The Pennsylvania State University
The Graduate School

LOW-COST WIND SENSING FOR DYNAMIC SOARING UAVS

A Thesis in
Aerospace Engineering
by
John Francis Quindlen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2012

The thesis of John Francis Quindlen was reviewed and approved* by the following:

Jacob W. Langelaan
Associate Professor of Aerospace Engineering
Thesis Advisor

Mark D. Maughmer
Professor of Aerospace Engineering

George A. Lesieutre
Professor of Aerospace Engineering
Head of the Department of Aerospace Engineering

*Signatures are on file in the Graduate School.

Abstract

Small UAVs have tremendous potential in military and civil applications, but are limited by their size in payload capacity and flight endurance. Extra batteries or fuel can be added to increase endurance, but at the expense of mission payload. Autonomous soaring techniques directly inspired by birds extract energy from the environment and offer the ability to increase UAV endurance without sacrificing payload.

Dynamic soaring exploits naturally-occurring spatial wind gradients to gain energy and has been used by albatrosses to routinely fly hundreds of miles out at sea. This technique requires precise knowledge of the wind vector and albatrosses are the only birds capable of it due to specialized sensory organs in their beak nostrils. A flush air data sensing (FADS) system located on the nose of a soaring-capable UAV operates similar to an albatross beak and measures the aircraft's orientation with respect to the wind. The approach aims to create a small, versatile, low-cost sensing system in comparison to the large, expensive, commercial-grade wind sensors currently available.

The design of the system was developed from potential flow and panel method models. Flush pressure ports drilled into the surface of a detachable nosecone feed airspeed, angle of attack, and angle of sideslip measurements to an Arduino microcontroller acting as a programmable air data computer. The entire FADS system cost less than \$130 to construct and instrument, but still measures the complete wind vector.

Neural networks are implemented to compute the wind vector. Wind tunnel tests collected training and validation data for airspeeds from 9 m/s to 27 m/s and aerodynamic angles from -15° to +15°. Five neural networks of 3, 9, 15, 30 and 45 neurons were created for each wind parameter from the training data. Validation results confirm the more complex networks outperform the simpler 3, 9, and 15 neuron models, although not by considerably more. Ultimately, a 30 neuron network model is chosen as it performs just as well as the 45 neuron model, but with a simpler structure.

Numerous flight tests conducted with a FADS equipped sailplane demonstrate the capability of the system in real world flight regimes. Pilot-induced longitudinal and lateral-directional maneuvers verified the system's accuracy in dynamic maneuvers and steady glides. The preliminary flight test results prove the neural network-based FADS system's ability to accurately compute the wind vector for use in dynamic soaring research.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my family above all for their support and encouragement over the years. My earliest memories are of my family fostering my childhood interest in airplanes whether it be my father taking me to all the Reading and Willow Grove airshows, my grandfather meticulously building me a C-47 model of his aircraft and then rebuilding it after I thought it could fly, or my parents trusting my 5 year old self with a hammer to nail blocks of wood together and call it an airplane. I guess this is just an adult continuation of tossing me that hammer.

I am grateful for my advisor, Dr. Jack Langelaan, and his support over the past years. I'm always blown away by the way he manages to come up with perfect ideas seemingly on a whim. His ability to successfully balance on the precarious line between too little and too much oversight is also beyond me. Through his assistance, I've grown to become a clearer, more precise and thorough thinker both in the lab and outside of it.

All the faculty and staff here at Penn State have been tremendously helpful in many ways. Dr. Maughmer's advice and assistance over the past 6 years have always given me better insights. Sometimes he has pointed out the most obvious things that I could not see myself. Dr. Horn's inputs and suggestions for neural network training and modeling saved me needless frustration and improved my results significantly. Rick Auhl's help in the wind tunnel testing was invaluable.

I would like to thank my labmates for their assistance during the last two years. All of their support, research advice, homework help, or sympathetic ears have enabled me to finish this thesis. I will always consider them some of my strongest friends. How they managed to put up with me is beyond my comprehension.

In the same vein, a big thanks is due to my friends for dealing with me while I've completed my research work. My friends, both here and at home, always have the presence of mind to know exactly when I've reached the critical mass of stress and need a break. Drinks on me.

Lastly, I would like to thank Dr. Langelaan and the Department of Defense National Defense Science & Engineering Graduate Fellowship Program for providing funding for my research.

Dedication

To my parents and grandparents.

Chapter 1

Introduction

This thesis presents a system for computing the wind vector for use in small, soaring-capable uninhabited aerial vehicles (UAVs). The purpose of this research is to develop a sensing technique to address guidance and control issues associated with dynamic soaring. Dynamic soaring exploits naturally-occurring wind shear gradients to extract energy from the environment. This method can be used to gain or maintain energy to extend of the endurance of small UAVs, especially for unpowered gliders. Small UAVs are defined as aircraft with wingspans less than 5 meters and a total mass below 10 kilograms. Dynamic soaring offers enormous potential, but requires precise knowledge of the wind for successful application.

There are many different techniques for sensing the wind vector from an aircraft. Multi-holed probe configurations extending into the freestream airflow are frequently used, as are wind vanes for measuring local wind angles on the aircraft [1]. Probes and vanes are the most common, but other techniques such as null pressure swiveling heads and trailing cones can be used to measure wind parameters. Figure 1.1 illustrates some of these different sensing configurations. These methods are mostly intended for use on manned aircraft or comparably sized UAVs and are therefore large, expensive, and susceptible to damage when compared to a small UAV. Flush sensing systems present another solution without these issues. Flush pressure ports keep the sensing system internal to the aircraft, or at most to the surface of the airframe. The pressure measurements taken from the ports can then be coupled with a computational method to calculate the wind vector.

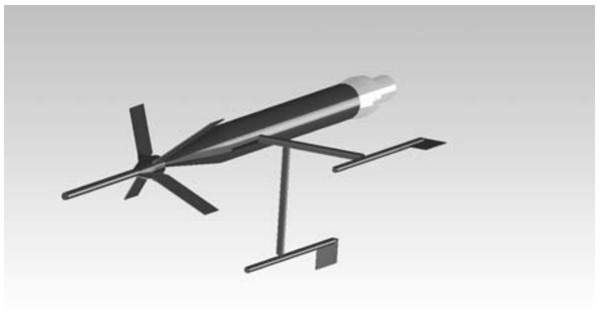
This thesis will:

- Provide the background for pressure sensing over an aircraft nosecone. Potential flow equations are used to predict pressure ranges over a nosecone.

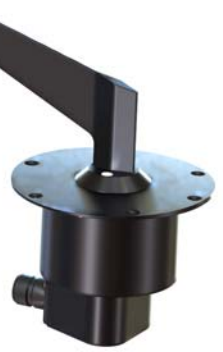
- Design and implement a nosecone and sensing system using low-cost Arduino board and sensor components.

- Present wind tunnel results taken with measurements from the nosecone system.

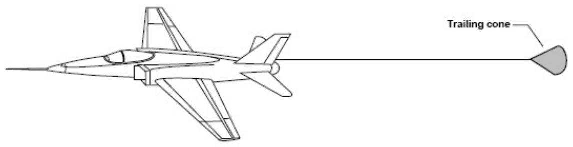
(a) 5 Hole Probe



(b) Swivel Point Head [2]



(c) Wind Vane [2]



(d) Trailing Cone [1]

Figure 1.1. Various wind sensing techniques.

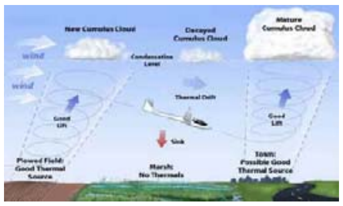
- Develop a neural network based approach to compute airspeed, angle of attack, and angle of sideslip. The wind tunnel measurements are normalized and adapted to train networks for each of the wind parameters.
- Apply the neural networks to flight tests with the aircraft and completed system. The performance of the networks with the real-world data is presented and discussed.

1.1 Motivation

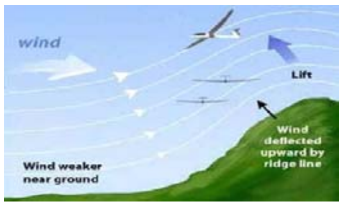
Small UAVs offer tremendous potential applications to both civil and military missions due to their small footprint. These aircraft are often hand-launchable, difficult to detect, and are generally low-cost and expendable. They have opened up new applications previously impossible with large manned aircraft, like urban search and rescue or flight through forests. Due to their small nature, these UAVs are also limited in their useful payloads. After the necessary autopilot and onboard flight control systems, there is only a tiny fraction of the space available for payload and batteries or fuel. This lack of fuel limits the endurance of the vehicles to a span of a few minutes to an hour or two at most. Fuel and battery space can sometimes be added, but at the cost of payload capacity for sensors, cameras, and other mission equipment.

Autonomous soaring techniques can be applied to small UAVs to extend their endurance without sacrificing mission payload. The various soaring methods exploit naturally occurring air and wind phenomena to extract energy from the environment. These techniques take their inspiration from nature as they have been successfully demonstrated for thousands of years by birds.

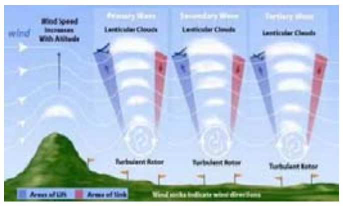
Static soaring is the most common and widely known technique which utilizes upwards moving regions of air to gain altitude. Birds and sailplane pilots routinely rely upon thermal soaring in regions of rising warm air to remain aloft for extended periods of time. A perfect example can be readily observed on warm days as hawks circle over fields, roads, or parking lots without ever flapping their wings. In addition to thermal soaring, sailplanes can use ridge soaring to take advantage of rising air on the windward side of ridges for lift. Certain areas downwind of tall mountains can be used as a source of lift for mountain wave soaring. All of these methods use known sources of rising air to gain energy, as illustrated in Figure 1.2.



(a) Thermal Soaring [3]



(b) Ridge Soaring [3]



(c) Mountain Wave Soaring [3]

Figure 1.2. Static soaring techniques.

A less common energy harvesting technique is gust soaring. Gust soaring methods use temporal gradients from random wind gusts as a source of energy [4]. A gust soaring aircraft must be able to quickly sense and respond to changes in wind speed and direction indicative of a gust without prior knowledge. Gust soaring techniques involve rapid changes in pitch rate with the gust, which make it difficult to implement on manned aircraft; however, birds appear able to employ gust soaring [5].

Dynamic soaring exploits shear layer gradients to harvest energy from wind. Though it is commonly used at low altitude shear layers near the surface of the Earth, dynamic soaring can be used anywhere a spatial wind gradient exists. During dynamic soaring, the bird or aircraft climbs into the wind, upwards through the gradient, before descending with the tailwind back through the shear layer with a negative flight path angle. The flight path can be completed as a continuous oval shape, or traverse across the wind in the manner seen in Figure 1.3.

While the potential of dynamic soaring is clearly understood, it is not as widely used as static soaring techniques for both aircraft and birds. Pilots of manned aircraft are wary to fly the tight turns required for dynamic soaring, especially at extremely low altitudes. Very few birds are seen dynamic soaring either. According to Pennyquick [7], the only birds to demonstrate dynamic soaring are albatrosses and petrels of the Procellariiform order. With their unconventional flight technique, these birds routinely fly hundreds of kilometers without flapping their wings, unlike

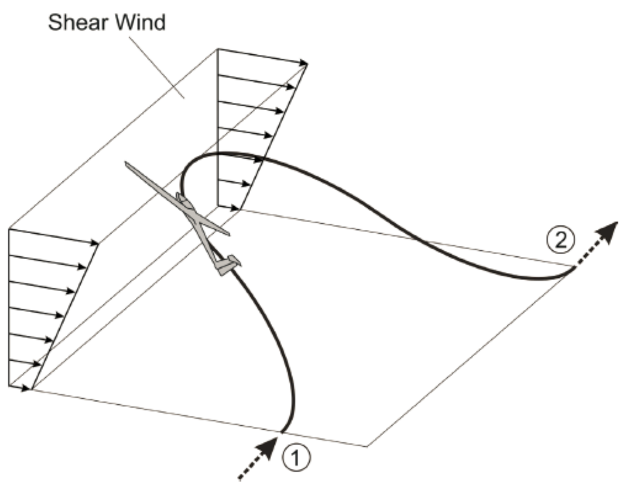


Figure 1.3. Dynamic soaring flight path. The aircraft begins dynamic soaring at point 1 when it climbs into the direction of the wind. After bursting through the shear layer, the aircraft turns around and dives back through the layer until it bottoms out at point 2. This process can then be repeated as long as the shear layer exists. [6]

most birds. The size and weight of some Procellariiform species are listed in Table 1.1. Radio-controlled aircraft pilots have also successfully demonstrated dynamic soaring in the shear layers downwind of hills. The success of the R/C pilots proves the feasibility of autonomous dynamic soaring. This type of energy harvesting method could easily be adapted as a solution to the limited endurance of small UAVs.

Table 1.1. Properties of various Procellariiform species [7].

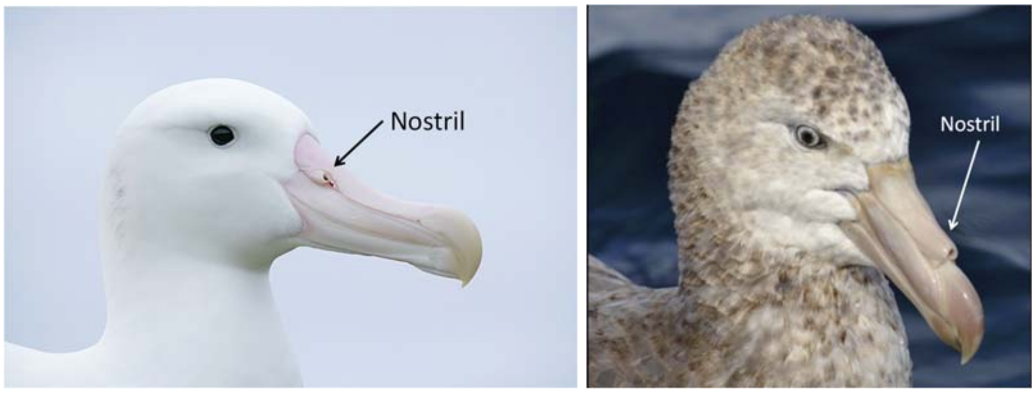
species	**mass (kg)**	**span (m)**	**chord (m)**	**area (m^2)**	**AR**
Wandering Albatross	8.46	3.01	0.197	0.592	15.3
Black-browed Albatross	3.08	2.19	0.162	0.354	13.5
Grey-headed Albatross	3.68	2.18	0.153	0.334	14.2
White-chinned Petrel	1.08	1.41	0.118	0.167	11.9

1.2 Problem Description

Dynamic soaring has obvious potential that makes it desirable for implementation on small UAVs. For autonomous dynamic soaring to successfully work; however, the autopilot requires precise knowledge of the aircraft's orientation in relation to the wind. The aircraft must constantly change lift coefficient, airspeed, and bank angle to perform the necessary maneuvering. This also requires noticeable amounts of crab angle to keep the aircraft aligned with the wind vector, a difficult task to perform without knowledge of the wind direction and speed [8].

While many birds employ thermal soaring, very few species are observed dynamic soaring.

In fact, out of all the species of birds, only certain species of albatrosses and petrels within the Procellariiform order are capable of dynamic soaring [9]. Pennyquick [7] proposes that this is due to a particular adaption that only the Procellariiform order possesses: tubular nostrils. These special nostrils allow them to measure their exact airspeed from dynamic pressure. The nostrils on the front of the beak act as pitot tubes for total pressure while sensory glands within their mouths act as static pressure sensors. The Diomedeidae family of larger birds have two nostrils on either side their beaks, which Pennyquick concludes are also capable of sensing yaw angles. These natural adaptations are what enable the Procellariiform species to successfully exploit the wind gradients. In order for a small aircraft to harvest energy in this manner, it must possess the same sensing capabilities of the petrels and albatrosses.



(a) Wandering Albatross [10] (b) Giant Petrel [11]

Figure 1.4. Procellariiform bird beaks. The Wandering Albatross displays two nostrils on either side of the beak while the Giant Petrel has a single large nostril on top of the beak.

1.3 System Overview

The wind sensing system developed in this thesis is meant to be implemented along with a fully capable autopilot for dynamic soaring research. The sensing system can also be used for standalone measurements in a piloted R/C aircraft. An aircraft can easily fly without knowledge of the complete wind vector, but it will not be capable of autonomous dynamic soaring. The wind vector information provided by the system is intended to become the enabling factor allowing the flight computer to perform the necessary control decisions.

The aircraft being used for the soaring research is an Omega II 2-meter sailplane with a mass of 1.2 kilograms. The aircraft was originally built as a motor glider equipped with an electric motor, although this has been removed. The sailplane is equipped with a V-tail empennage, with ruddervators for both pitch and yaw control. Roll control is accomplished with outboard ailerons while inboard flaps are used to slow the airplane for landing. The unmodified aircraft is shown in Figure 1.5. Additional aircraft parameters are located in Table 3.1.

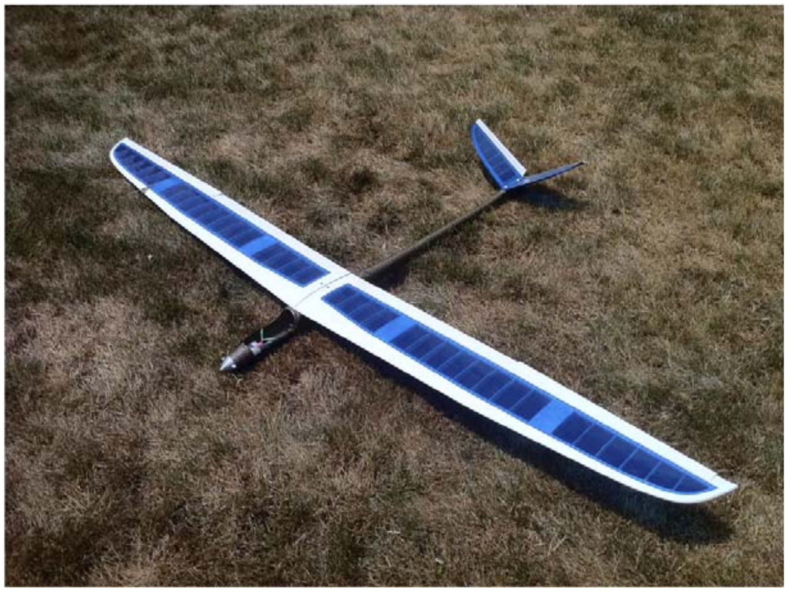


Figure 1.5. Omega II sailplane.

Right in line with the beaks of albatrosses, pressure measurements around the aircraft are used to determine wind axes and speed. Flush pressure ports located on the nose and fuselage detect changes in wind speed and direction with differential readings between ports. Flush air data systems (FADS) minimize the external footprint of the sensing system and protect the components by keeping everything internal to the airframe. This is especially important for the Omega II UAV because it lands on its belly and rough landings are almost inevitable.

The differential pressure readings from the flush ports are converted to the wind vector using computational methods. For this research, neural networks are used to calculate airspeed (V_a), angle of attack (α), and angle of sideslip (β). Other methods such as lookup tables or curve fitting functions could be used, but neural networks are chosen for their ease of use, especially for complex systems. The output from the neural networks can then be directly used by the autopilot for dynamic soaring control.

1.4 Related Work

There has been a multitude of research endeavors into autonomous soaring, air data sensing, and neural network models. Each aspect has seen extensive research within various subfields and applications. Autonomous soaring has an active research focus and generally breaks down into three categories corresponding to static, gust, and dynamic soaring.

1.4.1 Static Soaring

Thermal, ridge, and mountain wave soaring techniques each have an extensive collection of literature analyzing feasibility and optimal solutions. Chakrabarty and Langelaan combine these three soaring methods together and apply them to path planning for energy harvesting with gliders [12]. The path planning methods utilize regional wind fields measured above central Pennsylvania to plan autonomous soaring flights incorporating ridge and wave flight.

Flight test results of autonomous static soaring have proven the capabilities of thermals for energy harvesting. Allen and Lin [13] investigated thermal identification and exploitation with the Cloud Swift small UAV. They recorded multiple flights over 60 minutes in length using thermal soaring on the UAV. Andersson et al. [14, 15] have also flight tested thermal centering controllers based upon energy estimation. The SBXC glider, the same model as the Cloud Swift, made numerous flight tests with a switch mode enabled for thermal recognition. Lastly, Edwards and Silverberg [16, 17] used an SBXC-based ALOFT UAV to identify thermals and implement MacCready's Speed-to-Fly theory on a small UAV. The ALOFT UAV was able to stay airborne for 3.5 hours in autonomous soaring mode. The aircraft also placed 3rd in the Montague Cross Country Challenge against expert R/C glider pilots. These flight test demonstrations have all proven the feasibility and worth of static soaring for use with small UAVs.

1.4.2 Gust Soaring

Autonomous gust soaring has also been successfully demonstrated with small UAVs. Depenbusch and Langelaan [5] examined gust soaring with a receding horizon controller to compute pitch rates necessary for energy harvesting. Their work simulated gust and thermal soaring using an Omega II 1.8-meter sailplane, similar to the slightly larger Omega II used herein, and observed distinct energy savings possible with such an aircraft. Patel et al. [18] flight tested a gust soaring capable small UAV and recorded consistent results for both simulation and test data. Their simple, low cost UAV demonstrated energy savings of 36% in simulation and 46% in autonomous soaring flight tests. These results solidify the potential performance improvements possible with gust soaring techniques.

