(14);2171 fig14.Name = 'MHP vs. Pix Angles'; 2172 plt = plot(NKF1(:,2),windANGLEavg,NKF2(:,2),windANG); 2173 legend('MHP Wind Angle Estimation', 'PixWind Estimation') 2174title({'MHP vs. Pix Angle (Orbit Averaged)'}, ... 2175'Location', 'Best') xlabel('time (ms)') 2176ylabel('wind speed (deg)') 21772178 fig17 = figure(17);2179 fig17.Name = 'MHP vs. Pix Wind Vector'; 2180 plt = plot(NKF2(:,2), VWN, NKF2(:,2), VWE, ... 2181NKF1(:,2),mhpVNavg(:,1), NKF1(:,2), mhpVEavg(:,2)); legend({'VWN Pix', 'VWE Pix', 'VWN MHP', 'VWE MHP'}, ... 2182'Location', 'Best') title('MHP vs. Pix Vector') 2183 2184 xlabel('time(ms)') ylabel('wind speed (m/s)') 2185 2186 2187

```
2188
                     fig21 = figure(21);
                     fig21.Name = 'MHP Wind Vector';
2189
                     plt = plot(NKF1(:,2), mhpVNavg(:,1), NKF1(:,2), ...
2190
                         mhpVEavg(:,2),NKF1(:,2),mhpVDavg(:,3));
                     legend({'North Wind', 'East Wind', 'Down Wind'}, ...
2191
                         'Location', 'Best')
                     title('MHP Ground Based Wind Speed Averaged')
2192
                     xlabel('time(ms)')
2193
                     ylabel('wind speed (m/s)')
2194
2195
                     end
2196
                 if(strcmpi(sensorOut, 'Yes'))
2197
                     if(strcmpi(DFL_NewOld, 'New'))
2198
                         % Output parsed TPH data
2199
                         baseFileName = sprintf('%s_Parsed_TPH.csv', ...
2200
                             baseNameNoExtDFL);
                         fullOutputMatFileName = fullfile(folderDFL, ...
2201
                             baseFileName);
                         % Write data to text file
2202
                         writetable(TPH_table, fullOutputMatFileName);
2203
2204
                         % Get the name of the input.mat file and save as ...
2205
                             input_Parsed_MHP.csv
                         baseFileName = sprintf('%s_Parsed_MHP.csv', ...
2206
                             baseNameNoExtDFL);
                         fullOutputMatFileName = fullfile(folderDFL, ...
2207
                             baseFileName);
                         % Write data to text file
2208
2209
                         writetable(MHP_table, fullOutputMatFileName);
                     else
2210
                         % Output parsed TPH data
2211
```

```
2212
                         baseFileName = sprintf('%s_TPH_NS.csv', ...
                             baseNameNoExtDFL);
                          fullOutputMatFileName = fullfile(folderDFL, ...
2213
                             baseFileName);
                          % Write data to text file
2214
                          writetable(TPH_table, fullOutputMatFileName);
2215
2216
                          % Get the name of the input.mat file and save as ...
2217
                             input_Parsed_MHP.csv
                         baseFileName = sprintf('%s_MHP_NS.csv', ...
2218
                             baseNameNoExtDFL);
                          fullOutputMatFileName = fullfile(folderDFL, ...
2219
                             baseFileName);
                          % Write data to text file
2220
                         writetable(MHP_table, fullOutputMatFileName);
2221
                     end
2222
2223
                 end
2224
            elseif(strcmpi(mhpValue,'Yes') && (strcmpi(tphValue,'No') | ¬...
2225
                exist('THSense', 'var')))
2226
                 fig4=figure(4);
2227
                 fiq4.Name = 'Parsed 5HP Data';
2228
                 set(fig4, 'defaultLegendAutoUpdate', 'off');
2229
                 plt3 = ...
2230
                    plot (MHP(:,3), MHP(:,4), 'r', MHP(:,3), MHP(:,5), 'b', MHP(:,3), MHP(:,6), 'g');
                 title('u, v, w(raw)')
2231
                 legend({'u', 'v', 'w'}, 'Location', 'northwest')
2232
2233
                 xlabel('Time (sec)');
                 ylabel('Velocity (m/s)');
2234
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2235
2236
```

```
2237
                 fig5=figure(5);
                 fiq5.Name = 'MHP vs Pix Airspeeds';
2238
                 set(fig5, 'defaultLegendAutoUpdate', 'off');
2239
                 subplot (2, 1, 1);
2240
                 plt1 = plot(MHP(:,3),MHP(:,4),'r',CTUN(:,3),CTUN(:,4),'k');
2241
                 title('MHP Pitot and Pix Airspeeds with Time')
2242
                 legend({'MHP Pitot', 'Pix Arspd'}, 'Location', 'northwest')
2243
                 ylabel('Airspeed (m/s)');
2244
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2245
2246
                 subplot(2,1,2)
2247
                 plt2 = ...
2248
                     plot (MHP(:,3), MHP(:,4), 'r', MHP(:,3), MHP(:,5), 'b', MHP(:,3), MHP(:,6), 'g', C
2249
                 title('MHP Pitot, Alpha, Beta, and Pix Airspeeds with Time')
                 legend({'MHP-u', 'MHP-v', 'MHP-w', 'Pix ...
2250
                     Arspd'}, 'Location', 'northwest')
                 ylabel('Airspeed (m/s)');
2251
                 xlim([(min(MHP(:,3))) (max(MHP(:,3)))])
2252
2253
                 figl1 = figure(11);
2254
                 fig11.Name = 'MHP vs. Pix Windspeeds';
2255
                 plt = plot(NKF1(:,2),mhpSPEED,NKF2(:,2),wind);
2256
                 legend({'MHP Wind Estimation', 'PixWind Estimation'}, ...
2257
                     'Location', 'Best')
                 title('MHP vs. Pix Windspeeds')
2258
                 xlabel('time (ms)')
2259
                 ylabel('wind speed (m/s)')
2260
2261
                 fig14 = figure(14);
2262
                 fig14.Name = 'MHP vs. Pix Wind Angles';
2263
                 plt = plot(NKF1(:,2),mhpANG,NKF2(:,2),windANG);
2264
```

```
legend({'MHP Wind Angle Estimation', 'PixWind Estimation'}, ...
2265
                    'Location', 'Best')
                 title('MHP vs. Pix Wind Angles')
2266
                xlabel('time (ms)')
2267
                ylabel('wind angle (deg)')
2268
2269
                fig16 = figure(16);
2270
                 fig16.Name = 'MHP vs. Pix Wind Vector';
2271
                plt = plot(NKF2(:,2), VWN, NKF2(:,2), VWE, ...
2272
                    NKF1(:,2), mhpVEL(:,1), NKF1(:,2), mhpVEL(:,2));
                 legend({'VWN Pix', 'VWE Pix', 'VWN MHP', 'VWE MHP'}, ...
2273
                    'Location', 'Best')
                title('MHP vs. Pix Vector')
2274
                xlabel('time(ms)')
2275
                ylabel('wind speed (m/s)')
2276
2277
                fig20 = figure(20);
2278
                 fig20.Name = 'MHP vs. Pix Wind Vector';
2279
                plt = plot(NKF1(:,2),mhpVEL(:,1), NKF1(:,2), ...
2280
                    mhpVEL(:,2),NKF1(:,2),mhpVEL(:,3));
                 legend({'North Wind', 'East Wind', 'Down Wind'}, ...
2281
                    'Location', 'Best')
                title('MHP Ground Based Wind Speed')
2282
                xlabel('time(ms)')
2283
                ylabel('wind speed (m/s)')
2284
2285
                 if(strcmpi(mhpAvg, 'Yes'))
2286
                     fig12 = figure(12);
2287
2288
                     fig12.Name = 'MHP vs. Pix Windspeeds';
                     plt = plot(NKF1(:,2),windSPEEDavg,NKF2(:,2),wind);
2289
                     legend({'MHP Wind Estimation', 'PixWind Estimation'}, ...
2290
                        'Location', 'Best')
```

```
2291
                     title('MHP vs. Pix Windspeeds (Orbit Averaged)')
                     xlabel('time (ms)')
2292
                     ylabel('wind speed (m/s)')
2293
2294
                     fig15 = figure(15);
2295
                     fig15.Name = 'MHP vs. Pix Windspeeds';
2296
                     plt = plot(NKF1(:,2),windANGLEavg,NKF2(:,2),windANG);
2297
                     legend({'MHP Wind Estimation', 'PixWind Estimation'}, ...
2298
                         'Location', 'Best')
                     title('MHP vs. Pix Wind Angles (Orbit Averaged)')
2299
                     xlabel('time (ms)')
2300
                     ylabel('wind angle (deg)')
2301
2302
                     fig17 = figure(17);
2303
                     fig17.Name = 'MHP vs. Pix Wind Vector';
2304
                     plt = plot(NKF2(:,2), VWN, NKF2(:,2), VWE, ...
2305
                        NKF1(:,2),mhpVNavg(:,1), NKF1(:,2), mhpVEavg(:,1));
                     legend({'VWN Pix', 'VWE Pix', 'VWN MHP', 'VWE MHP'}, ...
2306
                         'Location', 'Best')
                     title('MHP vs. Pix Vector')
2307
                     xlabel('time(ms)')
2308
                     ylabel('wind speed (m/s)')
2309
2310
                     fig21 = figure(21);
2311
                     fig21.Name = 'MHP Wind Vector';
2312
                     plt = plot(NKF1(:,2),mhpVNavg(:,1), NKF1(:,2), ...
2313
                        mhpVEavg(:,1),NKF1(:,2),mhpVDavg(:,1));
                     legend({'North Wind', 'East Wind', 'Down Wind'}, ...
2314
                         'Location', 'Best')
                     title('MHP Ground Based Wind Speed Averaged')
2315
                     xlabel('time(ms)')
2316
2317
                     ylabel('wind speed (m/s)')
```

2318 2319 end 2320 if(strcmpi(sensorOut, 'Yes')) 2321 if(strcmpi(DFL\_NewOld, 'New')) 2322 % Get the name of the input.mat file and save as ... 2323 input\_Parsed\_MHP.csv baseFileName = sprintf('%s\_Parsed\_MHP.csv', ... 2324 baseNameNoExtDFL); fullOutputMatFileName = fullfile(folderDFL, ... 2325 baseFileName); % Write data to text file 2326 writetable(MHP\_table, fullOutputMatFileName); 2327 2328 else % Get the name of the input.mat file and save as ... 2329 input\_Parsed\_MHP.csv baseFileName = sprintf('%s\_MHP\_NS.csv', ... 2330 baseNameNoExtDFL); fullOutputMatFileName = fullfile(folderDFL, ... 2331baseFileName); % Write data to text file 2332 writetable(MHP\_table, fullOutputMatFileName); 2333 end 2334 2335 end 2336 2337 elseif(strcmpi(mhpValue,'No') && strcmpi(tphValue,'Yes') && ... 2338 exist('THSense', 'var')) 2339 fig4=figure(4); 2340 fig4.Name = 'Parsed TPH Data'; 2341

```
2342
                 plt2 = ...
                    plot (TPH(:,2)/1000, TPH(:,3), 'r-', TPH(:,2)/1000, TPH(:,4), 'b-', TPH(:,2)/10
                 title('Temp and Humidity vs Time')
2343
                 legend({'Temp 1', 'Temp 2', 'Temp 3', 'Humid 1', 'Humid 2', ...
2344
                     'Humid 3'}, 'Location', 'southeast')
                 xlabel('Time (sec)');
2345
                 ylabel('Temp ( C ) and Humidity (%)');
2346
                 xlim([(min(TPH(:,2)/1000)) (max(TPH(:,2)/1000))])
2347
2348
                 if(strcmpi(sensorOut, 'Yes'))
2349
                     if(strcmpi(DFL_NewOld, 'New'))
2350
                          % Output parsed TPH data
2351
                         baseFileName = sprintf('%s_Parsed_TPH.csv', ...
2352
                             baseNameNoExtDFL);
                          fullOutputMatFileName = fullfile(folderDFL, ...
2353
                             baseFileName);
                          % Write data to text file
2354
                          writetable(TPH_table, fullOutputMatFileName);
2355
                     else
2356
                          % Output parsed TPH data
2357
                          baseFileName = sprintf('%s_TPH_NS.csv', ...
2358
                             baseNameNoExtDFL);
                          fullOutputMatFileName = fullfile(folderDFL, ...
2359
                             baseFileName);
                          % Write data to text file
2360
                          writetable(TPH_table, fullOutputMatFileName);
2361
                     end
2362
2363
                 end
2364
            end
2365
        end
        %% CSV Output of GPS Data
2366
2367
        if (strcmpi(gpsOut, 'Yes'))
```

