EXPERIMENTAL CHARACTERIZATION OF WINGS FOR A HAWKMOTH-SIZED MICRO AIR VEHICLE

THESIS

Zachary R. Brown, Lieutenant Commander, USN

AFIT-ENY-14-M-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-14-M-10

EXPERIMENTAL CHARACTERIZATION OF WINGS FOR A HAWKMOTH-SIZED MICRO AIR VEHICLE

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Zachary R. Brown, B.S.M.E.S.

Lieutenant Commander, USN

March 2014

AFIT-ENY-14-M-10

EXPERIMENTAL CHARACTERIZATION OF WINGS FOR A HAWKMOTH-SIZED MICRO AIR VEHICLE

Zachary R. Brown, B.S.M.E.S.
Lieutenant Commander, USN

Approved:

AFIT-ENY-14-M-10

Abstract

Considerable prior research has been conducted on many aspects of Hawkmoth-sized, piezo-driven Flapping Wing Micro Air Vehicles (FWMAVs), but the majority utilized a common structural wing to conform with a biomimetic design. In this research, six alternate wing designs with the same planform and size, but different structures were built and explored. Finite Element Analysis (FEA) code was used to determine the location of maximum stress, and then mass was removed from minimally stressed areas, under the premise that equal force production with a lighter wing would improve the Micro Air Vehicle (MAV) design. The main metric for this research was vertical force generation per mass; high speed video provided complementary insight. The angle stop setting was fixed at 60° based on prior studies, and tests were executed by mounting the mechanism on an ATI Nano-17 Titanium force transducer. As part of this effort, several manufacturing processes enhancing repeatability and efficiency during testing and assembly were developed. The new method for wing and Piezo Ceramic Material actuator (PZT) attachment allow for constant drive linkage geometry, and non-destructive wing replacement.

The alternate designs created a 3.8 mg to 19.4 mg (6% to 31% of original wing and 0.5% to 2.5% of a typical Hawkmoth) mass reduction, and generated vertical forces approaching the original design. Combining the mass and natural frequency into a Relative Deduced Stiffness (RDS), an effective wing design can be predicted. The best design produced 14% less vertical force than the original design, however, it resulted in a 15% mass reduction from the original wing. High speed video suggested small additional changes to the wing motion could improve performance.

I dedicate this thesis this to my family. Their support has been integral to the completion of this work. My wife has taken on role of both Mother and Father during this time and for that I am grateful. Without her absolute dedication, none of this would be possible.

Acknowledgments

I must thank my thesis advisor, Dr. Mark Reeder, immensely for the guidance provided during this effort. He gave expert “stick and rudder” for this research while allowing me to really explore the topic. Dr. Reeder was always thoughtful with his direction and the results are clear. I would also like to thank Captain Garrison Lindholm, who showed me the AFIT MAV ropes. He taught me how to fabricate the MAVs in addition to spending a significant amount of time walking me through the ins and outs of the previously developed computer software. Lieutenant Colonel Anthony DeLuca was instrumental in providing guidance on the computer software in addition to his vast expertise in the field of Flapping Wing Micro Air Vehicles. His knowledge was key to teasing out some significant insights identified during the research. Dr. Richard Cobb provided critical insight at critical times during the research, helping to provide a solid foundation upon which the rest of the thesis stands.

Zachary R. Brown

Table of Contents

Page

List of Figures

Figure Page

Figure Page

List of Tables

List of Symbols

List of Acronyms

Acronym	Definition	
FWMAV	Flapping Wing Micro Air Vehicle	1
MAV	Micro Air Vehicle	1
US	United States	1
DOD	Department of Defense	1
ISR	Intelligence, Surveillance, and Reconnaissance	1
SOF	Special Operations Force	1
CBRN	Chemical, Biological, Radiological, or Nuclear	1
Re	Reynolds number	2
DARPA	Defense Advanced Research Projects Agency	2
NAV	Nano Air Vehicle	2
AFIT	Air Force Institute of Technology	3
CF	carbon fiber	3
LEV	Leading-Edge-Vortex	5
AOA	Angle of Attack	6
FEA	Finite Element Analysis	9
FE	Finite Element	9
DOF	Degree of Freedom	10
PZT	Piezo Ceramic Material actuator	11
IC	Bob Smith Industries Insta-Cure™ Cyanoacrylate	11
BABM	Bi-harmonic Amplitude and Bias Modulation	12
IGES	Initial Graphics Exchange Specification	17
PIV	Particle Image Velocimetry	21
DXF	Drawing Interchange Format	24

Acronym	Definition
SLDPRT	Solidworks Part Format 24
ODB	OpenDocument Database 27
PLU	LPKF Proto Laser U 30
CKC	Carbon-Kapton-Carbon 31
NI Box	National Instruments USB-6229 Data Acquisition Device 47
USB	Universal Serial Bus 47
GUI	Graphical User Interface 50
EZERA	State-Space Model Identification using the Eigensystem Realization Algorithm Toolbox for MATLAB 50
RDS	Relative Deduced Stiffness 64
AVI	Audio Video Interleave 70
PNG	Portable Network Graphics 70
FRF	Frequency Response Function 108

EXPERIMENTAL CHARACTERIZATION OF WINGS FOR A HAWKMOTH-SIZED MICRO AIR VEHICLE

I. Introduction

1.1 Motivation

THE United States (US) Department of Defense (DOD) constantly seeks to improve mission effectiveness while limiting costs. The Intelligence, Surveillance, and Reconnaissance (ISR) mission is no exception to that paradigm. Micro Air Vehicle (MAV) platforms provide a means by which that goal can be accomplished.

MAVs fit perfectly into the dull, dirty, and dangerous mission environment. In particular a Flapping Wing Micro Air Vehicle (FWMAV) is well suited to blend into dangerous environments that currently force the US to send a Special Operations Force (SOF) to investigate. Sending a small unmanned vehicle into the environment removes soldiers from danger, reducing the threat of detection. A single operator could launch multiple MAVs to conduct ISR missions safely away from the action [23].

The term dirty refers to a Chemical, Biological, Radiological, or Nuclear (CBRN) contaminated environment. A MAV with a full sensor loadout, including a camera, microphone, and CBRN detection equipment, could survey the area and provide critical information long before it is safe to even contemplate sending a human to investigate. This knowledge is gained well ahead of existing time lines, and will help focus the restoration efforts. With a smaller and focused response, the emerging crisis may be handled at a relatively low cost.

Dull environments are those that would require constant attention while waiting for some critical enemy action. Being constantly on guard while combating monotony is not an endeavor for which humans are well suited, generally speaking. An autonomous MAV can maintain a high-alert posture for long periods of time without concern of distraction or boredom. The Global Hawk (RQ-4) currently provides high-altitude long term (up to 28 hours) surveillance, however, MAVs are well suited to providing that same coverage below the rooftops of an urban environment. The MAV would not be in constant flight, but would only use flight as a transition between observation vantage points. The factor limiting on station time would be battery life for which there are a number of possible solutions. For example, perching on high power lines to inductively charge the battery is one answer. In observation mode, power consumption is expected to be low and solar power provides an alternate power source [5].

In 1996 the Defense Advanced Research Projects Agency (DARPA) created the MAV initiative to investigate the new realm of low Reynolds number (Re) operations [32]. The MAV initiative limited the wingspan to 15 cm with no other constraints on the design space. AeroVironment designed the Black Widow which was among the first working solutions. The Black Widow had a mass of 85 grams and a wingspan of 15 cm. Thirty minutes of flight time at an altitude of 240 meters was achieved by the Black Widow [1].

The DARPA initiative is now referred to as the Nano Air Vehicle (NAV) program, and a 20 gram mass constraint was placed on the designs [11]. This program led to a number of imaginative and novel designs. For one example of the creativity brought about by the program, the AeroVironment Hummingbird was the cover photo of the Top 50 Invention issue of Time Magazine in 2011 [18]. At the small end of the scale, Harvard researchers designed and flew a 60 mg FWMAV [33] capable of lifting its own weight, albeit with a power supply external to the MAV. The successful Harvard, Black Widow, and Hummingbird MAVs have opened the door to investigating varying sizes of MAVs.

1.2 Problem Statement

The AeroVironment Hummingbird has a wingspan of 16 cm and a mass of 19 grams [2]. In order to maximize usage for military environments, a smaller MAV size is required. The Air Force Institute of Technology (AFIT) research has been focused on a Hawkmoth-sized MAV through the efforts of a number of prior researchers [4–7, 12, 25–29]. Mass of the flight vehicle has been a limitation to self propelled flight to date. The current level of technology limits mass reduction of certain MAV components. The power supply and transformer masses cannot be reduced at present. Mass must be reduced and performance increased by focusing on those areas which are most accessible.

1.3 Research Objectives

Previous research at AFIT on a Hawkmoth-sized MAV mostly used a bioinspired wing design. The wing structure was built to match, as closely as possible, the structural dynamic performance of the adult Hawkmoth. The reason for this limitation was to place focus on the numerous other parameters which influence MAV performance and a tacit assumption that the biological wing was optimal. It has yet to be experimentally shown, however, whether or not the bioinspired MAV wing, made of carbon fiber (CF) and Mylar®, yields the optimum solution. Removing this constraint allowed for a concentration of effort on characterization of the aerodynamic performance. The present research effort seeks to investigate this area of wing optimization, primarily through force and mass measurements. Seeking to identify any possible gains in efficiency (force / mass), manipulation of the wing design is considered to be of significant value in achieving the ultimate goal of independent flight for the AFIT MAV. These gains will be quantified in terms of increased vertical force production and mass reduction.

The goal of this research is to reduce the mass of the wing while maintaining or increasing the vertical force production. In addition, the lighter mass may reduce the power required to drive the wings. In turn, the battery mass can then be reduced or flight time increased, so even small reductions in wing mass may lead to substantial improvements in overall MAV performance.

1.4 Thesis Organization

This thesis begins, with Chapter II, by detailing the basic mechanics of flapping flight followed by the chronological history of previous AFIT FWMAV research. Chapter III explains the methodology used to design and test the revised wings. Next, Chapter IV contains a detailed look at the computational and experimental results obtained as a result of the research. The final portion of the thesis, Chapter V, reviews the conclusions drawn and highlights some areas for future research.

II. Literature Review and Background

2.1 Flapping Flight Mechanics

The MAV category is defined by DARPA as a vehicle with a maximum dimension of 15 cm [9, 11]. There are two different flight mechanisms below this threshold dimension. These are most easily compared with their biological counterparts, birds and insects.

Ellington [17] illustrates the differences between these two mechanisms. Insects generally fly under laminar flow conditions where the Leading-Edge-Vortex (LEV) plays a prominent role. Birds, having larger wings, typically fly at higher Re under transitional or turbulent conditions. Only hummingbirds can fly across both spectrums. Ellington sums up the difference in saying that, "turbulence is the great dividing line between them [birds] and the insects [17]." Since the AFIT FWMAV is insect sized and will rely on the aerodynamics that apply to insects, the discussion will be concentrated on those laminar mechanisms. The low Re operation of insects is complicated but necessary to understand when designing an insect sized MAV.

To allow comparison of different insects, the Reynolds number must be defined. The flapping flight definition is derived from the Re used for traditional fixed wing flight given in Equation 2.1. The kinematic viscosity (ν) is defined as the ratio of the dynamic viscosity to the density (μ/ρ) of the fluid.

$$\mathrm{Re} = \frac{VL}{\nu} \tag{2.1}$$

Conn, et al. [9] defined the velocity (V) as being the mean wingtip velocity ($\overline{V}_{tip}$) and the characteristic length (L) as being the mean chord ($\overline{c}$). The mean wingtip velocity is found with Equation 2.2, where Φ is the wing beat amplitude (in radians), f is the wing beat frequency (in Hz), and R is the wing length (half of the wing span). The mean chord length is found with Equation 2.3. The aspect ratio (AR) is defined for flapping wings in Equation 2.4 where S represents the wing area. Combining these terms, the Reynolds number for flapping flight can be found with Equation 2.5.

$$\overline{V}_{tip} = 2\Phi f R \tag{2.2}$$

$$L = \overline{c} = \frac{2R}{AR} \tag{2.3}$$

$$AR = \frac{4R^2}{S} \tag{2.4}$$

$$\mathrm{Re} = \frac{4\Phi f R^2}{\nu AR} \tag{2.5}$$

Ellington [17] demonstrated that large LEV form the basis for lift generation of insects that fly at low Re (1,000 - 10,000). He explains how this allows a high Angle of Attack (AOA) to generate lift before stall occurs.

Insect flight is characterized by Conn, et al. as “high speed movement, in terms of both normalized flight speed and wing beat frequency, at low Reynolds numbers [9].” To generate wing motion the thorax is mechanized and moves using the indirect flight muscles as shown in C and D of Figure 2.1.

The wing stroke is characterized by four distinct phases: downstroke, upstroke, pronation, supination [13]. Conn, et al. indicate that the four phases are fully described by

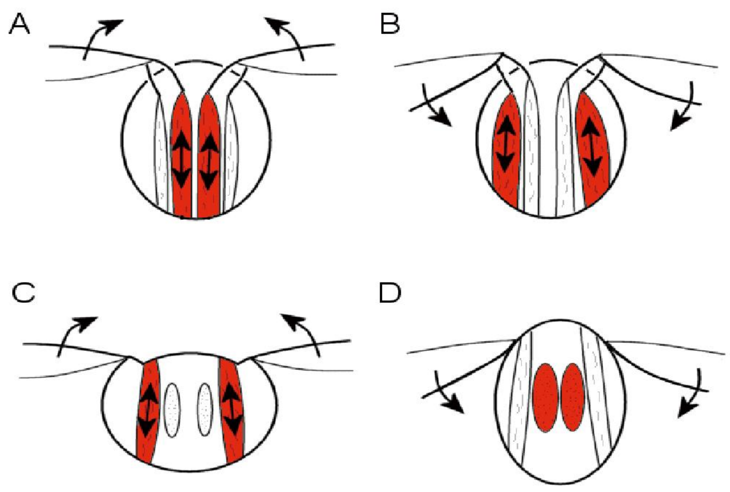


Figure 2.1: Mechanization of Hawkmoth Thorax [9].

several parameters: "stroke amplitude, wing beat frequency, wing angle of attack, stroke plane angle, downstroke/upstroke ratio, wing tip trajectory, timing for wing rotation [9]." The resultant trajectory varies widely even among different insects. Several examples are shown in Figure 2.2, they range from a simple in-plane stroke (Fruit Fly) to a complicated figure eight pattern (Hawkmoth).

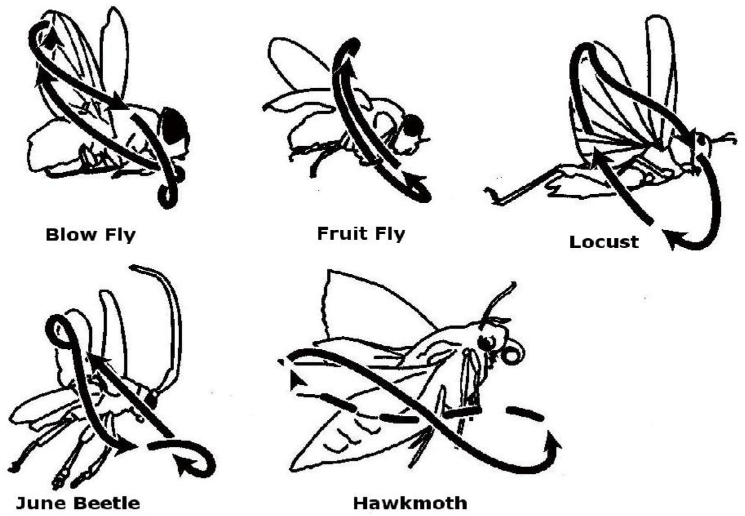


Figure 2.2: Different Insect Wingtip Trajectories [3].