1.4.3 Dynamic Soaring

Just like all soaring techniques, dynamic soaring got its inspiration from birds. The phenomena was first scientifically described by Lord Rayleigh in 1883 [19]. Ocean-going petrels and albatrosses have been widely observed dynamic soaring across the ocean in the separated regions behind the crests of large waves as described by Pennyquick [7]. As mentioned before, only the Procellariiform order is observed dynamic soaring. Pennyquick proposes that these are the only types of birds capable due to their unique nostrils that act as sensitive pitot tubes. Other birds possess nearly identical wing and size features, but do not have tubular nostrils and are not seen dynamic soaring. The soaring-capable petrels and albatrosses use this technique to traverse hundreds of kilometers out at sea, shown by Sachs et al. [20]. Sachs' GPS loggers attached to albatrosses have demonstrated that the birds are even capable of penetrating into the wind using dynamic soaring alone. These long flights show the potential for extended cross country flights with dynamic soaring methods.

The success of the albatrosses inspired research investigations into dynamic soaring techniques for small UAVs. The similar size and weight between small UAVs and Procellariiform birds make dynamic soaring an attractive solution to extending aircraft endurance. Sukumar and Selig [8] analyzed dynamic soaring over sample fields in the mid-western United States using a 3-meter sailplane. They compared the UAV flight path results to albatross paths and concluded that energy harvesting would be feasible with a small UAV. Sachs and da Costa [6] explored the use of jet streams at high altitudes for dynamic soaring. High altitude shear layers offer much higher wind speeds and pronounced shear layers that might be exploited. Meanwhile, both Zhao and Qi [21, 22] and Deittert et al. [23, 24] took a different approach from bird soaring. They formulated optimal control patterns featuring engine-assisted as well as unpowered soaring strategies. In both cases, the level of success was strongly dependent upon wind gradient and speed. Lawrance and Sukkarieh [25, 26] combined static and dynamic soaring approaches to explore a hybrid soaring technique. All of this research focused on dynamic soaring methods with small UAVs and predicted potential energy savings for long duration flight.

While all these papers demonstrated the feasibility and optimal control solutions to dynamic soaring, there has not been the same level of flight test validation in small UAVs that there has been with autonomous thermal and gust soaring. This does not mean aircraft have not successfully employed dynamic soaring before. Radio-control pilots routinely reach high speeds dynamically soaring on the backsides of hills. Gordon [27] implemented dynamic soaring in manned L-23 gliders above the desert. His pilots used 5-hole probes for precision feedback and control, although safety and airframe limitations reduced the effectiveness of the techniques. Manned aircraft are severely restricted by the necessary precautions, but small UAVs are ideally suited for dynamic soaring. Deittert et al. [24] acknowledged that dynamic soaring implementation requires advances in flight control and sensing in order to be successful. Such sensors could also prove useful to other flight aspects, such as gust soaring [4]. These sensors would need to accurately measure wind speeds to estimate the wind gradient and determine the optimal path. The previous research in dynamic soaring has proven its feasibility but issues in sensing and control must be addressed before extensive implementation can occur.

1.4.4 Flush Air Data Sensing Systems

A variety of methods can be used for wind vector sensing on an aircraft or rotorcraft. Figure 1.1 illustrates some of the common approaches. Flush air data sensing (FADS) systems are a particular method based upon pressure measurements at locations on the surface of the aircraft. This technique has an extensive background of research available. NASA research reports [28,29] designed flush systems for air data measurements onboard the Space Shuttle and X-33 space vehicles during re-entry where traditional techniques were not feasible. Later flight tests validated the FADS technique with manned aircraft like the F-14 [30] and KC-135 [31] military aircraft. These results demonstrate the feasibility of a FADS approach for accurate measurement of the wind axes, even in fast, maneuvering flight.

Later research implemented neural networks alongside FADS systems with notable success. The networks were employed to compute aircraft parameters from the FADS measurements. Crowther and Lamont [32] explored different pressure port configurations over a generic stealth fighter fuselage. Neural networks were used to computer V, α, and β given wind-tunnel training data. Calia et al. [33] calculated static pressure and Mach number for a M-346 jet trainer using a FADS system and neural networks. The networks were trained and validated using wind tunnel tests on the front of the aircraft and demonstrated success when applied in flight tests of the system. Rohloff et al. [34] employed a FADS on the nosecone of an F-18 to compute wind parameters. Neural networks were specifically chosen in this case because they can handle large sets of test data and did not require explicit knowledge of the airflow model. The success of these flight tests show the capability of FADS sensing in a real-world environment.

While this research in neural network based FADS systems proves the accuracy of the approach, they have all been implemented in large, fast air vehicles with expensive instrumentation. Few papers look at adapting this method for use in smaller aircraft. Meanwhile, the commercial-grade sensors available for small UAVs tend to protrude outward from the aircraft and cost more than the aircraft itself [2]. The FADS approach from the larger aircraft can be scaled down in the hope of creating small, accurate, low-cost wind sensors. A similar application has been used before by Samy et al. [35] for FADS on the leading edge of a micro air vehicle wing. Their neural network FADS successfully worked for angle of attack sensing at low airspeeds. A modified nosecone incorporating a FADS would employ the same principles and prove indispensable for small autonomous gliders.

1.5 Contributions

The primary contributions are described below:

- Design and implementation of a low-cost flush air data sensing nosecone for measuring differential pressure around a small UAV.
- Creation of neural networks for computing airspeed, angle of attack, and angle of sideslip. The networks are trained using wind-tunnel test data from the FADS nosecone.
- Performance verification of the wind sensing system through sample flight tests of the aircraft. Specific maneuvers are performed to test the expected dynamic response of the system.

1.6 Reader's Guide

The remainder of the thesis is organized as follows:

- **Chapter 2** describes the theoretical background of the nosecone. Potential flow models are used to demonstrate the feasibility of the system.
- **Chapter 3** presents the hardware implementation of the nosecone. The fabrication and hardware selection processes are shown. Wind-tunnel tests collect pressure data for the air data computer.
- **Chapter 4** defines the neural network models employed for the sensing system. The wind-tunnel surface pressures are used to train the neural networks to output the relevant air data.
- **Chapter 5** applies the neural networks trained for the air data computer to various pilot controlled glide tests for validation.
- **Chapter 6** reviews and concludes the work. Suggestions and improvements for future research are included.

Chapter 2

Theoretical Background

The chapter provides the theoretical background used to analyze the pressure measurements over a nosecone and prove the feasibility of the flush air data sensing method. Section 2.1 describes the components required to define a complete wind vector and wind sensing system. The location of the FADS system is discussed in the section as well. In Section 2.2, the nosecone location of the FADS system is approximated as a hemisphere and a spherical potential flow model is applied. The surface pressures over the nosecone are calculated in this section and the feasibility of such a system is argued. The results from Sections 2.1-2.3 directly influence the work presented in Chapter 3.

2.1 Problem Statement

The objective of the wind sensing system is to determine the wind vector on a glider for use by a dynamic soaring controller. Dynamic soaring techniques for extracting energy from naturally-occurring wind gradients offer huge potential for extending the endurance of small UAVs. In order to be successful, these techniques require precise knowledge of the aircraft's speed and orientation in the wind field. A practical wind sensor must provide airspeed and angles of attack and sideslip to the soaring controller with adequate accuracy and precision to meet those requirements. The dynamic soaring controller will then use the wind measurements to estimate the wind gradient of the shear layer.

Many different forms of wind sensing techniques exist, but the nature of small UAVs limits the size and weight than can be appropriated to an air data system. For this reason, a flush air data system presents the best method for implementation on a small, gliding UAV. Flush air data sensing uses pressure measurements taken on the surface of the aircraft to compute aircraft parameters, namely, the wind axes in this situation. FADS pressure ports located along the aircraft measure changes in airflow as the aircraft maneuvers and the wind field varies. These measurements correspond to the aircraft's orientation and are passed to the air data computer to obtain V_a, α, and β.

The wind axes parameters are defined in Figure 2.1 along with the body axes. The wind velocity vector can be broken into the body axes components U', V', and W' in Equations 2.1-2.4:

$$U' = V_a \cos(\alpha)\cos(\beta) \tag{2.1}$$

$$V' = V_a \sin(\beta) \tag{2.2}$$

$$W' = V_a \sin(\alpha)\cos(\beta) \tag{2.3}$$

$$V_a = \sqrt{(U')^2 + (V')^2 + (W')^2} \tag{2.4}$$

The airspeed, V_a, is the magnitude of the velocity vector while α defines the angle in the XZ plane projection and β defines the angle in the XY plane projection.

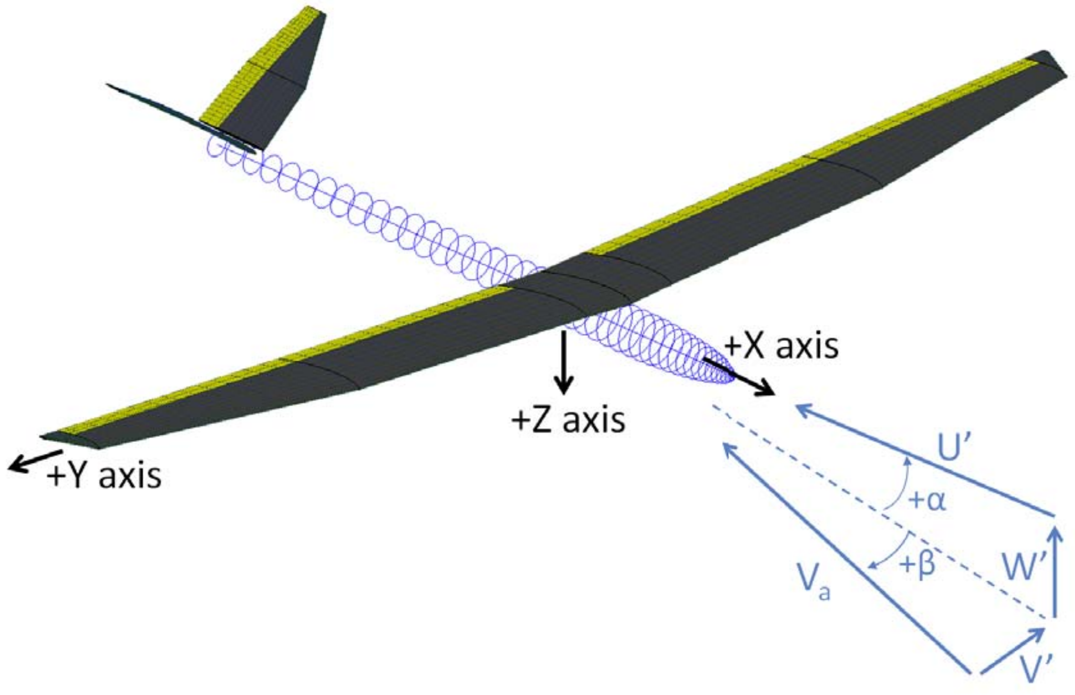


Figure 2.1. Body and wind axes definitions.

The flush sensing system must be designed in a way to minimize weight and necessary equipment within the small airframe while still providing the accuracy required by the soaring controller. In order to reduce the weight, only a small number of flush pressure ports can be used, so these ports must be arranged to produce the largest pressure differentials between them. Large pressure differences will only improve the accuracy of the pressure sensors. If the difference between the measurements is too small, only specialized, expensive transducers will be capable of detecting changes with any sort of accuracy.

The ideal location for the FADS system is the aircraft's nosecone. This location is the easiest area to implement the air data system as it takes advantage of existing mounting points on the airframe. Previously, the aircraft's spinner had been located at this location before it was removed, leaving four open screw holes in the firewall. These holes can definitely be used for mounting the nosecone. The nosecone location also keeps the pressure ports out of other regions of the aircraft that are more susceptible to separated or disrupted airflow. Generally, the nosecone ports will always receive laminar, undisrupted flow at the front of the aircraft in comparison to wing or fuselage mounted ports on the far side of the fuselage during a crosswind. If the crosswind is too strong, the fuselage or wing ports will be noticeably affected while the nosecone ports will not. These advantages make the nosecone a desirable location for the FADS system.

A main advantage of the the nosecone location is the fact that the nosecone can be accurately approximated as a hemisphere. This allows a spherical potential flow model to estimate the surface pressures around the nosecone. This approach has been used before to estimate pressure when Samy et al. [35] used spherical potential flow models to analyze different FADS pressure port arrangements for fuselage and wing leading edge air data sensing. The potential flow equations will give a surface pressure difference between the ports and determine the feasibility of using the nosecone for air data sensing before any implementation.

2.2 Surface Pressures in Potential Flow

The nosecone is approximated as a hemisphere and analyzed with a spherical potential flow model. Since differential pressures are used to determine angle of attack and angle of sideslip, determining port locations that maximize pressure differential for a give angle will allow the design of a system with maximum sensitivity. The pressure at any point on the surface of the nosecone is modeled as:

$$P_s = P_\infty + q_\infty \left(1 - \frac{9}{4}\sin^2\theta\right) \tag{2.5}$$

where P$_s$ is the surface pressure, P$_\infty$ is the freestream static pressure, q$_\infty$ is the freestream dynamic pressure, and θ_i is the angle from the stagnation point on the front of the sphere to the location i on the sphere. A 2-dimensional layout is seen in Figure 2.2. The nondimensionalized static pressure coefficients over the surface of the hemisphere are given in Figure 2.3.

Next, the pressure differentials between two pressure ports are identified. The surface pressure at a single port is already listed in Equation 2.5 and this can now be expanded to include two pressure ports. The differential pressure readings between these ports at various flow angles is defined in Equation 2.8.

$$P_{s1} = P_\infty + q_\infty \left(1 - \frac{9}{4}\sin^2(\theta + \alpha)\right) \tag{2.6}$$

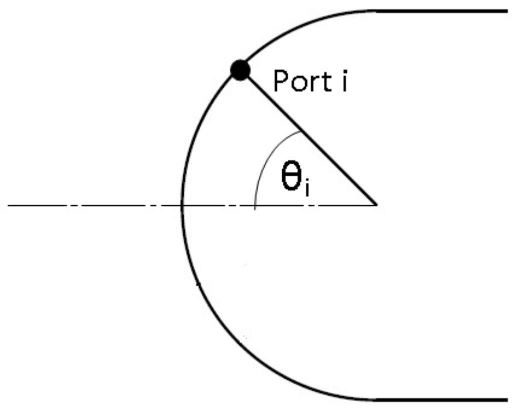


Figure 2.2. 2-dimensional view of the hemispherical nosecone with a pressure port at location i.

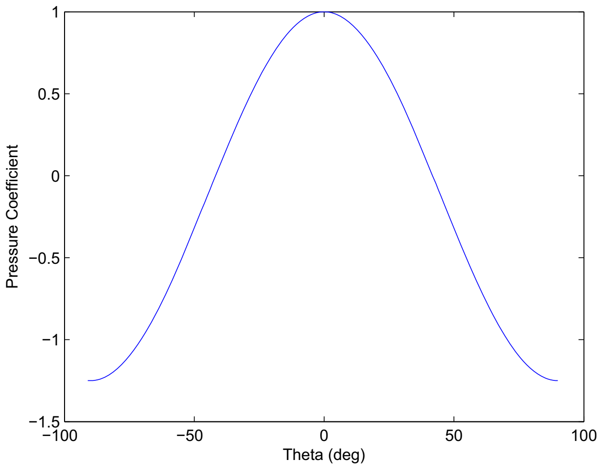


Figure 2.3. Pressure coefficients over the hemispherical nosecone model.

$$P_{s2} = P_\infty + q_\infty \left(1 - \frac{9}{4}\sin^2(\theta - \alpha)\right) \tag{2.7}$$

$$\Delta P = P_{s1} - P_{s2} = -\frac{9}{4} q_\infty \sin(2\theta)\sin(2\alpha) \tag{2.8}$$

Pressure ports 1 and 2 are set to mirror locations at angle θ from the centerline of the nosecone. The wind flows over the nosecone at incidence angle α in the same plane as the two ports. In this case, angle of attack α is shown, but the same equations apply to β just the same. Figure 2.4 depicts a 2-dimensional view of the hemispherical nosecone with multiple ports.

The port location θ angle that results in the largest pressure differentials can be found through the derivative of Equation 2.8, seen in Equation 2.9.

$$\frac{\partial(\Delta P)}{\partial\theta} = -\frac{9}{2} q_\infty \cos(2\theta)\sin(2\alpha) \tag{2.9}$$

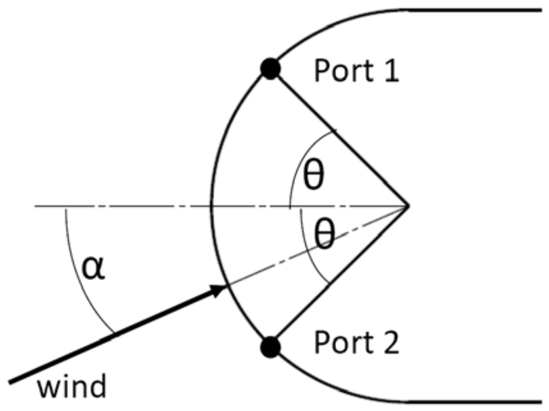


Figure 2.4. Hemispherical nosecone with airflow entering at angle α.

Assuming nonzero q_∞ and α, the θ portion of the equation, $\cos(2\theta)$, reaches zero at $\theta = 45°$. This corresponds to the maximum pressure difference between ports. Therefore, the potential flow model suggests the pressure ports should both be set to $\theta = +/\text{-}\ 45°$, or 90° apart from each other, to give the largest pressure differential and subsequent sensitivity.

Another useful insight from Equation 2.8 demonstrates that nondimensionalized readings for the pressure differential between ports are independent of dynamic pressure. When the differential pressure readings are divided by dynamic pressure, the resulting relation is independent of airspeed, assuming angle of attack doesn't change:

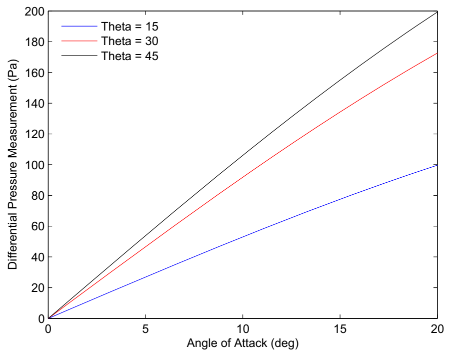
$$\frac{\Delta P}{q_\infty} = -\frac{9}{4}\sin(2\theta)\sin(2\alpha) \tag{2.10}$$

While it doesn't appear too important by itself, this relation in Equation 2.10 can be used to compare real pressure measurements to the potential flow model predictions. Even without knowing the actual values, the same general behavior by wind tunnel or flight test measurements would validate the applicability of the potential flow model for surface pressure measurements.

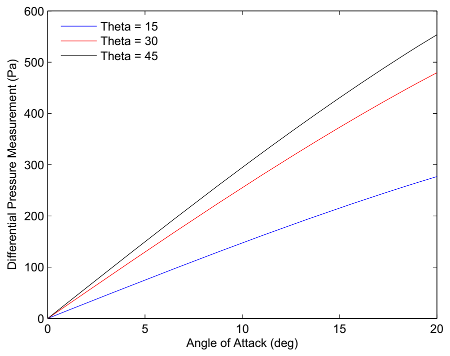
2.2.1 Pressure Differentials

Differential pressure values are plotted in Figure 2.5 for three separate port locations of θ =15°, 30°, and 45°. The airspeeds used for the surface pressure measurements are 15 m/s and 25 m/s, which fall well within the 10 - 30 m/s nominal operating range of the Omega II sailplane. Angular deflections from $0° \leq \alpha \leq 20°$ cover the maximum angles in a single direction the aircraft is likely to encounter.

In practice, the angle of attack will probably never exceed $\alpha = +/\text{-}\ 10°$, with an even smaller range for angle of sideslip. At these lower angles, the port location of $\theta = 45°$ produces significantly lower pressure differentials than seen in the upwards limits of Figure 2.5. At angle $\alpha =$ 10°, the pressure difference drops to 106 Pa at 15 m/s and 295 Pa at 25 m/s. These pressure differentials are definitely within the capabilities of available sensors. At $\alpha = 1°$, the difference



(a) Differential pressure at 15 m/s.



(b) Differential pressure at 25 m/s.

Figure 2.5. Differential pressure measurements for various port locations.

falls to 11 Pa at 15 m/s and 30 Pa at 25 m/s. The low differentials at this angle are more difficult to accurately measure with pressure sensors than the higher angles. Angles near 0° will push the limits of the sensors' capabilities, but overall, the measurements suggest air data sensing is possible using flush pressure ports.

2.3 Summary

The air data system must compute V_a, α, and β to determine the wind vector in relation to the aircraft. These wind components will then be sent to the dynamic soaring controller to estimate the wind shear gradient and aerodynamic parameters. Flush air data ports offer a desirable method to measure the wind parameters, but only if the pressure differentials between ports are large enough to be accurately measured by pressure transducers. The nosecone of the aircraft is approximated as a hemisphere and a spherical potential flow model is then used to calculate the surface pressure measurements. This enables an accurate prediction of the pressure differentials between various port locations. The results of the surface pressure differences confirm the nosecone and a FADS system are a feasible method for measuring the wind vector.