if(strcmpi(DFL\_NewOld, 'New')) 2368 % Get the name of the input.mat file and save as input\_GPS.csv 2369 baseFileName = sprintf('%s\_GPS.csv', baseNameNoExtDFL); 2370 2371elseif(strcmpi(DFL\_NewOld, 'Old')) % Get the name of the input.mat file and save as ... 2372 input\_GPS\_NS.csv baseFileName = sprintf('%s\_GPS\_NS.csv', baseNameNoExtDFL); 2373 end 2374fullOutputMatFileName = fullfile(folderDFL, baseFileName); 2375% Write data to text file 2376 writetable(GPS\_table, fullOutputMatFileName); 2377 2378 end %% CSV Output of Aircraft Attitude Data 2379 if (strcmpi(attitudeOut, 'Yes')) 2380 if(strcmpi(DFL\_NewOld, 'New')) 2381% Get the name of the input.mat file and save as ... 2382input\_Attitude.csv baseFileName = sprintf('%s\_Attitude.csv', baseNameNoExtDFL); 2383 elseif(strcmpi(DFL\_NewOld, 'Old')) 2384% Get the name of the input.mat file and save as ... 2385 input\_Attitude\_NS.csv baseFileName = sprintf('%s\_Attitude\_NS.csv', baseNameNoExtDFL); 2386 end 2387 fullOutputMatFileName = fullfile(folderDFL, baseFileName); 2388 % Write data to text file 2389 writetable(ATT\_table, fullOutputMatFileName); 2390 end 2391 %% CSV Output of IMU Data 2392 2393 if (strcmpi(stateSpace, 'Yes')) if(strcmpi(DFL\_NewOld, 'New')) 2394% Get the name of the input.mat file and save as input\_IMU.csv 23952396baseFileName = sprintf('%s\_IMU.csv', baseNameNoExtDFL);

```
elseif(strcmpi(DFL_NewOld, 'Old'))
2397
                 % Get the name of the input.mat file and save as ...
2398
                     input_IMU_NS.csv
                 baseFileName = sprintf('%s_IMU_NS.csv', baseNameNoExtDFL);
2399
            end
2400
            fullOutputMatFileName = fullfile(folderDFL, baseFileName);
2401
            % Write data to text file
2402
            writetable(IMU_table, fullOutputMatFileName);
2403
        end
2404
        %% Plot Data According To User Input
2405
        if (strcmpi(graphToggle, 'Yes'))
2406
            % Converted timestamps to time (in seconds) from TO to LND
2407
            t_GPS = table2array(GPS_table(:,3));
2408
            t_NKF = table2array(NKF2_table(:,3));
2409
            t_RCOU = table2array(RCOU_table(:,3));
2410
            t_BARO = table2array(BARO_table(:,3));
2411
            t_ATT = table2array(ATT_table(:,3));
2412
2413
            if(strcmpi(arduPilotType, 'ArduCopter'))
2414
            elseif(strcmpi(arduPilotType, 'CopterSonde'))
2415
                 t_CSIMET = table2array(CSIMET_table(:,3));
2416
                 CSIMET_mat = table2array(CSIMET_table);
2417
            else
2418
                 t_ctun = table2array(CTUN_table(:,3));
2419
                 CTUN_mat = table2array(CTUN_table);
2420
            end
2421
2422
            t_low = min(GPS_table{:,3});
2423
2424
            t_high = max(GPS_table{:,3});
            GPS_mat = [table2array(GPS_table(:,1:3)) ...
2425
                table2array(GPS_table(:,6:end))];
            ATT_mat = table2array(ATT_table);
2426
```

```
BARO_mat = table2array(BARO_table);
2427
2428
             IMU_mat = table2array(IMU_table);
             if(exist('MHP_table','var') && ¬exist('MHP','var'))
2429
                 MHP = [table2array(MHP_table(:,1:3)) ...
2430
                     table2array(MHP_table(:,6:end))];
             end
2431
             if(exist('TPH_table', 'var') && ¬exist('TPH', 'var'))
2432
                 TPH = [table2array(TPH_table(:,1:3)) ...
2433
                     table2array(TPH_table(:,6:end))];
2434
             end
            if(exist('PixData_table', 'var') && ¬exist('PixData', 'var'))
2435
                 PixData = table2array(PixData_table);
2436
             end
2437
             if(exist('iMet_table', 'var') && ¬exist('iMet', 'var'))
2438
                 iMet = [table2array(iMet_table(:,1)) ...
2439
                     table2array(iMet_table(:,4:end))];
2440
             end
             NKF2_mat = table2array(NKF2_table);
2441
             RCOU_mat = table2array(RCOU_table);
2442
2443
2444
2445
             if (strcmpi(overlay, 'Yes'))
2446
2447
                 fiq6=figure(6);
2448
                 fig6.Name = 'Data Plots from Parsed Sensor data and ...
2449
                     Autopilot DFL';
2450
                 if(strcmpi(mhpValue, 'Yes'))
2451
                     if(exist('MHP', 'var'))
2452
                          % Groundspeed plot
2453
2454
                          plt1 = subplot(5, 1, 1);
```

2455	<pre>plot(t_GPS,GPS_mat(:,8),'k',t_ctun,CTUN_mat(:,4),'b',t_NKF,NKF2_mat</pre>
2456	title('Groundspeed (black), Airspeed (blue),
	Windspeed (red), MHP-Pitot (green) vs Time')
2457	<pre>ylabel({'Velocity (m/s)'})</pre>
2458	end
2459	<pre>elseif(strcmpi(arduPilotType, 'ArduCopter')  </pre>
	<pre>strcmpi(arduPilotType, 'CopterSonde'))</pre>
2460	% Groundspeed plot
2461	plt1 = subplot(5,1,1);
2462	<pre>plot(t_GPS,GPS_mat(:,8),'k',t_NKF,NKF2_mat(:,6),'r')</pre>
2463	<pre>title('Groundspeed (black), Windspeed (red) vs Time')</pre>
2464	<pre>ylabel({'Velocity (m/s)'})</pre>
2465	else
2466	% Groundspeed plot
2467	plt1 = subplot(5,1,1);
2468	<pre>plot(t_GPS,GPS_mat(:,8),'k',t_ctun,CTUN_mat(:,4),'b',t_NKF,NKF2_mat(:,6</pre>
2469	title('Groundspeed (black), Airspeed (blue), Windspeed
	(red) vs Time')
2470	<pre>ylabel({'Velocity (m/s)'})</pre>
2471	end
2472	
2473	<pre>if(strcmpi(iMetValue, 'Yes'))</pre>
2474	<pre>if(exist('iMet_table','var'))</pre>
2475	% Barometric Pressure of Pixhawk and Sensor Packages
2476	plt4 = subplot(5,1,4);
2477	plot(t_BARO,
	BARO_mat(:,4), 'b-', iMet(:,1), iMet(:,2), 'r-');
2478	title('Pixhawk (internal) vs iMet (external)
	Atmospheric Pressure Measurements')
2479	%xticks([150:20:370])
2480	<pre>ylabel({'Pixhawk Pressure (blue)';'iMet Pressure</pre>
	(red)';'(mbar)'});