Dickinson, et al. [13] explored a “Rotational Circulation” factor that increases lift during the rotation phases (pronation and supination). The circulation is compared to that of a spinning baseball which draws air into the boundary layer as it spins increasing the flow velocity on one side. The increased lift during these phases is attributable to the injection of air flow into the boundary layer which causes a higher local velocity.

A second mechanism is offered as explanation by Dickinson, et al. [13] for the increased lift independent of the Rotational Circulation. There is a peak in lift at the completion of stroke reversal which the shed vortex from the previous stroke increases the velocity of the flow for the next stroke. This is termed “Wake Capture” and its timing is independent of wing rotation phase, however, the direction and magnitude are heavily dependent up rotational timing.

Conn, et al. indicate that insects employ “LEVs with delayed stall, rotational lift, and wake recapture [9],” to generate lift via unsteady aerodynamics in excess of that which can be explained by traditional steady flow theory.

Harvard researchers implemented the first ever solution to an insect-scale MAV [33]. The 60 mg FWMAV generated sufficient lift to take off with an external power source and five constrained body degrees of freedom. The smart composite microstructures, which became the basis for the AFIT manufacturing technique, are discussed. Additionally, the vein and membrane airfoil methodology is discussed.

The *Manduca Sexta*, more commonly know as the Hawkmoth, is an ideal candidate for study. The Hawkmoth displays excellent aerodynamic characteristics. Forward flight and hovering are both exhibited by the Hawkmoth even when the hind wings are removed as indicated by the Daniel Lab at the University of Washington [25]. This reduces the complexity of the aerodynamics to study when undertaking design of a MAV. There are numerous data sets available that provide specifics on the kinematics of Hawkmoth flight. Willmott and Ellington [30, 31] experimentally quantified controlled flight, and provide a succinct reference to other experimental and computational characterizations of *Manduca Sexta* flight. They found during flapping flight, the Hawkmoth operates with a Re ranging from 3150 to 15220, a reference velocity of 1 m/s and 5 m/s, respectively [30, 31]. The size of the Hawkmoth, shown compared with the size of coins and a commercially available micro digital camera, is shown in Figure 2.3. The Hawkmoth may provide an ideal platform size for current state of the art surveillance equipment.

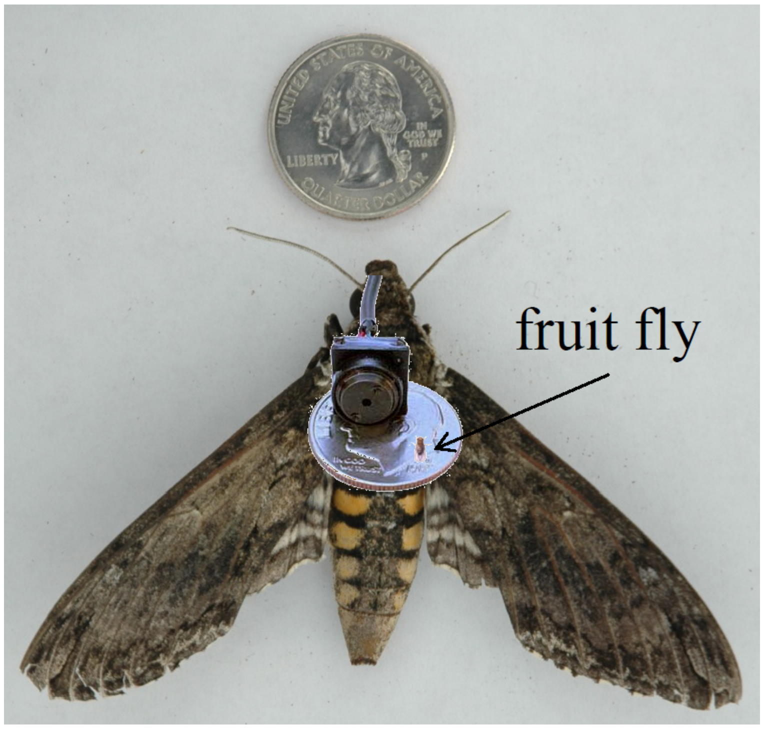


Figure 2.3: Top View of Hawkmoth Compared to US Quarter [25].

2.2 Basic Principles of Finite Element Analysis

Since Finite Element Analysis will be used in this research, an introduction to the fundamental basis for its formulation is presented. Finite Element Analysis (FEA) is concerned with finding a solution to a field equation. A field equation relates some value to its spatial distribution. In the case of the wing, the Finite Element (FE) program solves for the displacements, and then converts them to average stresses on the structure. The basis of finite elements is that each structure can be broken down into a number of pieces and then solved algebraically. Each piece has end points called nodes, which are connected together to form elements. These elements are predefined by the FE program, and allow for varying degrees of freedom. For example, a beam element can allow for six degrees of freedom. When such complexity is not necessary, the user may restrict the active Degree of Freedom (DOF) to allow for only in-plane motion. By reducing the number of DOF, the complexity of the system, and thereby, the solution time are reduced. The important takeaway is that FEA does not provide an exact solution, because the actual differential equations which mathematically describe the structure must be solved. With any sort of complex structure, this can prove extremely difficult at best, and sometimes impossible at worst. Cook, Malkus, Plesha, and Witt [10] provide an excellent in-depth explanation of Finite Element Analysis for the reader interested in learning more.

2.3 Previous AFIT Research

Design of the AFIT FWMAV progressed when Anderson and Sladek worked together in an attempt to create a Hawkmoth size vehicle. Sladek [29] worked extensively on quantifying various manufacturing techniques applicable to this scale of MAV. The four wings shown in Figure 2.4 were all designed with an area of 369 mm^2. The designs used Kapton® for the membrane, and the wing venation was created out of unidirectional CF.

After the initial designs, the unidirectional carbon fiber manufacturing techniques were not satisfactory. Several issues with Kapton® delamination and CF venation breakage were experienced. Based on further investigation of four different manufacturing methods, Sladek found that a three layer CF layup was the most effective. Using a 0° - 90° - 0° layering pattern, and then curing in an autoclave for four hours at 100 psi produced repeatable results. The wings could be fabricated with minimum deviation between

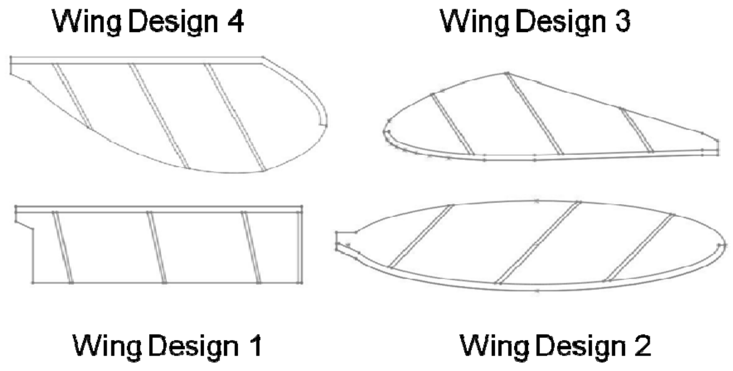


Figure 2.4: Four Initial Wing Designs [29].

batches. In addition, the three layer carbon fiber produced superior results from a structural dynamics perspective.

The four bar linkage in Figure 2.5 was implemented as the transmission between the Piezo Ceramic Material actuator (PZT) and wing. The PZT is substituted for the driving linkage (L_1). A maximum flapping stroke of 60° is used. The maximum angular travel of the linkage is based on a ± 1 mm deflection of the PZT. The linkage is manufactured by sandwiching a flexible Kapton® layer between two three-layer carbon fiber layups.

Sladek measured the forces and moments produced by the four wing designs using an ATI Automation Nano-17R force balance [29]. Wing designs three and four produced the best aerodynamic efficiency. Additionally, the designs had the same 1st and 3rd modes based on this early aerodynamic analysis.

During Sladek's testing, a unique method of wing change out was developed. First, the wing was glued onto the linkage using Bob Smith Industries Insta-Cure™ Cyanoacrylate (IC). Testing was conducted, and then a heat gun was used to heat the IC, and the wing was removed with an X-Acto knife. Next, the wing holder joint was sanded to remove the excess IC. Finally, the next wing was glued in place with IC, and testing was conducted

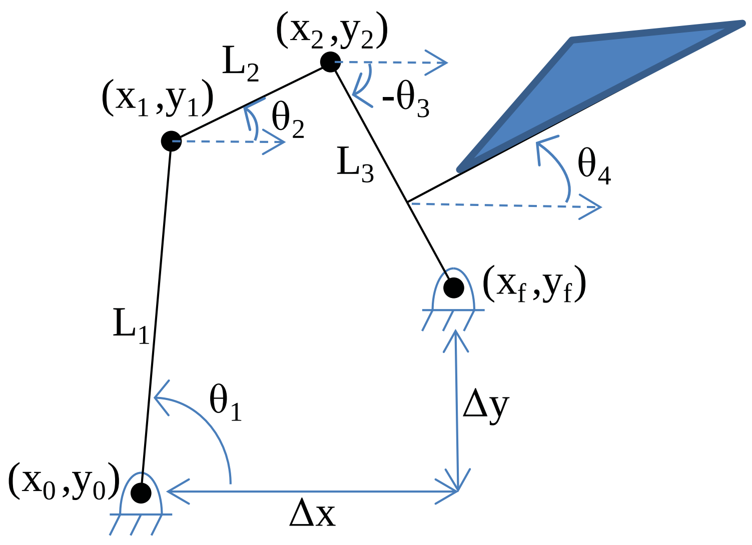


Figure 2.5: Four Bar Linkage [6].

on the newly attached wing. However, in his thesis, Sladek recommended further research into wing change out methods.

Anderson [4] sought to develop a method of providing control to the *Manduca Sexta* sized MAV. The work followed and extended the research of Doman, Oppenheimer, and Sigthorsson. A new control technique called Bi-harmonic Amplitude and Bias Modulation (BABM) was introduced and compared with other control techniques. Five degree of freedom control, using only two actuators, was shown to be possible. Pitch and yaw control with BABM were demonstrated using a constrained model. The BABM control method works by prescribing the wing stroke as a function of amplitude (A), split cycle parameter (τ), and bias (η). The wing rotation is passively controlled, and a constant wing angle of attack is assumed through each half stroke.

The effect of changing the various BABM control parameters was demonstrated by Carl [7]. In Figures 2.6, 2.7, and 2.8 the baseline drive signal is shown with a dashed blue line, while the modified signal is shown by the solid red line. Figure 2.6 shows how doubling the amplitude from one to two increases the peak amplitude linearly by a factor of two. The x-axis shows one flapping cycle, while the y-axis is the voltage applied to the drive signal. Figure 2.6 is only for illustrative purposes, the actual driving voltage for the flapping wing is significantly larger. The change of τ from zero to one-fourth is shown in Figure 2.7, again the time has been normalized to show one cycle, however, the drive signal has also been normalized to illustrate more clearly the effects of τ. The downward slope of the drive signal is decreased, and the upward slope is increased. While the flapping cycle period is kept constant, the intra-period duration of the downstroke compared with that of the upstroke is changed. The bias parameter shifts the drive signal away from the x-axis as seen in Figure 2.8 where both time and drive signal have been normalized. The bias is used to shift the center of pressure (average) of the wing, creating a pitching moment.

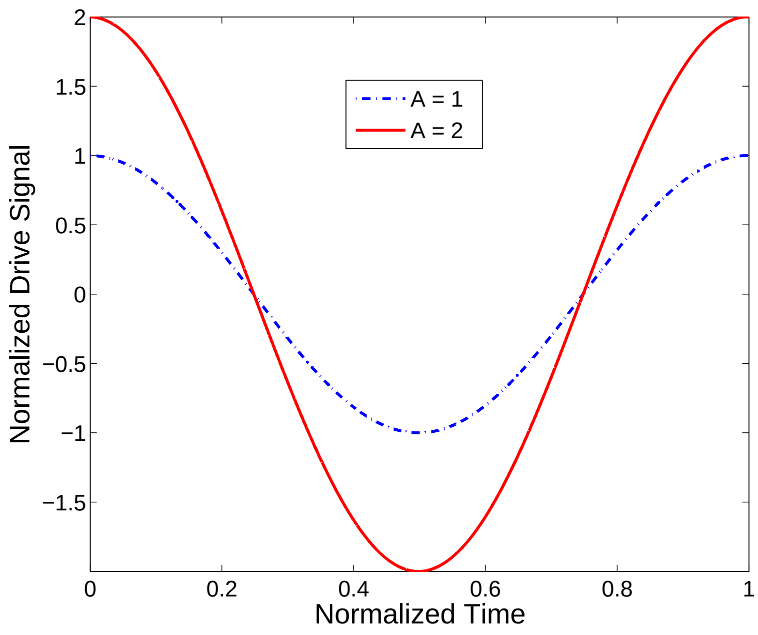


Figure 2.6: BABM Control Parameter: Amplitude,$\tau = 0$, $\eta = 0\ V$, Adapted from [7] Based on Research in [4].

Using the same CF four bar linkage as Sladek, Anderson was able to implement a quasi-steady blade-element based theory to accurately predict BABM control parameters.

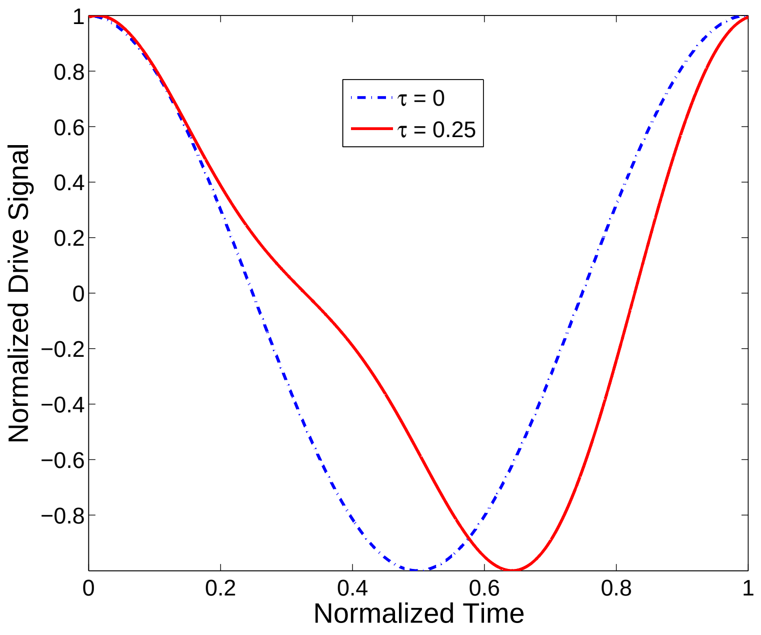


Figure 2.7: BABM Control Parameter: Split-Cycle Parameter, $A = 1\ V$, $\eta = 0\ V$, Adapted From [7] Based on Research in [4].

Wing design was beyond the scope of his research, and he relied on a biomimetically inspired design since wings were still required for testing.