Chapter 3

Implementation

This chapter details the physical implementation of the flush air data sensing system and its components. The results from the potential flow analysis in Chapter 2 form a basis for the design and configuration of the pressure ports. Section 3.1 discusses the Omega II airframe and the requirements it places on the FADS. An XFLR model of the aircraft is developed to place a static pressure port to obtain dynamic pressure readings. In Section 3.2, the hardware used for the system is described. The fabrication process of the nosecone is detailed along with the selection of the pressure transducers and air data computer. The nosecone developed in Sections 3.1 and 3.2 can then be tested to collect pressure data.

Sections 3.3-3.5 cover the wind tunnel testing process and the pressure measurements collected during the tests. The wind tunnel testing procedures are described in Section 3.3. The resulting pressure measurements are compared against the potential flow model in Section 3.4. In Section 3.5, the resolution of the onboard analog-to-digital converter (ADC) is verified against an external ADC to confirm the sensitivity of the pressure measurements. These wind tunnel measurements from Section 3.3 can be used to train neural networks in Chapter 4.

3.1 Omega II Airframe

The Omega II aircraft used for the dynamic soaring research has a mass of 1.2 kg with a 2-meter wingspan. The main fuselage is made of carbon fiber with some plywood reinforcements in critical locations inside the fuselage. Originally, the aircraft was equipped with an electric motor for powered self-launches, although this has been disconnected and removed. The aircraft is also equipped with a V-tail empennage rather than a conventional tail. The specific Omega II aircraft is pictured in Figure 1.5 and the relevant aircraft specifications are listed in Table 3.1.

The Omega II airframe includes a number of features that ease the implementation of a FADS nosecone. Most notably, the aircraft's front fuselage section is readily adaptable. Before it was disconnected, the electric motor mounted to a plywood firewall at the front of the fuselage. This

Table 3.1. Omega II aircraft properties.

variable	value (units)	description
m	1.2 (kg)	mass
b	1.989 (m)	wing span
c	0.1532 (m)	mean aerodynamic chord
S	0.3046 (m^2)	wing area
x_{cg}	0.0572 (m)	c.g. location behind wing leading edge

left four 2.5 mm through-holes for screws and a 10 mm center hole. These holes can easily be adopted to securely mount the nosecone and run pressure tubes back into the fuselage. The loss of the heavy electric motor also freed up weight in the nose for the FADS system. The weight distribution already assumed the motor's significant mass in the front, so the location of the FADS system actually helps alleviate a weight and balance problem rather than add to it. The spinner for the propeller can also be used as a template for the nosecone shape. The $3\frac{1}{4}$ cm long, 4 cm diameter cone is narrower and shorter than desired, but serves as a good starting point for the nosecone shape. These aspects from the electric motor system will prove useful in the design of the prototype nosecone configuration for the FADS system.

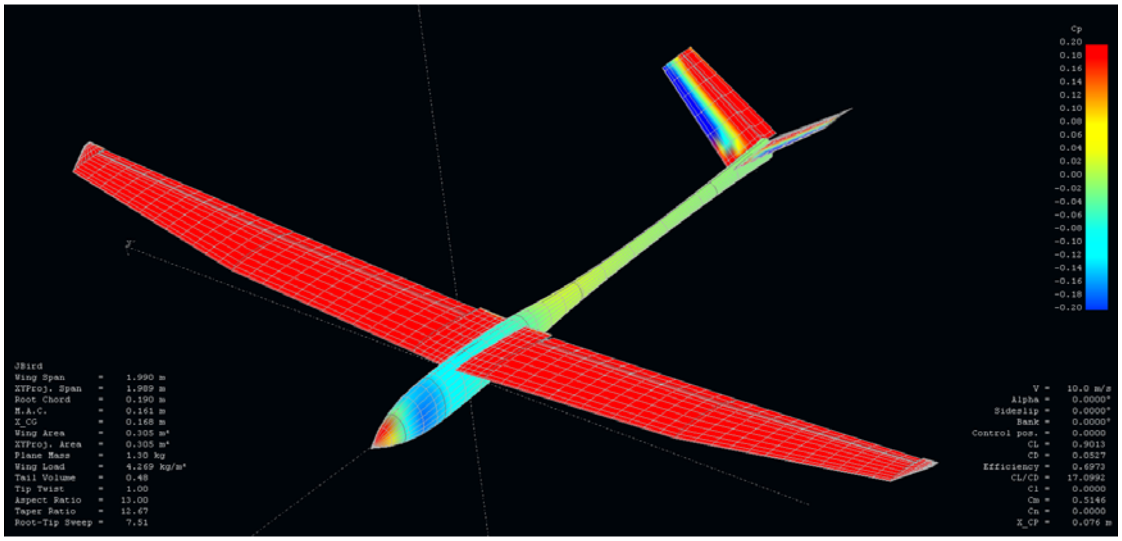
The only real downside associated with a FADS system is the propeller can't be employed on the front of the aircraft since the motor and spinner are replaced. This restriction will not scuttle the effort because the glider is capable of hi-start launches, which use elastic tubing staked to the ground to slingshot the aircraft airborne. Hi-start launches enable the unpowered Omega II to match the altitudes the self-launching version can reach, although it does eliminate the possibility of semi-powered dynamic soaring. While the ability to power launch is lost, a FADS equipped Omega II is fully capable of efficient dynamic soaring.

3.1.1 Static Pressure Measurement

Pressure ports distributed over the nosecone are a feasible method for sensing pressure differentials at various flow angles, but they can't be used to measure static pressure. A pressure port on the front of the nosecone could measure the total pressure, but a static pressure measurement is necessary to obtain dynamic pressure and subsequently compute airspeed. Static pressure readings also give the air density when combined with static temperature readings. Since pressure ports located on the nose are not capable of static pressure measurements, static ports will have to be set at some other location along the aircraft.

A new model of the entire Omega II aircraft is created using XFLR5 aircraft analysis software. XFLR5 employs panel method models to incorporate the contributions of the fuselage. Although this program is not intended for complex analysis of the fuselage like a Computational Fluid Dynamics model, it does give a clear prediction of the best regions for the placement of pressure ports. The XFLR5 model is pictured in Figure 3.1.

The best static port location appears to be slightly in front of and below the main wing. This area has a static pressure coefficient of C_p = -0.08, which is the closest to C_p = 0.0 in the entire



```
JBird
Wing Span       =    1.990 m
XYProj. Span    =    1.989 m
Root Chord      =    0.190 m
M.A.C.          =    0.161 m
X_CG            =    0.168 m
Wing Area       =    0.305 m²
XYProj. Area    =    0.305 m²
Plane Mass      =    1.30 kg
Wing Load       =    4.269 kg/m²
Tail Volume     =    0.48
Tip Twist       =    1.00
Aspect Ratio    =   13.00
Taper Ratio     =   12.67
Root-Tip Sweep  =    7.51
```

```
             V =   10.0 m/s
         Alpha =    0.0000°
      Sideslip =    0.0000°
          Bank =    0.0000°
  Control pos. =    0.0000
            CL =    0.9013
            CD =    0.0527
    Efficiency =    0.6973
         CL/CD =   17.0992
            Cl =    0.0000
            Cm =    0.5146
            Cn =    0.0000
          X_CP =    0.076 m
```

Figure 3.1. XFLR5 model of the Omega II at an airspeed of V_a = 10 m/s and α,β = 0°. Note: the visible static pressure coefficient values range from C$_p$ = +0.20 (red) to C$_p$ = -0.20 (blue).

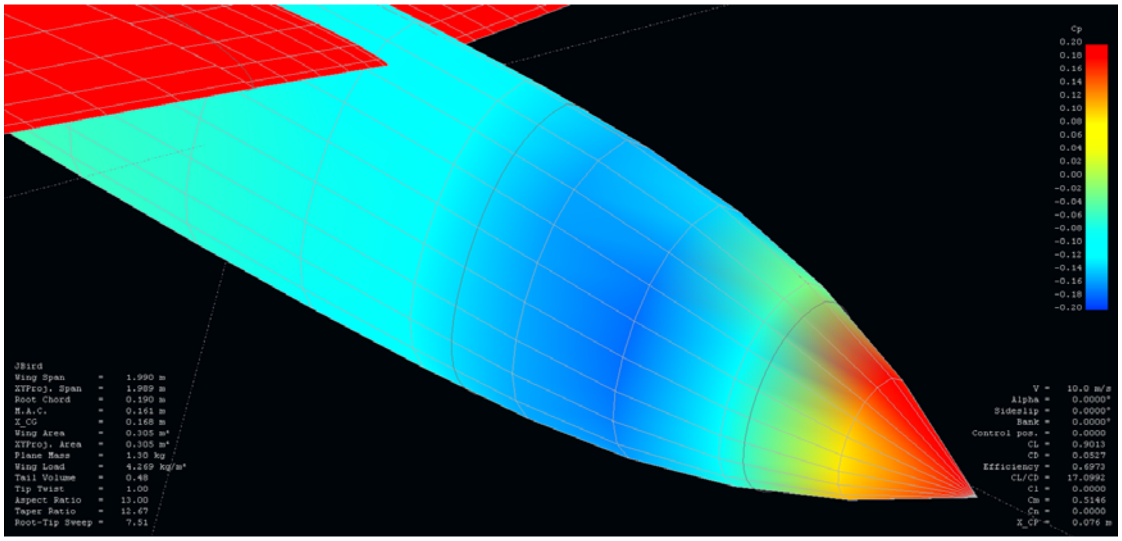
front fuselage area. The pressure coefficient value also remained roughly constant for angles of attack from $-5° < \alpha < +5°$ and airspeeds 10 m/s $\leq V_a \leq$ 20 m/s. Other areas on the aircraft might be slightly closer to C$_p$ = 0.0, but limiting the ports to the front fuselage clusters the FADS equipment close together and prevents pressure tubes from running all over the fuselage. The slight bias in C$_p$ measurements can also be calibrated out in the air data computer. A closeup of the region of interest can be seen in Figure 3.2. Rather than a single port, two ports at the same location on either side of the fuselage are employed. These can be averaged together to help remove the effects of sideslip angles on pressure readings. The static pressure is then obtained from these two ports and used in conjunction with total pressure and other readings.

3.2 Hardware

The results from Chapter 2 and Section 3.1 form the basis to produce a functional FADS nosecone and air data computer. First, the information is used to select appropriate pressure transducers to convert surface pressures into electrical signals. A physical prototype nosecone can then be created and connected to the transducers. The output signals are sent to an Arduino microcontroller acting as the air data computer. This combination forms a complete wind sensing system.

3.2.1 Hardware Selection

The surface pressure values must be measured using some form of pressure sensor. For this purpose, the system relies upon pressure transducers to convert the pressure measurements into electrical signals. There are two main types of applicable pressure transducers: single port



```
JBird
Wing Span        =   1.990 m
XYProj. Span     =   1.989 m
Root Chord       =   0.190 m
M.A.C.           =   0.161 m
X_CG             =   0.168 m
Wing Area        =   0.305 m²
XYProj. Area     =   0.305 m²
Plane Mass       =   1.30 kg
Wing Load        =   4.269 kg/m²
Tail Volume      =   0.48
Tip Twist        =   1.00
Aspect Ratio     =  13.00
Taper Ratio      =  12.67
Root-Tip Sweep   =   7.51
```

```
             V =   10.0 m/s
         Alpha =    0.0000°
      Sideslip =    0.0000°
          Bank =    0.0000°
  Control pos. =    0.0000
            CL =    0.9013
            CD =    0.0527
    Efficiency =    0.6973
         CL/CD =   17.0992
            Cl =    0.0000
            Cm =    0.5146
            Cn =    0.0000
          X_CP =    0.076 m
```

Figure 3.2. Closeup view of the region around the static pressure port location on the Omega II fuselage. Note: the visible static pressure coefficient values range from C_p = +0.20 (red) to C_p = -0.20 (blue).

transducers and dual port, differential pressure transducers. In a single port transducer, only one pressure measurement from the pressure port is output. A differential pressure, dual port transducer measures the difference between two pressure ports and outputs that difference. The actual readings from both ports are not seen in the output, only the difference between them. The advantage of using a differential pressure sensor is the space and approximate cost can be halved with fewer transducers. Most of the surface measurements are not useful by themselves anyways, as only static pressure can be used independently elsewhere. For these reasons, the wind sensing system relies upon dual port, differential pressure transducers.

Two differential pressure transducers were considered for use in the sensing system: Honeywell HSCDRRN002NDAA3 and Freescale MPXV7002DP. The Honeywell sensor is a high performance sensor with an operating range from +/- 500 Pa. This falls right in line with the maximum pressure differentials expected at $\alpha = 15°$and $V_a = 30$ m/s. The full specifications are listed in Table 3.2 and Figure 3.3 shows the sensor. Due to the higher performance, the Honeywell transducer has a limited availability and a relatively high cost of $51.25 per sensor at Digi-Key [36] or even higher at other distributors. This sensor possesses the capabilities needed for the wind sensing system, but at an increased cost.

The other pressure transducer considered is a Freescale MPXV7002DP. These sensors have been successfully used before in wind vector sensing system in the same manner [37, 38]. Specifications of the sensor are seen in Table 3.3 and the sensor is pictured in Figure 3.4. The real advantages of the Freescale transducer is its simplicity, wide availability, and low cost. The sensor is available at multiple distributors online, like DIY Drones, for a cost of $20 each [39]. The sensors are also self contained breakout boards that can be cheaply replaced when one is damaged or destroyed. The downside of them is their 2 kPa range is too broad for the pressure values

Table 3.2. Honeywell HSCDRRN002NDAA3 differential pressure transducer pertinent specifications.

parameter	**value (units)**
price	\$51.25
operating range	+/- 500 (Pa)
supply voltage	3.3 (V)
supply current	1.6 (mA)
signal output	0.33 - 2.97 (V)
accuracy	0.066 (V)
response time	0.46 (ms)

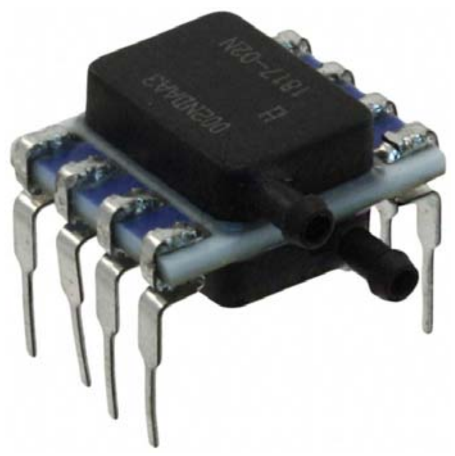


Figure 3.3. Honeywell HSCDRRN002NDAA3 differential pressure sensor [36].

seen by the nosecone. While the sensors will still detect the entire range of pressure changes, sensors with a smaller 500 Pa range like the Honeywell transducer will generally produce a better resolution than this sensor with a very broad range of pressure values. Despite this, the Freescale differential sensors have established success for use in wind vector sensing techniques.

Table 3.3. Freescale MPXV7002 differential pressure transducer pertinent specifications.

parameter	**value (units)**
price	\$19.99
operating range	+/- 2000 (Pa)
supply voltage	5 (V)
supply current	10 (mA)
signal output	0.5 - 4.5 (V)
accuracy	0.1 (V)
response time	1 (ms)

Ultimately, the Freescale differential transducers are chosen based upon their low cost and proven track record. The Honeywell sensors fit the pressure operating range more closely, but at 250-300% the cost of Freescale sensors. The purpose of the nosecone FADS system is to replace expensive \$5000 probes for use on a \$400 sailplane, so cost should be kept low. The previous work with the Freescale sensors has shown them to be not only capable, but rugged and

Figure 3.4. Freescale MPX7002DP differential pressure sensor [39].

interchangeable. With the belly landing Omega II, the cost of replacing sensitive transducers would rise extremely quickly. Despite the loss in performance, the Freescale sensors are used in the wind vector system to measure differential pressure.

3.2.2 Fabrication

The spinner from the electric motor was used as a template for the prototype nosecone. The hemispherical shape of the potential flow model was stretched into a curved conical shape with a 4 $\frac{1}{8}$ cm diameter and 4 cm length. Because of the small size, only five pressure ports could feasibly fit on the nosecone. One of the ports is placed at the tip of the nose with the remaining four distributed over the rest of the nosecone. The stretched shape also required the pressure ports to be moved forward from the location at $\theta = 45°$ to $\theta = 38°$. This should not have too much effect on the pressure readings, but simplified the machining process. These modifications were done to give more access to the inside of the nosecone shell for the assembly of the pressure ports.

A number of different port configurations are possible, but aligning the ports with the body axes maximizes resolution. Other arrangements like a 'x' shape with the ports aligned 45° off the body axes would enable multiple transducers to directly record the changes in pressure associated with angular deflections. This allows for more redundancy in the measurements as more than two transducers would record angular movement, but it reduces the separation between pressure ports in the direction of the body axes. The reduced separation limits the differential pressure reading on the transducer and degrades the resolution when compared to the '+' arrangement. Other configurations allow more than two ports to record α and β deflections, but sacrifice the resolution available.

The nosecone was machined using a 3-dimensional rapid prototyping machine. The updated design was laid out in Solidworks Solid Modeling software before being sent to the machine. The fabrication of the nosecone cost less than \$8 in total at The Pennsylvania State University's

Learning Factory. The plastic material used by the rapid prototyping machine did not present an issue as the material is durable enough to survive hard landings and can be easily drilled and sanded. The only problem encountered is an inconsistent error with small deviations in wall thickness. The deviations caused a few of the nosecones to come out either too large or small to match the shape of the fuselage and form a smooth contour. There is no real fix to this problem, but it didn't drastically change any results.

The nosecone had to be separated into two distinct parts during the machining process to allow access to the inside while still maintaining the solid mount to the fuselage. The larger piece contains the shell of the cone shape, including the pressure port holes. The second piece forms the base of the cone that meets the firewall on the main fuselage. The areas that overlap between the two parts are cut with a jig-saw pattern, as seen in Figure 3.5. When the work on the inside of the nosecone is complete, the two halves snap snugly together.

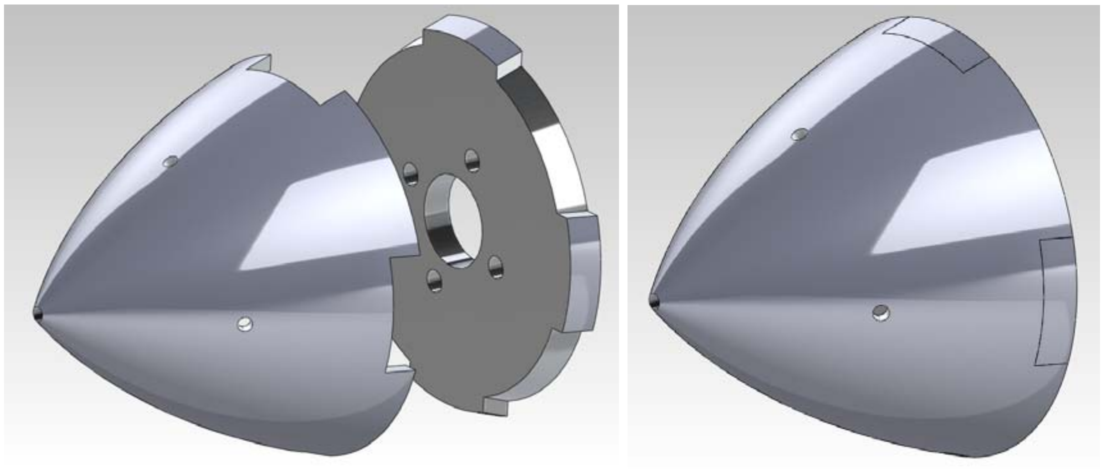

(a) Exploded view. (b) Assembled view.

Figure 3.5. Solidworks model. The first view shows the two separate pieces. The second view shows the pieces snapped together.

The main shell defines the curvature of the nosecone and contains the five pressure ports. The pilot holes for the pressure ports were created by the rapid prototyping machine as $\frac{1}{8}$ inch (3.175 mm) holes through the surface. Small $\frac{5}{32}$ inch (3.969 mm) OD, $\frac{3}{32}$ inch (2.381 mm) ID by 5 cm long brass tubes were inserted into the pilot holes, flush with the external surface. The tubes were sanded down to match the curvature of the nosecone and protrude normal from the surface. Flexible $\frac{1}{8}$ inch (3.175 mm) ID tygon tubes were then attached to each of the brass tubes. Thick layers of epoxy were applied to outside of the tygon tubes where they fit over top of the brass tubes. The thick globs of epoxy were then spread out to the inside wall of the nosecone to make a complete seal between the surface, brass tubes, and tygon tubes. This process is visible in Figure 3.6. The $\frac{1}{8}$ inch (3.175 mm) tube size was chosen to take up less space within the nosecone and Paces et al. [37] already demonstrated that this size of pressure port will have minimal impact on the output signal when compared to larger ports. The same process for making port connections is used for the two static pressure ports on the fuselage, although those tygon tubes are mated together with a 3-way connector.