2481end 2482 else % Barometric Pressure of Pixhawk and Sensor Packages 2483plt4 = subplot(5, 1, 4);2484plot(t\_BARO, BARO\_mat(:,4), 'b-'); 2485title('Pixhawk (internal) Atmospheric Pressure ... 2486 Measurements') %xticks([150:20:370]) 2487ylabel({'Pixhawk Pressure (blue)';'(mbar)'}); 2488 2489 end 2490if(strcmpi(arduPilotType, 'CopterSonde')) 2491plt11 = subplot(2,1,1); 2492 colors = { 'k', 'r', 'b', 'g', 'k-', 'r-', 'b-', 'g-' }; 2493 for i=0:3 2494 if (min (CSIMET\_mat(:,i+4))>0) 2495hold on 2496 plot(t\_CSIMET,CSIMET\_mat(:,i+4)/100,colors{i+1}); 2497 plot(t\_CSIMET,CSIMET\_mat(:,i+8)-273.15,colors{i+5}); 2498 end 2499 end 2500 hold off 2501title('CopterSonde Temperature and Humidity'); 2502 ylabel({'Temp ( C ) and Humidity (%)'}); 25032504plt12 = subplot(2, 1, 2);2505plot(BARO\_mat(:,3),BARO\_mat(:,4),'k'); 2506title('Pixhawk (internal) Pressure Readings'); 2507 ylabel({'Pressure (mbar)'}); 2508 xlabel({'Time (sec)'}); 2509linkaxes([plt11 plt12],'x') 2510 xlim([min(t\_CSIMET) max(t\_CSIMET)]) 2511

```
2512
                 else
                      % Throttle Output
2513
                      plt2 = subplot(5, 1, 2);
2514
                      plot(t_RCOU,RCOU_mat(:,6),'b')
2515
                      title('Throttle vs Time')
2516
                      ylabel({'Throttle';'(%)'})
2517
                      %xticks([150:20:370])
2518
                      vlim([0 100])
2519
2520
                      % For the dotted line along x-axis of pitch plot
2521
                      zero=int8(zeros(length(ATT_mat(:,5)),1));
2522
2523
                      % Aircraft Pitch angle: Can change ylim to something ...
2524
                         more relevant.
                      % TIV uses -20 to 50 to see high AoA landing
2525
                      plt3 = subplot(5, 1, 3);
2526
                      plot(t_ATT,ATT_mat(:,5),'b',t_ATT,zero,'r:')
2527
                      title('Aircraft Pitch Angle vs Time')
2528
                      ylabel({'Aircraft Pitch';'Angle ( )'})
2529
                      %xticks([150:20:370])
2530
                      ylim([-20 50])
2531
2532
                      % Altitude plot
2533
                      plt5 = subplot(5, 1, 5);
2534
                      plot(t_GPS,GPS_mat(:,7),'b')
2535
                      title('Altitude vs Time')
2536
                      %xticks([150:20:370])
2537
                      ylabel({'Altitude AGL';'(m)'})
2538
2539
                      xlabel('Time (seconds)')
2540
                      linkaxes([plt1 plt2 plt3 plt4 plt5],'x')
2541
                      xlim([t_low t_high])
2542
```

```
2543
                 end
2544
                 if (strcmpi(tphValue, 'Yes') && strcmpi(mhpValue, 'Yes') && ...
2545
                     strcmpi(iMetValue, 'Yes'))
                     if(exist('TPH','var') && exist('MHP','var') && ...
2546
                         exist('iMet', 'var'))
                          fig7=figure(7);
2547
                          fig7.Name = 'Pixhawk, iMet, TPH, and MHP Data ...
2548
                             Comparisons';
2549
                          if(strcmpi(arduPilotType, 'ArduCopter') | ...
2550
                              strcmpi(arduPilotType, 'CopterSonde'))
                              plt1 = subplot(3, 1, 1);
2551
                              plot (MHP(:,3), MHP(:,4), 'r', MHP(:,3), MHP(:,5), 'b', MHP(:,3), MHP(:,
2552
                              title('5HP Alpha, Beta, and Pitot Velocities');
2553
                              ylabel({'Velocity (m/s)'});
2554
                          else
2555
                              plt1 = subplot(3, 1, 1);
2556
                              plot(t_ctun,CTUN_mat(:,4), 'k',MHP(:,3),MHP(:,4), 'r',MHP(:,3),MH
2557
                              title('Pixhawk Airspeed vs 5HP Alpha, Beta, and ...
2558
                                  Pitot Velocities');
                              ylabel({'Velocity (m/s)'});
2559
                          end
2560
2561
                          plt2 = subplot(3, 1, 2);
2562
                          plot(iMet(:,1),iMet(:,3),'k',iMet(:,1),iMet(:,4),'k-',TPH(:,3),TPH(
2563
                          title('iMet vs Sensor Package Temperature and ...
2564
                             Humidity');
                          ylabel({'Temp ( C ) and Humidity (%)'});
2565
2566
                          plt3 = subplot(3, 1, 3);
2567
2568
                          plot(BARO_mat(:,3),BARO_mat(:,4),'k',iMet(:,1),iMet(:,2),'r');
```

```
2569
                          title('Pixhawk (internal) vs iMet (external) ...
                              Pressure Readings');
                          ylabel({'Pressure (mbar)'});
2570
                          xlabel({'Time (sec)'});
2571
                          linkaxes([plt1 plt2 plt3],'x')
2572
                          xlim([min(t_BARO) max(t_BARO)])
2573
                      end
2574
                 end
2575
2576
2577
             else
2578
                 fiq6=figure(6);
2579
                 fig6.Name = 'Data Plots from Parsed Autopilot DFL';
2580
2581
                 if(strcmpi(arduPilotType, 'ArduCopter') | ...
2582
                     strcmpi(arduPilotType, 'CopterSonde'))
                      % Groundspeed plot
2583
                      plt1 = subplot(4, 1, 1);
2584
                      plot(t_GPS,GPS_mat(:,8), 'b-',t_NKF,NKF2_mat(:,6), 'g-')
2585
                      title('Groundspeed, and Windspeed vs Time')
2586
                      ylabel({'Groundspeed (blue)';'Windspeed (green)';'(m/s)'})
2587
                 else
2588
                      % Groundspeed plot
2589
                      plt1 = subplot(4, 1, 1);
2590
                      plot(t_GPS,GPS_mat(:,8), 'b-',t_ctun,CTUN_mat(:,4), 'r-',t_NKF, NKF2_mat(:,
2591
                      title('Groundspeed, Airspeed, and Windspeed vs Time')
2592
                      ylabel({'Groundspeed (blue)';'Airspeed ...
2593
                          (red)';'Windspeed (green)';'(m/s)'})
2594
                 end
2595
                 % Throttle Output
2596
                 plt2 = subplot(4, 1, 2);
2597
```

```
plot(t_RCOU, RCOU_mat(:, 6), 'b')
2598
                 title('Throttle vs Time')
2599
                 ylabel({'Throttle';'(%)'})
2600
                 ylim([0 100])
2601
2602
                 % For the dotted line along x-axis of pitch plot
2603
                 zero=int8(zeros(length(ATT_mat(:,5)),1));
2604
2605
                 % Aircraft Pitch angle: Can change ylim to something more ...
2606
                     relevant.
                 % TIV uses -20 to 50 to see high AoA landing
2607
                 plt3 = subplot(4, 1, 3);
2608
                 plot(t_ATT,ATT_mat(:,5),'b',t_ATT,zero,'r:')
2609
                 title('Aircraft Pitch Angle vs Time')
2610
                 ylabel({'Aircraft Pitch';'Angle ( )'})
2611
                 ylim([-10 40])
2612
2613
2614
                 % Altitude plot
2615
                 plt4 = subplot(4, 1, 4);
2616
                 plot(t_GPS,GPS_mat(:,7),'b')
2617
                 title('Altitude vs Time')
2618
                 ylabel({'Altitude AGL';'(m)'})
2619
                 xlabel('Time (seconds)')
2620
2621
                 linkaxes([plt1, plt2, plt3, plt4],'x')
2622
                 xlim([t_low t_high])
2623
2624
2625
2626
             end
2627
2628
             if (strcmpi(indvToggle, 'Yes'))
```

```
2629
                 fig8=figure(8);
2630
                 fig8.Name = 'Interactive Plot - Right click or press ...
2631
                     Return/Enter when finished';
2632
                 % Groundspeed plot
2633
                 subplot(4,1,1);
2634
                 plot(t_GPS,GPS_mat(:,8), 'b-',t_ctun,CTUN_mat(:,4), 'r-',t_NKF,NKF2_mat(:,6),
2635
                 title('Groundspeed and Airspeed vs Time')
2636
                 ylabel({'Groundspeed (blue)';'Airspeed (red)';'Windspeed ...
2637
                     (green) '; '(m/s) '})
                 xlim([min(t_GPS) max(t_GPS)])
2638
2639
                 % Throttle Output
2640
                 subplot(4,1,2);
2641
                 plot(t_RCOU,RCOU_tabl(:,6),'b')
2642
                 title('Throttle vs Time')
2643
                 ylabel({'Throttle';'(%)'})
2644
                 xlim([min(t_RCOU) max(t_RCOU)])
2645
                 ylim([0 100])
2646
2647
                 % For the dotted line along x-axis of pitch plot
2648
                 zero=int8(zeros(length(ATT_mat(:,5)),1));
2649
2650
                 % Aircraft Pitch angle: Can change ylim to something more ...
2651
                     relevant.
                 % TIV uses -20 to 50 to see high AoA landing
2652
                 subplot(4,1,3);
2653
                 plot(t_ATT,ATT_mat(:,5),'b',t_ATT,zero,'r:')
2654
                 title('Aircraft Pitch Angle vs Time')
2655
                 ylabel({'Aircraft Pitch';'Angle ( )'})
2656
2657
                 xlim([min(t_ctun) max(t_ctun)])
```

```
ylim([-20 50])
2658
2659
                 % Altitude plot
2660
                 subplot(4,1,4);
2661
                 plot(t_GPS,GPS_mat(:,7),'b')
2662
                 title('Altitude vs Time')
2663
                 xlim([min(t_GPS) max(t_GPS)])
2664
                 ylabel({'Altitude AGL';'(m)'})
2665
                 xlabel('Time (seconds)')
2666
2667
                 n = 0;
                 while true
2668
                      [horiz, vert, button] = ginput(1);
2669
                     if isempty(horiz) || button(1) == 3; break; end
2670
2671
                     n = n+1;
                      x_n(n) = horiz(1); % save all points you continue getting
2672
                     hold on
2673
                      y_n1(n) = GPS_mat(find(t_GPS \ge x_n(n), 1), 8);
                                                                      % Groundspeed
2674
                      y_n2(n) = CTUN_mat(find(t_ctun > x_n(n), 1), 4);
                                                                        % Airspeed
2675
                      y_n3(n) = NKF2_mat(find(t_NKF \ge x_n(n), 1), 6);
                                                                        % Windspeed
2676
                      y_n4(n) = RCOU_mat(find(t_ctun \ge x_n(n), 1), 6);
                                                                         8 . . .
2677
                         Throttle Percent
                      y_n5(n)=ATT_mat(find(t_ctun≥x_n(n),1),5); % Aircraft Pitch
2678
                      y_n6(n) = GPS_mat(find(t_GPS \ge x_n(n), 1), 7); % Altitude
2679
                      y_n7(n)=RCOU_mat(find(t_ctun≥x_n(n),1),4); % Pitch PWM
2680
2681
                      % Groundspeed plot
2682
                      subplot(4,1,1);
2683
                      plot(t_GPS,GPS_mat(:,8), 'b',t_ctun,CTUN_mat(:,4), 'r',t_NKF,NKF2_mat(:,6)
2684
                     title('Groundspeed, Airspeed, and Windspeed vs Time')
2685
                      ylabel({'Groundspeed (blue)';'Airspeed ...
2686
                         (red)';'Windspeed (green)';'(m/s)'})
                      xlim([min(t_GPS) max(t_GPS)])
2687
```

```
2688
                      % Throttle Output
2689
                      subplot(4,1,5);
2690
                      plot(t_RCOU, RCOU_mat(:, 6), 'b', x_n, y_n4, 'kx')
2691
                     title('Throttle vs Time')
2692
                     ylabel({'Throttle';'(%)'})
2693
                      xlim([min(t_RCOU) max(t_RCOU)])
2694
                      vlim([0 100])
2695
2696
                      % For the dotted line along x-axis of pitch plot
2697
                      zero=int8(zeros(length(ATT_mat(:,5)),1));
2698
2699
                      % Aircraft Pitch angle: Can change ylim to something ...
2700
                         more relevant.
                      % TIV uses -20 to 50 to see high AoA landing
2701
                      subplot(4,1,3);
2702
                      plot(t_ATT,ATT_mat(:,5),'b',t_ATT,zero,'r:',x_n,y_n5,'kx');
2703
                      title('Aircraft Pitch Angle vs Time');
2704
                      ylabel({'Aircraft Pitch';'Angle ( )'});
2705
                      xlim([min(t_ctun) max(t_ctun)]);
2706
                      ylim([-20 50]);
2707
2708
                      % Altitude plot
2709
                      subplot(4,1,4);
2710
                      plot(t_GPS,GPS_mat(:,7), 'b', x_n, y_n6, 'kx')
2711
                     title('Altitude vs Time')
2712
                     xlim([min(t_GPS) max(t_GPS)])
2713
                     ylabel({'Altitude AGL';'(m)'})
2714
2715
                      xlabel('Time (seconds)')
2716
                      drawnow
2717
2718
```

2719	end
2720	
2721	if (strcmpi(throttleToggle,'Yes')) &
	(strcmpi(pitchToggle, 'No'))
2722	
2723	<pre>thrT = interpl(throttleMap(:,1),throttleMap(:,2),y_n4);</pre>
2724	
2725	% Drag = Thrust
2726	C_D = (thrT.*2)./((y_n2.^2)*S*rho);
2727	% Lift = Weight
2728	C_L = (GTOW*2)./(S*rho*y_n2.^2);
2729	% Ratio of C_L/C_D
2730	$L_D = C_L./C_D;$
2731	
2732	% Flight Analysis
2733	<pre>baseFileName = sprintf('%s_metricsFull.csv',</pre>
	<pre>baseNameNoExtDFL);</pre>
2734	<pre>fullOutputMatFileName = fullfile(folder, baseFileName);</pre>
2735	% Save file with parsed data as the original filename
	plus the added portion
2736	% Create a table with the data and variable names
2737	$T = table(round(y_n1.',1), round(y_n2.',1), \ldots$
	<pre>round(y_n3.',1), round(y_n4.',1), round(y_n5.',1),</pre>
	<pre>round(y_n6.',1), round(y_n7.',1), round(thrT.',2),</pre>
	round(C_D.',6), round(C_L.',6), round(L_D.',4),
	'VariableNames', {'Groundspeed (m/s)','Airspeed
	<pre>(m/s)','Windspeed (m/s)','Throttle (%)','Aircraft</pre>
	Pitch ( )','Altitude (m, AGL)','Pitch PWM','Thrust
	Output (units of Throttle Curve input
	<pre>file)','C_D','C_L','CD_CL'} )</pre>
2738	% Write data to text file
2739	<pre>writetable(T, fullOutputMatFileName)</pre>

```
2740
                     fig9 = figure(9);
2741
                     fig9.Name = 'Lift vs Drag Coefficients';
2742
                     subplot (1, 2, 1)
2743
                     scatter(C_D.',C_L.')
2744
                     title('C_L vs C_D')
2745
                     ylabel('C_L')
2746
                     xlabel('C_D')
2747
2748
                     subplot(1,2,2)
2749
                     scatter(y_n2.',L_D.')
2750
                     title('L_D Ratio vs Airspeed')
2751
                     ylabel('L/D')
2752
                     xlabel('Airspeed (m/s)')
2753
2754
                 elseif (strcmpi(throttleToggle, 'Yes')) & ...
2755
                     (strcmpi(pitchToggle, 'Yes'))
2756
                     % Mapping throttle Percent from User Input to Thrust Output
2757
                     thrT = interp1(throttleMap(:,1),throttleMap(:,2),y_n4);
2758
                     % Mapping pitch PWM from Pixhawk to true angular deflection
2759
                     pitchT = interp1(pitchMap(:,1),pitchMap(:,2),y_n7);
2760
2761
                     % Drag = Thrust*cos(angle)
2762
                     C_D = (thrT.*cos(pitchT/180*3.14)*2)./((y_n2.^2)*S*rho);
2763
                     % Lift = Weight + Drag*sin(angle)
2764
                     C_L = ...
2765
                         ((GTOW+(thrT.*sin(pitchT/180*3.14)))*2)./(S*rho*y_n2.^2);
                     % Ratio of C_L/C_D
2766
                     L_D = C_L./C_D;
2767
2768
2769
                     % Flight Analysis
```

2770	<pre>baseFileName = sprintf('%s_metricsFull.csv',</pre>
	<pre>baseNameNoExtDFL);</pre>
2771	<pre>fullOutputMatFileName = fullfile(folder, baseFileName);</pre>
2772	% Save file with parsed data as the original filename
	plus the added portion
2773	% Create a table with the data and variable names
2774	$T = table(round(y_n1.',1), round(y_n2.',1),$
	<pre>round(y_n3.',1), round(y_n4.',1), round(y_n5.',1),</pre>
	<pre>round(y_n6.',1), round(y_n7.',1), round(thrT.',2),</pre>
	<pre>round(pitchT.',1), round(C_D.',6), round(C_L.',6),</pre>
	<pre>round(L_D.',4), 'VariableNames', {'Groundspeed</pre>
	(m/s)', 'Airspeed (m/s)', 'Windspeed (m/s)', 'Throttle
	(%)','Aircraft Pitch ( )','Altitude (m,
	AGL)','Pitch PWM','Thrust Output (units of Throttle
	Curve input file)','Servo Angular Deflection
	( )','C_D','C_L','CD_CL'} )
2775	% Write data to text file
2776	<pre>writetable(T, fullOutputMatFileName)</pre>
2777	
2778	<pre>fig9 = figure(9);</pre>
2779	<pre>fig9.Name = 'Lift vs Drag Coefficients';</pre>
2780	subplot(1,2,1)
2781	<pre>scatter(C_D.',C_L.')</pre>
2782	<pre>title('C_L vs C_D')</pre>
2783	<pre>ylabel('C_L')</pre>
2784	<pre>xlabel('C_D')</pre>
2785	
2786	subplot(1,2,2)
2787	<pre>scatter(y_n2.',L_D.')</pre>
2788	title('L_D Ratio vs Airspeed')
2789	<pre>ylabel('L/D')</pre>
3790	<pre>xlabel('Airspeed (m/s)')</pre>

```
2791
2792
                 end
2793
            end
2794
        end
2795
        %% Animation of Flight
2796
2797
        if(strcmpi(animateToggle, 'Yes'))
2798
            % Get animation plot titles and output file names for animations
2799
            if(strcmpi(animateToggle, 'Yes') | strcmpi(animation, 'Yes'))
2800
                plotTitle = input('<strong>Enter Plot Title for animation ...
2801
                    function: </strong>','s');
                 % Get the name of the intput.mat file and save as ...
2802
                    input_parsed.mat
                userFileName = input('<strong>Enter an output file name for ...
2803
                    the animation sequence to save as: </strong>','s');
                 if(strcmpi(userFileName,''))
2804
                     userFileName = 'defaultAnimationOutput';
2805
                end
2806
                vidFileName = regexprep(userFileName, ' +', ' ');
2807
                videoOutputFileName = fullfile(startingFolderDFL, vidFileName);
2808
2809
                animVid = VideoWriter(videoOutputFileName, 'MPEG-4');
2810
                 animVid.FrameRate = animFrameRate; %can adjust this, 5 - ...
2811
                    10 works well for me
                 animVid.Quality = 100;
2812
            end
2813
            if(strcmpi(tripleAnimateTPH, 'Yes'))
2814
2815
                 %% Animation Setup
                 animVid = VideoWriter('Kessler 8-4-2020, ...
2816
                    TriplePlot', 'MPEG-4'); %open video file
```