Norris, Palazotto, and Cobb [26] sought to characterize the structural dynamics of the *Manduca Sexta* forewing. Several traits of different insects were examined to find their suitability for use in developing a FWMAV. The wing was settled as the "only true common attribute across them [flying insects] [26]." Furthermore, the wing was found to be common across flying insects in that the rotation is controlled passively, which differs from birds and bats which can "actively control wing shape through muscular flexure and actuation of joints [26]." The Hawkmoth wings were tested using a scanning laser vibrometer to determine their natural frequencies and modeshapes. These tests were conducted in both quiescent air and in a vacuum to determine the effects of air mass on the results. Modal ratios for the Hawkmoth were found to be normally distributed as shown in Figure 2.9

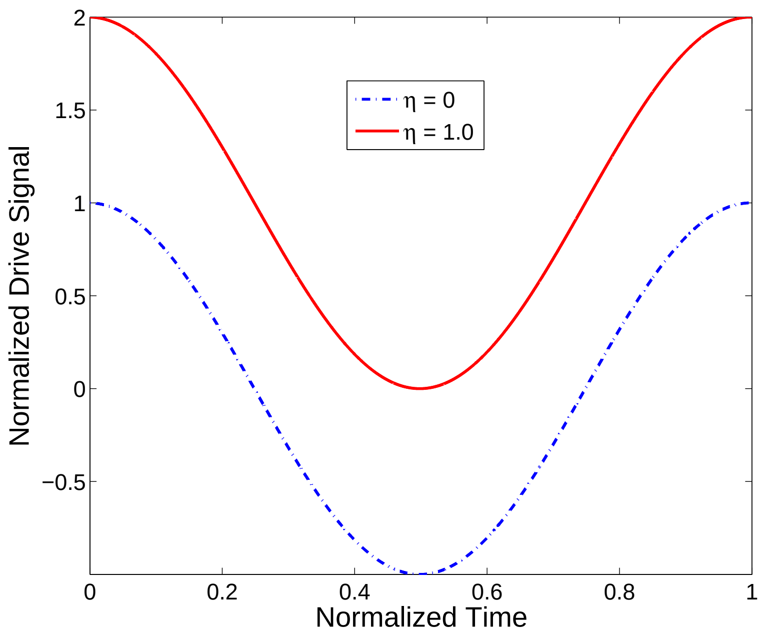
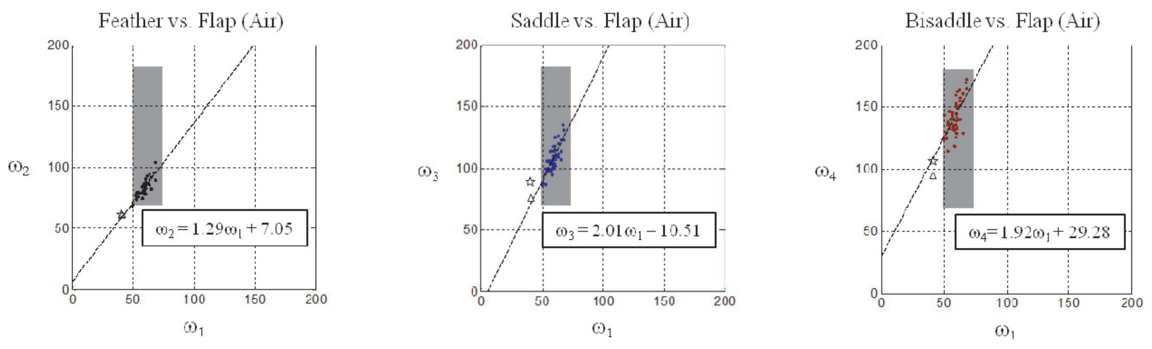


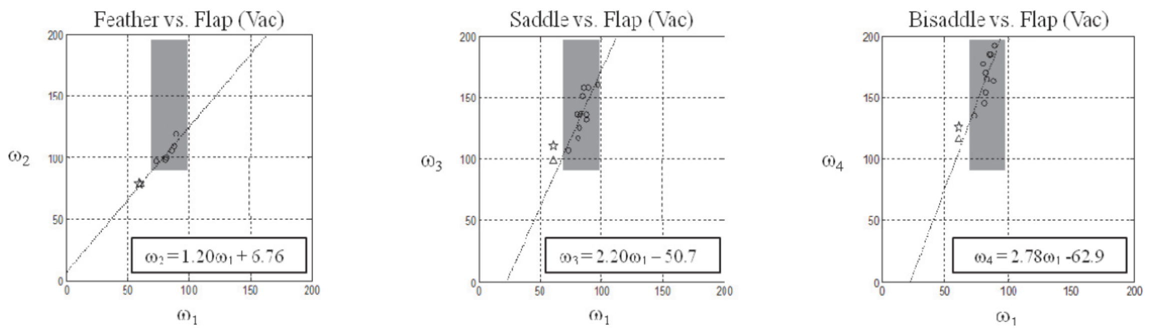
Figure 2.8: BABM Control Parameter: Stroke Bias, $A = 0\ V$, $\tau = 0$, Adapted From [7] Based on Research in [4].

which plots the “feather, saddle, and bisaddle modes [26]” against the flap mode in air (Figure 2.9a) and vacuum (Figure 2.9b).

To alleviate any possibility that placing the wings in a vacuum would cause damage through a “drying or brittling effect”, some of the wings were retested in air following the vacuum test [26]. No difference was observed between the results produced from testing in air before and after the vacuum test, and they concluded that the vacuum testing did not have any appreciable negative effects. In addition, an “age sensitivity” test was conducted, and they concluded the results were minimally affected (<3%) on tests conducted less than one hour after wing removal from the Hawkmoth. Tests were also conducted on a Monarch, Swallowtail, and Skipper butterfly where the modeshapes were found to be identical with the Hawkmoth, lending more credence to the assertion that the wing modal ratios, regardless of planform shape, are a common design across uncommon insect species.



(a) Air.



(b) Vacuum.

Figure 2.9: Ratio of Hawkmoth Feather, Saddle, and Bisaddle to Flap Mode [26].

O'Hara [28] undertook a morphological study of the *Manduca Sexta* to provide an accurate model for the biomimetically engineered FWMAV. Cataloging the individual properties of the Hawkmoth led to a suggestive proportioning of FWMAV components shown in Table 2.1. The losses due to dissection of the Hawkmoth are only 3% of the total mass of the specimen.

Table 2.1: *Manduca Sexta* Properties and Engineered Components, Adapted from [28].

Component	Analog	Average Mass (mg)	Portion of Total Mass
Entire Hawkmoth	Total MAV	1553	100
Forewing	Wings	34.6	2.23
Hindwing	None	12.2	0.78
Head	CPU	106	6.81
Thorax	Driving Mechanism	584	37.6
Abdomen	Payload/Power Gen	722	46.47

The wings were found to have an average length (R) of 45-55 mm, planform area (S) of 715 mm^2, and an aspect ratio ($\mathcal{R}$) of 14-15. The inner and outer diameters of the veins were determined by slicing the specimens at 2.5 mm intervals as shown in Figure 2.10 resulting in Figure 2.11.

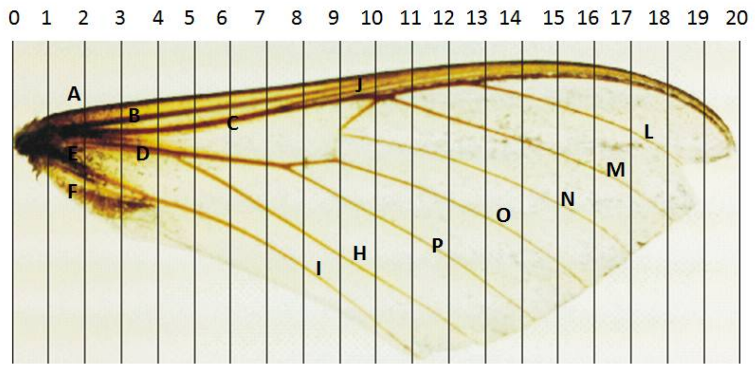


Figure 2.10: Sectioning Locations of *Manduca Sexta* [28].

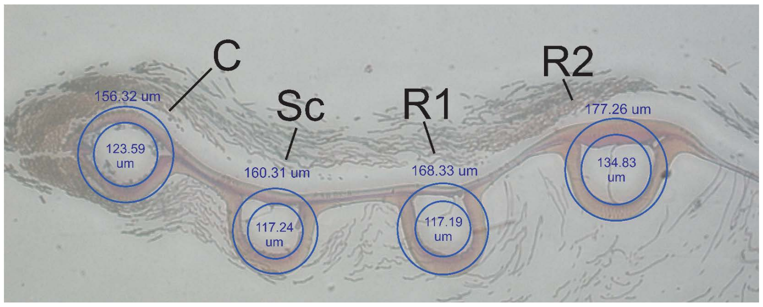


Figure 2.11: Vein Diameter Measurements of *Manduca Sexta* [28].

A sketch of the *Manduca Sexta* forewing was created in Solidworks 3D CAD, and from there an Initial Graphics Exchange Specification (IGES) file was exported. The IGES file was imported to Abaqus Finite Elements Program. In Abaqus, a 3D model was created from the sketch using Timoshenko B32 beam elements for the venation and S8R/STRI65 shell elements for the membrane. A clamped boundary condition was used at the root of the wing. MATLAB® was used to iterate on the model after Abaqus had created the input file in addition to calling Abaqus from the command line to identify the modal (eigenvalues / vectors) properties of the wing. The process is shown in Figure 2.12. The data from the FEA were compared with the experimental data from the biological wing. The MATLAB® iteration continued until the engineered wing structural response matched that of the biological wing.

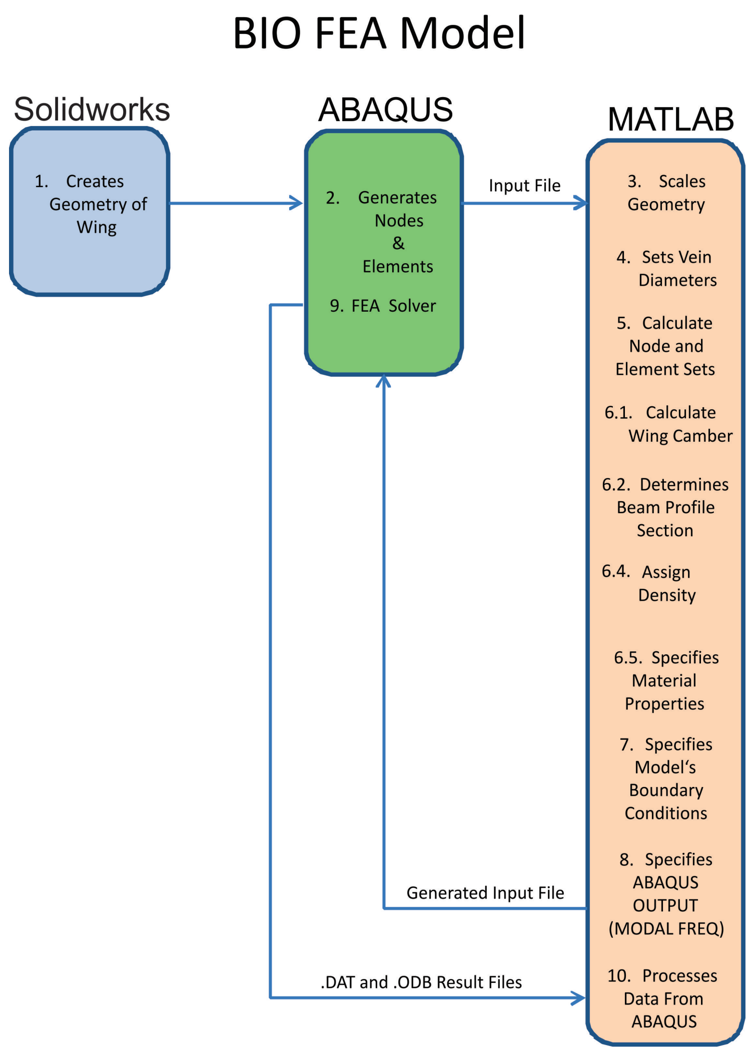

BIO FEA Model
Solidworks
1. Creates Geometry of Wing
ABAQUS
2. Generates Nodes & Elements
9. FEA Solver
Input File
MATLAB
3. Scales Geometry
4. Sets Vein Diameters
5. Calculate Node and Element Sets
6.1. Calculate Wing Camber
6.2. Determines Beam Profile Section
6.4. Assign Density
6.5. Specifies Material Properties
7. Specifies Model's Boundary Conditions
8. Specifies ABAQUS OUTPUT (MODAL FREQ)
10. Processes Data From ABAQUS
Generated Input File
.DAT and .ODB Result Files

Figure 2.12: Finite Element Wing Property Identification Loop [28].

Having obtained a matched response, the focus shifted to the manufacture of an actual engineered wing. Several materials were investigated for the venation. Stainless steel and titanium were found to have a low stiffness to density ratio, however, they were significantly heavier than the biological equivalents. Nitinol was considered for active camber control, but was too heavy. FullCure 720 and 840 from the Objet Geometries Ltd. Eden 500V, a 3D rapid prototyping machine, were able to print repeatable high quality representations, however the elastic modulus of the different materials was not significant enough to produce different stiffness from one feature to the next.

The final material investigated was a sample of YSH-50A carbon fiber obtained from the Harvard Microrobotics Laboratory. Upon further investigation, it was determined this material was no longer produced by the manufacturer, and YSH-70A was used instead. O'Hara found the properties of YSH-70A were comparable, and suitable for the engineered wing. Differences between the materials are shown in Table 2.2. Overall the CF provided a dramatic mass reduction (approximately 100 milligrams) compared with the previously investigated metals.

Table 2.2: Differences Between Carbon Fiber Materials [28].

Material	Tensile Modulus	Tensile Strength	Density	Fiber Dia.	Yield
YSH-50A	520 GPa	3.9 GPa	$2.10\,\mathrm{g\,cm^{-3}}$	$10\,\mu\mathrm{m}$	$30/1000\,\mathrm{g\,m^{-1}}$
YSH-70A	720 GPa	3.6 GPa	$2.14\,\mathrm{g\,cm^{-3}}$	$7\,\mu\mathrm{m}$	$75/1000\,\mathrm{g\,m^{-1}}$

A three ply laminate with fibers oriented in a 0°-90°-0°layup as shown in Figure 2.13 was determined to provide an appropriate balance between strength, weight, and flexibility. A notable improvement from the construction methods of Sladek [29] and Anderson [4] was the use of a newly purchased LPKF Multipress S pressure plate system for curing instead of the autoclave. This change in the curing process allowed for more repeatability in wing production. The venation patterns of the Hawkmoth were kept; however, it is important to note that they are a 2-D structure extruded to be 3-D; while the Hawkmoth veins are 3-D tapering cylinders.