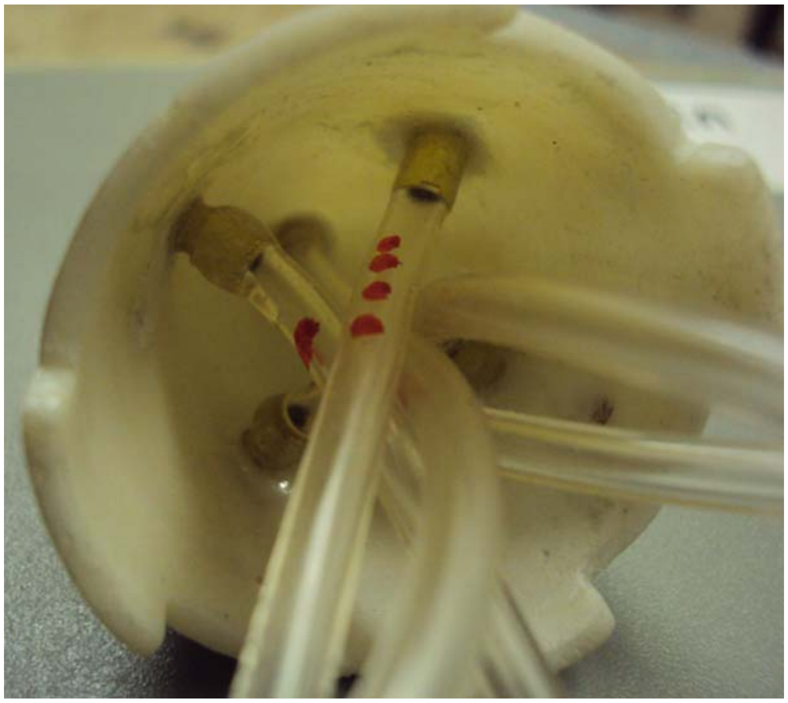


Figure 3.6. Internal view of the nosecone shell. The 5 tygon tubes are epoxied to 5 brass tubes that are in turn epoxied to the inside of the shell. This forms a complete seal through both connections.

Rather than contain pressure ports, the nosecone backing's purpose is to mount the FADS cone to the fuselage. This is done through four 2.5 mm holes drilled into the firewall that previously held the electric motor in place. The holes in the backing piece match the fuselage holes and 2.5 mm nuts were epoxied onto the inside face of the backing. This enables a blind fit when screwing the nosecone into place after the two pieces have been snapped together. The backing also contains a 10 mm hole in the middle that matches a similar hole in the firewall. The tygon tubes from the shell ports can be passed through this hole to the pressure transducers on the inside of the fuselage. After being screwed into place, 2.5 mm screws hold the backing securely against the firewall.

Once the two pieces of the nosecone are finished and the tygon tubes have been passed through the 10 mm hole in the center, the two parts are epoxied together. This permanently seals the pieces together, but is necessary to ensure they remain held together in flight and upon landing. The whole external surface must then be sanded down to smooth out the epoxy bumps and any ridges left from the rapid prototyping machine. This smooth nosecone is the finished product and can be screwed into place with 2.5 mm screws. The attached nosecone is pictured in Figure 3.7.

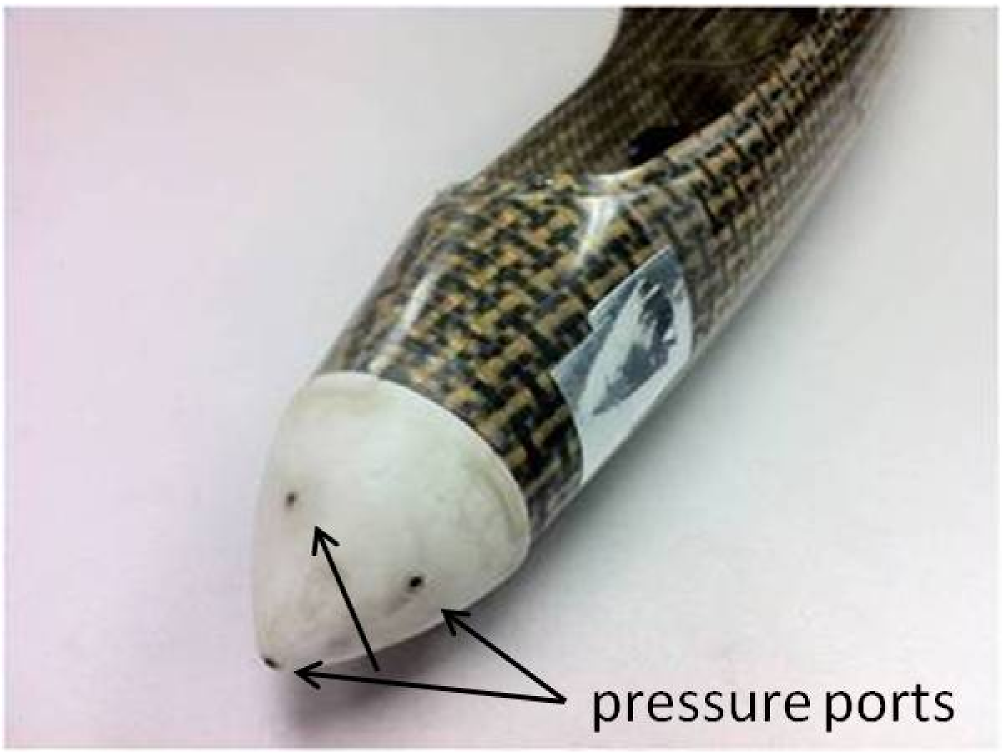


Figure 3.7. View of the finished nosecone after the pieces have been epoxied together and sanded smooth.

3.2.3 System Layout

The signals from the differential pressure transducers are fed into the air data computer, the last component needed for a complete wind sensing system. The electrical signals are converted into digital readings by the computer and sent to model functions that compute the wind vector parameters. The wind vector parameters can then be sent to transmitters, computers, or autopilots, as well as the dynamic soaring controller it's intended for. A diagram of the system is pictured in Figure 3.8.

An ArduPilot Mega serves as the programmable air data computer of the FADS system. The ArduPilot Mega is an Arduino programming language based, single-board microcontroller specifically optimized for use in small UAVs. Arduino-based controllers are widely used in many research projects and there exists a large, well-documented reference and support community. ArduPilot Mega's low cost of $64 and wide availability makes it an attractive solution [40]. The ArduPilot board is easily adapted and can be used as a fully functional autopilot when combined with RC equipment and sensors. The specifications of the ArduPilot Mega are found in Table 3.4 and the board is pictured in Figure 3.9.

Once the computational methods in the air data computer have calculated the wind vector components, this information is sent to the desired controller or transmitter location. The ArduPilot handles the communication, although the exact form of the communication protocol

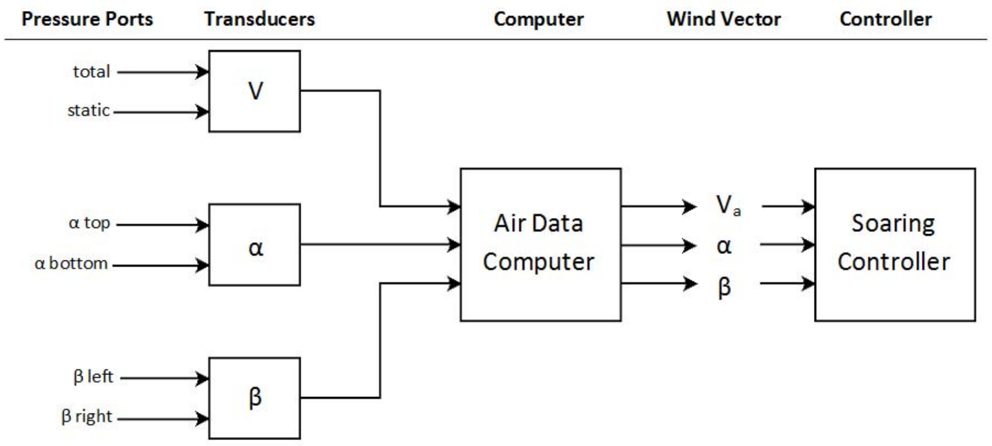


Figure 3.8. System block diagram

Table 3.4. ArduPilot Mega microcontroller specifications.

parameter	**value (units)**
processor speed	16 (MHz)
memory	256 (KB)
battery voltage	7-11 (V)
supply output	5 (V)
digital I/O	40
analog inputs	16
ADC resolution	10 (bit)
serial ports	4
RC channels	8

depends on the system interface and sensors attached. This complete FADS system cost \$8 to manufacture and \$124 to instrument compared to thousands of dollars for a commercial probe system. The next step is to create the parameter computing functions, which first require training data.

3.3 Wind Tunnel Tests

Raw training data was collected from wind-tunnel tests performed in The Pennsylvania State University's low-turbulence, subsonic wind tunnel with a 3.25 x 5 ft (1 x 1.5 m) test section. The data attained in the tests trains the wind-modeling computational methods for the air data computer. Testing was performed across multiple weeks, although only the last week's data is used for training. The other sets of data are still used for data analysis. All of the wind tunnel tests make a large and robust data collection for training the air data computer.

For the wind-tunnel tests, the airframe was mounted to an armature connected to stepper motors. Since the 2-meter wing can not fit within the test section in either direction, two carbon

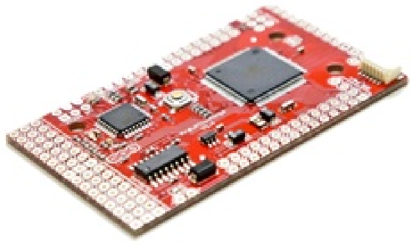


Figure 3.9. ArduPilot Mega microcontroller [40].

fiber reinforced foam wings, 52 and 79 cm span, were cut out with the aircraft's MH-32 airfoil. The aircraft was also mounted in a 90° roll orientation to allow more wingspan to fit within the section. The wing must be included as the inner wing section near the wing root will affect the static pressure ports; however, the outer wing tip sections are assumed to have negligible effect on the readings and can be truncated. Figure 3.10 depicts the aircraft as it is mounted during the tests.

The tests were performed at airspeeds of 9 m/s to 27 m/s in 1 m/s increments. The airspeed themselves were not directly controlled, but rather indirectly through manual control of the wind tunnel's throttle. This airspeed range corresponds to the expected flight speeds of the aircraft while dynamic soaring. At each airspeed, pressure readings were taken from $-15° \leq \alpha, \beta \leq +15°$ in 1° increments. The stepper motors moved in standardized rows controlled by the computer. Dynamic tests consisting of rate and dynamic step tests were performed along with the steady state step tests, but were not included in any training data.

The stepper motor movement and data collection synchronization was commanded through LabVIEW. The LabVIEW program also recorded measurements from pressure transducers mounted in the wind tunnel. These transducers were calibrated to read the dynamic pressure at the aircraft's test section. Static pressure and temperature were measured in 5 minute intervals and used to calculate air density, then combined with the dynamic pressure readings to obtain true airspeed. At each particular commanded location, the LabVIEW program automatically recorded the true α and β locations, collected 101 data points from the wind tunnel mounted sensors, and saved 101 data packets from the ArduPilot containing V_a, α, β, and checksum values. The entire process only lasts about 2 seconds at each location. The wind-tunnel pressure and temperature sensors along with the computer-controlled stepper motors give full knowledge of the true wind vector. This true data can then be used to train the functions with the corresponding V_a, α, and β measurements.

The two reinforced wings were intended to also deal with the large lift forces expected at

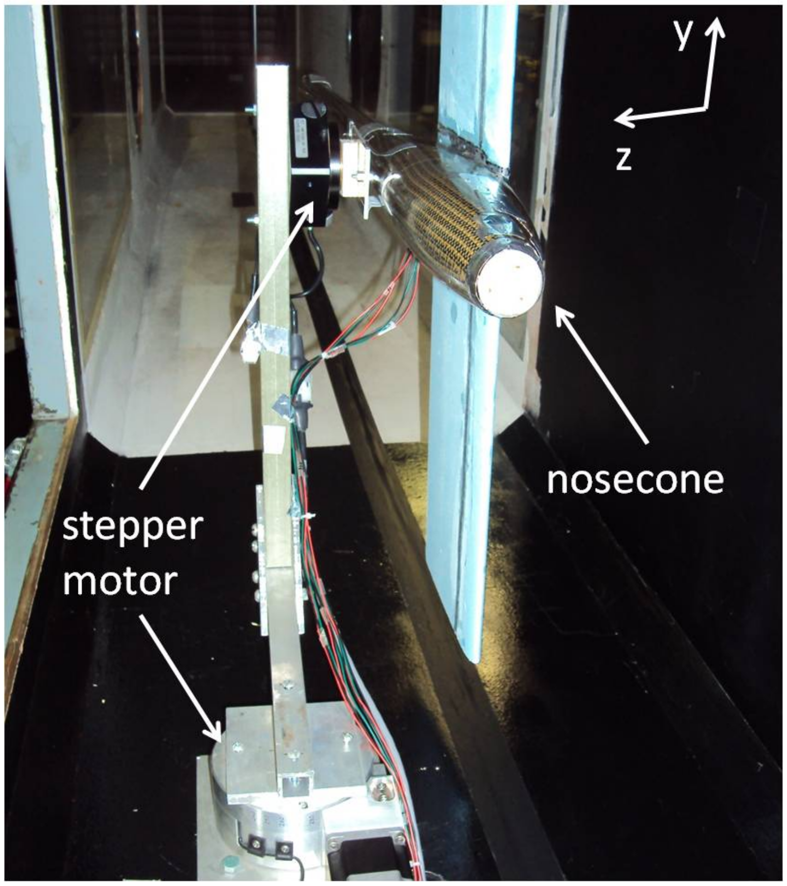
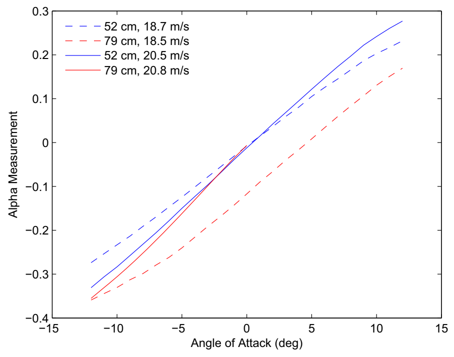


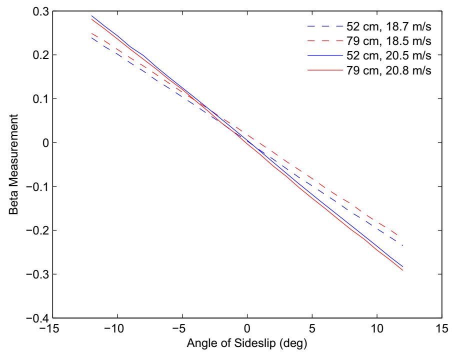
Figure 3.10. The aircraft mounted in the wind tunnel.

high angles of attack at the higher airspeeds. The wings were reinforced with carbon fiber due to the large stresses expected near V_a =30 m/s and $\alpha = 15°$. The shorted wingspans also lessen the torque placed on the stepper motors by the lift force. The pitch PK-266 and yaw PK-245 stepper motors max out at 1.17 and 0.43 N m respectively [41]. Even the truncated 72 cm wing produces too much torque on the pitch motor at high airspeeds, so the 52 cm wing reduces the torque on the motor without being too narrow to affect the static pressure reading. These two wings were necessary to take the place of the 2-meter wing and its effect on the static pressure ports' readings.

During testing, the majority of the runs were completed with the 79 cm span wing. At $V_a =$ 20 m/s, the wing produces too much torque for the pitch motor, causing it to stall at $\alpha = 10°$ and stop moving any further. The α deflections were limited to a maximum of $\alpha = 0°$ until $V_a =$ 24.5 m/s to increase the airspeed range while avoiding stalling the motors. After this, the 52 cm wing was installed on the airframe and the runs from 19 m/s to 25 m/s were repeated to overlap the data for comparison. At V_a = 27 m/s, even the 52 cm wing produced too much torque for the pitch motor so the runs from V_a = 25 to 27 m/s had the α deflections truncated again. The intentions were to explore the full range of α, β deflections until 30 m/s, but the stepper motors could not exceed 27 m/s without stalling. Figure 3.11 shows some of the results for both the 52 cm and 79 cm wings. There are no noticeable differences between the data and both sets of wind-tunnel test data should be included in training data.



(a) Angle of attack comparison.



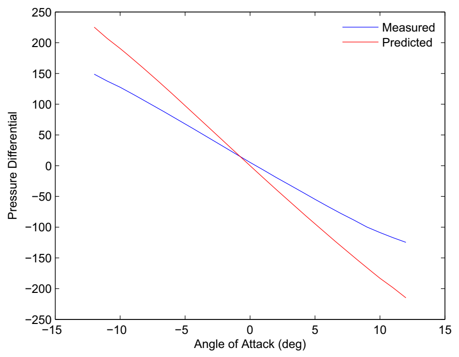
(b) Angle of sideslip comparison.

Figure 3.11. Comparison of similar results from 52 and 79 cm wings.

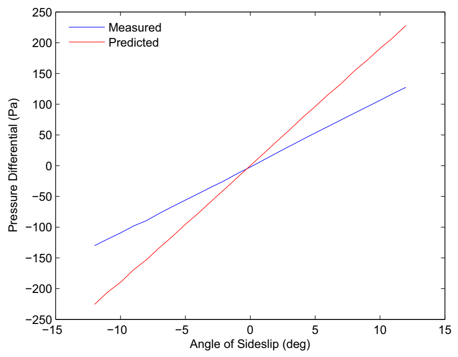
3.4 Comparison to Potential Flow

The results from the wind-tunnel testing can be compared against the estimates from the potential flow model. This gives an indication of the accuracy and applicability of the models for design use. Even if the measurements do not specifically match the model, similar trends in the data indicate the behavior predicted by the model is still valid. Real-world airflow are susceptible to many unforeseen influences and measurements are also affected by sensor bias and noise. Comparison between the actual readings and the predicted readings serves to validate the assumptions made in the design of the FADS system and check for large deviations that adversely affect the results.

A direct comparison between the measured pressure differentials and the difference predicted by the potential flow model shows similar behavior but different magnitudes. In Figure 3.12, both plots depict relatively straight lines that decrease with increasing α or increase with increasing β for both measured and predicted data. Although they share the same behavior trends, it's clear the magnitudes of the potential model are much higher than seen in the actual measurements. A good portion of the deviation can be attributed to assumptions made for both the potential flow model and the pressure measurements. The potential flow model assumes multiple parameters are constant or negligible and doesn't account for interference and viscous effects. The pressure differentials from the wind-tunnel measurements are actually estimates from the digital signals and could easily be inaccurate due to changes in the sensor and analog-to-digital converter parameters. Despite difference in magnitudes, the plots confirm the nosecone measurements behave in the expected manner and share roughly the same intercepts.



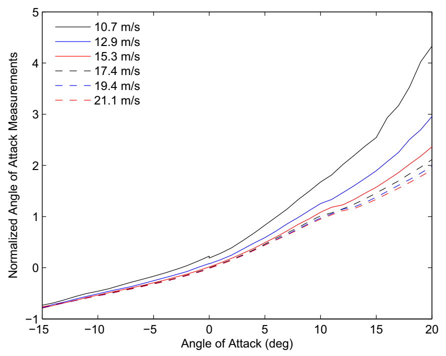
(a) Angle of attack comparison.



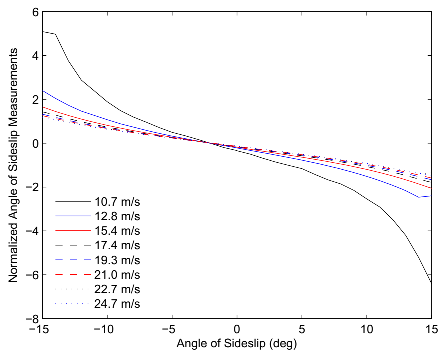
(b) Angle of sideslip comparison.

Figure 3.12. Pressure differentials in wind-tunnel measurements versus potential flow predictions.

Normalized pressure differentials are also used to compare the nosecone measurements with the potential flow model predictions. Equation 2.10 from the previous chapter showed that the α and β pressure differentials are independent of airspeed when divided by q_∞, forming a simple method to compare the measurements. Both plots in Figure 3.13 demonstrate the normalized pressure differences are not totally independent of airspeed, but do begin to collapse onto one another as airspeed increases. The divergent behavior at low airspeed appears to be due to separation and strong viscous effects at the low Reynolds numbers. Additionally, the wing has a pronounced effect upon the α differential and especially the q_∞ measurement. The effect of the wing propagates upstream to the nosecone. Likewise, the q_∞ measurement really measures the U' component of the wind vector from Equation 2.4 as α or β change, not the magnitude, V_a, so the plots are not perfect comparisons to the model. Regardless, the normalized measurements show the same definite trend predicted by the potential flow model.



(a) Angle of attack measurements.



(b) Angle of sideslip measurements.

Figure 3.13. Pressure differentials normalized with dynamic pressure measurements.

Despite the differences in specifics, the general trends of the wind-tunnel measurements match the predictions from the potential flow model. The measurements support the assumptions made by the model and verify that nothing unexpected has occurred. This verification clears the measurements to be fully capable of training the computational methods in the air data computer.

3.5 Sensor Resolution

Before proceeding, the resolution of the ArduPilot's 10-bit analog-to-digital converter (ADC) is examined. The wind sensing system needs high sensitivity from the pressure sensors and ADC in order to maintain high accuracy. Since the Freescale transducers already have too broad an operating pressure range, the associated sensitivity of their signals is suboptimal. The ArduPilot's 10-bit ADC ought to be compared against a higher bit converter to verify the lower resolution is not further degrading the sensitivity.