2817	animVid.FrameRate = 20; %can adjust this, 5 - 10 works
	well for me
2818	animVid.Quality = 100;
2819	
2820	<pre>interval = 20;</pre>
2821	
2822	<pre>timer = table2array(BARO_table(1:2:8597-3,3));</pre>
2823	<pre>cAnim1 = [nan;table2array(TPH_table(1:2:end,6))];</pre>
2824	<pre>cAnim2 = [nan;table2array(TPH_table(1:2:end,9))];</pre>
2825	<pre>cAnim3 = [nan;table2array(BARO_table(1:2:8597-3,4))];</pre>
2826	<pre>xAnim=[nan;table2array(GPS_table(1:4297,6))];</pre>
2827	<pre>yAnim=[nan;table2array(GPS_table(1:4297,7))];</pre>
2828	<pre>zAnim=[nan;table2array(GPS_table(1:4297,9))];</pre>
2829	<pre>lx = length(xAnim);</pre>
2830	<pre>ly = length(yAnim);</pre>
2831	<pre>lz = length(zAnim);</pre>
2832	
2833	<pre>fig10 = figure(10);</pre>
2834	fig10.Position = [360 380 1180 400];
2835	%% Temperature Plot %%%%%%%%%%%%
2836	<pre>subplot(1,3,1);</pre>
2837	<pre>ax1 = gca;</pre>
2838	<pre>title('Temperature ( C )');</pre>
2839	<pre>xlim(ax1, [min(xAnim(2:lx)) max(xAnim(2:lx))]);</pre>
2840	<pre>ylim(ax1, [min(yAnim(2:ly)) max(yAnim(2:ly))]);</pre>
2841	<pre>zlim(ax1, [min(zAnim(2:lz)) max(zAnim(2:lz))]);</pre>
2842	view(ax1, 3)
2843	grid on
2844	<pre>zl = zlabel('Altitude (m, AGL)');</pre>
2845	<pre>xticks(min(xAnim):((max(xAnim)-min(xAnim))/4):max(xAnim));</pre>
2846	<pre>yticks(min(yAnim):((max(yAnim)-min(yAnim))/4):max(yAnim));</pre>
2847	<pre>zticks(min(zAnim):((max(zAnim)-min(zAnim))/5):max(zAnim));</pre>
```
xtickformat('%.3f')
2848
                ytickformat('%.3f')
2849
                ztickformat('%.0f')
2850
                set(ax1, 'Color', 'k', 'xcolor', 'w', 'ycolor', 'w', 'zcolor', 'w', 'LineWidth',2)
2851
2852
                data_range1 = ceil(max(max(cAnim1(2:end)))) - ...
2853
                    floor(min(min(cAnim1(2:end)))) + 1;
                colormap(jet(data_range1*10));
2854
                caxis([min(min(cAnim1(2:end))) max(max(cAnim1(2:end)))])
2855
2856
                cbh1 = colorbar();
                cbh1.Ticks = ...
2857
                    round (min (min (cAnim1 (2:end))): ((max (max (cAnim1 (2:end))) - min (min (cAnim1 (2
                ticks1 = strsplit(num2str(cbh1.Ticks));
2858
                ax01 = axes('Position', cbh1.Position);
2859
                edges1 = linspace(0,1,numel(ticks1)+1);
2860
                centers1 = edges1(2:end) - ((edges1(2) - edges1(1))/2);
2861
                text(ones(size(centers1))*0.5, centers1, ticks1, ...
2862
                    'FontSize', ...
                    cbh1.FontSize, 'BackgroundColor', 'w', 'Margin', 1, 'HorizontalAlignment', ...
                    'Center', 'VerticalAlignment', 'Middle');
                ax01.Visible = 'off'; %turn off new axes
2863
                cbh1.Ticks = [];
2864
                2865
                subplot(1,3,2);
2866
                ax2 = qca;
2867
                title('Relative Humidity (%)');
2868
                xlim(ax2, [min(xAnim(2:lx)) max(xAnim(2:lx))]);
2869
                ylim(ax2, [min(yAnim(2:ly)) max(yAnim(2:ly))]);
2870
                zlim(ax2, [min(zAnim(2:lz)) max(zAnim(2:lz))]);
2871
                view(ax2, 3)
2872
                grid on
2873
2874
                xticks(min(xAnim):((max(xAnim)-min(xAnim))/4):max(xAnim));
```

```
yticks(min(yAnim):((max(yAnim)-min(yAnim))/4):max(yAnim));
2875
                zticks(min(zAnim):((max(zAnim)-min(zAnim))/5):max(zAnim));
2876
                xtickformat('%.3f')
2877
                ytickformat('%.3f')
2878
                ztickformat('%.0f')
2879
                set(ax2,'Color','k','xcolor','w','ycolor','w','zcolor','w','LineWidth',2)
2880
2881
                data_range2 = ceil(max(max(cAnim2(2:end)))) - ...
2882
                    floor(min(min(cAnim2(2:end)))) + 1;
2883
                colormap(jet(data_range2*10));
                caxis([min(min(cAnim2(2:end))) max(max(cAnim2(2:end)))])
2884
                cbh2 = colorbar();
2885
                cbh2.Ticks = ...
2886
                    round (min (min (cAnim2 (2:end))): ((max (max (cAnim2 (2:end))) - min (min (cAnim2 (2
                ticks2 = strsplit(num2str(cbh2.Ticks));
2887
                ax02 = axes('Position', cbh2.Position);
2888
                edges2 = linspace(0,1,numel(ticks2)+1);
2889
                centers2 = edges2(2:end) - ((edges2(2) - edges2(1))/2);
2890
                text(ones(size(centers2))*0.5, centers2, ticks2, ...
2891
                    'FontSize', ...
                    cbh2.FontSize, 'BackgroundColor', 'w', 'Margin', 1, 'HorizontalAlignment', ...
                    'Center', 'VerticalAlignment', 'Middle');
                ax02.Visible = 'off'; %turn off new axes
2892
                cbh2.Ticks = [];
2893
                2894
                subplot(1,3,3);
2895
                ax3 = qca;
2896
                title('Pressure (mbar)');
2897
                xlim(ax3, [min(xAnim(2:lx)) max(xAnim(2:lx))]);
2898
                ylim(ax3, [min(yAnim(2:ly)) max(yAnim(2:ly))]);
2899
                zlim(ax3, [min(zAnim(2:lz)) max(zAnim(2:lz))]);
2900
2901
                view(ax3, 3)
```

2902	grid on
2903	<pre>xticks(min(xAnim):((max(xAnim)-min(xAnim))/4):max(xAnim));</pre>
2904	<pre>yticks(min(yAnim):((max(yAnim)-min(yAnim))/4):max(yAnim));</pre>
2905	<pre>zticks(min(zAnim):((max(zAnim)-min(zAnim))/5):max(zAnim));</pre>
2906	<pre>xtickformat('%.3f')</pre>
2907	<pre>ytickformat('%.3f')</pre>
2908	<pre>ztickformat('%.0f')</pre>
2909	<pre>set(ax3,'Color','k','xcolor','w','ycolor','w','zcolor','w','LineWidth',2)</pre>
2910	
2911	<pre>data_range3 = ceil(max(max(cAnim3(2:end))))</pre>
	<pre>floor(min(min(cAnim3(2:end)))) + 1;</pre>
2912	<pre>colormap(jet(data_range3*10));</pre>
2913	<pre>caxis([min(min(cAnim3(2:end))) max(max(cAnim3(2:end)))])</pre>
2914	cbh3 = colorbar();
2915	cbh3.Ticks =
	<pre>round(min(cAnim3(2:end)):((max(cAnim3(2:end))-min(cAnim3(2:end)))/5):max</pre>
2916	<pre>ticks3 = strsplit(num2str(cbh3.Ticks));</pre>
2917	<pre>ax03 = axes('Position', cbh3.Position);</pre>
2918	<pre>edges3 = linspace(0,1,numel(ticks3)+1);</pre>
2919	centers3 = edges3(2:end)-((edges3(2)-edges3(1))/2);
2920	<pre>text(ones(size(centers3))*0.5, centers3, ticks3,</pre>
	'FontSize',
	cbh3.FontSize, 'BackgroundColor', 'w', 'Margin', 1, 'HorizontalAlignment',
	<pre>'Center', 'VerticalAlignment', 'Middle');</pre>
2921	<pre>ax03.Visible = 'off'; %turn off new axes</pre>
2922	cbh3.Ticks = [];
2923	%% Position Control
2924	<pre>subfig = get(gcf,'children');</pre>
2925	<pre>set(subfig(1),'position',[.98115 0 2.1]); % Pressure</pre>
	color bar labels
2926	<pre>%set(subfig(1),'position',[0.98 0.19 0 .62]);</pre>

2927	<pre>set(subfig(2),'position',[.97 .2125 .02 .58]); %</pre>
	Pressure color bar
2928	<pre>set(subfig(3),'position',[.7025 .06 .265 .88]); %</pre>
	Pressure mian plotting area
2929	<pre>set(subfig(4), 'position', [.6545 0.19 0 .62]); % Humidity</pre>
	color bar labels
2930	<pre>set(subfig(5),'position',[.645 .2125 .02 .58]); %</pre>
	Humidity color bar
2931	<pre>set(subfig(6),'position',[.3775 .06 .265 .88]); %</pre>
	Humidity main plotting area
2932	<pre>set(subfig(7),'position',[.33 0.19 0 .62]); % Temperature</pre>
	color bar labels
2933	<pre>set(subfig(8),'position',[.32 .2125 .02 .58]); %</pre>
	Temperature color bar
2934	<pre>set(subfig(9),'position',[.053 .06 .265 .88]); %</pre>
	Temperature main plotting area
2935	
2936	<pre>view(subfig(3),-45,20);</pre>
2937	<pre>view(subfig(6),-45,20);</pre>
2938	<pre>view(subfig(9),-45,20);</pre>
2939	
2940	AnimPos = [.025 .88 .09 .1];
2941	zl.Position = [34.9796 -97.51672 744.2306];
2942	%% Set Axis Colors
2943	<pre>ax1.ZTickLabel(1) = {'0'};</pre>
2944	<pre>ax2.ZTickLabel(1) = {'0'};</pre>
2945	<pre>ax3.ZTickLabel(1) = {'0'};</pre>
2946	
2947	<pre>for i=1:length(ax1.XTickLabel)</pre>
2948	ax1.XTickLabel{i} = ['\color[rgb] $\{0,0,0\}$ '
	<pre>ax1.XTickLabel{i}];</pre>