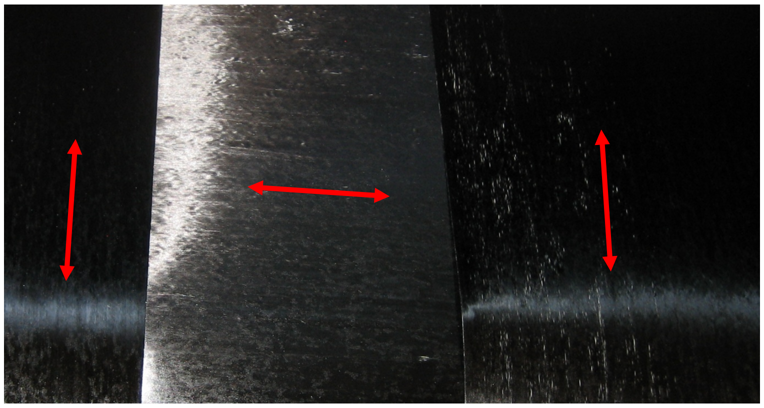


Figure 2.13: Direction of Primary Fibers in Three Layer Carbon Fiber Layup.

To further reduce mass, the membrane was switched from Kapton® to Mylar® construction. Mylar® is available in a wide range of thickness and a 2.5 μm was chosen. The Mylar® provided a weight savings of 18.5 mg from the Kapton®. In addition the heat shrink abilities of the Mylar® are well suited for adherence to the wing venation structure without the application of separate adhesive. The results of the final engineered wing are shown in Figure 2.14.

After creation of the engineered wing, the FEA model was updated from a tubular structure to composite laminate, and finally to I-Beam elements as shown in Figure 2.15. A 1 mg point load at the leading and trailing edge tip nodes was used to conduct an explicit analysis. These settings ensured the FEA model of engineered wing matched as closely as possible to the production engineered wing.

In controlled laboratory experiments, the engineered wing produced comparable lift, more efficiently than its biological analog when operated with 45° angle stops [28]. The engineered wing had a Lift-to-Power ratio almost 30% greater than the biological wing

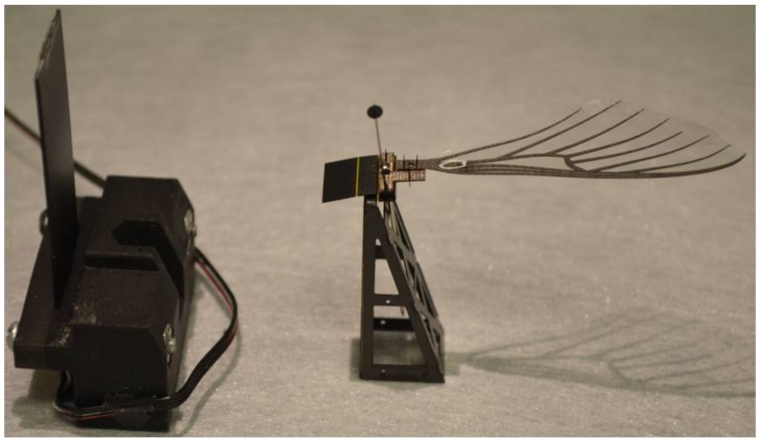


Figure 2.14: Biomimetic Engineered Wing Created by O'Hara [28].

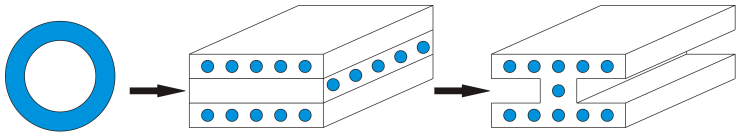


Figure 2.15: Evolution of FEA Elements [28].

when measured without scales[1]. Table 2.3 shows the respective biological and engineered wing properties. The morphological study included two variations of the biological wing specimens. The first was with the liberated wing including the scales. Those scales were then removed to determine what affect, if any, they had on lift production and power consumption. As seen in the Table 2.3, the lift was slightly reduced, while power consumption decreased by 0.8 mW with the scales removed.

In an effort to characterize the aerodynamic performance of the AFIT biomimetic wing and drive system, DeLuca [12] conducted six DOF force balance testing using a ATI Nano-17 Titanium force transducer and phase averaged stereo Particle Image Velocimetry (PIV). The investigation used the same wing design as O'Hara, while delving into the details of the

[1]The *Manduca Sexta* wings are covered in 'scales', which can be removed by lightly brushing the liberated wings [28].

Table 2.3: Biological and Engineered Wing Properties [28]

	Bio W/Scales	Bio W/O Scales	Eng
Lift (mgF)	1042.3	969.8	893.3
Power (mW)	3.9	3.1	2.2
Lift to Power (mgF/W)	0.267	0.312	0.401

fluid dynamics and effect of the passive angle stop, which directly relates the wing's AOA with respect to the relative wind. Since the flapping wing rotation is passively controlled, some mechanism to limit that rotation is required. Angle stops are the solution to limit that travel. Figure 3.21b shows the angle stops installed on the engineered wing. DeLuca explored several different angle stops in his research. The research determined that resonant flapping was necessary to generate the greatest displacement from the PZT, which led to the greatest lift generation developed by the engineered wings. Furthermore, the 60° angle stop produced the highest velocities above and below the wing, which contributed significantly to lift generation. In other words, neglecting aeroelastic effects and mechanical slop, the wing would travel with an angle of 60° relative to the x-y plane (where x is in a chordwise direction, and y is in the spanwise direction) of the wing. Issues with delamination of the Mylar® from the wing venation presented during his research, providing an area for further improvement.

2.4 Chapter Summary

This chapter gave a brief primer on fundamental the mechanics of flapping flight necessary to understand the remainder of the thesis. A short introduction to FEA was discussed. A significant portion of this chapter is dedicated to providing the history of previous FWMAV research at AFIT. The context of the present research and how it fits into the greater collective of AFIT FWMAV research is discussed to enhance understanding of the experimental methods presented in Chapter III.

III. Methodology

3.1 Wing Finite Element Analysis

THE primary goal of this research is to investigate approaches to wing weight reduction while maintaining or increasing vertical force. Using the same convention as previous research [4, 12, 28], lift and vertical force are defined along the positive x-axis of Figure 3.1.

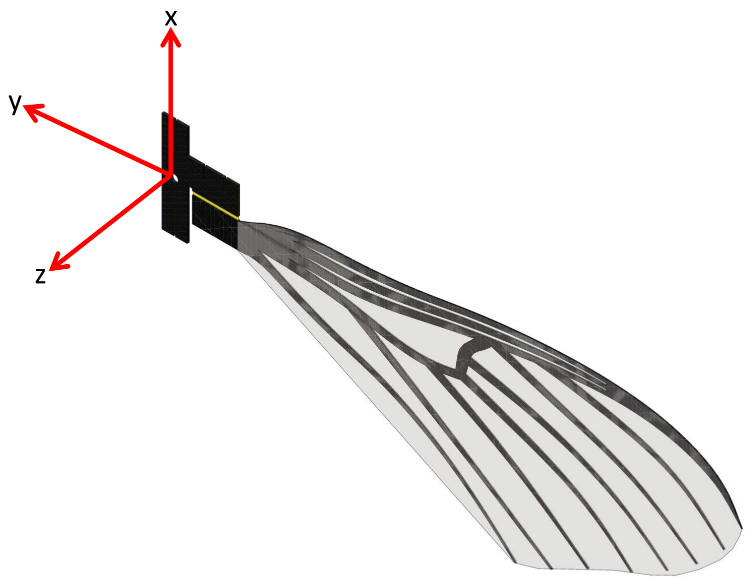


Figure 3.1: Axis System for the AFIT MAV Wing.

Under the assumption of resonant flapping [4], the system's[2] flapping frequency is based on the square root of the system stiffness to system mass ratio and is approximated in Equation 3.1. To increase the flapping frequency, it is critical for the mass of the wing to decrease more quickly than the system stiffness. As the mass-to-stiffness ratio is reduced, the flapping frequency is increased, and the vertical force produced should increase.

[2]The system refers to all components attached directly to the wing including the wing itself, hinge, angle stops, linkages, and PZT. The wing resonant frequency is much higher than the system's resonant frequency.

$$\omega_n \propto \sqrt{\frac{k}{m}} \tag{3.1}$$

To remove wing mass, a natural starting place is to use FEA to identify regions of low stress on the wing structure and remove material from those locations. The primary goal of this limited FEA study was to determine if wings would break when produced.

3.1.1 Solidworks.

The first step was to take the existing wing design and get it in a usable format. The design existed in a Drawing Interchange Format (DXF) file for use in CorelDraw. This file was imported into Solidworks as a 2-D sketch. The sketch was converted to splines, and any open constructions or misinterpreted splines were corrected, while excess lines were removed. The 2-D sketch of the wing shape exists in a Solidworks Part Format (SLDPRT) shown in Figure 3.2.

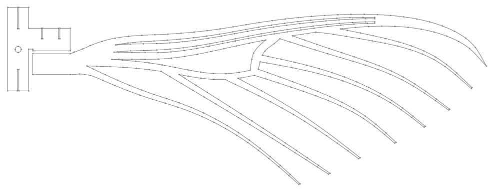


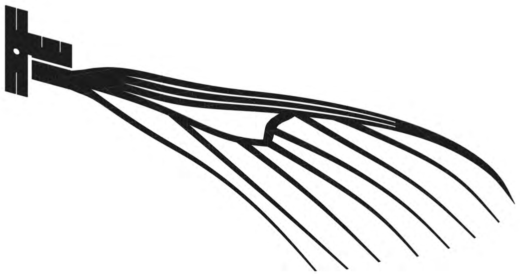
Figure 3.2: Solidworks Imported Sketch.

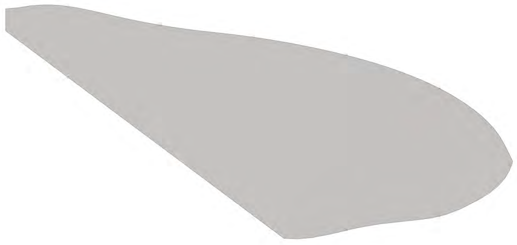
The sketch is copied to three other SLDPRT files. The first of the copied sketches is transformed to match the shape of the Kapton® layer for the wing. The second copied sketch is adjusted into the covering layer of CF. A spline is created around the outside of the wing shape on the third and final sketch. The spline forms the shape of the Mylar® wing membrane. Each of the four total SLDPRT files forms one layer of the wing structure.

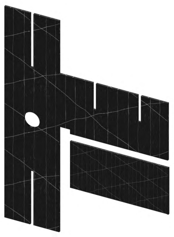
The four layers are extruded to the thickness shown in Table 3.1. Figure 3.3a shows the wing carbon fiber layer. Figure 3.3b shows the Mylar® layer. Figures 3.3c and 3.3d show the covering carbon fiber layer and Kapton® layer respectively.

Table 3.1: Extrusion Thickness for Individual Layers.

Layer	Thickness	Units
Wing CF	0.1525	mm
Mylar®	0.0025	mm
Covering CF	0.1525	mm
Kapton®	0.025	mm


(a) Wing CF.


(b) Mylar®.


(c) Covering CF.

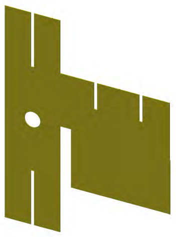
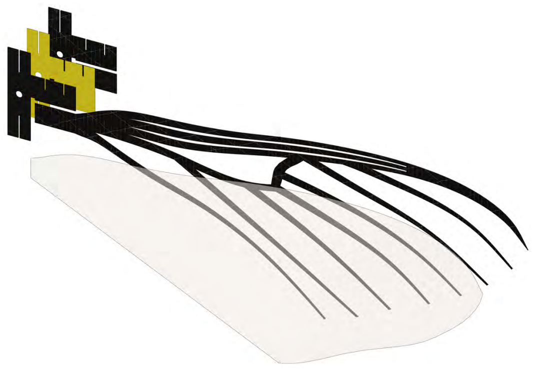

(d) Kapton®.

Figure 3.3: Individual Wing Layers.

The layers are shown assembled in Figure 3.4 for illustrative purposes only. The layers are exported to IGES files individually.


(a) Exploded View.

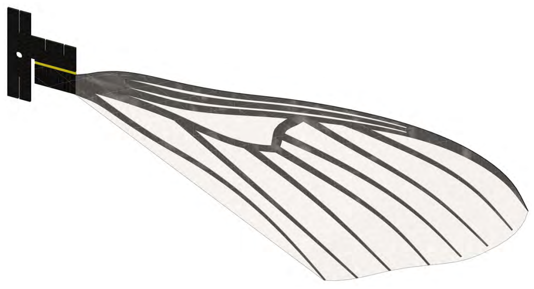

(b) Collapsed View.

Figure 3.4: Solidworks Assembled Wing Layers.

3.1.2 Abaqus.

The individual IGES files were imported as individual parts into Abaqus. The next step was to create the individual layer materials and associate their respective properties. The material properties are obtained from various locations and are summarized in Table 3.2. The Mylar® [16] properties are obtained directly from manufacturer distributed information. The carbon fiber properties were defined by O'Hara [28, Section 5.1.4]. Each part is assigned the appropriate section of material.

Table 3.2: Individual Material Properties.

Material	Density (g/cm^3)	Young's Modulus (GPa)	Poisson's ratio
Carbon Fiber	2.14	520	0.25
Mylar®	1.38	3.7	0.38

Next, each individual part is meshed as shown in Table 3.3. The mesh is generated automatically by Abaqus. The number of elements varies with the surface area of each wing revision, and those shown in Table 3.3 are for Revision 0. The mesh for the wing carbon fiber is significantly more refined than that of the Mylar®. Since the Mylar® is a constant throughout the different designs, the important stress information occurs on the wing venation.

Table 3.3: Element Type and Number for Individual Part Mesh.

Part	Type	Number
Wing CF	C3D4	27302
Mylar®	SC8R	1403
Mylar®	SC6R	40

A copy, or instance as it is known in Abaqus, of each part is then created to form the beginnings of an assembly. This method ensures that the original part remains separate from the instance of the part used in the assembly and allows changing the properties of the part without affecting previously designed parts. The parts are assembled as shown in Figure 3.4. Position constraints are added to ensure proper part alignment. Tie constraints are added to act as a bond between the surfaces of the individual layers. After the layers are tied together and constrained the result is a wing model that appears as shown in Figure 3.5. Note that the complexity has been reduced from the solidworks model for solution by Abaqus. The CF covering layer and Kapton® layer have been removed to simplify the model.

The model is now constrained in a clamped boundary configuration in a similar method to that described by O'Hara [28]. A point load of 500 mgF is applied on the leading edge of the wing, which is similar to the methodology O'Hara used in his analysis of the biological and engineered wing specimens. While the point load does not accurately model the true pressure distribution on the wing during flapping, it is used under the assumption that the accuracy is high enough to identify those regions of high and low stress. The loaded and constrained model is then submitted to Abaqus CAE for analysis. The results of the analysis are saved to an OpenDocument Database (ODB) file, which can be analyzed using

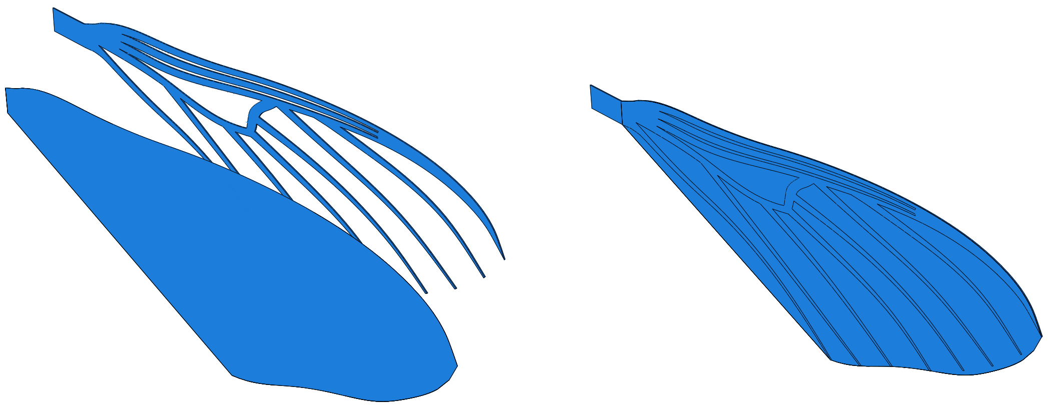

(a) Exploded View. (b) Collapsed View.