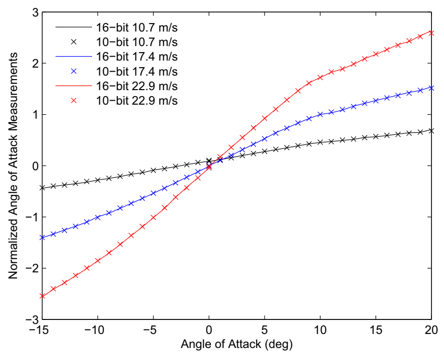
To check the effect of the Arduino's ADC resolution, the analog signals from the transducers are also connected to an external ADC. The wiring connecting the transducers to the ArduPilot are split and connected to a Texas Instruments PCI-6036E 16-bit ADC. The 10- and 16-bit readings are normalized onto the same scale by dividing each reading by the respective value at the same predetermined position index. Three airspeeds at or around $V_a = 10.7$, 17.5, and 22.9 m/s are used to compare the results for different airspeeds. The data is then collected in α and β sweeps at these airspeeds. Comparison between the 10- and 16-bit results indicate the effect of ADC resolution on the measurements.

The analog-to-digital converter mean and standard deviations results are the same regardless of the ADC resolution. Figure 3.14 depicts the mean readings for the 101 data points at each particular step. There is virtually no difference between the 10- and 16-bit mean measurements on the normalized scale for both α and β sweeps. More likely, the difference in sensor resolution would manifest itself in the standard deviation of the measurements. Lower resolution signals would show larger fluctuations with the more discrete jumps. Figure 3.15 plots the standard deviation of the measurements, but there is no difference in magnitude between the 10- and 16-bit signals at each airspeed. The standard deviations don't exactly match due to random noise, but the averages are consistent at each airspeed. The 10-bit ADC signals demonstrate the same mean and standard deviation as the 16-bit higher resolution ADC.

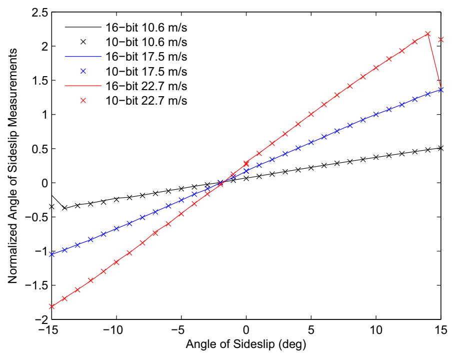
This similar performance shows the signal resolution will not be improved with a higher bit ADC. The noise in the readings is due to external sources beyond control, such as air turbulence or noise in the transducer and its wiring. The 10-bit ADC is not hampering accuracy of the system and its resolution is ample for use with the Freescale sensors.

3.6 Conclusion

This chapter explored the physical implementation of the flush air data system. The design of the nosecone and pressure port configuration was adapted for the Omega II airframe from the

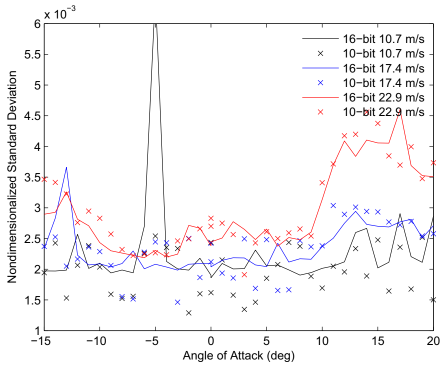


(a) Angle of attack measurements.

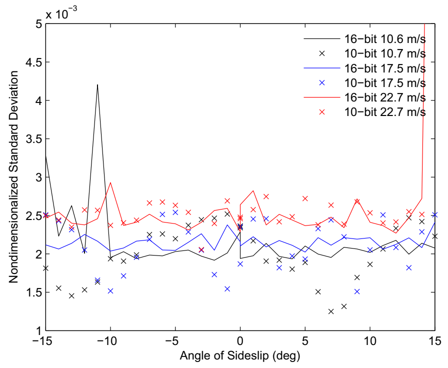


(b) Angle of sideslip measurements.

Figure 3.14. 10-bit versus 16-bit analog-to-digital converter mean readings.



(a) Angle of attack standard deviation.



(b) Angle of sideslip standard deviation.

Figure 3.15. 10-bit versus 16-bit analog-to-digital converter standard deviation.

potential flow analysis in Chapter 2. A rapid prototyping machine created the nosecone in two sections along with five pressure ports. Two static ports were drilled into the fuselage for static pressure measurement. These six pressure readings identify the full wind vector. Freescale differential pressure transducers were chosen for their simplicity and low cost despite their suboptimal sensing range. Output signals from the pressure transducers are sent to an ArduPilot Mega, which acts as a programmable air data computer. This system was mounted onboard the Omega II airframe and tested in a wind tunnel. The results showed agreement with the potential flow model and verified the resolution of the system components.

The results collected in the wind tunnel can now be used to train the air data computational methods. Large sets of data were collected during the wind-tunnel tests; now the question is what to do with all the data. The data itself can't be directly used to find the wind parameters since the pressure measurements and wind vector do not have a one-to-one relationship. The data can instead train multiple data models to compute the full wind vector. Neural networks are chosen for this purpose, detailed in Chapter 4.

Chapter 4

Surface Pressure to Air Data

This chapter covers the selection, creation, and training of computational methods used to obtain the V_a, α, and β air data parameters from surface pressure measurements. Section 4.1 discusses various computational methods and details the selection of neural networks for this purpose. The basic structure and methodology of neural networks is explained as well. The neural network training process is described in Section 4.2. The training data is taken as a subset of the wind-tunnel test data from Chapter 3. Multiple neural networks of different sizes are trained to each compute the wind vector parameters. The results from Section 4.2 are then used in Section 4.3 to examine the performance of the various neural network models. Comparisons at cruise flight operating ranges and at high angle deflections identify the accuracy of the networks over the entire dynamic soaring cycle. The best networks are selected for implementation on the air data computer.

4.1 Air Data Methods

With the large amount of wind tunnel testing data, a robust, efficient method must convert the pressure measurements into wind vector components. This is made difficult by the complex nature of the airflow over the nosecone when it's at various angles and airspeeds. A simple mathematical model is not possible since the potential flow equations in Chapter 2 used to calculate surface pressure from wind components are not invertible. The equations are not one-to-one functions since different airspeeds and flow angles produce the same surface pressure differential. For example, according to Equation 2.8, the nosecone at $V_a = 29$ m/s and $\alpha = 2°$ has the same differential pressure of 81 Pa between the α ports as $V_a = 10$ m/s and $\alpha = 18°$. There is a huge difference between the wind vectors and a misinterpretation by the air data computer could prove seriously detrimental, if not disastrous. This only becomes more problematic as the β angle is incorporated and still further complicated with the actual measurements. The air data conversion from surface pressures to wind vector components is not a trivial matter and requires complex methods.

Multiple methods can be used to obtain the wind vector from surface pressure measurements. One particular method is curve or function fitting techniques. The measurement equation is assumed to be known or well-defined to determine the appropriate coefficients. In some cases, this method is combined with Kalman filters for closer approximation [42]. Lookup tables are another method to accurately compute the wind vector from a large collection of data. The air data is interpolated from a 3-dimensional table that includes all the measurements and truth data. This method has been used before for FADS or probe system, sometimes in conjunction with polynomial functions and Kalman filters [43]. These methods have worked successfully in the past, but have their own disadvantages that limit their effectiveness in this case. Function fitting requires a predetermined knowledge of the system and its order or additional methods to account for error and variations. Lookup tables don't need this previous knowledge, but require at least one large table to be stored in the flight computer. The potential flow model from Chapter 2 gives good estimates for the measurements, but the actual 3-dimensional viscous flow over the nosecone is not well understood. Additionally, while the ArduPilot Mega does contain an adequate amount of memory for a lookup table, smaller Arduinos that could also be used do not. Processing larger tables will also take computation time and slow down the ArduPilot. These two methods are not ideal for implementation on this FADS air data computer.

4.1.1 Neural Networks

Neural networks present an ideal solution to compute the wind vector parameters from the surface pressure input. Neural networks are complex models made up of numerous neurons of certain weights and biases. The data collected in the wind tunnel would then be used to train the network and determine the appropriate weights associated with the neurons. Neural networks have been used many times before in FADS or multi-hole probe system due to their applicability [32–35]. They are particularly well suited compared to lookup tables or other functions because they excel when trained with large data sets, but don't need any preceding information on even the most complex systems. The resulting trained networks are capable of high accuracy without prior knowledge and large tables stored in the air data computer.

The readings from the pressure transducers are sent to three separate neural networks rather than a single network. Each network is responsible for computing V_a, α, or β from the three pressure readings. Attempts were made to combine all three networks into a single network with three outputs, but the root-mean-square error increased considerably afterwards. Both feedforward and radial basis neural networks were investigated, but the feedforward networks produced a higher performance despite significantly less neurons. Only feedforward neural networks are used hereafter.

The feedforward neural networks all follow the same general structure. The three readings from the pressure transducers are sent into each network's input layer with N number of neurons. Each input layer neuron is then connected to every neuron in the hidden layer, which is made of N neurons as well. All the neurons in the hidden layer are connected to a single neuron in the output layer that generates the designated wind parameter. The N number of neurons in both the input and hidden layers are all individually weighted. The weighting for each neuron is determined during the training process. The architecture results in a N by 3 weighting matrix for the input layer since it has three inputs, a N by 1 weighting vector in the hidden layer, as well as a N by 1 bias vector for the input layer and a 1 by 1 scalar for the hidden layer. The airspeed neural network's architecture is pictured in Figure 4.1 and the remaining two α and β networks follow the same exact structure.

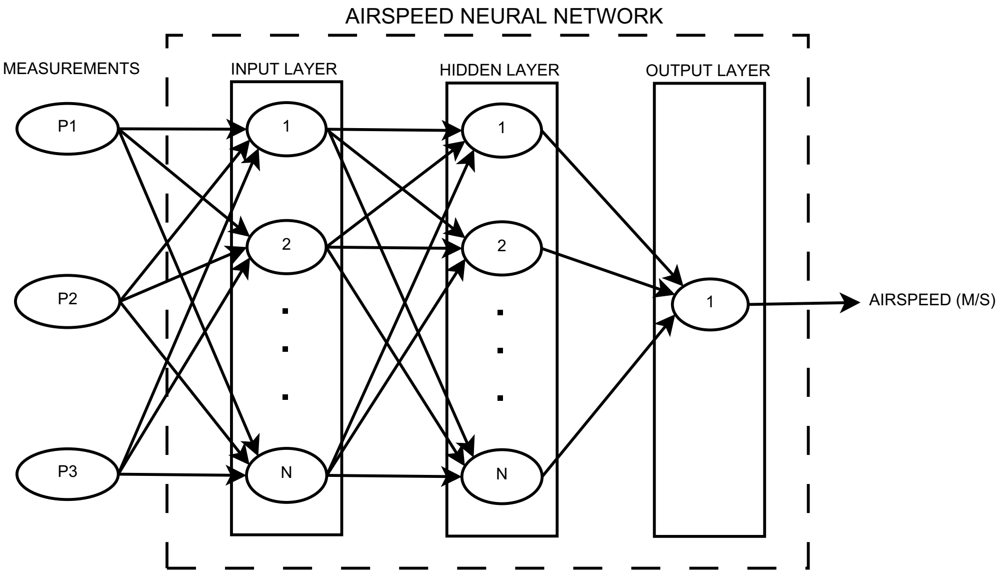


Figure 4.1. The generalized structure of the airspeed neural network with N neurons in the input and hidden layers. The α and β networks follow the same structure.

Different feedforward networks with varying numbers of neurons were compared. Within each network, the N number of neurons in the input and hidden layers was kept consistent, but multiple versions with N = 3, 9, 15, 30, and 45 neurons were analyzed. With each increase in the number of neurons, the complexity and memory requirement also increased. Table 4.1 shows the number of neurons in each layer along with the number of parameters to be stored for both the single network and the total system. Large numbers of neurons will generally improve the accuracy, but at the risk of slowing down the computer or even exceeding memory capability of the Arduino if too many parameters are stored. All of the networks are still fully capable of converting the surface pressure measurements to air data components.

4.2 Neural Network Training

The training data used for the neural networks is taken as a subset of the wind-tunnel surface pressure measurements and truth data. Data from all the airspeeds are used with the aerodynamic angles truncated to $-5° \leq \alpha \leq 12°$ and $-6° \leq \beta \leq 6°$. These angles correspond to

Table 4.1. The number of parameters stored in each network. The number of neurons affects the parameters stored in the input and hidden layers for each network. The total number of parameters accounts for three networks with the same structure.

Neurons	Input Layer	Hidden Layer	Network Total	System Total
3	12	4	16	48
9	36	10	46	138
15	60	16	76	228
30	120	31	151	453
45	180	46	226	678

the usual operating range of the Omega II sailplane. Data points outside the expected operating range would expand the capabilities of the sensor at extreme flight conditions, but these capabilities are not necessary for this particular soaring application. Also, the extraneous data adversely affects the training process and degrades the eventual network's effectiveness within normal flight regimes.

Before use, the training data set was preprocessed to remove bad measurements and fit within the training process bounds. ArduPilot data packets that failed to pass a simple checksum were removed. Anomalous measurements that otherwise passed the check were also deleted. For example, a 2000 Pa reading among one-hundred 38 Pa readings would have been identified as anomalous. This anomalous reading probably occurred as the sensor momentarily disconnected as it bounced around within the aircraft. The deleted anomalies were obviously incorrect readings, not unfavorable data readings. This data verification proved important because the sensor readings are averaged together. Rather than use all 101 data points, the mean of the 101 or remaining data points for a particular V_a, α, and β location is used as the single measurement for this location in the training data. This use of mean values reduced the number of training points drastically, but didn't skew the results.

In addition to removing bad or anomalous measurements, the training data was preprocessed to adjust the scale and offset of the subset. All the data was shifted until a reading of zero corresponded to zero pressure differential. This helped center the data since the three pressure transducers each have their own slight bias that sometimes changed as temperature or airspeed varied. The data was also normalized to a scale of +/- 1 since the neural network training process requires the input data to fall within these bounds. To normalize the measurements, every mean data point was divided by the single largest magnitude measurement from the entire set of wind-tunnel measurements. This ensured that in all subsequent use of the networks, the data wouldn't exceed the bounds as long as this same value was divided from the measurements. The normalized training data subset has a resulting range of -0.2726 to 0.8288. After preprocessing, the training set was stored as a 4648 by 3 input matrix.

The true V_a, α, and β values corresponding to the 4648 by 3 training subset were saved in three separate target vectors. One vector stored the airspeed values from the wind tunnel as the targets for the airspeed neural network. Unlike the input data, this vector is not normalized and the data is kept in units of m/s. The α and β target vectors are made up of the true positions of the aircraft recorded from the position of the stepper motor when the measurements were taken. These vectors were converted to radians to prevent any unforeseen issues with angles in degrees. The large training data input matrix and the three corresponding target vectors are then ready for training the neural networks.

4.2.1 Training Method

The feedforward neural networks were trained using the Levenberg-Marquardt algorithm. The Matlab Neural Network Toolbox includes the specific Levenberg-Marquardt function used for training the networks [44]. The algorithm uses a backpropagation, gradient-based approach to minimize the mean-square error. Matlab's Levenberg-Marquardt training function, trainlm, is one of the easiest methods to implement. According to the toolbox's comparison of methods, the algorithm performs particularly well on nonlinear regression fitting and is the fastest at function approximation [44]. The method does require more memory than the other quasi-Newton or Resilient Backpropagation techniques available, but this is not an issue since the networks themselves were already kept small for their Arduino implementation. After each training process, the networks are retrained heuristically until performance doesn't improve.

The training data matrices are further broken up into three sections for training, testing, and validation. The data is randomly separated into 70% for training, 15% for validation, and 15% for independent testing. The first two sections are used in the training method itself while the third section tests the trained networks as an external performance verification technique. The results of the training and testing are shown in the next section.

4.3 Results of Wind Tunnel Based Training

The airspeed, angle of attack, and angle of sideslip neural networks are trained with the methods previously described. The overall results of the training are seen in the root-mean-square error (RMSE) of the training output, listed in Table 4.2. These results include the training, validation, and independent testing RMSE for the entire training data set.

Not surprisingly, the most complex systems demonstrate the lowest error. The airspeed RMSE only improves slightly between the 15, 30, and 45 neuron networks. The 30 neuron V_a network performs almost as well as the 45 neuron model and either can be used in the final configuration. The α and β networks both demonstrate diminishing returns with increasing neuron count, although not to the level of the airspeed networks. The α networks may have benefited most from the increased number of neurons since the aircraft is asymmetric in the y axis and the effect of the wing factors into the measurements more. With that same logic, it's surprising the V_a RMSE doesn't improve more drastically with more neurons. The airspeed ports really measure U' from Equation 2.4, not V_a, so a higher number of neurons would seemingly account for this complex relationship better. The fact that it didn't might reflect the difficulty in airspeed sensing or the accuracy has actually reached a bottom limit with the measurements.

Regardless of the airspeed limit, all these results point to accurate wind sensing through the neural network computational method.

Table 4.2. Root-mean-square error of neural network output. The performance corresponds to the entire training set containing the training, validation, and testing subsets.

Neurons	Airspeed (m/s)	Angle of Attack (deg)	Angle of Sideslip (deg)
3	0.2551	1.3594	0.4026
9	0.2272	0.7429	0.2685
15	0.1910	0.7193	0.2252
30	0.1855	0.5120	0.1744
45	0.1844	0.3648	0.1497

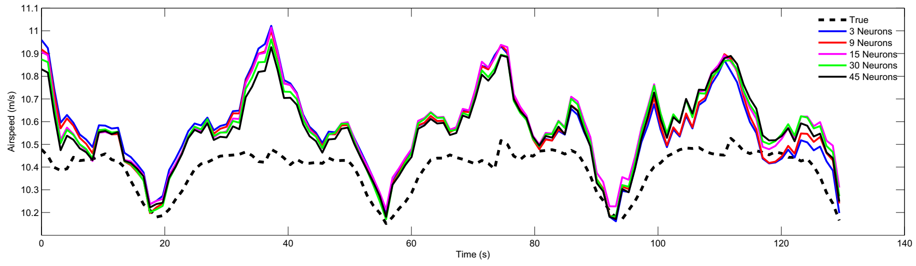
4.3.1 Full Operating Range

The results of the neural networks are compared to the true V_a, α, and β values over the entire nominal operating range. Figure 4.2 depicts the performance of the V_a, α, and β networks for measurements taken at the lower end of the airspeed spectrum at $V_a = 10$ m/s. The performance of the same networks for measurements taken at a higher airspeed of $V_a = 20$ m/s is shown in Figure 4.3. The true angle of attack covers the full range of the training set from $-5^\circ \leq \alpha \leq 12^\circ$ while only one half of the β range is used since it can be assumed to be symmetric.

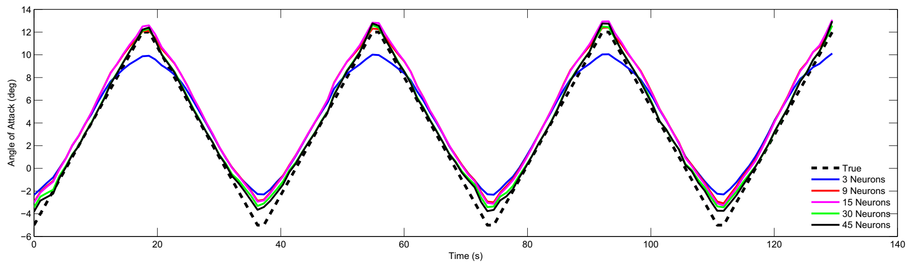
Both figures show the same general behavior for the V_a, α, and β plots. All the neural networks behave rather consistently with similar overshoot or undershoot at the same spots. The higher neuron count models match the true values closer than the fewer neuron models, although not by much. The trends also demonstrate the higher count networks don't seem to be overfitting the data either as once feared. Overall, the networks perform better at higher airspeeds due to the larger pressure differential. The higher pressure values give the pressure transducer more sensitivity and enable finer measurements. The training data taken from these measurements will result in finer network outputs. The airspeed bias due to angle of attack is higher at the lower airspeeds. The bias decreases from $V_{\alpha,bias} = 0.6$ m/s in Figure 4.2(a) to $V_{\alpha,bias} = 0.25$ m/s in Figure 4.3(a). Angle of attack predictions show improvement in overshoot and undershoot in Figure 4.3(b) over Figure 4.2(b). The β networks also show better performance at higher β positions in Figure 4.3(c) at the higher airspeed. These results definitely suggest the limiting factor is the pressure transducer sensitivity. Despite this issue, the networks all work as intended and further plots take closer observations at their performance in particular instances.