2949	ax2.XTickLabel{i} = ['\color[rgb] $\{0,0,0\}$ '
	<pre>ax2.XTickLabel{i}];</pre>
2950	ax3.XTickLabel{i} = ['\color[rgb] $\{0,0,0\}$ '
	<pre>ax3.XTickLabel{i}];</pre>
2951	end
2952	<pre>for j=1:length(ax1.YTickLabel)</pre>
2953	ax1.YTickLabel{j} = ['\color[rgb] $\{0,0,0\}$ '
	<pre>ax1.YTickLabel{j}];</pre>
2954	ax2.YTickLabel{j} = ['\color[rgb]{0,0,0}'
	<pre>ax2.YTickLabel{j}];</pre>
2955	<pre>ax3.YTickLabel{j} = ['\color[rgb]{0,0,0}'</pre>
	<pre>ax3.YTickLabel{j}];</pre>
2956	end
2957	<pre>for k=1:length(ax1.ZTickLabel)</pre>
2958	ax1.ZTickLabel{k} = ['\color[rgb]{0,0,0}'
	<pre>ax1.ZTickLabel{k}];</pre>
2959	ax2.ZTickLabel{k} = ['\color[rgb]{0,0,0}'
	<pre>ax2.ZTickLabel{k}];</pre>
2960	ax3.ZTickLabel{k} = ['\color[rgb]{0,0,0}'
	<pre>ax3.ZTickLabel{k}];</pre>
2961	end
2962	
2963	ax1.XTickLabel(i) = $\{ ' ' \};$
2964	ax2.XTickLabel(i) = $\{ ' ' \};$
2965	ax3.XTickLabel(i) = { ' '};
2966	ax1.YTickLabel(j) = $\{ ' ' \};$
2967	$ax2.YTickLabel(j) = { ' ' };$
2968	ax3.YTickLabel(j) = {' '};
2969	
2970	<pre>zl.Color = 'k';</pre>
2971	%% Animation Function
2972	open(animVid);

```
for jj=1:interval:length(xAnim);
2973
2974
                      try
                          p1 = ...
2975
                              patch(ax1,xAnim(1:jj),yAnim(1:jj),zAnim(1:jj),cAnim1(1:jj),'Edge
                              view(-45,45)
                          p2 = ...
2976
                              patch(ax2,xAnim(1:jj),yAnim(1:jj),zAnim(1:jj),cAnim2(1:jj),'Edge
                              view(-45,45)
                          p3 = ...
2977
                              patch(ax3,xAnim(1:jj),yAnim(1:jj),zAnim(1:jj),cAnim3(1:jj),'Edge
                             view(-45,45)
2978
                          delete(findall(fig10, 'type', 'annotation'));
2979
                          Seconds = floor(mod(timer(jj,1),60));
2980
                          Minutes = fix(timer(jj, 1)/60);
2981
2982
                          if(Seconds<10)</pre>
2983
                               String = {sprintf('Time ...
2984
                                  (min:sec) \n%.0f:0%.0f', Minutes, Seconds) };
                          else
2985
                               String = {sprintf('Time ...
2986
                                  (min:sec) \n%.0f:%.0f', Minutes, Seconds) };
                          end
2987
                          annotation(fig10,'textbox',AnimPos,'String',String,'HorizontalAlign
2988
2989
2990
                          pause(1/50); %Pause and grab frame
2991
                          frame = getframe(gcf); %get frame
2992
                          writeVideo(animVid, frame);
2993
                          cla(ax1);
2994
                          cla(ax2);
2995
2996
                          cla(ax3);
```

```
catch
2997
                           pause(1/50); %Pause and grab frame
2998
                           frame = getframe(gcf); %get frame
2999
                           writeVideo(animVid, frame);
3000
                           close(animVid);
3001
                           break
3002
3003
                      end
3004
                  end
3005
                  close(animVid);
3006
             elseif(strcmpi(isometric, 'Yes'))
3007
                  %% Animation Setup
3008
3009
                  if(strcmpi(plotTitle,''))
3010
                      plotTitle = 'Default';
3011
                  end
3012
                  WindSpacing = NKF2_table(1:5:end,:);
3013
3014
                  if (height (GPS_table) < height (WindSpacing))</pre>
3015
                      WindSpacing = WindSpacing(1:height(GPS_table),:);
3016
                  else GPS_table = GPS_table(1:height(WindSpacing),:);
3017
3018
                  end
3019
                  Lat = table2array(GPS_table(:, 6));
3020
                 Long = table2array(GPS_table(:,7));
3021
                  Alt = table2array(GPS_table(:,9));
3022
                 minAlt = min(Alt);
3023
                  if(minAlt<0)</pre>
3024
                      Alt=Alt+abs(min(Alt));
3025
                  elseif(minAlt>0)
3026
                      Alt=Alt-min(Alt);
3027
3028
                  end
```

```
NW = table2array(WindSpacing(:,4));
3029
                 EW = table2array(WindSpacing(:,5));
3030
                 WindMag = round(table2array(WindSpacing(:, 6)), 1);
3031
3032
                 limHigh = max(max(abs(EW), abs(NW)));
3033
                 limLow = -limHigh;
3034
3035
                 fiq10 = fiqure(10);
3036
                 fig10.Position=[130 130 800 600];
3037
                 fiq10.Resize = 'off';
3038
3039
                 xAnim1=[nan;Long(:,1)];
3040
                 yAnim1=[nan;Lat(:,1)];
3041
                 zAnim1=[nan;Alt(:,1)];
3042
3043
                 lx = length(xAnim1);
3044
                 ly = length(yAnim1);
3045
                 lz = length(zAnim1);
3046
                 3047
                 subplot (1, 2, 1)
3048
                 % Wind Plotting Data
3049
3050
                 ax1 = qca;
                 % Modified compass figure with higher radial limit
3051
                 c1=compass(ax1,[0 0],[limHigh 0]);
3052
                 set(c1(1), 'visible', 'off')
3053
                 set(ax1, 'ydir', 'reverse')
3054
                 labels = findall(ax1, 'type', 'text');
3055
                 % Altering the angular label
3056
                 set(findall(ax1, 'String', '0'),'String', 'N');
3057
                 set(findall(ax1, 'String', '90'),'String', 'E');
3058
                 set(findall(ax1, 'String', '180'),'String', 'S');
3059
                 set(findall(ax1, 'String', '270'),'String', 'W');
3060
```

```
set(labels(3:6),'visible','off');
3061
                 set(labels(9:12),'visible','off');
3062
3063
                 view(ax1, -90,90)
3064
                title({'Pixhawk-Estimated Wind Rose';''});
3065
                 3066
                 subplot (1, 2, 2)
3067
                 % GPS Plotting data
3068
                 ax2 = qca;
3069
                 xlim(ax2, [min(Long) max(Long)]);
3070
                ylim(ax2, [min(Lat) max(Lat)]);
3071
                 zlim(ax2, [min(Alt) max(Alt)]);
3072
                xl = xlabel('Longitude');
3073
                yl = ylabel('Lattitude');
3074
                 zl = zlabel('Alt (m, AGL)');
3075
3076
                 if(strcmpi(plotTitle, 'Default'))
3077
                     plotTitle = baseNameNoExtDFL;
3078
                 end
3079
3080
                 t = title({plotTitle, ' '}, 'Interpreter', 'none');
3081
                 t.FontSize = 16;
3082
3083
                 view(ax2, 3)
3084
                 xticks(min(Long):((max(Long)-min(Long))/4):max(Long));
3085
                 yticks(min(Lat):((max(Lat)-min(Lat))/4):max(Lat));
3086
                 zticks(min(Alt):((max(Alt)-min(Alt))/4):max(Alt));
3087
                 xtickformat('%.3f')
3088
                 ytickformat('%.3f')
3089
                 ztickformat('%.0f')
3090
                 grid
3091
                 set(ax2,'Color','k','xcolor','w','ycolor','w','zcolor','w','LineWidth',2)
3092
```

```
3093
                 axis(ax2,[min(Long) max(Long) min(Lat) max(Lat) min(Alt) ...
3094
                    max(Alt)]);
                 %% Set Axis Colors
3095
                 for i=1:length(ax2.XTickLabel)
3096
                     ax2.XTickLabel{i} = ['\color[rgb]{0,0,0}' ...
3097
                         ax2.XTickLabel{i}];
                 end
3098
                 for j=1:length(ax2.YTickLabel)
3099
                     ax2.YTickLabel{j} = ['\color[rgb]{0,0,0}' ...
3100
                         ax2.YTickLabel{j}];
3101
                 end
                 for k=1:length(ax2.ZTickLabel)
3102
                     ax2.ZTickLabel{k} = ['\color[rgb]{0,0,0}' ...
3103
                         ax2.ZTickLabel{k}];
                 end
3104
3105
                 %ax2.XTickLabel{i} = []; ax2.YTickLabel{j} = [];
3106
3107
                 xl.Color = 'k';
3108
                 yl.Color = 'k';
3109
                 zl.Color = 'k';
3110
                 %% Position Control
3111
                 subfig1 = get(gcf, 'children');
3112
                 set(subfig1(1),'position',[.1 .075 .8 .8]); % GPS ...
3113
                    animation space
                 set(subfig1(2),'position',[.725 .725 .225 .225]); % Wind rose
3114
3115
3116
                 view(subfig1(1),-45,45);
3117
                 display('Beginning animation sequence');
3118
                 %% Animation Function
3119
```

3120	<pre>if(strcmpi(recordAnimation, 'Yes'));</pre>
3121	open(animVid);
3122	
3123	<pre>for i=1:animateSpeed:length(EW);</pre>
3124	<pre>cl=compass(ax1,[0 EW(i)],[limHigh NW(i)]);</pre>
3125	<pre>set(c1(1),'visible','off');</pre>
3126	<pre>labels = findall(ax1,'type','text');</pre>
3127	<pre>view(ax1, -90,90);</pre>
3128	<pre>set(ax1,'ydir','reverse');</pre>
3129	<pre>set(labels(13),'visible','off');</pre>
3130	<pre>set(labels(14),'visible','off');</pre>
3131	
3132	<pre>set(c1(2),'LineWidth',1.5,'color','red');</pre>
3133	<pre>%annotation('textbox',[.645 .55 .3</pre>
	.3],'String',sprintf('%s
	<pre>%s',num2str(WindMag(i)),'m/s'),'FitBoxToText','on');</pre>
3134	
3135	%removedthis
3136	if ( $i \leq (animateTailLength+1)$ )
3137	pH1 =
	<pre>patch(ax2,'XData',xAnim1(1:i),'YData',yAnim1(1:i),'ZData',zA</pre>
3138	pH2 =
	<pre>patch(ax2,'XData',xAnim1(i),'YData',yAnim1(i),'ZData',zAnim1</pre>
3139	else
3140	pH1 = patch(ax2, 'XData', [nan;
	<pre>xAnim1((i-animateTailLength):i)],'YData',[nan;</pre>
	<pre>yAnim1((i-animateTailLength):i)],'ZData',[nan;</pre>
	<pre>zAnim1((i-animateTailLength):i)],'EdgeColor','r','FaceColor'</pre>
3141	pH2 =
	<pre>patch(ax2,'XData',xAnim1(i),'YData',yAnim1(i),'ZData',zAnim1</pre>
3142	end
3143	

3144	<pre>pause(1/(50)) %Pause and grab frame</pre>
3145	<pre>frame = getframe(gcf); %get frame</pre>
3146	<pre>writeVideo(animVid, frame);</pre>
3147	<pre>delete(findall(fig10, 'type', 'annotation'));</pre>
3148	<pre>cla(ax2);</pre>
3149	
3150	end
3151	
3152	<pre>close(animVid);</pre>
3153	else
3154	<pre>for i=1:animateSpeed:length(EW);</pre>
3155	
3156	<pre>cl=compass(ax1,[0 EW(i)],[limHigh NW(i)]);</pre>
3157	<pre>set(c1(1),'visible','off');</pre>
3158	<pre>labels = findall(ax1,'type','text');</pre>
3159	<pre>view(ax1, -90,90);</pre>
3160	<pre>set(ax1,'ydir','reverse');</pre>
3161	<pre>set(findall(ax1, 'String', '0'),'String', 'N');</pre>
3162	<pre>set(findall(ax1, 'String', '90'),'String', 'E');</pre>
3163	<pre>set(findall(ax1, 'String', '180'),'String', 'S');</pre>
3164	<pre>set(findall(ax1, 'String', '270'),'String', 'W');</pre>
3165	<pre>set(labels(3:6),'visible','off');</pre>
3166	<pre>set(labels(9:14),'visible','off');</pre>
3167	
3168	<pre>set(c1(2),'LineWidth',1.5,'color','red');</pre>
3169	annotation('textbox',[.645 .55 .3
	.3],'String',sprintf('%s
	<pre>%s',num2str(WindMag(i)),'m/s'),'FitBoxToText','on');</pre>
3170	
3171	<pre>if(i&lt;(animateTailLength+1))</pre>
3172	pH1 =
	<pre>patch(ax2,'XData',xAnim1(1:i),'YData',yAnim1(1:i),'ZData',zA</pre>

```
3173
                               pH2 = ...
                                   patch(ax2, 'XData', xAnim1(i), 'YData', yAnim1(i), 'ZData', zAnim1
                           else
3174
                               pH1 = patch(ax2, 'XData', [nan; ...
3175
                                   xAnim1((i-animateTailLength):i)], 'YData', [nan; ...
                                   yAnim1((i-animateTailLength):i)], 'ZData', [nan; ...
                                   zAnim1((i-animateTailLength):i)],'EdgeColor','r','FaceColor'
                               pH2 = ...
3176
                                   patch(ax2, 'XData', xAnim1(i), 'YData', yAnim1(i), 'ZData', zAnim1
3177
                           end
3178
                           pause(1/(50)) %Pause and grab frame
3179
3180
                           delete(findall(fig10, 'type', 'annotation'));
3181
                           cla(ax2);
3182
3183
                      end
3184
                  end
3185
             else
3186
                  %% Animation Setup
3187
3188
                  if(strcmpi(plotTitle,''))
3189
                      plotTitle = 'Default';
3190
                  end
3191
                 WindSpacing = NKF2_table(1:5:end,:);
3192
3193
                 if (height (GPS_table) < height (WindSpacing))</pre>
3194
                      WindSpacing = WindSpacing(1:height(GPS_table),:);
3195
                  else GPS_table = GPS_table(1:height(WindSpacing),:);
3196
3197
                  end
3198
3199
                 Lat = table2array(GPS_table(:, 6));
```

```
Long = table2array(GPS_table(:,7));
3200
                 Alt = table2array(GPS_table(:,9));
3201
                 minAlt = min(Alt);
3202
                 if(minAlt<0)</pre>
3203
                     Alt=Alt+abs(min(Alt));
3204
                 elseif(minAlt>0)
3205
                     Alt=Alt-min(Alt);
3206
                 end
3207
                 NW = table2array(WindSpacing(:,4));
3208
                 EW = table2array(WindSpacing(:, 5));
3209
                 WindMag = round(table2array(WindSpacing(:,6)),1);
3210
3211
                 limHigh = max(max(abs(EW), abs(NW)));
3212
                 limLow = -limHigh;
3213
3214
                 fig10 = figure(10);
3215
                 fig10.Position=[130 130 800 600];
3216
                 fiq10.Resize = 'off';
3217
3218
                 xAnim1=[nan;Long(:,1)];
3219
                 yAnim1=[nan;Lat(:,1)];
3220
                 zAnim1=[nan;Alt(:,1)];
3221
3222
                 lx = length(xAnim1);
3223
                 ly = length(yAnim1);
3224
                 lz = length(zAnim1);
3225
                 %% USGS Mapping Data
3226
        baseURL = "https://basemap.nationalmap.gov/ArcGIS/rest/services";
3227
3228
        usgsURL = baseURL + "/BASEMAP/MapServer/tile/${z}/${y}/${x}";
        basemaps = ["USGSImageryOnly" "USGSImageryTopo" "USGSTopo" ...
3229
            "USGSShadedReliefOnly" "USGSHydroCached"];
```

```
displayNames = ["USGS Imagery" "USGS Topographic Imagery" "USGS ...
3230
            Shaded Topographic Map" "USGS Shaded Relief" "USGS Hydrography"];
        attribution = 'Credit: U.S. Geological Survey';
3231
                3232
                % GPS Plotting data
3233
                ax2 = geoaxes();
3234
                geobasemap('satellite')
3235
                basemapx = basemaps(2);
3236
                url = replace(usgsURL, "BASEMAP", basemapx);
3237
                view(ax2,2)
3238
3239
                if(strcmpi(plotTitle, 'Default'))
3240
                     plotTitle = baseNameNoExtDFL;
3241
3242
                 end
3243
                 t = title({plotTitle, ' '}, 'Interpreter', 'none');
3244
                t.FontSize = 16;
3245
3246
                display('Beginning animation sequence');
3247
                %% Animation Function
3248
                if(strcmpi(recordAnimation, 'Yes'));
3249
                     open(animVid);
3250
3251
                     for i=1:animateSpeed:length(EW);
3252
3253
                         %annotation('textbox',[.645 .55 .3 ...
3254
                             .3], 'String', sprintf('%s ...
                             %s',num2str(WindMag(i)),'m/s'),'FitBoxToText','on');
3255
                         if(i < (animateTailLength+1))</pre>
3256
                             hold on
3257
```