Figure 3.5: Abaqus Assembled Wing Layers.

the visualization tool integrated within the Abaqus environment. Viewing these results allows for identification of stress locations and their values.

3.1.3 FEA Design Update Process.

After the results are reviewed, the initial Solidworks model is modified by removing structure from the wing CF layer. The remaining three layers do not change during design iterations. The updated SLDPRT is again exported to an IGES file so that it can be imported into Abaqus. The new wing design is imported and the wing CF layer is replaced with the updated design. The analysis is re-accomplished with the new design and the locations of maximum and minimum stress are identified. Figure 3.6 illustrates a sample set of results. The contour scale indicates the maximum stress areas at the top (red) and the minimal stress areas at the bottom (blue). In the example set of results, some stress can be seen throughout the leading edge venation, while most of the stress resides at the attachment point. The process continued removing mass from the structure (in the low stress areas) to isolate different regions of interest on the wing. Seven total wing designs (including the original) were created, those designs were analyzed using FEA to ensure that a critically stressed location had not been added. These seven designs were then built and experimentally tested.

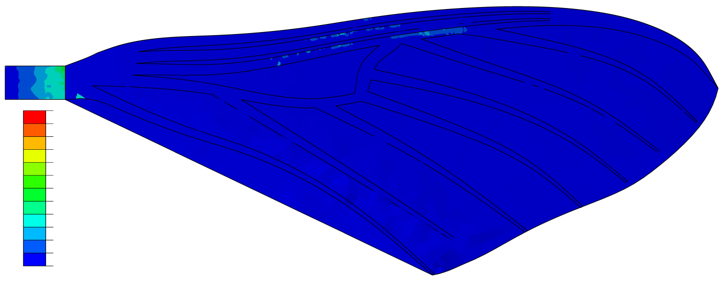
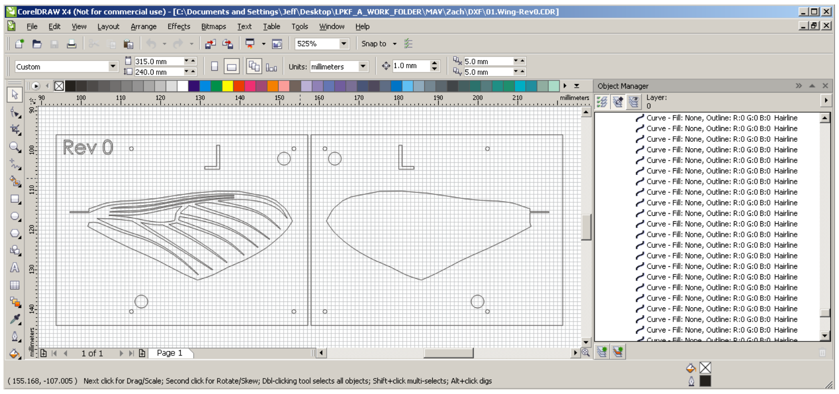


Figure 3.6: Sample Results from FEA.

3.2 Wing Construction

3.2.1 Cutout and Pressing.

The individual wing designs were exported from Solidworks back into a DXF file. The DXF file is imported into CorelDraw where the drawing was manipulated if needed for cutting on the LPKF Proto Laser U (PLU). This process consists of creating openings where the Kapton® layer will act as flexure joints and changing the spline thickness to hairline[3]. Once the first model was completed the updated file was used as the basis for future designs. The results of this manipulation are shown in Figure 3.7.



```
CorelDRAW X4 (Not for commercial use) - [C:\Documents and Settings\Jeff\Desktop\LPKF_A_WORK_FOLDER\MAV\Zach\DXF\01.Wing-Rev0.CDR]
```

```
Custom   315.0 mm   240.0 mm   Units: millimeters   1.0 mm   5.0 mm   5.0 mm
```

```
Object Manager
Layer:
0
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
Curve - Fill: None, Outline: R:0 G:0 B:0  Hairline
```

```
( 155.168, -107.005 )  Next click for Drag/Scale; Second click for Rotate/Skew; Dbl-clicking tool selects all objects; Shift+click multi-selects; Alt+click digs
```

Figure 3.7: CorelDraw Sketch.

After correcting any discrepancies (such as open contours) in CorelDraw, the DXF file is imported to CircuitCAM used for computer-aided manufacturing. For subsequent designs using the newly baselined DXF file a step was removed from the process. For the updated design the DXF file was imported directly from Solidworks into CircuitCAM bypassing CorelDraw altogether.

In CircuitCAM the wings are placed on a contour layer and the Laser Path Scanning Tool is run. This uses the known configuration of the PLU to divide the wing into sections that correspond to the size that the PLU can cut in one pass before needing to move the

[3]Hairline refers to the thinnest possible line that can be represented on the output device. In this case, the hairline is a one pixel line since the output "device" is a file

material. The CircuitCAM drawing is then exported to CircuitMaster which is the software that interfaces directly with the PLU. The tools are assigned in CircuitMaster depending on the type of material to be cut. O'Hara [28, Table 22] summarizes the laser settings for several common material types.

The three-layer 0°-90°-0° carbon fiber layup was constructed in accordance with the methodology described by O'Hara [28, Section 6.5.2] and is laid flat on the PLU as seen in Figure 3.8. The nonporous Teflon rectangles are applied around the edges of the carbon fiber to act as a vacuum seal. This helps keep the CF flat on the PLU table during cutting.

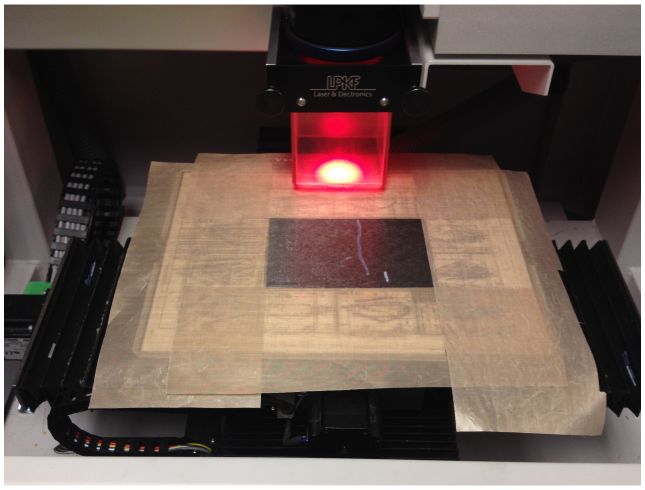


Figure 3.8: Carbon Fiber on PLU.

The first set of cuts are made to cut the top and bottom layers of CF. This cuts the outline of the wing and the wing venation in two separate rectangles as shown in Figure 3.9. The venation and outline are shown after separation from the surrounding material in Figure 3.10. Next, a Kapton® layer is cut on the PLU as shown in Figure 3.11. The Kapton® layer is placed between the two CF layers. Figure 3.12 shows the Kapton® placed on the vein side of the assembly. The wrinkles in the Kapton® are smoothed out and the outline layer is glued into place as shown in Figure 3.13 creating a Carbon-Kapton-

Carbon (CKC) sandwich. The glue used is a simple Elmer's glue stick and serves only to tack the three layers together in preparation for pressing.

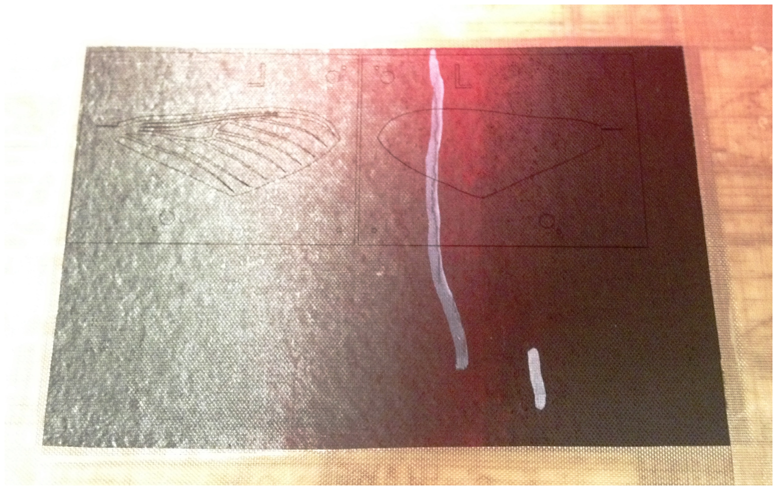
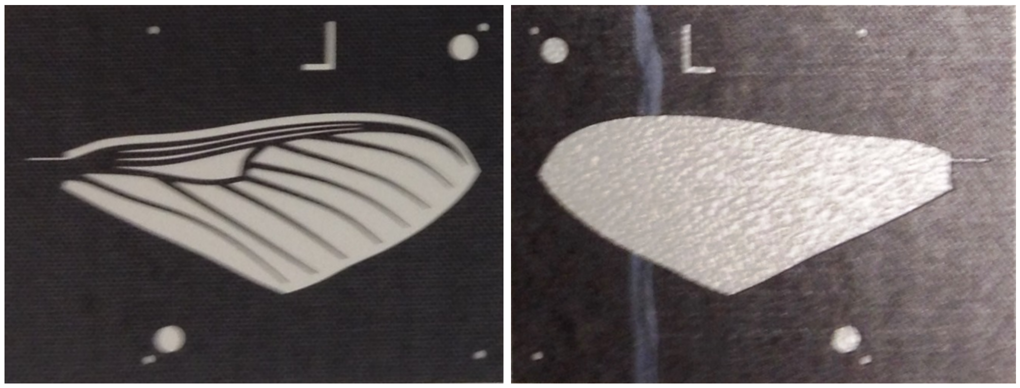


Figure 3.9: Carbon Fiber Cutout on PLU.



(a) Wing Venation. (b) Wing Outline.

Figure 3.10: Carbon Fiber Cutout Separated from Surroundings.

The CKC layup is placed facedown on 2.5 micron Mylar® film. The Mylar® film is shown by the blue arrow in Figure 3.14. The Mylar® is the same diameter as the white

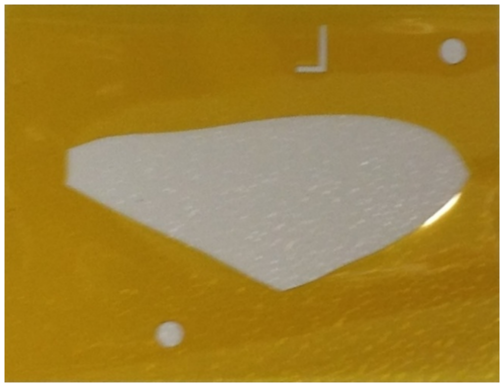


Figure 3.11: Kapton® Layer.

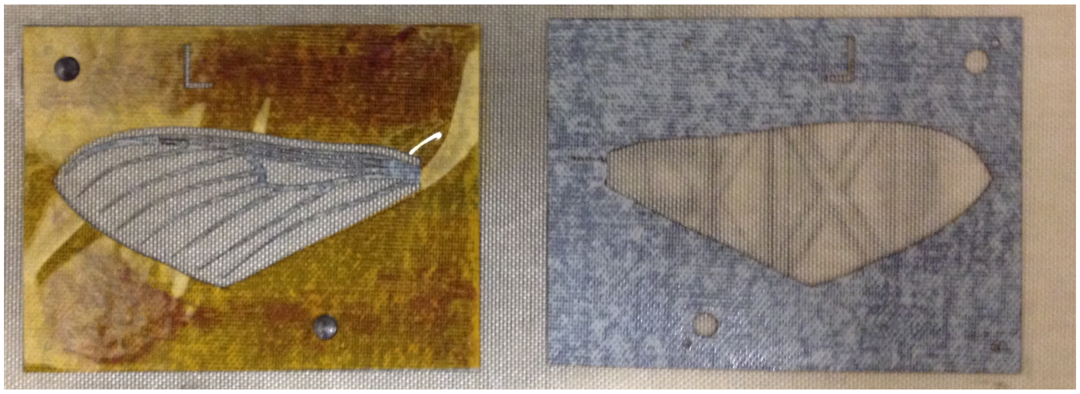


Figure 3.12: Partial CKC Layup.

circle just above it. The white circle is the protective cloth for the Mylar® and removed before pressing.

Figure 3.15 shows the layout of materials used in the LPKF Multipress S. The other half of the layout is a mirror image of these layers with the material to be pressed in the middle. The layers, working from the middle layer (closest to the wing) outward indicated by the numbers in the upper right of Figure 3.15 are as follows:

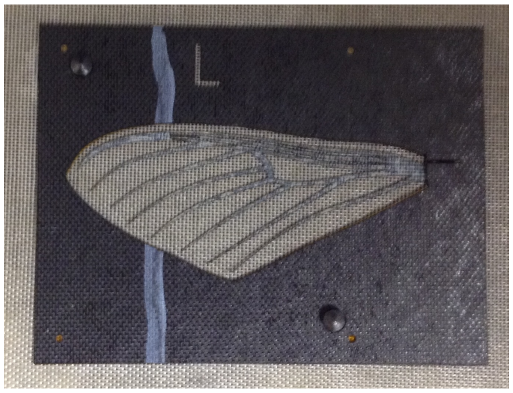


Figure 3.13: Carbon-Kapton®-Carbon Layup.

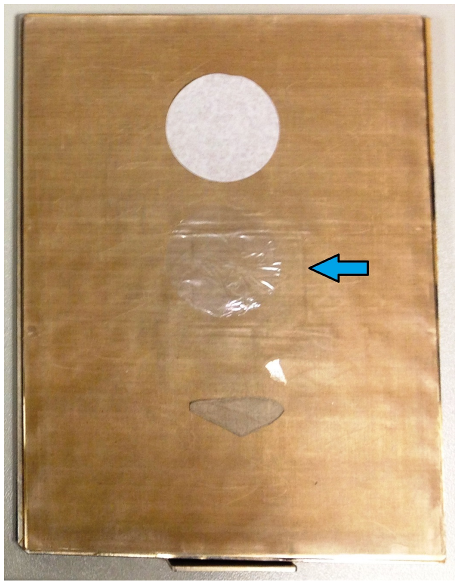


Figure 3.14: Mylar® Film Laid Flat on Press Plate.