4.3.2 Cruise Regime

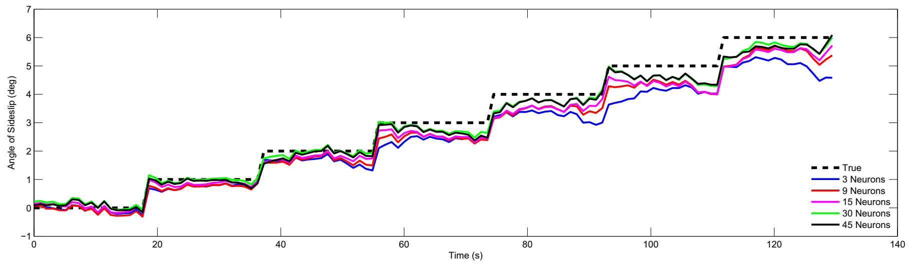
While Figures 4.2 and 4.3 show the general performance behavior of the networks over the entire training set range, it's difficult to compare the accuracy of the networks on such a large scale. For this reason, a smaller data set consisting of flight parameters in the cruise flight regime is used. The cruise regime is defined as angular positions of $2^\circ \leq \alpha \leq 6^\circ$ and $0^\circ \leq \beta \leq 2^\circ$. Angle of attack $\alpha = 2^\circ$ corresponds to trimmed flight at 20 m/s while $\alpha = 6^\circ$ corresponds to trimmed



(a) Airspeed Performance



(b) Angle of Attack Performance



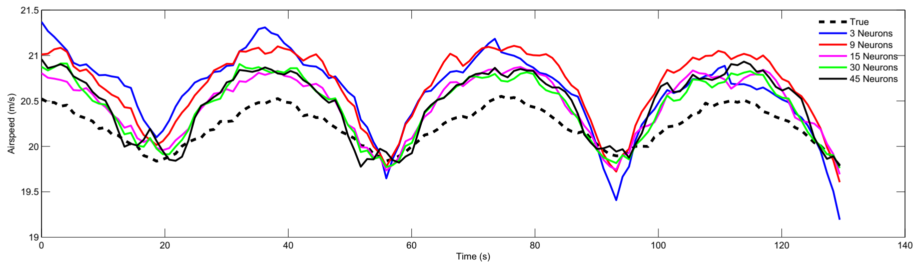
(c) Angle of Sideslip Performance

Figure 4.2. Performance of the neural networks for the full range of data at 10 m/s with $-5^\circ \leq \alpha \leq 12^\circ$ and $0^\circ \leq \beta \leq 6^\circ$.

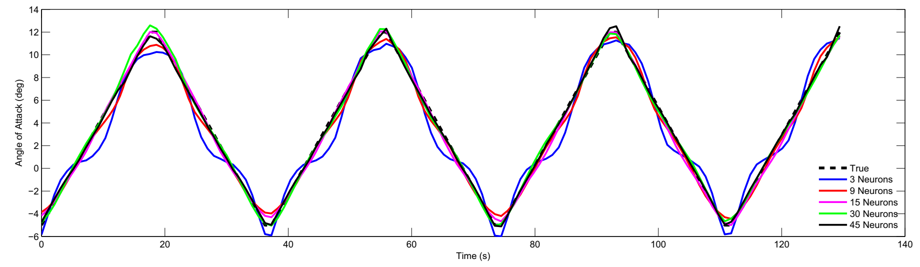
flight at 10 m/s. The aircraft does not generally slip drastically in this case, so β is limited to $\beta \leq 2^\circ$. A closer analysis of the cruise parameters allows a more in depth comparison of the networks' performance in the expected operating range.

The cruise regime analysis depicts a clear improvement in accuracy attributed to larger neuron layers. Lower airspeed results at $V_a = 10$ m/s are shown in Figure 4.4 along with higher airspeed plots at $V_a = 20$ m/s in Figure 4.5. In both figures, the α and β biases decrease noticeably with an increased number of neurons in the networks.

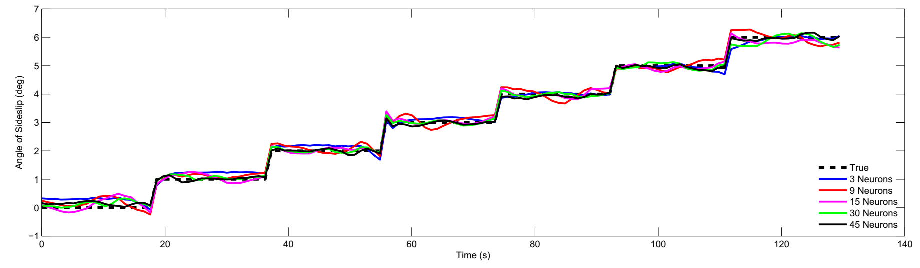
The angle of attack bias decreases from an almost constant 1° offset for the 3, 9, and 15 neuron networks to a max of 0.3° for the 30 and 45 neuron networks in Figure 4.4(b). Figure 4.5(b) has the same behavior. The angle of sideslip networks in Figures 4.4(c) and 4.5(c) show an improvement in accuracy in the higher neuron models, but not as distinct as in the α performance plots.



(a) Airspeed Performance



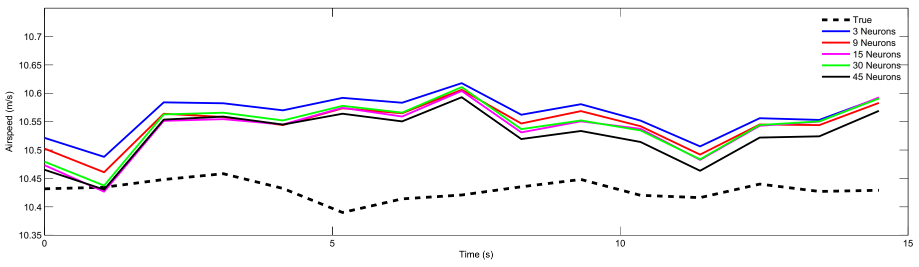
(b) Angle of Attack Performance



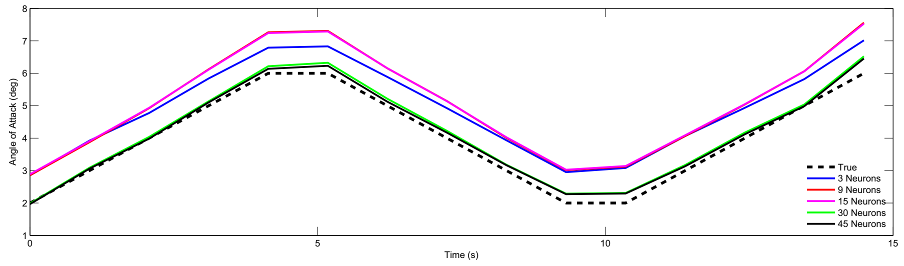
(c) Angle of Sideslip Performance

Figure 4.3. Performance of the neural networks for the full range of data at 20 m/s with $-5^\circ \leq \alpha \leq 12^\circ$ and $0^\circ \leq \beta \leq 6^\circ$.

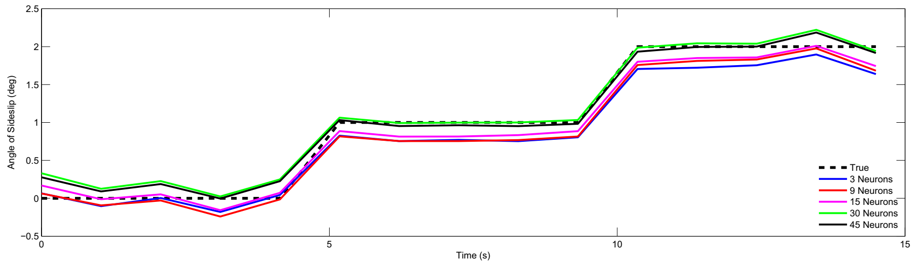
The airspeed figures show less success than the α and β plots. In Figure 4.4(a), there is only slight increases in accuracy with increasing number of neurons. There is an improvement in accuracy with the 15, 30, and 45 neuron models at $V_a = 20$ m/s in Figure 4.5(a); however, there is still a 0.3 m/s bias between the predicted and true airspeeds. This noticeable bias appears to be another result of the sensitivity in the pressure transducers and in the true airspeed measurements. The bias might result from inconsistencies in the measurements and tradeoffs made in the training process to account for these inconsistencies. Even though the V_a performance is less pronounced, all three wind vector parameters' performance in the cruise regime demonstrate a clear boost from more complex neural networks.



(a) Airspeed Performance



(b) Angle of Attack Performance

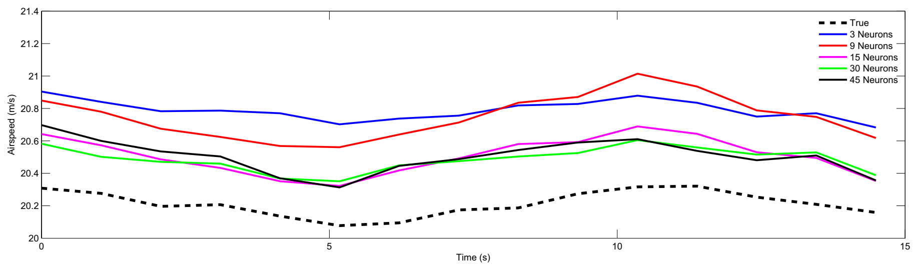


(c) Angle of Sideslip Performance

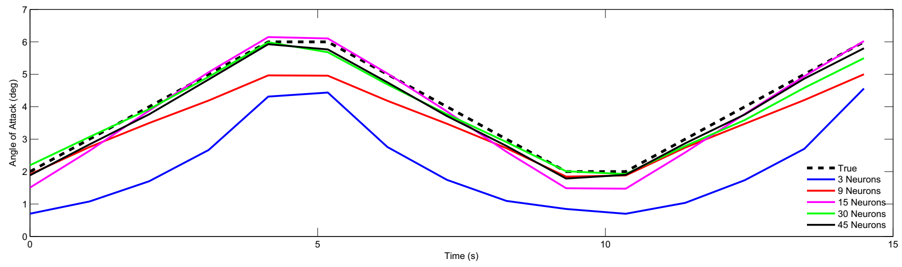
Figure 4.4. Performance of the neural networks for a cruise operating range at 10 m/s with $2^\circ \leq \alpha \leq 6^\circ$ and $0^\circ \leq \beta \leq 2^\circ$.

4.3.3 High Angle Deflections

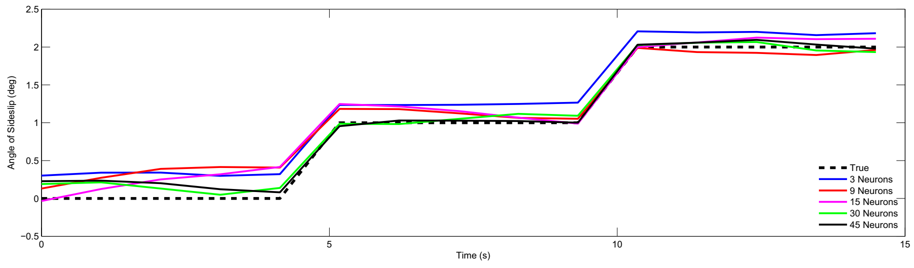
During dynamic soaring maneuvers, the aircraft will have to perform sharp climbs, dives, and turns to correctly orient with the wind as commanded by the soaring controller. Even if the aircraft spends most of its time cruising in steady glides, the dynamic soaring maneuvers are the important sections of the flights. The complex control parameters are the reason a wind sensing system is needed so the system better perform well during these maneuvers. While the exact airspeeds and angles required for dynamic soaring vary, certain portions of the flight path will require rapid changes in lift coefficient and heading angles, recorded in optimal control paths [21, 22]. At these portions, the lift coefficient reaches values from $1 \leq C_L \leq 1.5$ and the associated angle of attack will be near maximum. During these rapid lift coefficient changes, the heading angle fluctuates at almost 10°/sec and the angle of sideslip will vary as the aircraft



(a) Airspeed Performance



(b) Angle of Attack Performance



(c) Angle of Sideslip Performance

Figure 4.5. Performance of the neural networks for a cruise operating range at 20 m/s with $2^\circ \leq \alpha \leq 6^\circ$ and $0^\circ \leq \beta \leq 2^\circ$.

rapidly changes direction and orientation into the wind. A training data subset at high α and β values will give an indication of neural network performance during these important portions of the flight path. Since the previous section examined the performance at low α and β, this comparison will analyze the performance at seemingly worst-case α and β values. A reasonable conclusion about neural network performance during actual dynamic soaring maneuvers can be inferred from the comparisons.

Another truncated training data subset with $8^\circ \leq \alpha \leq 12^\circ$ and $4^\circ \leq \beta \leq 6^\circ$ is used to examine the performance of the neural networks at the extremes of the training data. Likewise, these correspond to the most aggressive orientations the aircraft would experience. Just as in the previous comparisons, the neural network output all appear consistent with no extraneous behavior. The same increase in accuracy at higher airspeeds is observed between Figures 4.6 and 4.7. The lower neuron count models more accurately compute α and β at $V_a = 20$ m/s than at $V_a = 10$ m/s where the plots appear closer together. In general, the higher neuron number networks are more accurate at α and β prediction, just as before.

In both Figures 4.6(b) and4.7(b), the 30 neuron networks have slightly more accuracy than the 45 neuron models. Meanwhile in Figures 4.6(c) and 4.7(c), the 30 and 45 neuron networks exhibit the same performance. With the same accuracy, it appears the 30 neuron network might be a better model for implementation in the air data computer.

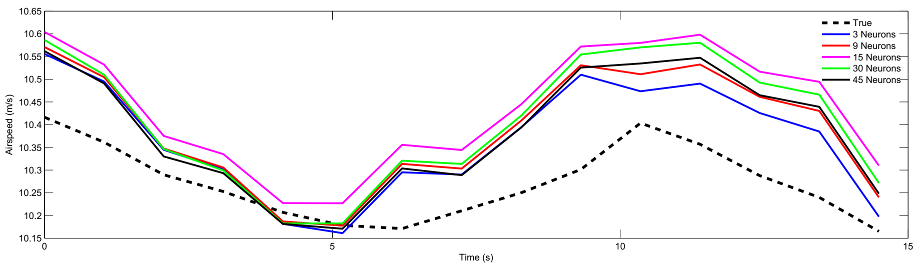
The airspeed plots in Figures 4.6(a) and 4.7(a) experience the same issues as before. All the networks in Figure 4.6(a) perform roughly the same while the 30 and 45 neuron networks display slightly better performance in Figure 4.7(a). The bias observed in all the previous figures disappears in Figure 4.7(a). Again, this seems to indicate a tradeoff in performance during the training process. Despite the higher angular positions, the neural networks all demonstrate the same level of accuracy seen in the cruise operating range. All of the models manage to predict the values with varying levels of success attributed to their complexity.

4.3.4 Network Comparison

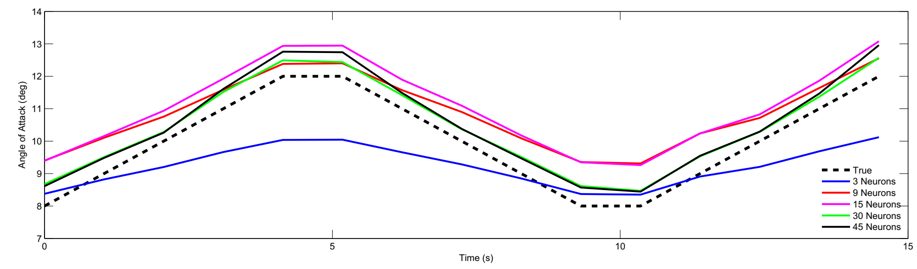
Not surprisingly, the more complex networks outperform the simpler networks across the board. This is most noticeable in the α predictions as the 3 and 9 neuron network predictions overshoot or undershoot the true α values by up to $2°$ or $3°$, particularly at the lower airspeeds. There is quite an improvement in performance from the 3 to 9 neuron networks and from 9 to 15 neurons. Despite the noticeable inferiority to the 30 and 45 neuron α networks, the simpler models perform much better than expected at high angle deflections. If the 30 or 45 neuron networks cannot be implemented, the 15 neuron α network serves as a good replacement computational method.

The angle of sideslip plots show similar improvement in accuracy with more neurons, although at a lower scale. The β networks were already more accurate than the α networks, so the more complex models did not improve the performance as drastically as in the α networks. This could be due to a number of factors. The β networks were trained with a smaller training range that varied much less than the 17 α positions used to train the α networks. The β networks also benefited from the symmetry across the aircraft's z axis to improve performance. Angle of sideslip measurements see little to no effect from the wing due to this symmetry, unlike the V_a and α measurements. These predispose the β networks to less variation in measurements and finer predictions as a result.

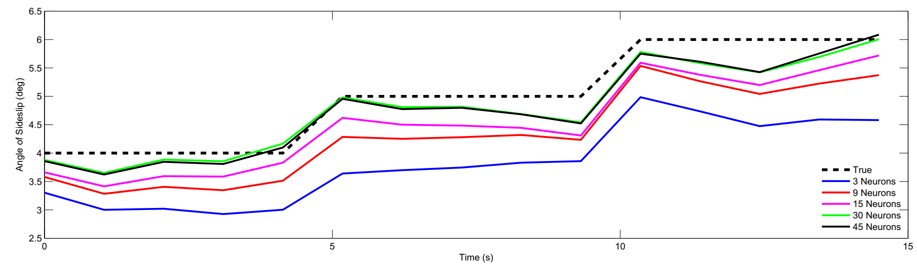
Predictions from the airspeed neural networks display a consistency similar to the β network outputs. This consistency suggests a simpler model might be sufficient and the measurements are noisy. Since the wing affects the V_a measurements more than the α measurements, it appears the measurements are more likely noisy than the complex wing airflow interactions are accurately captured by simple models. Separation or fluctuations due to the wing propagate upstream and affect the static ports, resulting in larger standard deviations. The calibrated pressure transducers in the wind tunnel used to detect true V_a could also present noisy signals and account for the bias observed between predictions and true values in some of the figures. The other possible source



(a) Airspeed Performance



(b) Angle of Attack Performance

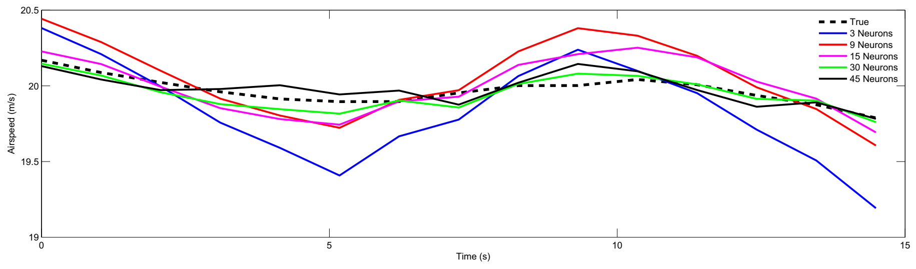


(c) Angle of Sideslip Performance

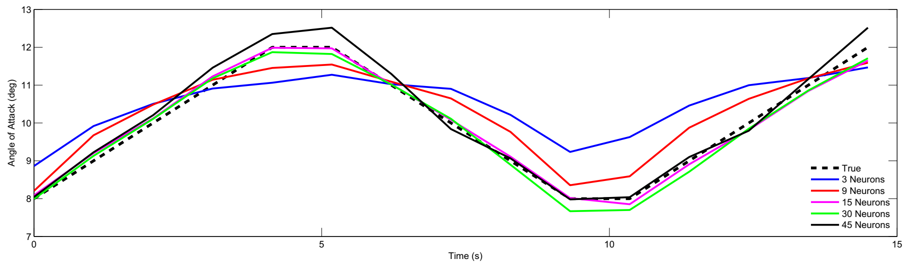
Figure 4.6. Performance of the neural networks for high angle deflections at 10 m/s with $8^\circ \leq \alpha \leq 12^\circ$ and $4^\circ \leq \beta \leq 6^\circ$.

of error is a discrepancy in the airspeed readings. Bias in some portions of the true V_a values or large jumps in airspeed data would force a trade-off in the neural network training process. The V_a would have to accept a certain level of bias to minimize error to the entire set. This type of error would also account for the bottoming out of airspeed RMSE with increased number of neurons. Regardless of the reason, it's only at higher airspeeds that the more complex neural networks demonstrate noticeable improvement over simpler models.

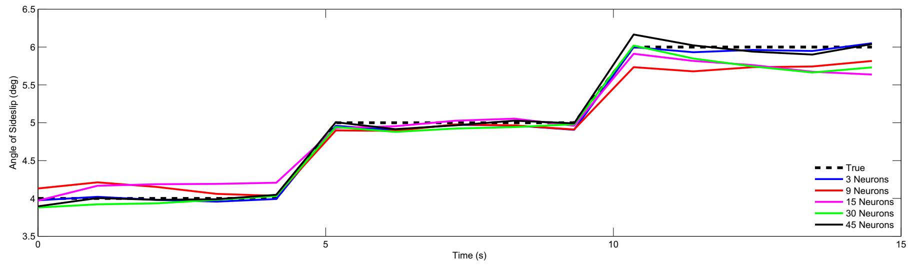
For subsequent implementation, the 30 neuron neural networks are the best models. In all the figures, the 30 neuron networks match the performance of the 45 neuron networks, and exceed it in a few cases. Even though memory capacity is not a problem with the ArduPilot Mega based air data computer, 225 fewer parameters for the 30 neuron networks will ease implementation. If a smaller capacity computer was used, the 15 neuron models would work for one or all of the networks in the computer. The 30 neuron V_a, α, and β networks all demonstrate a high level of



(a) Airspeed Performance



(b) Angle of Attack Performance



(c) Angle of Sideslip Performance

Figure 4.7. Performance of the neural networks for high angle deflections at 20 m/s with $8^\circ \leq \alpha \leq 12^\circ$ and $4^\circ \leq \beta \leq 6^\circ$.

accuracy in the predictions and are selected for the air data computer.