3258	pH1 =
	<pre>geoplot(ax2,yAnim1(1:i),xAnim1(1:i),'r','LineWidth',animateT</pre>
3259	pH2 =
	<pre>geoplot(ax2,yAnim1(i),xAnim1(i),'Marker','o','MarkerSize',an</pre>
3260	else
3261	hold on
3262	pH1 = geoplot(ax2,[nan;
	<pre>yAnim1((i-animateTailLength):i)],[nan;</pre>
	<pre>xAnim1((i-animateTailLength):i)],'r','LineWidth',animateTail</pre>
3263	pH2 =
	<pre>geoplot(ax2,yAnim1(i),xAnim1(i),'Marker','o','MarkerSize',an</pre>
3264	end
3265	
3266	<pre>geobasemap('satellite')</pre>
3267	geolimits([min(yAnim1)0005
	<pre>max(yAnim1)+.0005],[min(xAnim1)0005</pre>
	<pre>max(xAnim1)+.0005]);</pre>
3268	
3269	<pre>pause(1/(50)) %Pause and grab frame</pre>
3270	<pre>frame = getframe(gcf); %get frame</pre>
3271	<pre>writeVideo(animVid, frame);</pre>
3272	<pre>delete(findall(fig10,'type','annotation'));</pre>
3273	<pre>cla(ax2);</pre>
3274	hold off
3275	
3276	end
3277	
3278	<pre>close(animVid);</pre>
3279	else
3280	<pre>for i=1:animateSpeed:length(EW);</pre>
3281	
3282	<pre>if(i≤(animateTailLength+1))</pre>

3283	hold on	
3284	pH1 =	
	<pre>geoplot(ax2,yAnim1(1:i),xAnim1(1:i),'r','LineWidth',anima</pre>	tel
3285	pH2 =	
	<pre>geoplot(ax2,yAnim1(i),xAnim1(i),'Marker','o','MarkerSize'</pre>	, an
3286	else	
3287	hold on	
3288	pH1 = geoplot(ax2,[nan;	
	<pre>yAnim1((i-animateTailLength):i)],[nan;</pre>	
	<pre>xAnim1((i-animateTailLength):i)],'r','LineWidth',animateTailLength):i)</pre>	ail
3289	pH2 =	
	<pre>geoplot(ax2,yAnim1(i),xAnim1(i),'Marker','o','MarkerSize'</pre>	, an
3290	end	
3291		
3292	<pre>geobasemap('satellite')</pre>	
3293	geolimits([min(yAnim1)0005	
	<pre>max(yAnim1)+.0005],[min(xAnim1)0005</pre>	
	<pre>max(xAnim1)+.0005]);</pre>	
3294		
3295	<pre>pause(1/(50)) %Pause and grab frame</pre>	
3296	<pre>delete(findall(fig10,'type','annotation'));</pre>	
3297	<pre>cla(ax2);</pre>	
3298	hold off	
3299		
3300	end	
3301	end	
3302	end	
3303	end	
3304	%% Hover/Profile Averaging	
3305	<pre>if(strcmpi(hoverProfile, 'Hover')   strcmpi(hoverProfile, 'Profile'))</pre>	
3306	<pre>warning('off','MATLAB:ISMEMBER:RowsFlagIgnored');</pre>	
3307	<pre>warning('off','MATLAB:table:RowsAddedExistingVars');</pre>	

```
%% Merge Suite and GPS data
3308
             for i=1:height(TPH_table)
3309
                 currentTPH = table2cell(TPH_table(i,5));
3310
                 currentTPH = currentTPH{1};
3311
                 A(i,1) = cellstr(currentTPH(1:end-2));
3312
             end
3313
3314
             for i=1:height(GPS_table)
3315
                 currentGPS = table2cell(GPS_table(i,5));
3316
                 currentGPS = currentGPS{1};
3317
                 B(i,1) = cellstr(datetime(currentGPS, 'Format', 'HH:mm:ss.S'));
3318
3319
             end
3320
             idx1 = ismember(A, B, 'rows');
3321
3322
             count=1;
3323
             for i=1:length(idx1)
3324
                 if(idx1(i) == 1)
3325
                      iteration1(count,1) = i;
3326
                      tempVals(count, 1) = A(i);
3327
                      count=count+1;
3328
3329
                 end
             end
3330
3331
             GPS_time_alone = cellstr(GPS_full_time(:,1:end-2));
3332
             idx2 = ismember(GPS_time_alone,tempVals,'rows');
3333
3334
             count=1;
3335
3336
             for i=1:length(idx2)
                 if(idx2(i) == 1)
3337
                      iteration2(count,1) = i;
3338
3339
                      count=count+1;
```

```
3340
                 end
3341
            end
3342
            GPS_table_new = GPS_table(iteration2,:);
3343
3344
            count=1;
3345
            for i=1:height(GPS_table_new)
3346
                 test = ...
3347
                     find(BARO(:,1) > table2array(GPS_table_new(i,1)),1,'first');
3348
                 if (test > 1)
                     iteration3(count,1) = test;
3349
3350
                     count=count+1;
                 end
3351
3352
            end
3353
            itLen = [length(iteration1) length(iteration2) length(iteration3)];
3354
3355
            finalTable = [GPS_table(iteration2(1:min(itLen)),4:10) ...
3356
                BARO_table(iteration3(1:min(itLen)),4:5) ...
                TPH_table(iteration1(1:min(itLen)),6:11)];
3357
            baseFileName = sprintf('%s_Parsed_Pix_Suite.csv', ...
3358
                baseNameNoExtDFL);
            fullOutputMatFileName = fullfile(folderDFL, baseFileName);
3359
            % Write data to text file
3360
            writetable(finalTable, fullOutputMatFileName);
3361
            %% Hover with 10 second averaging
3362
            if(strcmpi(hoverProfile, 'Hover'))
3363
                 count=1;
3364
                 for i=1:50:(height(finalTable)-50)
3365
                     tempArray1(count,1) = finalTable(i,1);
3366
                     tempArray1(count,2) = finalTable(i,2);
3367
```

3368	<pre>tempArray1(count,3:7) = finalTable(i,3:7);</pre>
3369	<pre>tempArray1(count,8:15) =</pre>
	<pre>array2table(mean(table2array(finalTable(i:(i+50),8:15)));</pre>
3370	<pre>count=count+1;</pre>
3371	end
3372	
3373	<pre>tempArray1.Properties.VariableNames =</pre>
	finalTable.Properties.VariableNames;
3374	<pre>baseFileName = sprintf('%s_Suite_10sec_Av.csv',</pre>
	<pre>baseNameNoExtDFL);</pre>
3375	<pre>fullOutputMatFileName = fullfile(folderDFL, baseFileName);</pre>
3376	% Write data to text file
3377	<pre>writetable(tempArray1, fullOutputMatFileName);</pre>
3378	end
3379	%% Profile with 10 meter averaging
3380	<pre>if(strcmpi(hoverProfile, 'Profile'))</pre>
3381	count=1;
3382	<pre>iter=1;</pre>
3383	<pre>startAlt = table2array(finalTable(iter,6));</pre>
3384	<pre>for i=1:height(finalTable)</pre>
3385	<pre>curAlt = table2array(finalTable(i,6));</pre>
3386	
3387	if((abs(curAlt - startAlt)) $\geq$ 10)
3388	<pre>tempArray2(count,1) =</pre>
	<pre>array2table(mean(table2array(finalTable(iter:i,1)));</pre>
3389	<pre>tempArray2(count,2) =</pre>
	<pre>array2table(mean(table2array(finalTable(iter:i,2)));</pre>
3390	$tempArray2(count, 3:15) = \dots$
	<pre>array2table(mean(table2array(finalTable(iter:i,3:15))));</pre>
3391	iter=i;
3392	<pre>count=count+1;</pre>
3393	<pre>startAlt = table2array(finalTable(iter,6));</pre>

```
3394
                     end
3395
                 end
3396
                 tempArray2.Properties.VariableNames = ...
3397
                    finalTable.Properties.VariableNames;
                 baseFileName = sprintf('%s_Suite_10m_Av.csv', ...
3398
                    baseNameNoExtDFL);
                 fullOutputMatFileName = fullfile(folderDFL, baseFileName);
3399
                 % Write data to text file
3400
                 writetable(tempArray2, fullOutputMatFileName);
3401
            end
3402
            warning('on', 'MATLAB:ISMEMBER:RowsFlagIgnored');
3403
            warning('on', 'MATLAB:table:RowsAddedExistingVars');
3404
3405
        end
3406 else
3407
        %% USGS Mapping Data
3408
        baseURL = "https://basemap.nationalmap.gov/ArcGIS/rest/services";
3409
        usgsURL = baseURL + "/BASEMAP/MapServer/tile/${z}/${y}/${x}";
3410
        basemaps = ["USGSImageryOnly" "USGSImageryTopo" "USGSTopo" ...
3411
            "USGSShadedReliefOnly" "USGSHydroCached"];
        displayNames = ["USGS Imagery" "USGS Topographic Imagery" "USGS ...
3412
            Shaded Topographic Map" "USGS Shaded Relief" "USGS Hydrography"];
        attribution = 'Credit: U.S. Geological Survey';
3413
        %% Load All GPS Files
3414
3415
        [baseFileName, folder]=uigetfile('*.mat', 'Select the INPUT DATA ...
3416
            FILE(s)', 'MultiSelect', 'on');
3417
        disp('Parsing and merging Pixhawk files.')
3418
        if(strcmpi(simultaneous, 'Yes'))
3419
3420
            for i=1:length(baseFileName)
```

```
3421
                 fullInputMatFileName(i) = fullfile(folder, baseFileName(i));
                 load(char(fullInputMatFileName(i)),'GPS');
3422
3423
                 Time{:,i} = GPS(2:end,4);
3424
                 Speed{:,i} = GPS(2:end,11);
3425
                 Lat{:,i} = GPS(2:end,8);
3426
                 Long{:,i} = GPS(2:end,9);
3427
                 Alt{:,i} = GPS(2:end,10);
3428
                 minTime(i) = min(GPS(2:end, 4));
3429
                 maxTime(i) = max(GPS(2:end, 4));
3430
                 clear GPS
3431
3432
             end
3433
3434
        else
             for i=1:length(baseFileName)
3435
                 fullInputMatFileName(i) = fullfile(folder, baseFileName(i));
3436
                 load(char(fullInputMatFileName(i)),'GPS');
3437
3438
                 rawLat{:,i} = GPS(2:end,8);
3439
                 rawLong{:, i} = GPS(2:end, 9);
3440
                 rawAlt(:,i) = GPS(2:end,10);
3441
                 rawSpeed{:,i} = GPS(2:end,11);
3442
                 clear GPS
3443
3444
             end
3445
3446
             for k=1:i
3447
                 Speed = [rawSpeed{k}];
3448
3449
                 tempLat = [rawLat{k}];
                 tempLong = [rawLong\{k\}];
3450
                 tempAlt = [rawAlt{k}];
3451
3452
                 Lat\{k\} = tempLat(Speed>.5);
```

```
3453
                 Long\{k\} = tempLong(Speed>.5);
                 Alt{k} = tempAlt(Speed>.5);
3454
                 clear tempLat tempLong tempAlt Speed
3455
             end
3456
        end
3457
3458
3459
        disp('Completed Pixhawk file manipulation.');
3460
        %% Data Merging
3461
        % Combine all datasets into arrays of same size
3462
        % find all takeoff times
3463
        % find first aircraft to takeoff
3464
        % convert times to basic plotting number (incriment of 1)
3465
        % keep lat/long of all grounded aircraft, but pad with plotting number
3466
        % plot all aircraft according to plotting number, with lat/long being
3467
          the only differntiating factor
        8
3468
3469
        if(strcmpi(simultaneous, 'Yes'))
3470
3471
            minT = min(minTime);
3472
            maxT = max(maxTime);
3473
            translate(:,1) = minT:200:maxT;
3474
            translate(:,2) = 1:length(minT:200:maxT);
3475
             startLat = cellfun((v)v(1), Lat);
3476
             startLong = cellfun(@(v)v(1),Long);
3477
             startAlt = cellfun(@(v)v(1), Alt);
3478
             endLat = cellfun(@(v)v(end),Lat);
3479
             endLong = cellfun(@(v)v(end),Long);
3480
3481
             endAlt = cellfun(@(v)v(end), Alt);
3482
             finLat(1:length(translate(:,1)),1:i)=0;
3483
3484
             finLong(1:length(translate(:,1)),1:i)=0;
```