1. Non-porous Teflon
2. Copper Backer Plate
3. Fiber Board Backer Plate
4. Aluminum Backer Plate
5. Cloth Insulative Layer

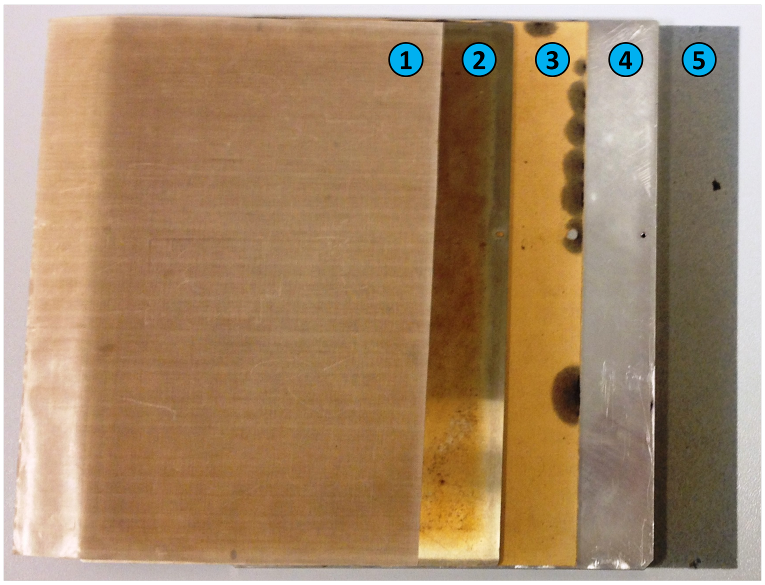


Figure 3.15: Press Plate Layout.

The Mylar® bonds with the wing venation to form the skin of the wing. This layup is then pressed in the LPKF Multipress S at 100 N/cm^2 and 192°C. This process takes approximately 1.75 hours which includes 1 minute of prepressing, 60 minutes of pressing, and 45 minutes of cooldown. The results are shown in Figure 3.16.

The newly pressed wing card is placed back in the PLU to be cutout from the surroundings. The card is placed on 1.5 mm alignment pins set in the PLU backer board to maintain the proper position while cutting. Figure 3.17 shows a closeup of the wing card placed on the PLU (viewed from the front) with the alignment pins circled in blue.

The resulting wing is liberated from the CKC surroundings shown in Figure 3.18. When the PLU does not cut all the way through the CKC layers, an X-Acto knife is gently used along with a dental pick to eliminate the remaining material using light pressure. There is always excess Mylar® covering a portion of the flexible joints which must also be removed. The excess CF and Mylar® to the left of the red dashed line in Figure 3.19 are

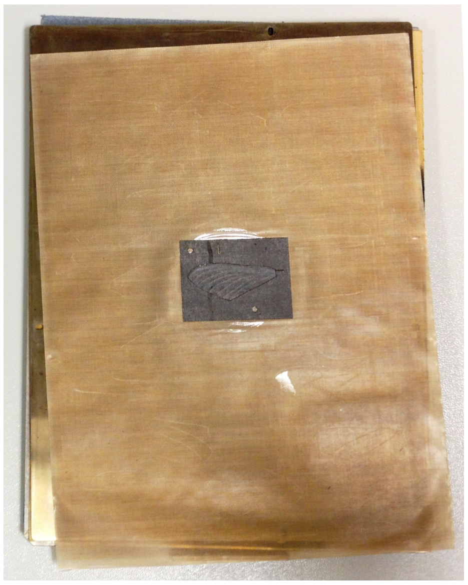


Figure 3.16: Wing Card After Pressing.

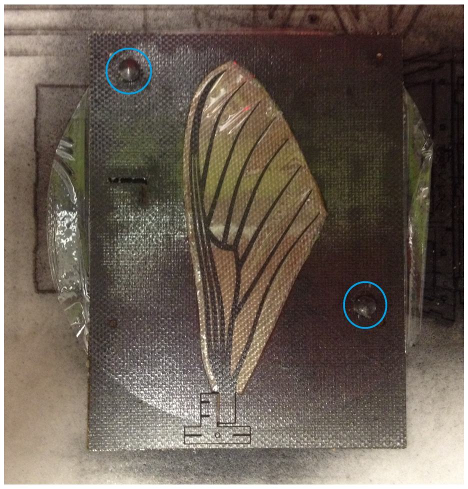


Figure 3.17: Wing Card Aligned on PLU For Cutout.

removed with an X-Acto knife. The liberated wing shown in Figure 3.20 is now ready for attachment to the base and experimental testing.

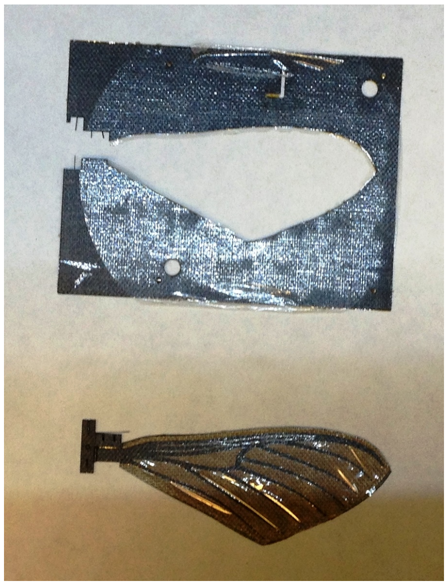


Figure 3.18: Wing Separated from Surroundings.

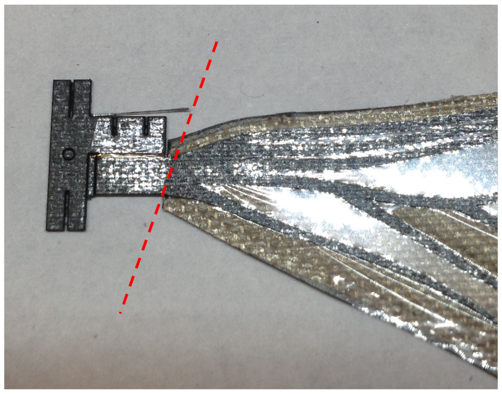


Figure 3.19: Excess Mylar® to be Removed.

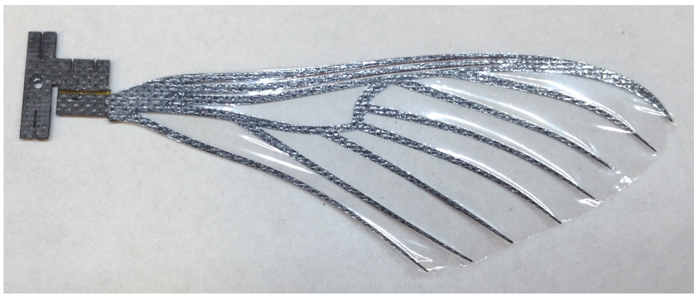
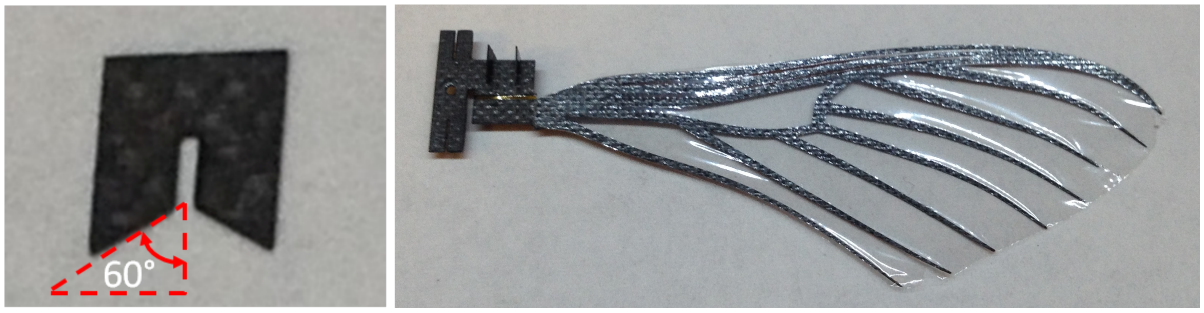


Figure 3.20: Fully Constructed Liberated Wing.

3.2.2 Angle Stop Attachment.

An example angle stop is shown in Figure 3.21a. Two stops are inserted as shown in Figure 3.21b, and glued into place with thin IC. The 60° angle stops were used in this research because they were found by DeLuca [12] to give the best aerodynamic results. Since these experiments are comparing efficiency of one wing design to the next, the 60° angle stops were used for all design revisions. The simplifying assumption that relative results between wing designs will be insensitive to angle stop is made, however, if the flexibility of the wing is significantly changed this assumption may need to be examined more closely.



(a) Single 60° Angle Stop.

(b) Two 60° Angle Stops Installed.

Figure 3.21: 60° Angle Stops.

3.2.3 Base Assembly.

O'Hara [28] provides a thorough description of the base assembly. The pertinent steps are repeated here. The process for cutting out and layering the individual base pieces is similar to that used in the wing construction. First, the sides of the base are folded at a 90° angle and supported in position between two metal blocks as shown in Figure 3.22.

Next, the front plate is snapped into position and bonded with IC. Alignment is controlled by the three teeth on either side of the front plate that slide into corresponding slots on the base. The second step is shown in Figure 3.23. Step three dictated by O'Hara,

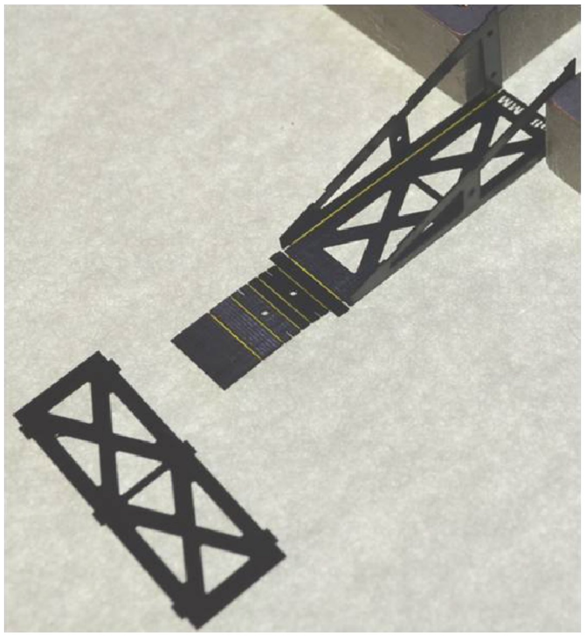


Figure 3.22: Fold Sides of Base [28].

in Figure 3.24, consists of folding the top over and gluing it to the base. Thick IC is used to ensure a secure bond.

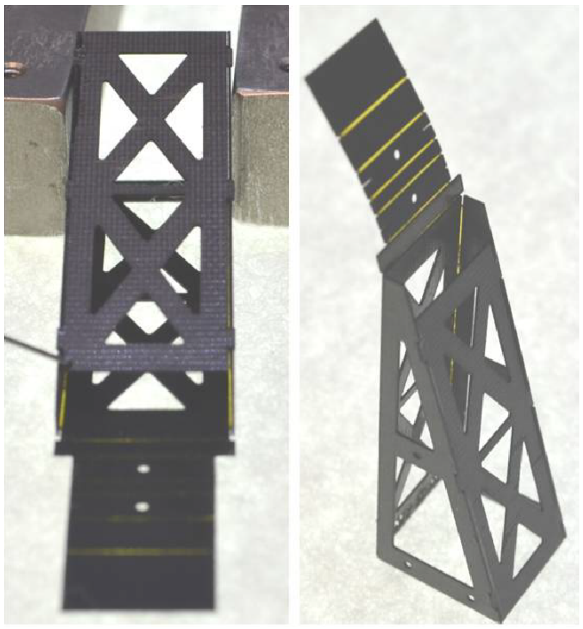


Figure 3.23: Glue Front Plate into Position [28].

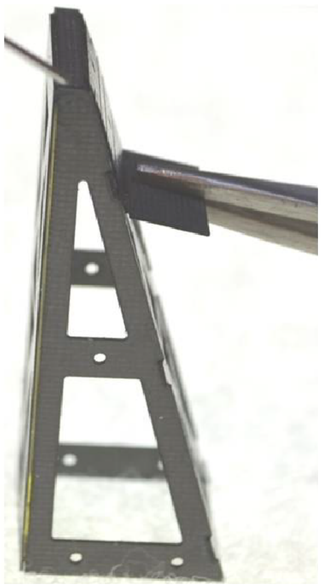


Figure 3.24: Glue Top to Base [28].

Figure 3.25 shows the top being folded back on top of itself to form a C-shape. The assembly is held in place by a pin. IC is placed on the internal portions of the folded Kapton® joints to provide stiffness.

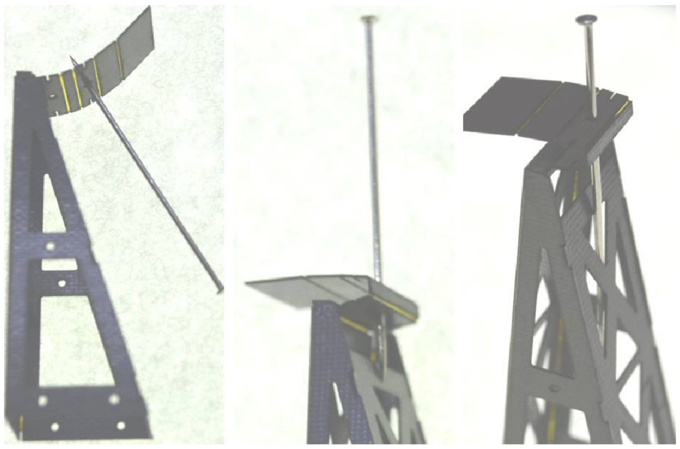


Figure 3.25: Fold Top in C-Shape [28].

Finally, the retaining E-clips are inserted into position. The E-clips are bonded to the base with IC. The large E-clip with a slot at the top shown in the left and right panels of Figure 3.26 was used for attachment of an optical tracking dot in O'Hara's experiments. In the current research both E-clips appear as in the middle panel of Figure 3.26 without the long tab attached as in the left- and right-hand images.

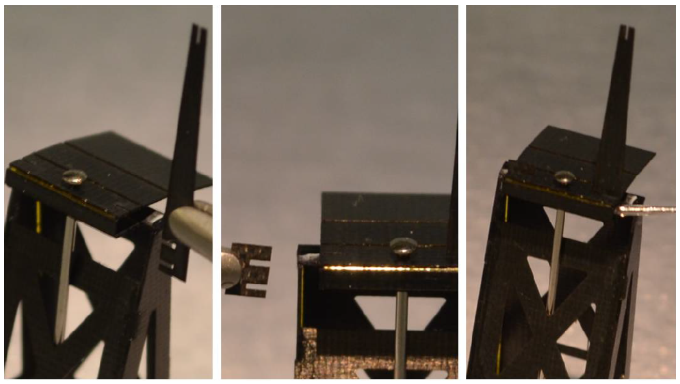


Figure 3.26: Glue E-Clips in Place [28].

3.2.4 PZT Attachment.

The base is attached to the mounting bracket which will connect to the test stand. The base is attached by sliding the open end onto the mounting bracket raised receptacle. The receptacle has two holes large enough for pins to slide in and secure the base to the mounting bracket. Figure 3.27 shows the front view of the base attached to the bracket with the pins inserted.

The PZT is attached by sliding it through the receiving slot on the the mounting bracket and tightening screws to clamp it in place. The rear view, shown in Figure 3.28, illustrates the mounting of the PZT. It is critical to adjust the height of the PZT so the top of the base and the top of the PZT lie in a plane parallel to the bottom of the mounting bracket. This is highlighted by the red dashed line Figure 3.29.