4.4 Conclusion

Computational methods to convert surface pressure measurements to air data parameters were discussed in this chapter. A neural network approach was chosen over lookup tables or other forms of function fitting. Neural networks do not require preexisting knowledge of the system and can handle large data sets without issue. The training data for the networks was taken from the wind tunnel test measurements for all airspeeds with angles from $-5^\circ \leq \alpha \leq 12^\circ$ and $-6^\circ \leq \beta \leq 6^\circ$. Various 3, 9, 15, 30, and 45 neuron networks were trained using the Levenberg-Marquardt process to compute V_a, α, and β. The results from the training and validation processes are then used to compare the performance of the networks.

The 30 neuron neural networks represent the best and most efficient computational methods for the wind vector parameters. The 30 neuron models demonstrate the same performance as the 45 neuron networks, but with hundreds of fewer required parameters. This smaller size makes it easier to implement on the air data computer. The three V_a, α, and β neural networks are now ready to be applied to actual flight tests for real-world performance validation.

Chapter 5

Gliding Flight Tests

The wind sensing system's performance during simple glide tests is covered in this chapter. Section 5.1 discusses the setup and instrumentation of the Omega II for the tests. GPS measurements are used for validation purposes, although more complex sensors to measure the full true wind vector are unfortunately unavailable. Section 5.2 details the systems response to multiple flights. One steady glide with minimal maneuvering identifies the baseline performance of the system since the GPS-derived estimates are more accurate in trimmed flight. Two other flights with elevator and rudder doublets demonstrate the longitudinal and directional responses of the system.

5.1 Testing Setup

For the flight tests, the ArduPilot Mega used as the air data computer also functions as the R/C controller for the Omega II. The ArduPilot is switchable between basic autopilot control and fly-by-wire R/C control. During these tests, the autopilot functionality was removed and all flights are flown under full manual control. The Arduino is responsible for managing all the digital input and output commands on the aircraft. The ArduPilot interprets PPM signals from a 72 MHz, 8-channel Castle Creations Berg4 receiver for pilot commands for aileron, flap, elevator, and rudder deflections. It then converts the pilot commands to servo PWM signals and determines the correct V-tail, flap crow, and differential aileron mixing. While the ArduPilot serves as the flight control system computer, it still handles the data acquisition of the the wind sensing system.

5.1.1 Sensor Integration

A number of components are attached to the ArduPilot Mega for the glide test validation flights. The most necessary components are the Freescale pressure transducers. The transducers are the same exact sensors from the wind-tunnel testing arranged in the same exact order. This will cut down on any issues caused by different bias or resolution between sensors. Unlike in the wind-tunnel tests, the pressure transducers are connected to a proto-board that fits over top of the ArduPilot board rather than directly to the ArduPilot itself. This connection saves premium space within the tightly packed airframe. All the Arduino pins from before are connected back to the same sensors and the proto-board slides over top of the ArduPilot as seen in Figure 5.1. These three pressure transducers are solely responsible for all the surface pressure measurements on the aircraft.

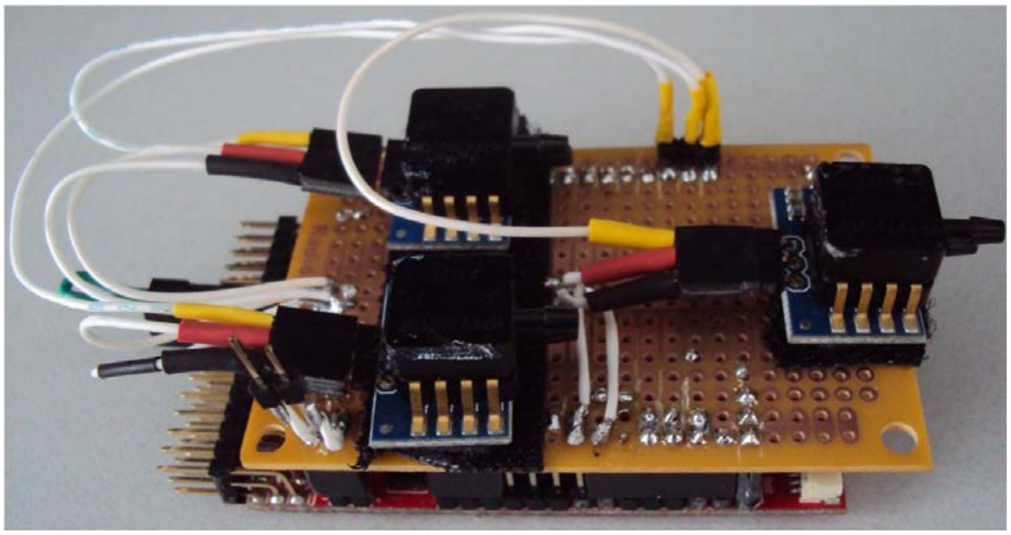


Figure 5.1. The proto-board shield mounted to the ArduPilot Mega.

GPS measurements from a MediaTek GPS receiver monitor the position and groundspeed of the aircraft. The GPS receiver used is a MediaTek MT3329 reciever with a breakout board specifically configured for use with an ArduPilot Mega [45]. Shown in Figure 5.2, the GPS receiver is mounted on the outside of the Omega II's canopy to prevent interference from the carbon fiber fuselage acting as a Faraday cage. Even though the GPS receiver does not measure airspeed or aerodynamic angles, the groundspeed and altitude readings give an indication of the aircraft's overall flight path. This provides some form of validation for the wind vector sensing system and groundspeed matches airspeed on windless days. The GPS can later be coupled with altitude sensors to determine pitch angle using vertical speed. GPS measurements form an integral part of the validation techniques to verify the air data sensor's performance.

Data transmission is handled with two 2.4 GHz XBee Pro wireless radios. One module connects to the Arduino proto-board shield while a second connects to a laptop ground station. Both modules are capable of transmitting and receiving data, although data is only transmitted from the Arduino to the laptop module during flights. The XBee antenna protrudes from the aircraft's canopy right behind the GPS receiver and can also be seen in Figure 5.2. The XBee's 2.4 GHz radio frequency was chosen since it doesn't conflict with the 72 MHz R/C receiver.

The XBee modules allow the measurements from the wind sensing system and the other

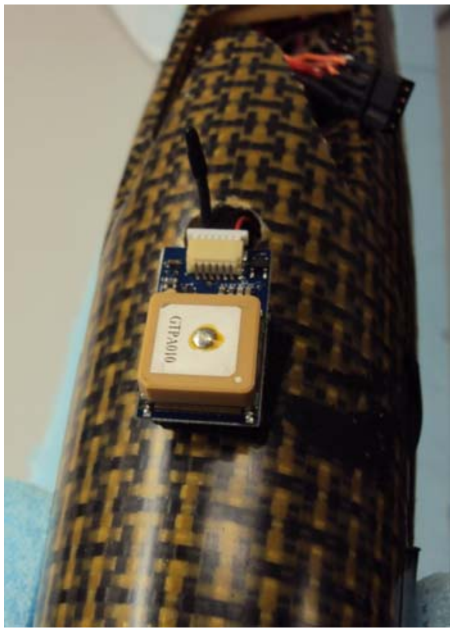


Figure 5.2. MediaTek GPS receiver mounted to airframe.

components to be stored externally on the laptop computer. The XBee radios are set to 57600 baud rate to allow large data packets to pass quickly to the ground station. Messages are also kept brief to save bits and aid in the transmission of messages. Faster data transmission gives finer measurements and prevents information from being lost or bottle-necked. All the data packets are tagged with the Arduino time to indicate the staggering of the messages and check for errors or time delays.

5.1.2 Data Packets

Four main data packet messages are sent over the XBee modules. These messages are listed below:

1. **GPS packet**
   The GPS data outputs in the following format:

   <Identifier; Arduino timer; latitude; longitude; altitude; groundspeed; course; # of SVs visible; GPS time>

   Note that all the units are in meters, meters/second, seconds, or degrees where appropriate. These measurements are limited by the GPS receiver and are sent as soon as they arrive for a speed of 1 Hz. An example packet is shown:

   <$GP; 1101; 23.0985721; 120.2843832; 120; 20; 123.1; 4; 03:35:23.10>

2. **Wind sensor packet**
   The wind data is sent in the following format:

   <Identifier; Arduino timer; airspeed; angle of attack; angle of sideslip; checksum>

   The wind sensor packet includes all the pressure readings as well as a checksum. Because the neural networks have not been verified, the raw measurements are sent to the computer and processed offline rather than on the Arduino itself. The checksum is the same $3 * V_a + 5 * \alpha + 7 * \beta$ formula from the wind-tunnel test data. An example packet is shown:

   <$PR; 1102; 512, 512, 512, 7221>

3. **Radio input packet**
   The pilot commands received by the R/C receiver are stored and repeated back in the following format:

   <Identifier; Arduino timer; L. aileron; R. aileron; elevator; rudder; autopilot mode; pitch trim; autopilot airspeed; flap setting>

   The numbers are sent in PPM format and reflect the commands before V-tail, crow, or differential aileron mixing. The autopilot mode and airspeed commands are currently disabled. An example packet is shown:

   <$RI; 1103; 1500; 1500; 1600; 1400; 2000; 1507; 1837; 1900>

4. **Servo output packet**
   The commands to the servos are also sent to the ground station in the following format:

   <Identifier; Arduino timer; L. aileron; L. flap; R. flap; R. aileron; L. ruddervator; R. ruddervator>

   These servo commands reflect the mixing and are in PWM format. The flap servos were disabled during the flights as they were not needed. An example packet is shown:

   <$SO; 1104; 1500; 1500; 1500; 1500; 1600; 1200>

5.2 Neural Network Results

The 30 neuron V_a, α, and β neural networks trained in Chapter 4 are applied to the flight test data. The neural networks are implemented offline rather than onboard the computer to speed up data analysis and because the best network model was not clearly identified on the dates the aircraft was flown. Raw pressure data is collected from the pressure transducers by the Arduino, sent to the laptop via the XBee, and received through Hyperterminal to externally save the data. Before each flight, the first 5 seconds of the data recordings are used to identify the calibration offset necessary to rezero the readings while the aircraft is at rest. Unexpectedly, this offset changed noticeably from the wind-tunnel tests and varied from flight to flight, even without wind. Because this discrepancy was unexpected, a better method was not in place and instead the data was shifted down manually until it matched the wind-tunnel measurements taken while at rest.

After post-processing the raw data with the correct tare, the measurements are sent to the neural networks. The tared data is normalized according to the same procedure as before with the same exact numbers. These normalized measurements are then fed into each neural network. The measurements are not filtered or pre-processed aside from the normalization scaling transformations described in order to represent a real-time implementation. The results from the neural networks should then behave in the exact same manner as a system onboard the Arduino.

A total of six glide tests were flown with the FADS-equipped Omega II sailplane. All of the flights were conducted during the early daylight hours to minimize wind and gusts. The glider was thrown off a 13-meter slope in a South-East heading, which was maintained for most of the flights. Each flight consisted of simple, deliberate maneuvers except during the first 5 seconds of flight. During these few initial seconds after hand-launching, large variations occurred as the pilot quickly stabilized the aircraft, often from dangerous orientations. According to the University Park Airport's ATIS weather report, the wind speed for that morning was 3 m/s North or North-East [46]. The testing site was surrounded by trees and slopes on three sides and no noticeable winds were observed during the tests, despite the ATIS reading 7 miles away.

Results from each of the glide tests are compared against the GPS measurements. The airspeed output is compared to the GPS groundspeed to give an indication of the neural network's accuracy, especially on a calm day. Since the wind at the test site appeared negligible, the groundspeed from the GPS can be assumed to be roughly equal to the airspeed. Angle of attack output is compared against an α estimate derived from a steady glide assumption and the groundspeed. Using the aircraft's 11.10 N weight and other aircraft parameters, the angle of attack can be estimated for groundspeed with Equation 5.1:

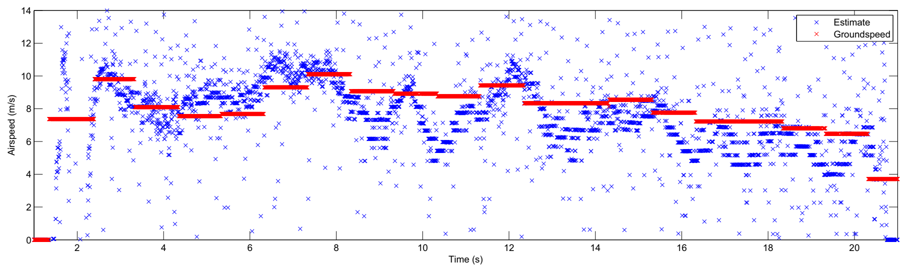
$$\alpha_g = \frac{W}{0.5 * \rho * {V_g}^2 * S * a_{3D}} \tag{5.1}$$

where $\rho = 1.2295$ kg/m^3 on that day and $a_{3D} = 0.1043/^\circ$. There are no equivalent comparisons for β from the GPS data, but it's assumed $\beta \approx 0^\circ$ unless excited by lateral-directional commands.

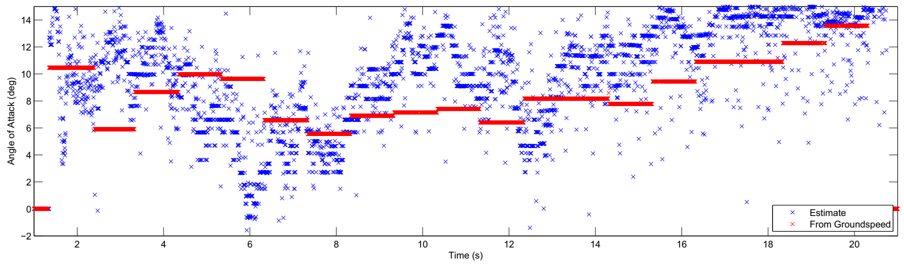
These methods provide a good validation technique for verifying the accuracy of the wind sensing system in the absence of a true validation sensor or technique.

5.2.1 Steady Glide

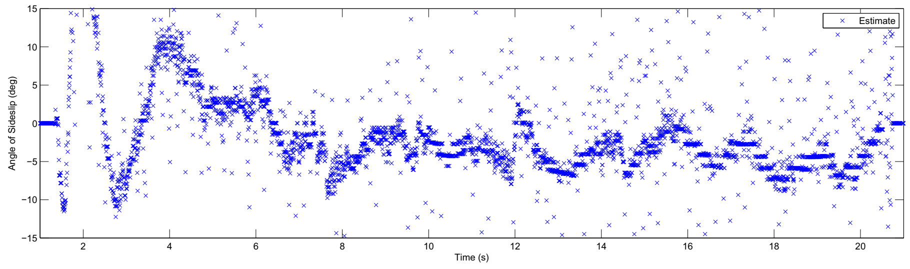
System output for a steady gliding flight is pictured in Figure 5.3. Aside from the first 5 seconds of flight as the pilot stabilizes the aircraft after the hand-launch, there is minimal pilot input. Only minor corrections to level the wings and slight nose up elevator to lightly flare for landing are sent to the computer.



(a) Airspeed Output



(b) Angle of Attack Output



(c) Angle of Sideslip Output

Figure 5.3. Wind vector output from the neural networks for a steady glide flight test.

Airspeed and groudspeed in Figure 5.3(a) display nearly identical behavior. Since there is minimal pilot input and the aircraft glides with no wind, the groundspeed should closely match airspeed. The V_a predictions show more fluctuations, although V_g would not pick up the transients at a slower 1 Hz rate. Surprisingly, the V_a neural network demonstrates good predictions even when the aircraft flies at or below the lower limit of the training data, $V_a = 9$ m/s.

Angle of attack and the associated groundspeed estimate are depicted in Figure 5.3(b). Just like the airspeed and groundspeed, the computed α also displays close agreement with the trend line of the expected α_g value for a steady glide. Towards the end of the flight, the α_g is slightly lower than α; however, this agrees with the airspeed comparison since $V_g > V_a$ during the same period. It makes sense that the α would be larger if the aircraft is flying at a lower airspeed. Additionally, the aircraft is near the ground and the pilot flared the aircraft in ground effect at this point. These conditions could further affect the V_a and α surface pressure measurements and the resulting computed parameters.

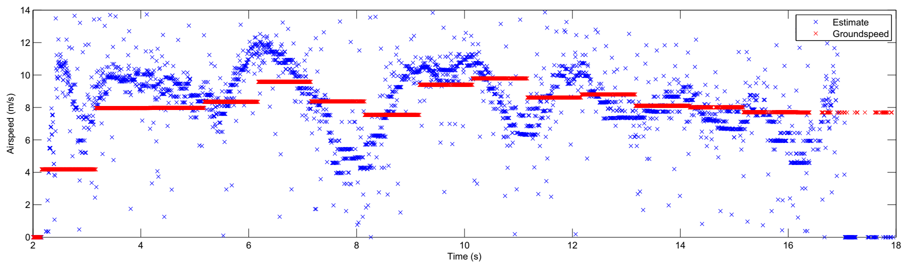
During the flight, β remains fairly constant with a slight negative bias. Only small rudder adjustments were sent so the slight bias is probably the result of incorrect calibration in the measurements that is manifesting in the results. Despite this bias, the computed β values remain close to the expected $0°$ during the flight. All the V_a, α, and β results suggest the neural networks are accurate computational models for the wind parameters during a steady glide. Although the exact true values are unknown, the GPS-derived estimates agree with the neural networks.

5.2.2 Longitudinal Response

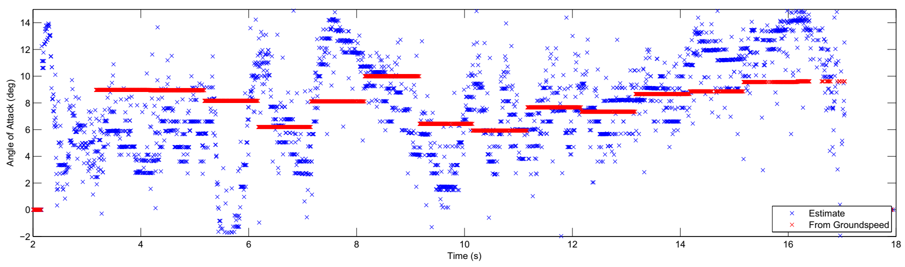
Longitudinal response of the system is displayed in Figure 5.4. The aircraft is excited with two elevator doublets to test the changes in V_a and α. For the doublet input, the elevator is deflected trailing-edge down for a nose down moment at 5.5 seconds into the flight followed immediately by an elevator up command. The entire process is repeated 9.5 seconds into the flight as well. After the doublet, the pilot stabilized the aircraft, partly accounting for the fluctuations in the V_a, α, and even β readings.

Figure 5.4(a) shows the airspeed and groundspeed for the test. In this case, the groundspeed measurements can't be compared as directly to V_a and α as in the steady glide test because the elevator doublet induces a highly dynamic response. Despite this, the trend in the groundspeed agrees with the trend in the airspeed. The groundspeed still decreases as the average V_a decreases and increases with the increasing airspeed. Since the GPS updates at 1 Hz, all the V_g measurements seem delayed compared to the widely varying V_a predictions. These variations depict the effect of the rapid pitching at the elevator doublets. The aircraft picks up speed when the nose is pitched down at t = 5.5 seconds and then slows as the nose is pitched back up. The magnitudes seem off as the aircraft would stall at an airspeed of $V_a = 4$ m/s, but the airspeed behavior definitely displays the intuitive response.

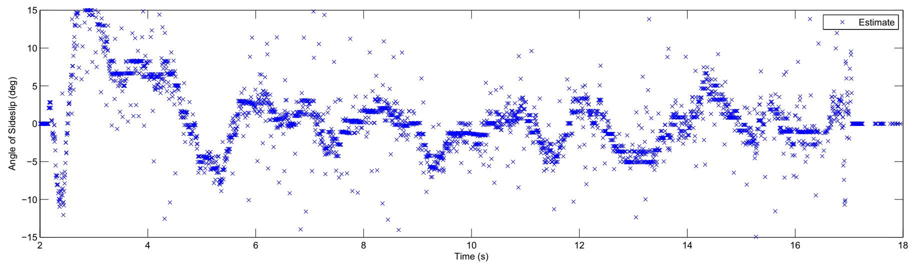
Angle of attack response is illustrated in Figure 5.4(b). The aircraft isn't in a steady glide for most of the flight so the groundspeed estimate, α_g, is inaccurate until the later part of the flight when the aircraft glides in for a landing. Even though elevator commands pitch rate, not α, the elevator doublet should be reflected in the angle of attack response. Assuming the flight path remains roughly constant immediately following the elevator excitation, the near



(a) Airspeed Output



(b) Angle of Attack Output



(c) Angle of Sideslip Output

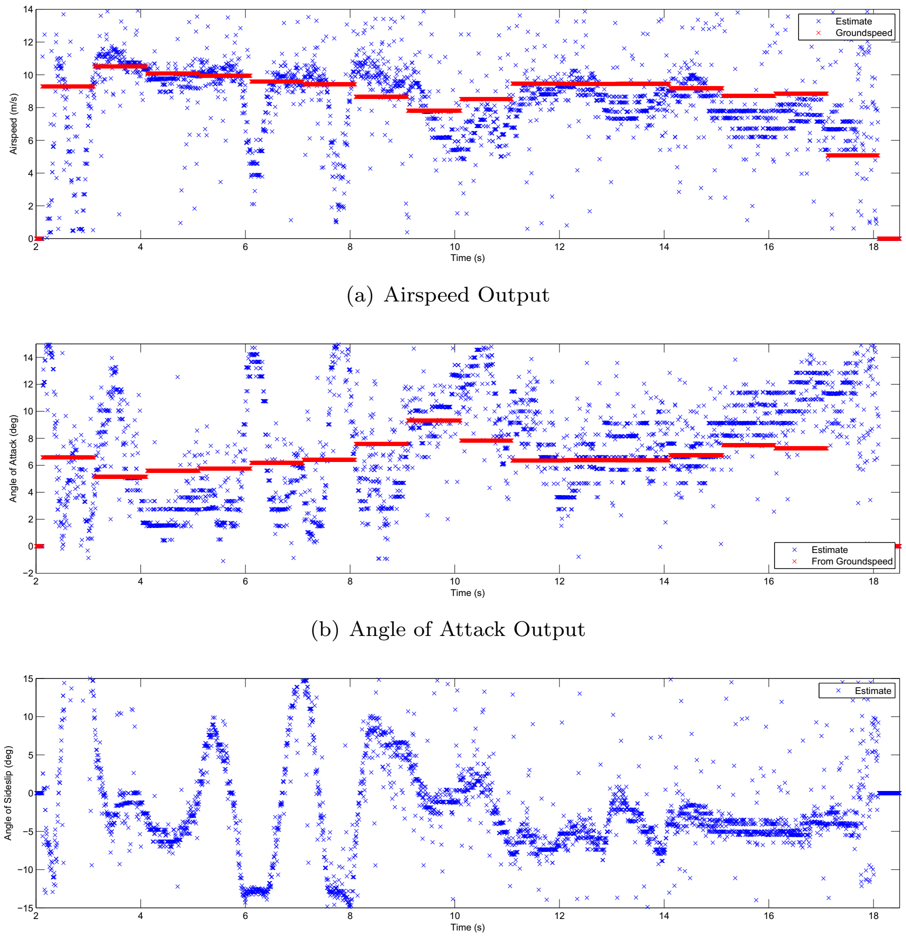
Figure 5.4. Wind vector output from the neural networks for a glide excited with elevator doublets.

instantaneous elevator commands should induce a rapid change in α for a few seconds while the aircraft responds. Indeed, α decreases following the nose-down pitching command and increases after the nose-up elevator input.