3485	<pre>finAlt(1:length(translate(:,1)),1:i)=0;</pre>
3486	<pre>finTime(:,1) = translate(:,2);</pre>
3487	<pre>finSpeed(1:length(translate(:,1)),1:i) = 0;</pre>
3488	
3489	<pre>for(j = 1:i)</pre>
3490	<pre>curTime = round(cell2mat(Time(j)),-2);</pre>
3491	<pre>curLat = cell2mat(Lat(j));</pre>
3492	<pre>curLong = cell2mat(Long(j));</pre>
3493	<pre>curAlt = cell2mat(Alt(j));</pre>
3494	<pre>curSpeed = cell2mat(Speed(j));</pre>
3495	<pre>count=0;</pre>
3496	<pre>for k = 1:length(curTime);</pre>
3497	<pre>Catch = find(curTime(k,1)==translate(:,1),1,'first');</pre>
3498	if(Catch≥0)
3499	<pre>count=count+1;</pre>
3500	if (count $\neq$ 1)
3501	<pre>finLat(Catch,j) = curLat(k,1);</pre>
3502	<pre>finLong(Catch,j) = curLong(k,1);</pre>
3503	<pre>finAlt(Catch,j) = curAlt(k,1);</pre>
3504	<pre>finSpeed(Catch,j) = curSpeed(k,1);</pre>
3505	else
3506	<pre>finLat(1:Catch,j)=curLat(k,1);</pre>
3507	<pre>finLong(1:Catch, j)=curLong(k, 1);</pre>
3508	<pre>finAlt(1:Catch,j)=curAlt(k,1);</pre>
3509	<pre>finSpeed(1:Catch,j) = curSpeed(k,1);</pre>
3510	end
3511	end
3512	end
3513	<pre>finLat(Catch:end,j) = curLat(end,1);</pre>
3514	<pre>finLong(Catch:end,j) = curLong(end,1);</pre>
3515	<pre>finAlt(Catch:end,j) = curAlt(end,1);</pre>
3516	<pre>finSpeed(Catch:end,j) = curSpeed(end,1);</pre>

```
3517
                 clear curTime curLat curLong curAlt
3518
             end
3519
             for col = 1:size(finLat,2)
3520
                 for row = 1:size(finLat,1)
3521
                      if finLat(row, col) == 0
3522
                           finLat(row,col) = finLat(row+1,col);
3523
                      end
3524
                 end
3525
3526
             end
3527
             for col = 1:size(finLong,2)
3528
                 for row = 1:size(finLong,1)
3529
                      if finLong(row, col) == 0
3530
                           finLong(row, col) = finLong(row+1, col);
3531
                      end
3532
                 end
3533
3534
             end
3535
             for col = 1:size(finAlt,2)
3536
                 for row = 1:size(finAlt,1)
3537
                      if finAlt(row, col) == 0
3538
                           finAlt(row, col) = finAlt(row-1, col);
3539
                      end
3540
                 end
3541
3542
             end
3543
             for(k = 1:i)
3544
                 try startSpeed(k) = find(finSpeed(:,k) ≥1,1,'first');
3545
                      endSpeed(k) = find(finSpeed(:,k) >1,1,'last');
3546
3547
3548
                 catch startSpeed(k) = -1;
```

```
endSpeed(k) = -1;
3549
                  end
3550
             end
3551
3552
             firstSpeed = min(startSpeed(startSpeed > 0));
3553
             lastSpeed = max(endSpeed(endSpeed > 0));
3554
3555
             startSpeed(startSpeed<0) = firstSpeed;</pre>
3556
             endSpeed(endSpeed<0) = lastSpeed;</pre>
3557
3558
             trimStart = min(startSpeed);
3559
             trimEnd = max(endSpeed);
3560
3561
             finLat = finLat(trimStart:trimEnd,:);
3562
             finLong = finLong(trimStart:trimEnd,:);
3563
             finAlt = finAlt(trimStart:trimEnd,:);
3564
             finTime = finTime(trimStart:trimEnd,:);
3565
         end
3566
         %% Animation Function
3567
3568
         % Color Setup
3569
         ColOrd = [1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1; \ 1 \ 1 \ 1; \ 0 \ 0 \ 0; \ 1 \ 1 \ 0; \ 1 \ 0 \ 1; \ 0 \ 1 \ 1];
3570
         [m, n] = size(ColOrd);
3571
3572
         % Animation of the Pixhawk Log from recorded GPS coordinates
3573
         if(strcmpi(simultaneous, 'Yes'))
3574
             minLat = min(min(finLat));
3575
             maxLat = max(max(finLat));
3576
             minLong = min(min(finLong));
3577
             maxLong = max(max(finLong));
3578
             minAlt = min(min(finAlt));
3579
             maxAlt = max(max(finAlt));
3580
```

```
3581
             if(strcmpi(animation, 'No'))
3582
3583
                  disp('Beginning plot setup.');
3584
3585
                 fig1=figure(1);
3586
                  fig1.Position = [10 50 560 725];
3587
                  fig1.Resize = 'off';
3588
3589
                  ax = geoaxes();
3590
                 geobasemap('satellite')
3591
                 basemapx = basemaps(2);
3592
                 url = replace(usgsURL, "BASEMAP", basemapx);
3593
                 view(ax,2)
3594
3595
                  disp('Starting the plotting process.');
3596
3597
                  for(j =1:i)
3598
                      ColRow = rem(j,m);
3599
                      if ColRow == 0
3600
                           ColRow = m;
3601
3602
                      end
                      Col = ColOrd(ColRow,:);
3603
3604
                      hold on
3605
3606
                      geoplot(ax,finLat(:,j),finLong(:,j),'Color',Col,'LineWidth',.8);
3607
                      geoplot(ax,finLat(end,j),finLong(end,j),'LineStyle','none','Marker','o',
3608
3609
                      geobasemap('satellite')
3610
                      geolimits([minLat-.0005 maxLat+.0005],[minLong-.0005 ...
3611
                          maxLong+.0005]);
```

```
3612
                 end
3613
                 title(plotTitle);
3614
3615
                 baseFileNamePic = sprintf('MergedData_Image10');
3616
                 fullParsedMatFileNamePic = fullfile(folder, baseFileNamePic);
3617
3618
                 saveas(fig,fullParsedMatFileNamePic,'png')
3619
3620
3621
                 disp('Successfully completed plot.');
3622
3623
             elseif (strcmpi(animation, 'Yes'))
3624
3625
                 disp('Beginning animation setup.');
3626
3627
                 animVid = VideoWriter('Kessler 8-4-2020, ...
3628
                     TriplePlot', 'MPEG-4'); %open video file
                 animVid.FrameRate = 20; %can adjust this, 5 - 10 works ...
3629
                     well for me
                 animVid.Quality = 100;
3630
3631
                 tot = ceil(length(finLat(:,1))/iterationSkip);
3632
3633
                 fig1 = figure(1);
3634
                 fig1.Position = [10 50 560 725];
3635
                 fig1.Resize = 'off';
3636
3637
                 if(strcmpi(isometric, 'No'))
3638
3639
                     ax = geoaxes();
3640
                      geobasemap('satellite')
3641
```

```
basemapx = basemaps(2);
3642
                      url = replace(usgsURL, "BASEMAP", basemapx);
3643
                      view(ax,2)
3644
                      title(plotTitle);
3645
3646
                      disp('Starting the animation process.');
3647
3648
                      open(animVid);
3649
3650
                      ptot = 0;
3651
3652
                      for jj=1:iterationSkip:length(finLat(:,1))
3653
                          hold on
3654
                           for k = 1:i
3655
                               ColRow = rem(k,m);
3656
                               if ColRow == 0
3657
                                   ColRow = m;
3658
                               end
3659
                               Col = ColOrd(ColRow,:);
3660
                               hold on
3661
3662
                               geoplot(ax,finLat(1:jj,k),finLong(1:jj,k),'Color',Col,'LineWidth
3663
                               geoplot(ax,finLat(jj,k),finLong(jj,k),'LineStyle', 'none', 'Marker
3664
3665
                               geobasemap('satellite')
3666
                               geolimits([minLat-.0005 ...
3667
                                   maxLat+.0005],[minLong-.0005 maxLong+.0005]);
                           end
3668
                           pause(.001);
3669
                           frame = getframe(gcf);
3670
                          writeVideo(animVid, frame);
3671
3672
                           ptot = ptot+1;
```

```
display = sprintf('Iteration: %d of %d, %s%% ...
3673
                              complete',ptot,tot,num2str(round(ptot/tot*100,1)));
                          disp(display);
3674
                          cla(ax);
3675
                          hold off
3676
                      end
3677
3678
                      close(animVid);
3679
3680
                      disp('Successfully completed animation.');
3681
                 else
3682
                      ax = qca();
3683
                      axis([minLat maxLat minLong maxLong minAlt maxAlt]);
3684
                      view(ax, 3)
3685
                      xticks(minLat:((maxLat-minLat)/4):maxLat);
3686
                      yticks(minLong:((maxLong-minLong)/4):maxLong);
3687
                      zticks(minAlt:((maxAlt-minAlt)/5):maxAlt);
3688
                      xtickformat('%.lf')
3689
                      ytickformat('%.1f')
3690
                      ztickformat('%.0f')
3691
                      title(plotTitle);
3692
                      grid
3693
3694
                      view(ax,[azimuthAngle elevationAngle]);
3695
3696
                      disp('Starting the animation process.');
3697
3698
                      open(animVid);
3699
3700
                      ptot = 0;
3701
3702
3703
                      for jj=1:iterationSkip:length(finLat(:,1))
```