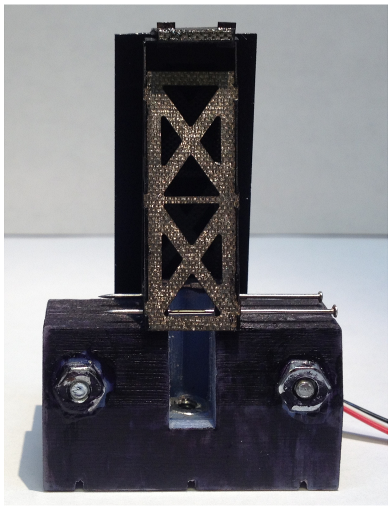


Figure 3.27: Front View of Base Attached to Mounting Bracket.

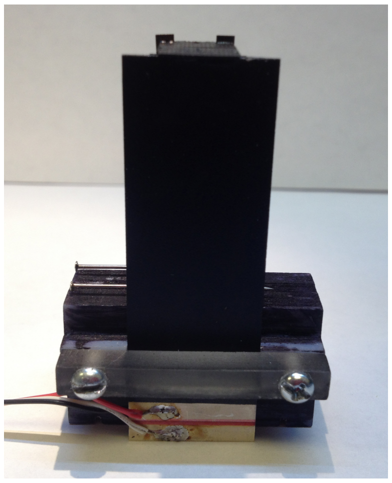


Figure 3.28: Rear View of PZT Attached to Mounting Bracket.

During this research, a new method for PZT attachment was developed leading to a significant improvement in the consistency of a test series when the wing is replaced multiple times during testing. The top of the base is attached to the PZT with thick IC at the location shown with the red dashed circle in Figure 3.30. This connection had previously been made with CrystalBond. Using CrystalBond on this critical joint allowed

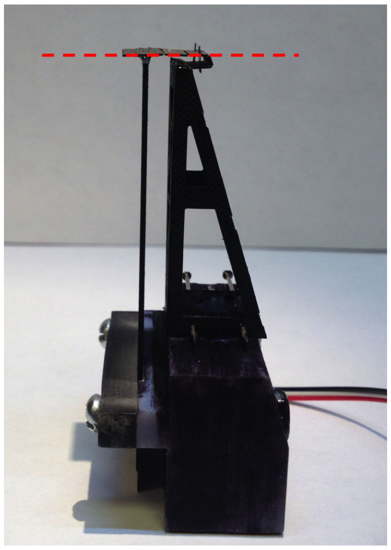


Figure 3.29: Required Plane for PZT Alignment.

for movement of the attachment when the wings were heated for removal. The thick IC is a similar consistency to the CrystalBond, however the bond is not weakened by the amount of heat used in wing removal. The original wing results with IC used to make the joint were consistent in shape with those where CrystalBond was used to make the connection, as seen in Figure 4.28. The alignment of this joint is critical since the thrust of this research is to compare the effects of wing design, if the alignment changes, the geometry of the four-bar-linkage changes which affects the results. The IC prevents this from changing during a set of tests where multiple wings are used.

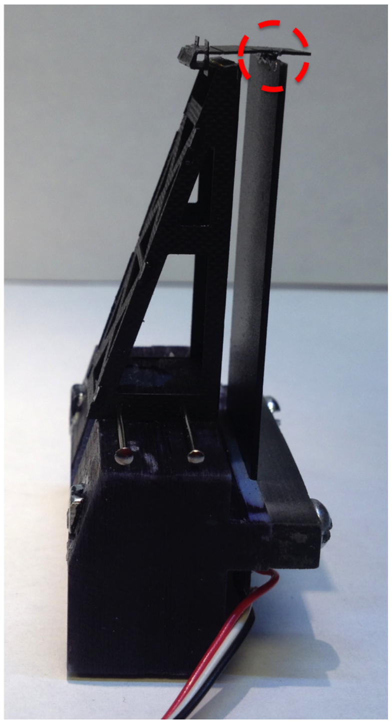


Figure 3.30: PZT Attachment to Base.

3.2.5 Base Attachment.

Once the wing and base are constructed, they must be assembled, and the preparation is given in Figure 3.31. As previously recommended [29], a new, improved wing/base attachment method was sought out. This new process, developed during the current research and described below, provides a significant gain in test consistency. CrystalBond adhesive is heated using a Weller WHA 900 heat gun, with an airflow setting of 0.5 (low) and heat setting of 10 (high). Once the CrystalBond is in liquid form[4], a small portion is applied to the two locations indicated in Figure 3.32. Extreme care was taken to avoid allowing any CrystalBond to flow into and foul the Kapton® flexures (left side of red circles in Figure 3.32) because this would ruin the capability of the flapping mechanism to have a consistent and smooth motion between up and down strokes. Once the CrystalBond returns

[4]The liquid form is very viscous, similar in consistency to the thick IC.

to room temperature (i.e. solid state), the wing is attached and ready for testing as shown in Figure 3.33. The solidification process takes approximately one minute.

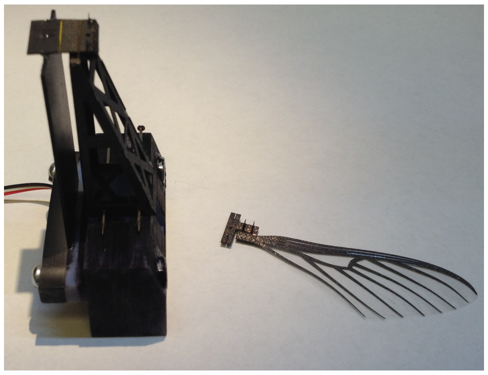


Figure 3.31: Wing Ready to be Attached to Base.

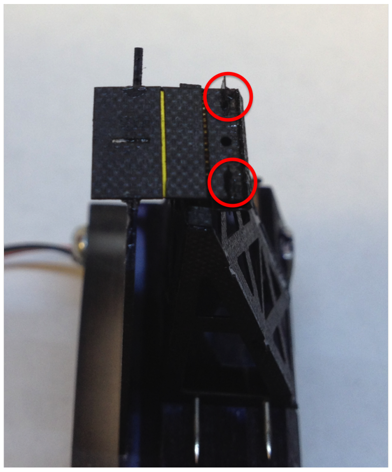


Figure 3.32: Locations for CrystalBond Adhesive.

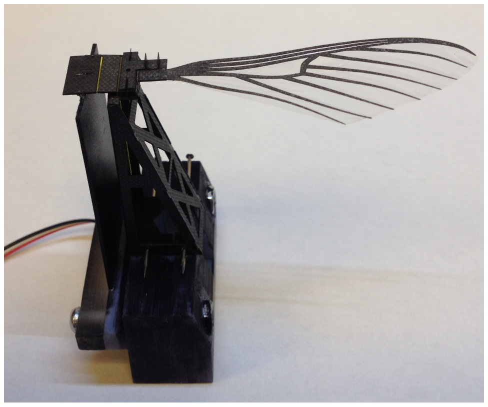


Figure 3.33: Wing Ready for Testing.

3.2.6 Wing Replacement.

The experimental setup is predicated upon testing multiple wings in succession. The changing of a wing must occur without a change to the configuration of the experiment. Using the newly developed process described in Section 3.2.5 enhances speed and accuracy of wing replacement and removes the risk of wing degradation that existed from sanding. An unintended, but nonetheless significant, benefit of the new process is that manufacturing requirements are decreased substantially. Previously, a new base would have to be manufactured for every wing tested due to the residue left by the IC on the attachment points. Since the new method allows for non-destructive wing replacement, one base and PZT can be used throughout a test sequence.

The wing CrystalBond attachment can be broken by heat allowing for multiple replacements of the wings while using the same base test stand. Previously, IC had been used to attach the wing to the base. The removal process required scraping and sanding the excess IC to remove it from the wing, and this process carried the inherent risk of removing carbon fiber material from the wing and/or the base. If the wing were to be subsequently tested with material removed, the properties are changed, and repeatability was compromised. If the CF were removed from the base, the geometry of the attachment would change, and subsequent wing revisions would be tested in a different configuration. The CrystalBond attachment carries the advantage of easy removal with no loss of CF from either the wing or base, leading to satisfactory repeatability.

To remove the wing, the CrystalBond locations shown in Figure 3.32 are heated using the settings previously identified for the heat gun. Once the CrystalBond is in liquid form, a pair of tweezers may be used to gently remove the wing as shown. Once the wing has been removed, the next wing design can be installed following the same process previously described. By using the same test stand attached to the same PZT construction, variations of both the geometry and PZT are removed as variables from the test sequence. The analysis from one test to another then only accounts for the differences between wing design, and not the support apparatus. High-speed video provided clear evidence the new approach to wing replacement was effective and repeatable. The wings that were not otherwise compromised exhibited the identical smooth flapping motion to their counterparts attached using legacy methods.

3.3 Experimental Setup

The experimental setup consists of several components. MATLAB® creates a signal to drive the wing using the code created by DeLuca [12] and Lindholm [21], shown in Appendix A. The MATLAB® signal is sent to the National Instruments USB-6229 Data Acquisition Device (NI Box) via Universal Serial Bus (USB). The signal is fed from the NI Box to the Trek PZD700A High-Voltage Power Amplifier / Piezo Driver. The amplifier supplies the signal to a breadboard, which is connected directly to the PZT. The PZT converts the electrical energy into mechanical energy, and causes the wing to flap. The 4-bar linkage converts the transverse bending motion of the PZT into the angular (rotary) motion of the wing. The output from the ATI Nano-17 Titanium (forces) and the Micro Optronic optoNCDT 1800 Laser Optic Displacement Sensor are fed back to the NI Box for recording by MATLAB®. The entire setup is shown in Figure 3.34. Figure 3.35 shows a closeup of the test stand wing in place ready to be tested.

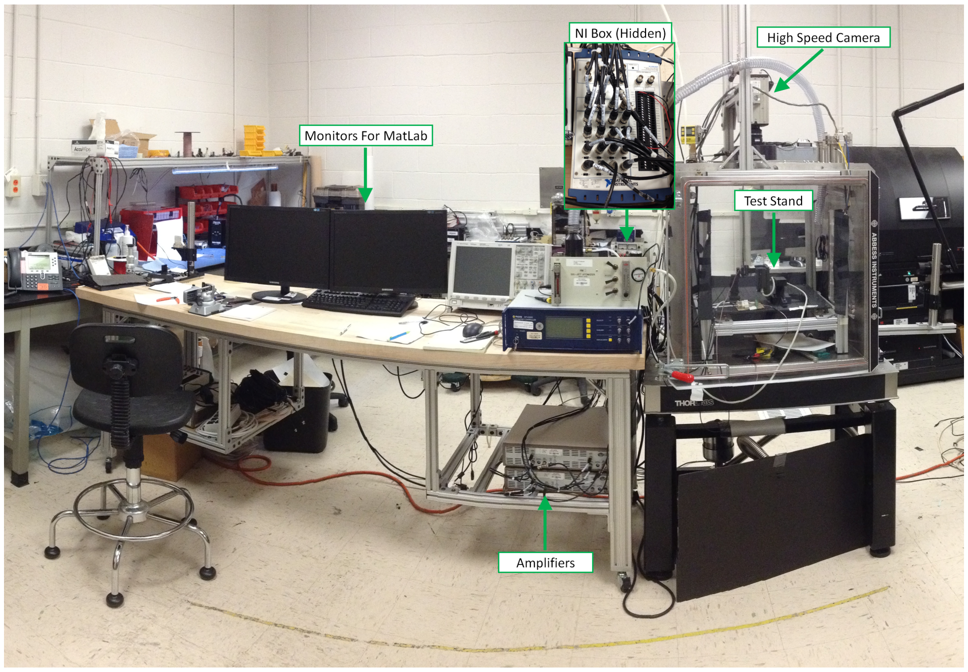


Figure 3.34: Experimental Setup.

The amplifier is connected to the PZT via three wires. The connections for single wing flapping are shown in Figures 3.36a and 3.36b.

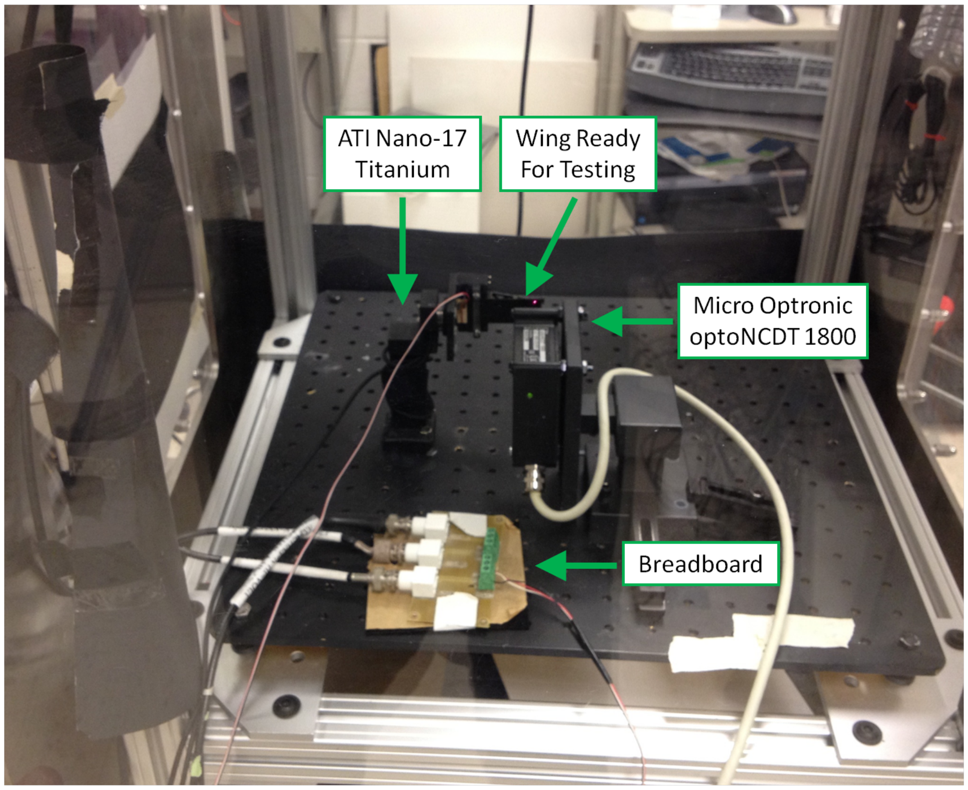
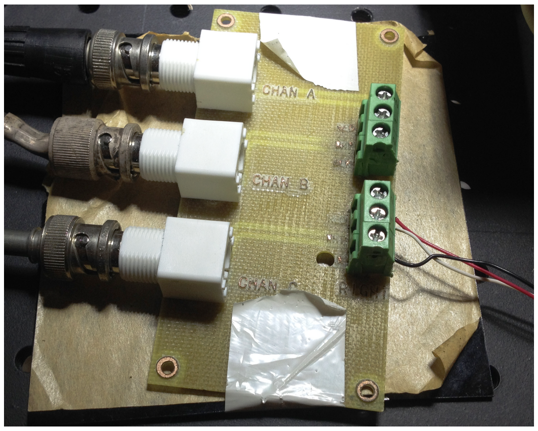
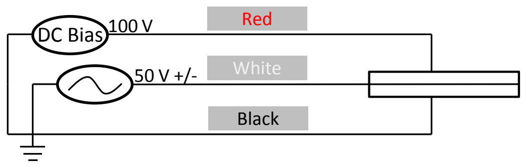


Figure 3.35: Test Stand Setup.





(a) Breadboard.

(b) Electrical.

Figure 3.36: Electrical Connections for Single Wing Flapping.