The longitudinal response of the system readily matches the expected trends in the V_a and α values. The elevator doublet is noticeably detected by the neural networks and seen in the results. The magnitudes of the parameters appear to have a slight bias, but the true values are unknown. Additionally, the biases from this particular test flight seem abnormal compared to the other flights and could slant the output. Overall, after some fine-tuning, the system appears fully capable of detecting longitudinal variations in the wind vector.

5.2.3 Directional Response

The Omega II is excited with rudder doublets to induce a lateral-directional response. The neural network output for this test is shown in Figure 5.5. A rudder doublet in this case consists of a trailing-edge left rudder command to yaw the aircraft left followed by opposite rudder to the right. In this flight, the two doublets are performed in rapid succession with one at t = 5 seconds and the second at t = 6.5 seconds into the flight. Just like the elevator in the longitudinal response, the rudder does not directly control β, but the rudder still induces a change in the angle of sideslip as the directional response.



(a) Airspeed Output

(b) Angle of Attack Output

(c) Angle of Sideslip Output

Figure 5.5. Wind vector output from the neural networks for a glide excited with rudder doublets.

The angle of sideslip response to the rudder doublet is plotted in Figure 5.5(c). The β results reflect the commands since β increases as the aircraft yaws left and decreases as it yaws right. The sideslip readings flatten out at β = -12° which is at least partly due to the measurements exceeding the training data limit at β = -6°. The rudder causes extremely large and rapid changes in sideslip so these β predictions outside the training range are questionable, but don't appear unreasonable. Immediately after the second doublet, the transients are expected as the aircraft stabilizes itself. All the β values closely mirror the intuitive results.

The airspeed and groundspeed plots in Figure 5.5(a) should not display large changes as a result of the rudder commands. Along with α, only token changes in V_a are recorded during the rudder commands, except for two particular points. The airspeed drops rapidly for a brief instant after the rudder inputs are sent, but quickly recover back to the previous values. This behavior is unexpected, although it could be due to unknown transients in the surface pressure measurements that manifest during the maneuver and affect the neural network output. Even with the airspeed drops and recoveries, the neural network readings follow expectations with little change in the mean value.

The neural network results agree with the expected directional response of the system. Airspeed and angle of attack predictions do not demonstrate any response to the rudder doublet aside from the extraneous transients. Angle of sideslip matches the change expected from the left and right rudder commands. The success of the system proves it can detect the trends in the directional response given pilot inputs.

5.3 Conclusion

The 30 neuron feedforward networks from Chapter 4 compute V_a, α, and β for multiple glide tests of the FADS-equipped Omega II. The aircraft is instrumented with available sensors to measure pressure and GPS packets and to wirelessly transmit that data to a ground station. Unfortunately, a more complex validation method was not available so the exact wind data parameters are unknown. Despite this, numerous flights are performed to compare the response of the system to longitudinal and directional inputs. The system outputs agree with GPS measurement derived estimates for a steady glide. Angle of attack and sideslip predictions follow the expected trends for elevator and rudder doublets to test the longitudinal and direction responses. Without knowing the exact true values, the air data results of the tests indicate the system's apparent accuracy in computing the wind vector during both steady flights and dynamic maneuvers.

Chapter 6

Conclusion

Small UAVs' low size, weight, and cost make them attractive for use in various applications. This small size and weight proves to be a double-edged sword as it also limits the amount of fuel or battery space available and the resulting flight endurance. More fuel or batteries can be added, but at the cost of mission payload. Energy harvesting techniques such as autonomous soaring offer the potential to alleviate the lack of endurance without additional fuel.

Dynamic soaring is a particular form of soaring that exploits naturally-occurring spatial wind gradients to extract energy from the environment. Spatial wind gradients occur at shear layers near the surface of the ground or sea, on the leeward side of hills, or even in high altitude jet streams. These are rather reliable and predictable sources of potential energy. Albatrosses routinely fly hundreds of miles out to sea without flapping their wings using dynamic soaring alone. Despite its large potential, this method has one daunting barrier: dynamic soaring requires precise knowledge of the local wind vector to be successfully employed. Out of all the species of birds, albatrosses and petrels are the only birds capable of dynamic soaring. These birds possess unique sensory organs in their beaks and nostrils that act as pitot static systems for measuring airspeed and orientation. If UAVs are going to mimic the albatross dynamic soaring method, they will also require similar sensing systems to determine the wind vector.

This thesis presents a method for wind vector sensing on a small UAV to enable it to employ dynamic soaring techniques. Different forms of wind sensing exists, but are ill-suited for application on small UAVs. Small aircraft are limited in the weight and space alloted to a sensor which prevents most wind vanes and booms from being implemented. The low-cost nature of the aircraft also precludes several thousand dollar probes from being mounted to a rough landing glider that only costs a few hundred dollars. To address these concerns, the approach uses surface pressure measurements taken with flush pressure ports on a nosecone to determine the wind parameters. This flush air data system (FADS) mimics the same method proven successful by albatrosses.

A potential flow analysis of a hemispherical nosecone model is performed to validate the concept of a FADS nosecone. The potential flow model predicts the differential pressure readings expected over the nosecone and identifies trends within the data. Results from the analysis point to the optimal configuration and location of the pressure ports to maximize the accuracy and sensitivity of the system. The design suggested from the potential flow analysis is adapted to create a functioning prototype. A rapid prototyper machined the nosecone while commercially available pressure transducers were used to keep the cost down. The entire functioning system cost less than $130 to produce, but is still fully capable of sensing the aircraft's speed and orientation in the local wind field.

Neural networks are implemented in the air data computer to calculate airspeed, angle of attack and angle of sideslip from the surface pressure readings. Wind-tunnel tests of the nosecone and FADS system collected data at various airspeeds and aerodynamic angles. Three separate feedforward neural networks to each compute V_a, α, and β are trained using the wind-tunnel measurements and Matlab's Neural Network Toolbox. Five different networks of increasing complexity and size are tested for each wind parameter with the remaining data. The 30 neuron networks are chosen over 45 neuron models because 30 neuron neural networks performed nearly the same as the more complex models, but with 225 fewer stored parameters.

Gliding tests with the FADS-equipped aircraft were flown to analyze the performance of the system during actual real-world flights. The performance of the neural networks and the system is evaluated for steady glides, longitudinal maneuvers, and directional slips. Although the exact true wind parameters can not be determined without more complex validation techniques, GPS measurement derived estimates are compared against the results as a baseline verification. The comparison and trends in the system output qualitatively show the neural-network-based FADS nosecone is capable of accurate wind sensing.

6.1 Summary of Contributions

6.1.1 Design of Flush Sensing Nosecone

A functional nosecone capable of determining the local wind field at airspeeds up to 30 m/s is developed. The system measures surface pressure at five flush ports distributed over the nosecone. These are coupled with two static pressure ports housed in the fuselage. Low-cost, commercial pressure sensors and microcontrollers are chosen to minimize the cost of the system.

6.1.2 Neural Network Based Air Data Computer

Feedforward neural network models are created to each compute the airspeed, angle of attack, and angle of sideslip wind parameters. The networks were trained with subsets taken from wind-tunnel measurements of the FADS nosecone. Output performance of the networks was verified using an independent testing subset of the wind-tunnel test data.

6.1.3 Performance Validation Through Glide Tests

The trained neural networks were applied to results taken during simple flight tests of the aircraft. Results from various flights testing steady glide and longitudinal and directional responses of the system demonstrated correct qualitative behavior. The system showed agreement with GPS-derived estimates of the true wind vector and intuitive expectations of the transient response.

6.2 Recommendations for Future Work

6.2.1 Sensor Selection

While the Freescale differential pressure sensors demonstrate they are capable of sensing changes in the wind vector, the limitation of their sensitivity is seen in Chapters 4 and 5. Before it can reach the level of precision required for dynamic soaring, the FADS system needs more sensitivity at lower airspeeds and aerodynamic angles. The aircraft will reach airspeeds at or in excess of 30 m/s during these maneuvers so the sensors still need an adequately matched pressure range. The Freescale transducers worked well, but it was clear from the results they had too large a pressure range to meet the necessary sensitivity. The wind-tunnel and flight test results suggest the Honeywell differential pressure sensor or similarly ranged products will produce clearer readings which would go a long way towards improving precision. Smaller pressure range sensors, possibly coupled with filters, would possess the performance necessary for dynamic soaring UAVs.

The results from the wind-tunnel tests and gliding flights have shown the airspeed can be correctly determined from the differential reading between the total and static pressure ports. As mentioned before, coupling the total and static measurements in one differential pressure transducer prevented them from being separated. Employing two single port sensors in place of the dual port transducer would enable static pressure measurements to be used for pressure altitude and density computation, especially when combined with a temperature sensor. This will not necessarily improve the accuracy of the wind vector sensing but would expand the capabilities of the system. If the space is available after a redesign of the electrical components and proto-board, independent static pressure measurements would serve as a convenient supplement to the full wind vector.

6.2.2 Testing

The initial glide tests demonstrated the viability of the neural network FADS system; however, more thorough evaluation testing is necessary before the system is ready for use with a dynamic soaring controller. Many more flights with hi-start launches for higher altitudes are necessary since the simple hand-launched glide tests were limited in the aggressiveness and duration of maneuvers due to the low altitude. Although the GPS-derived estimates for the wind parameters gave good indication of the accuracy, some independent form of true V_a, α, and β measurement is required. Ideally, this would come through limited use of a 5-hole probe to simultaneously record the wind data alongside the FADS system. Additional verification methods like tower flyby or dynamic maneuvers have been used to identify the true wind vector [1,42]. Longer flight tests with independent verification methods would prove indispensable and would quantify the accuracy of the system.

In addition to expanding flight tests, additional wind-tunnel tests would improve the system's performance. According to the glide test results, the aircraft does fly at airspeed below 10 m/s for extended periods. Since the training data stopped at 10 m/s, the system's accuracy for output below this limit can't be verified. Further wind-tunnel tests to collect additional low-speed training data would solidify the accuracy of the system in these regions. At the other end of the airspeed spectrum, increased tests at higher airspeeds would fill in gaps in the training data. Since the stepper motors used in the wind-tunnel tests jammed at high angle deflections at high airspeed, shortened ranges of data were collected at higher airspeeds. If the motor is replaced or rebuilt, full ranges of angular deflections would prevent any skewing in the neural network training process caused by an inadequate amount of training data at high α, high V_a combinations. Additional wind-tunnel testing would further enhance the neural network performance with the extra training data in underrepresented regions of the flight envelope.

Bibliography

[1] Haering, E. A. (1995) *Airdata Measurement and Calibration, Tech. Rep. 104316*, National Aeronautics and Space Administration.

[2] Space Age Control (2012), “State-of-the-Art Air Data Products Solutions Guide,” Online.
URL `http://www.spaceagecontrol.com/adpgal.htm`

[3] Dartmoor Gliding Society (2012), “What Is Gliding,” Online.
URL `http://www.dartmoorgliding.co.uk/html/what_is_gliding.html`

[4] Langelaan, J. W. and N. Roy (2009) “Enabling New Missions for Robotic Aircraft,” *Science*, **326**(5960), pp. 1642–1644.

[5] Depenbusch, N. T. and J. W. Langelaan (2010) “Receding Horizon Control for Atmospheric Energy Harvesting by Small UAVs,” in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-8177, Toronto, Ontario, Canada.

[6] Sachs, G. and O. da Costa (2006) “Dynamic Soaring in Altitude Region Below Jet Streams,” in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2006-6602, Keystone, Colorado.

[7] Pennyquick, C. J. (2002) “Gust Soaring as a Basis for the Flight of Petrels and Albatrosses (Procellariiformes),” *Avian Science*, **2**(1).

[8] Sukumar, P. and M. Selig (2010) “Dynamic Soaring of Sailplanes over Open Fields,” in *28th AIAA Applied Aerodynamics Conference*, AIAA Paper 2010-4953, Chicago, Illinois.

[9] Barnes, J. P. (2011), “How Flies the Albatross: The Flight Mechanics of Dynamic Soaring,” Online Presentation.
URL `http://www.howfliesthealbatross.com/downloadables/Albatross%20Dynamic%20Soaring.pdf`

[10] Morris, A. (2007), “Birds Art Bulletin,” Online.
URL `http://www.birdsasart.com/bn221`

[11] Zufelt, K. (2011), “Checklist of Seabirds of the World,” Online.
URL `http://www.pelagicodyssey.ca/page4`

[12] Chakrabarty, A. and J. W. Langelaan (2010) “Flight Path Planning for UAV Atmospheric Energy Harvesting Using Heuristic Search,” in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-8033, Toronto, Ontario, Canada.

[13] Allen, M. J. and V. Lin (2007) "Guidance and Control of an Autonomous Soaring Vehicle with Flight Test Results," in *45th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2007-867, Reno, Nevada.

[14] Andersson, K. (2009) "Extending Endurance for Small UAVs by Predicting and Searching for Thermal Updrafts," Mechanical and Astronautical Engineering Dept., Naval Postgraduate School, Monterey, California, 2009.

[15] Andersson, K., I. Kaminer, and K. D. Jones (2010) "Autonomous Soaring; Flight Test Results of a Thermal Centering Controller," in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-8034, Toronto, Ontario, Canada.

[16] Edwards, D. J. (2008) "Implementation Details and Flight Test Results of an Autonomous Soaring Controller," in *AIAA Guidance, Navigation, and Control Conference*, Reston, Virginia.

[17] Edwards, D. J. and L. M. Silverberg (2010) "Autonomous Soaring: The Montague Cross-Country Challenge," *Journal of Aircraft*, **47**(5), pp. 1763–1769.

[18] Patel, C. K., H.-T. Lee, and I. M. Kroo (2008) "Extracting Energy from Atmospheric Turbulence," in *XXIX OSTIV Congress*, OSTIV, Lsse-Berlin, Germany.

[19] Rayleigh (1883) "The Soaring of Birds," *Nature*, pp. 534–535.

[20] Sachs, G., J. Traugott, and F. Holzapfel (2011) "Progress Against the Wind with Dynamic Soaring - Results from In-Flight Measurements of Albatrosses," in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2011-6225, Portland, Oregon.

[21] Zhao, Y. J. (2004) "Optimal Patterns of Glider Dynamic Soaring," *Optimal Control Applications and Methods*, **25**, pp. 67 – 89.

[22] Zhao, Y. J. and Y. C. Qi (2004) "Minimum Fuel Powered Dynamic Soaring of Unmanned Aerial Vehicles Utilizing Wind Gradients," *Optimal Control Applications and Methods*, **25**, pp. 211–233.

[23] Deittert, M., A. Richards, C. A. Toomer, and A. Pipe (2009) "Dynamic Soaring Flight in Turbulence," in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2009-6012, Chicago, Illinois.

[24] Deittert, M., A. Richards, C. Toomer, and A. Pipe (2009) "Engineless Unmanned Aerial Vehicle Propulsion by Dynamic Soaring," *Journal of Guidance, Control, and Dynamics*, **32**(5).

[25] Lawrance, N. R. and S. Sukkarieh (2009) "Wind Energy Based Path Planning for a Small Gliding Unmanned Aerial Vehicle," in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2009-6112, Chicago, Illinois.

[26] ——— (2010) "Simultaneous Exploration and Exploitation of a Wind Field for a Small Gliding UAV," in *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-8032, Toronto, Ontario, Canada.

[27] Gordon, R. (2006) *Optimal Dynamic Soaring for Full Sized Sailplanes*, Master's thesis, Air Force Institute of Technology.

[28] Larson, T. and P. M. Siemers (1980) *Use of Nose Cap and Fuselage Pressure Orifices for Determination of Air Data for Space Shuttle Orbiter Below Supersonic Speeds, Tech. Rep. 1643*, National Aeronautics and Space Administration.

[29] Whitmore, S. A., B. R. Cobleigh, and E. A. Haering (1998) *Design and Calibration of the X-33 Flush Airdata Sensing (FADS) System*, *Tech. Rep. 206540*, National Aeronautics and Space Administration, Dryden Flight Research Center.

[30] Larson, T., S. A. Whitmore, L. J. Ehernberger, J. B. Johnson, and P. M. Siemers (1987) *Qualitative Evaluation of a Flush Air Data System at Transonic Speeds and High Angles of Attack*, *Tech. Rep. 2716*, National Aeronautics and Space Administration.

[31] Larson, T. J. and P. M. Siemers (1981) *Subsonic Tests of an All-Flush-Pressure-Orifice Air Data System*, *Tech. Rep. 1871*, National Aeronautics and Space Administration.

[32] Crowther, W. J. and P. J. Lamont (2000) "A Neural Network Approach to the Calibration of a Flush Air Data System," School of Engineering, University of Manchester.

[33] Calia, A., E. Denti, R. Galatolo, and F. Schettini (2008) "Air Data Compution Using Neural Networks," *Journal of Aircraft*, **45**(6), pp. 2078–2083.

[34] Rohloff, T. J., S. A. Whitmore, and I. Catton (1998) "Air Data Sensing from Surface Pressure Measurements Using a Neural Network Method," *AIAA Journal*, **36**(11), pp. 2094–2101.

[35] Samy, I., I. Postlethwaite, , and D.-W. Gu (2010) "Neural-Network-Based Flush Air Data Sensing System Demonstrated on a Mini Air Vehicle," *Journal of Aircraft*, **47**(1), pp. 18–31.

[36] Digi-Key (2012), "HSCDRRN002NDAA3 Honeywell Sensor," Online.
URL `http://www.digikey.ie`

[37] Paces, P., K. Draxler, V. Hanzal, T. Censky, and O. Vasko (2010) "A Combined Angle of Attack and Angle of Sideslip Smart Probe with Twin Differential Sensor Modules and Doubled Output Signal," in *IEEE Sensors 2010 Conference*, Institute of Electrical and Electronics Engineers.

[38] Chen, X. Q., Q. Ou, D. R. Wong, Y. J. Li, M. Sinclair, and A. Marburg (2009) "Flight Dynamics Modelling and Experimental Validation for Unmanned Aerial Vehicles," in *Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions* (X. Q. Chen, Y. Q. Chen, and J. G. Chase, eds.), chap. 9, InTech, pp. 177 – 202.

[39] DIY Drones (2012), "Breakout Board MPXV7002DP," Online.
URL `https://store.diydrones.com`

[40] ——— (2012), "ArduPilot Mega 1 - Arduino Compatible UAV Controller w/ ATMega2560," Online.
URL `https://store.diydrones.com`

[41] Oriental Motor (2012) "Stepping Motors," Oriental Motor General Catalog 2012/2013.

[42] Parameswaran, V., R. V. Jategaonkar, and M. Press (2005) "Five-Hole Flow Angle Probe Calibration from Dynamic and Tower Flyby Maneuvers," *Journal of Aircraft*, **42**(1), pp. 80–86.

[43] Wenger, C. W. and W. J. Devenport (1999) "Seven-Hole Pressure Probe Calibration Method Utilizing Look-Up Error Tables," *AIAA Journal*, **37**(6), pp. 675–679.

[44] MathWorks, Inc. (2012), "Neural Network Toolbox," Online.
URL `http://www.mathworks.com/help/toolbox/nnet/`

[45] DIY Drones (2012), “MediaTek MT3329,” Online.
URL `https://store.diydrones.com`

[46] Weather Underground (2012), “Daily Observations for University Park, PA,” Online.
URL `http://www.wunderground.com`