3704 hold on for k = 1:i3705 ColRow = rem(k,m); 3706 if ColRow == 0 3707 ColRow = m; 3708 end 3709 Col = ColOrd(ColRow,:); 3710 hold on 3711 3712plot3(ax,finLat(1:jj,k),finLong(1:jj,k),finAlt(1:jj,k),'Color',( 3713 plot3(ax,finLat(jj,k),finLong(jj,k),finAlt(jj,k),'LineStyle','nd 37143715 end 3716 pause(.001); 3717 frame = getframe(gcf); 3718 writeVideo(animVid, frame); 3719 ptot = ptot+1;3720 display = sprintf('Iteration: %d of %d, %s%% ... 3721 complete',ptot,tot,num2str(round(ptot/tot\*100,1))); disp(display); 3722 cla(ax); 3723 hold off 3724end 3725 3726 close(animVid); 3727 3728 disp('Successfully completed animation.'); 3729 3730 end 3731 end 3732 else 3733 % Animation of the Pixhawk Log from recorded GPS coordinates 3734

```
3735
             minLat = min(min(cell2mat(Lat(:))));
             maxLat = max(max(cell2mat(Lat(:))));
3736
             minLong = min(min(cell2mat(Long(:))));
3737
             maxLong = max(max(cell2mat(Long(:))));
3738
             minAlt = min(min(cell2mat(Alt(:))));
3739
             maxAlt = max(max(cell2mat(Alt(:))));
3740
3741
             if(strcmpi(animation, 'No'))
3742
3743
                 disp('Beginning plot setup.');
3744
3745
                 fig1=figure(1);
3746
                 fig1.Position = [10 50 750 600];
3747
                 fig1.Resize = 'off';
3748
3749
                 ax = geoaxes();
3750
                 geobasemap('satellite')
3751
                 basemapx = basemaps(2);
3752
                 url = replace(usgsURL, "BASEMAP", basemapx);
3753
                 view(ax,2)
3754
3755
                 disp('Starting the plotting process.');
3756
3757
                 for(j =1:i)
3758
                      ColRow = rem(j,m);
3759
                      if ColRow == 0
3760
                           ColRow = m;
3761
3762
                      end
3763
                      Col = ColOrd(ColRow,:);
3764
                      hold on
3765
3766
```

```
3767
                      geoplot(ax,cell2mat(Lat(j)),cell2mat(Long(j)),'Color',Col);
3768
                      geobasemap('satellite')
3769
                      geolimits([minLat-.0005 maxLat+.0005],[minLong-.0005 ...
3770
                         maxLong+.0005]);
                 end
3771
                 title(plotTitle);
3772
3773
                 baseFileNamePic = sprintf('MergedData_Image10');
3774
                 fullParsedMatFileNamePic = fullfile(folder, baseFileNamePic);
3775
3776
                 saveas(fig1,fullParsedMatFileNamePic,'png')
3777
3778
3779
                 disp('Successfully completed plot.');
3780
3781
             else
3782
3783
                 disp('Beginning animation setup.');
3784
3785
                 % Get the name of the intput.mat file and save as ...
3786
                     input_parsed.mat
                 baseFileName = sprintf('MergedData_Animation1');
3787
                 fullParsedMatFileName = fullfile(folder, baseFileName);
3788
3789
                 % Initialize total iteration count
3790
                 tot=0;
3791
                 for l=1:i
3792
                     tot = tot+numel(Lat{1});
3793
                 end
3794
3795
3796
                 tot=ceil(tot/iterationSkip)+1;
```

```
3797
                 animVid = VideoWriter(fullParsedMatFileName, 'MPEG-4');
3798
                 animVid.FrameRate = 20; %can adjust this, 5 - 10 works ...
3799
                     well for me
                 animVid.Quality = 100;
3800
3801
                 fig1 = figure(1);
3802
                 fig1.Position = [10 50 750 600];
3803
                 fig1.Resize = 'off';
3804
3805
                 ax = geoaxes();
3806
                 geobasemap('satellite')
3807
                 basemapx = basemaps(2);
3808
                 url = replace(usgsURL, "BASEMAP", basemapx);
3809
                 view(ax,2)
3810
                 title(plotTitle);
3811
3812
                 disp('Starting the animation process.');
3813
3814
                 open(animVid);
3815
3816
                 ptot = 0;
3817
                 for k=1:i
3818
                     hold on
3819
                      tLat = cell2mat(Lat(k));
3820
                      tLong = cell2mat(Long(k));
3821
                      for jj=1:iterationSkip:length(tLat)
3822
                          ColRow = rem(k,m);
3823
                          if ColRow == 0
3824
                               ColRow = m;
3825
                          end
3826
3827
                          Col = ColOrd(ColRow,:);
```

hold on 3828 3829 3830 geoplot(ax,tLat(1:jj),tLong(1:jj),'Color',Col); 3831 geoplot(ax,tLat(jj),tLong(jj),'LineStyle', 'none', 'Marker', 'o', 'Marker' 3832 3833 **if**(k>1) 3834for x = 1: (k-1)3835 ColRow = rem(x,m); 3836 if ColRow == 0 3837 ColRow = m; 3838 end 3839 Col = ColOrd(ColRow,:); 3840 geoplot(ax,cell2mat(Lat(x)),cell2mat(Long(x)),'Color',Col); 3841 end 3842 3843 end 38443845 geobasemap('satellite') 3846%geolimits([minLat-.0005 ... 3847 maxLat+.0005],[minLong-.0005 maxLong+.0005]); 3848 pause(.001); 3849 frame = getframe(gcf); 3850 writeVideo(animVid, frame); 3851 ptot = ptot+1;3852 display = sprintf('Iteration: %d of %s, %s%% ... 3853 complete',ptot,num2str(tot),num2str(round(ptot/tot\*100,1))); 3854disp(display); cla(ax); 3855 hold off 3856 3857 end

```
3858
                  end
3859
3860
                  close(animVid);
3861
3862
                  disp('Successfully completed animation.');
3863
3864
             end
3865
        end
3866
3867 end
3868 toc
3869 disp('All operations completed.');
3870
3871 %% Functions
3872
3873 % make a spectral analysis function
3874
3875 function [P1, f] = freq_this(X,Fs, L)
3876
                           % Sampling period
_{3877} T = 1/Fs;
3878 t = (0:L-1)*L; % Time vector
3879
_{3880} Y = fft(X);
3881
_{3882} P2 = abs(Y/L);
_{3883} P1 = P2(1:L/2+1);
3884 P1(2:end-1) = 2*P1(2:end-1);
3885
_{3886} f = Fs*(0:(L/2))/L;
3887
3888 end
3889
```
```
3890 % Implementation of the MHPA as displayed in Rautenberg2018 this is an
3891 % itteration on Axford1968
3892
3893
3894 %%%%% [VN, VE, VD ...
        (m/s)][Angle(Degrees)][U(m/s)]%%%%%%%%%%%%%%%%[U,alpha,beta][VN,VE,VD][phi,theta,psi]
3895 function [windVELOCITY, windDIRECTION, windSPEED] = ...
       MHPA_Rautenberg2018 (aDATA, gDATA, bDATA)
3896
3897 % tTEMP is singular does not need to be defined
3898
3899 % airdata
3900 u_a=aDATA(:,1);
3901 alpha=aDATA(:,2);
3902 beta= aDATA(:,3);
3903
3904 % ground data
3905 VN=gDATA(:,1);
3906 VE=gDATA(:,2);
3907 VD=qDATA(:,3);
3908
3909 %body data
3910 phi=bDATA(:,1);
3911 theta=bDATA(:,2);
3912 psi=bDATA(:,3);
3913
3914 R=287; %J kg^-1 K^-1
3915 cp=1004; % J kg^-1 K^-1
3916
3917 k=R/cp; % poisson number
3918
3919 for i=1:length(bDATA(:,1))
```

```
3920
                       RM(:,i) = ...
                                 [cosd(psi(i))*cosd(theta(i))+tand(alpha(i))*(sind(phi(i))*sind(psi(i))+ ...
                                 . . .
                       cosd (phi (i)) *cosd (psi (i)) *sind (theta (i))) +tand (beta (i)) * (cosd (psi (i)) *sind (phi (i)) *sind (phi (i)) * (cosd (psi (i)) * (cosd (psi (i))) * (cosd (ps
3921
                       cosd(phi(i))*sind(psi(i)));
3922
                       cosd (theta (i)) *sind (psi (i)) +tand (alpha (i)) * (cosd (phi (i)) *sind (psi (i)) *sind (theta (i))
3923
                                 . . .
                       tand(beta(i))*(cosd(phi(i))*cosd(psi(i))+sind(phi(i))*sind(psi(i))*sind(theta(i)));
3924
                       -sind(theta(i))+cosd(phi(i))*cosd(theta(i))*tand(alpha(i))+cosd(theta(i))*sind(phi(:
3925
3926
                       D(i,1) = sqrt(1+tand(beta(i))^2+tand(alpha(i))^2);
3927
3928
                       bodyU(i,1) = abs(u_a(i))/D(i);
3929
3930
           end
3931
           D = sqrt(1+tan(beta)^2+tan(alpha)^2);
3932
3933
           groundMHP = bodyU.*RM';
3934
3935
           windGROUND = abs(gDATA) - abs(groundMHP);
3936
3937
3938 windVELOCITY = windGROUND;
         for i=1:length(windGROUND)
3939
                       windDIRECTION(i) = atan2d(windVELOCITY(i,1),windVELOCITY(i,2))+90;
3940
3941 end
3942 %windDIRECTION=rad2deg(windDIRECTION);
windSPEED=sqrt (windVELOCITY(:,1).^2+windVELOCITY(:,2).^2+windVELOCITY(:,3).^2);
3944 end
3945
3946 % Implementation of the MHPA as derived in Lenschow1989 (incorporates
3947 % angular rates)
3948
```

```
3949
3950 %%%%%%[VN, VE, VD ...
        (m/s)][Angle(Degrees)][U(m/s)]%%%%%%%%%%%%%%%%[U,alpha,beta][VN,VE,VD][phi,theta,psi]
3951 function [windVELOCITY, windDIRECTION, windSPEED] = MHPA_wRATES(aDATA, ...
       gDATA, bDATA, rDATA, L)
3952
3953 % tTEMP is singular does not need to be defined
3954
3955 % airdata
3956 u_a=aDATA(:,1);
3957 alpha=aDATA(:,2);
3958 beta= aDATA(:,3);
3959
3960 % ground data
3961 VN=gDATA(:,1);
3962 VE=gDATA(:,2);
3963 VD=gDATA(:,3);
3964
3965 %body data
3966 phi=bDATA(:,1);
3967 theta=bDATA(:,2);
3968 psi=bDATA(:,3);
3969
3970 %rate data
3971
3972 p=rDATA(:,1);
3973 q=rDATA(:,2);
3974 r=rDATA(:,3);
3975
3976 R=287; %J kg^-1 K^-1
3977 cp=1004; % J kg^-1 K^-1
3978
```

```
3979 k=R/cp; % poisson number
3980
   for i=1:length(bDATA(:,1))
3981
        RM_gw(:,i) = \ldots
3982
           [cosd(psi(i))*cosd(theta(i))+tand(alpha(i))*(sind(phi(i))*sind(psi(i))+ ...
           . . .
        3983
        cosd(phi(i)) * sind(psi(i)));
3984
        cosd (theta (i)) *sind (psi (i)) +tand (alpha (i)) * (cosd (phi (i)) *sind (psi (i)) *sind (theta (i))
3985
           . . .
        tand (beta (i)) * (cosd (phi (i)) * cosd (psi (i)) + sind (phi (i)) * sind (psi (i)) * sind (theta (i)));
3986
        -sind (theta(i))+cosd(phi(i))*cosd(theta(i))*tand(alpha(i))+cosd(theta(i))*sind(phi(:
3987
3988
3989
   응
          RM_gb(:, i) = [cosd(psi(i)) * cosd(theta(i)) ...
       -(sind(phi(i))*sind(psi(i))+cosd(psi(i))*sind(phi(i))*sind(theta(i))) ...
       sind(psi(i))*sind(theta(i))+cosd(psi(i))*sind(theta(i))*cosd(phi(i));
3990 %
              sind(psi(i))*cosd(theta(i)) ...
       cosd(psi(i))*cosd(phi(i))*sind(psi(i))*sind(theta(i))*sind(phi(i)) ...
       sind(psi(i))*sind(theta(i))*cosd(phi(i))-cosd(psi(i))*sind(phi(i));
              -sind(theta(i)) cosd(theta(i))*sind(phi(i)) ...
3991 %
       cosd(theta(i))*cosd(phi(i))];
3992
3993 rates(i,1) ...
       =L*(deg2rad(q(i))*sind(theta(i))*cosd(psi(i))-r(i)*sind(psi(i)));
3994 rates(i,2) ...
       =L* (deg2rad(r(i))*cosd(psi(i))*cosd(theta(i))-q(i)*sind(psi(i))*sind(theta(i)));
_{3995} rates(i,3) =-L*(deg2rad(q(i))*cosd(theta(i)));
3996
        D(i,1) = sqrt(1+tand(beta(i))^2+tand(alpha(i))^2);
3997
3998
        bodyU(i,1) = abs(u_a(i))/D(i);
3999
4000 end
```

## VITA

## Kyle Hickman Candidate for the Degree of Master of Science

## Thesis: THIS BLOWS?: EVALUATION OF ADDITIVE MANUFACTURED FIVE HOLE PROBES FOR UNMANNED SYSTEM BASED WIND MEASUREMENTS

Major Field: Mechanical and Aerospace Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2021.

Completed the requirements for the Bachelor of Science in Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in 2019.

Completed the requirements for the Bachelor of Science in Mechanical Engineering at Oklahoma State University, Stillwater, Oklahoma in 2019.

Professional Memberships:

American Model Association, American Institute of Aeronautics and Astronautics, American Physical Society