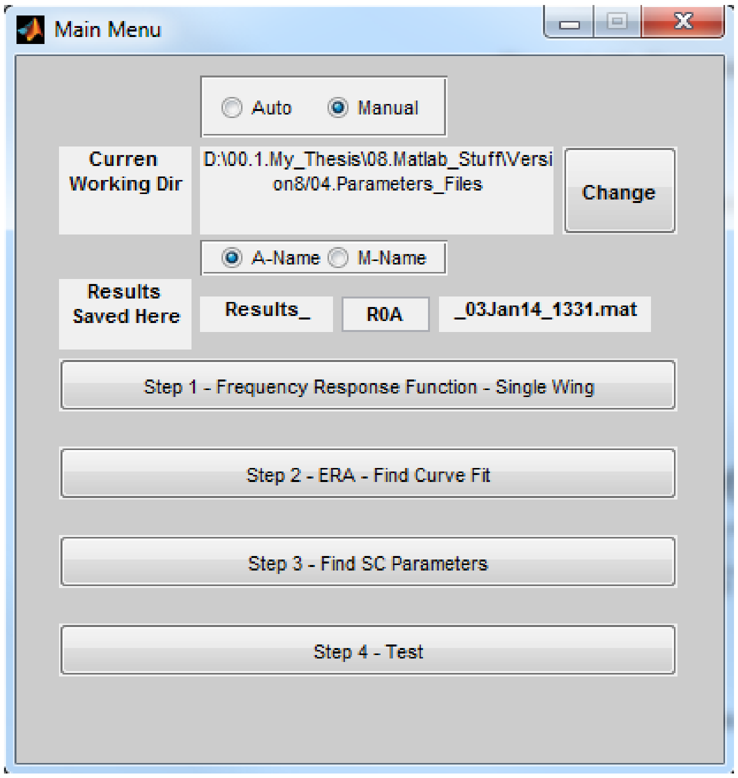
3.4 Experimental Methodology

A Graphical User Interface (GUI) was developed in MATLAB® to facilitate testing of the different wing revisions. The idea for the GUI was developed based on code written by DeLuca [12] and O'Hara [28], the initial intent was to modify that GUI for use in this project. Instead, a new GUI was developed from the ground-up to incorporate that functionality and so that a complete understanding of the process was obtained. Complete code for the GUI is available in Appendix B.

The process of testing is started by running the main menu which brings up the screen shown in Figure 3.37. The user has the option to choose automatic or manual testing. Manual testing allows greater control over the parameters and was used throughout this research. The user can designate a working directory where all files and settings are to be saved in addition to specifically changing the name of the results file. There are then four buttons corresponding to the four steps of testing. In the manual mode, the user is returned to the main menu after each step. In automatic mode, the progression of steps 1-4 happens in sequence with no user input.

After configuring the file options, the user selects Step 1 to obtain the natural frequency of the system. This brings up the options shown in Figure 3.38. The user can select any desired options for the chirp signal by moving the sliders or typing a new number in each respective box. There is also an option provided at the bottom of the screen to allow for loading a previous settings file. The settings for this research were left as shown in Figure 3.38. The GUI then stores the results into a MATLAB® data file. The GUI calls the previously developed code (modified to accept data file inputs) to generate various chirp frequencies to generate a profile for the wing being tested. The results are recorded and saved to a MATLAB® data file. The user is then returned to the main menu of the GUI.

The user then selects Step 2, which takes the user to the State-Space Model Identification using the Eigensystem Realization Algorithm Toolbox for MATLAB


Main Menu
Change
Step 1 - Frequency Response Function - Single Wing
Step 2 - ERA - Find Curve Fit
Step 3 - Find SC Parameters
Step 4 - Test

Auto [✓ Manual]

Curren Working Dir: D:\00.1.My_Thesis\08.Matlab_Stuff\Version8/04.Parameters_Files

[✓ A-Name] M-Name

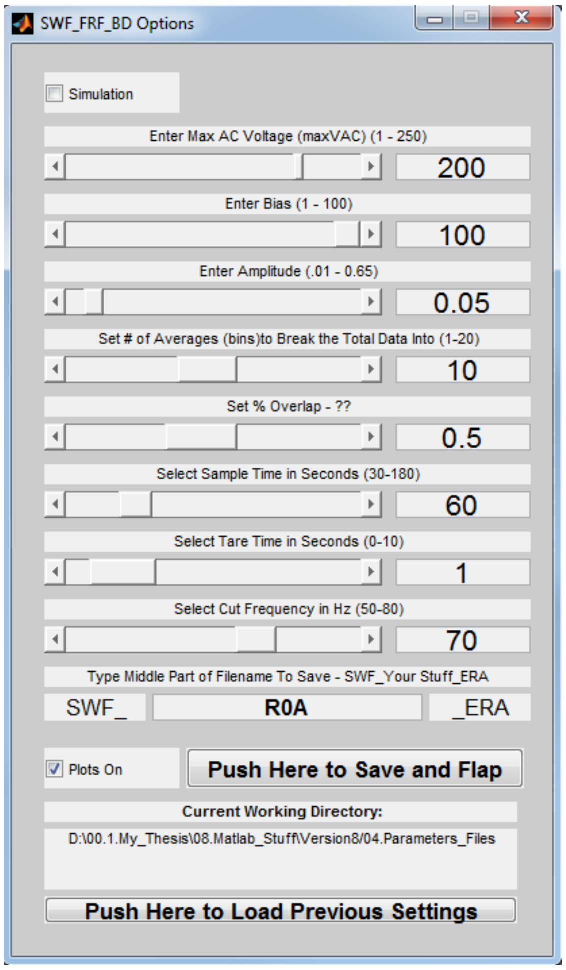
Results Saved Here: Results_ R0A _03Jan14_1331.mat

Figure 3.37: Graphical User Interface Main Menu.

(EZERA) developed by Cobb [8]. This program generates a state-space system based on the data obtained by the chirp test which can be used to find the natural frequencies of the system. The natural frequencies (most importantly the first) are saved and made available to the rest of the flapping program. The wing is then driven at system resonance which, as indicated by DeLuca [12], gives the largest wing deflection for a given amount of voltage.

Next, if desired the split cycle parameters are found by selecting Step 3. The *Find_SC_Parameters* function in Appendix A begins by constructing the state-space model of the system found by EZERA. The frequency response of the system is found at the first natural frequency of the system. Here, however, split-cycle testing was not performed.

Finally, the wing is ready to be tested and Step 4 is selected from the main menu. This brings up the dialog shown in Figure 3.39. There are several choices the user can select regarding testing, including a single frequency, a frequency range, or BABM test. While


SWF_FRF_BD Options

☐ Simulation

Enter Max AC Voltage (maxVAC) (1 - 250): 200

Enter Bias (1 - 100): 100

Enter Amplitude (.01 - 0.65): 0.05

Set # of Averages (bins)to Break the Total Data Into (1-20): 10

Set % Overlap - ??: 0.5

Select Sample Time in Seconds (30-180): 60

Select Tare Time in Seconds (0-10): 1

Select Cut Frequency in Hz (50-80): 70

Type Middle Part of Filename To Save - SWF_Your Stuff_ERA: SWF_ R0A _ERA

☑ Plots On

Push Here to Save and Flap

Current Working Directory:
D:\00.1.My_Thesis\08.Matlab_Stuff\Version8\04.Parameters_Files

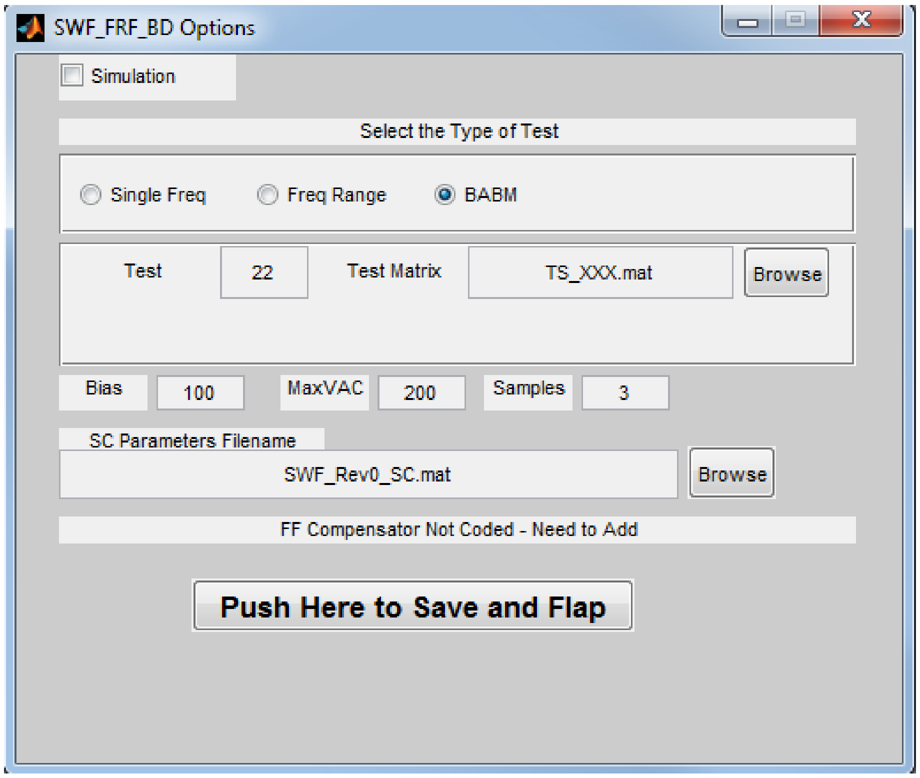
Push Here to Load Previous Settings

Figure 3.38: Graphical User Interface Chirp Options.

the BABM test portion of the GUI was used for this research, the settings for τ, and η were set equal to zero ($\tau = \eta = 0$). This specifies symmetric flapping. The amplitude[5] was varied from A = 0.30 - 0.45.

Using the code in Appendix A the drive signal is sent to the actuator in accordance with the test matrix specified. The testing is commenced, and MATLAB® provides progress indication in the command window along with saving the results to a data file.

[5]The drive amplitude setting does not directly specify the flap angle of the wing. Instead, amplitude specifies a percentage of maximum voltage to the PZT. If A = 0.30 and V_{max} = 200, then 60 V will be sent to the PZT. The end result is that a rough correlation between flap angle and amplitude exists, however, amplitude is not directly specified.


SWF_FRF_BD Options

Simulation

Select the Type of Test

Single Freq   Freq Range   [✓ BABM]

Test: 22   Test Matrix: TS_XXX.mat   Browse

Bias: 100   MaxVAC: 200   Samples: 3

SC Parameters Filename: SWF_Rev0_SC.mat   Browse

FF Compensator Not Coded - Need to Add

Push Here to Save and Flap

Figure 3.39: Flapping Wing Test Options.

The data was processed with several MATLAB® scripts using the saved results files as inputs. The scripts parse the results and compute the average, mean, and standard deviation for each different wing revision. These results are then written to a file that can be read by TecPlot or VisIt for graphical analysis.

3.5 Chapter Summary

The methods discussed in this chapter thoroughly detailed explanation of how the research was implemented. The use of the limited FEA study was discussed along with the construction process for the wings explained. The setup and methodology behind the planned experimental implementation was presented. Chapter IV will discuss at length, the results obtained during this research.

IV. Results and Analysis

The initial goal of this research was to investigate approaches to mass reduction while maintaining the vertical force production throughout different design revisions. Additionally, investigation into manufacturing methods for improving durability were sought. In undertaking this effort, it was not expected that all wing designs would prove effective; however, even from these "failed" designs, insight into the traits of a "good" wing can be gleaned. To meet this challenge, seven different designs were tested, and the results subsequently examined.

4.1 Finite Elements Analysis

The six new wing revisions were created, Finite Element Analysis was used to determine the location of minimum stress indicating those areas most suitable to mass reduction. The wings were held in a clamped boundary condition and subjected to a 500 mgF point load at the leading edge of the wing at 70% of the span. The overall trend in the results of the limited FEA was a lack of load support in the main planform venation. In addition, all of the designs carried substantial load at the attachment point. With those results at hand, the attachment point was not changed in any manner since it had effectively supported the load for the Revision 0 wing. Additionally, there were not indications that changing the individual venation throughout the wing would introduce any significant risk of breakage.

The original wing has a maximum stress location indicated by the light blue coloring located at the base of the wing and some portions of the main vein as seen in the stress contour plot of Figure 4.1.

Design 1 through 6 displayed results favorable to mass reduction, which are highlighted in Figure 4.2 as dark blue. Revisions 1, 3, 4, 5, and 6 exhibit stress along the

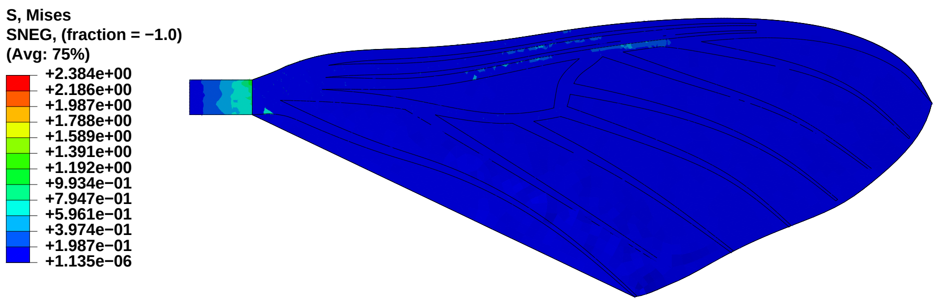
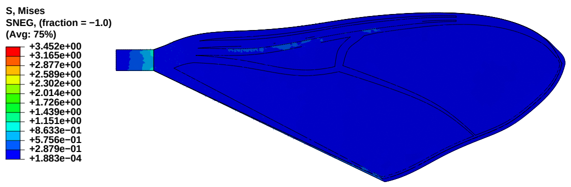
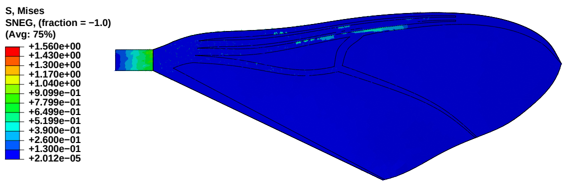


Figure 4.1: Results of Revision 0 Wing FEA.

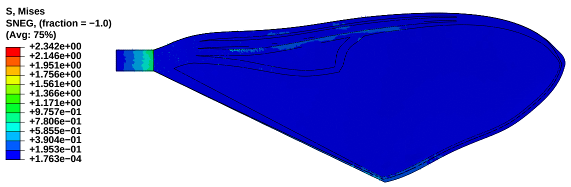
trailing edge member. This added member is sharing some of the stress load, and reducing it from the base of the wing. As more of the venation is removed, the trailing edge member supports more of the load on the wing. Revision 2 is the only modified design without the trailing edge member, and much of the stress is spread throughout the main vein and the arculus. What is consistent throughout is that the individual chordwise veins do not support much load and can be altered or removed as needed without a significant detrimental effect.



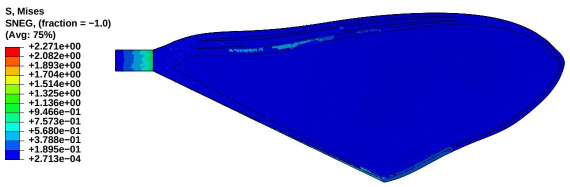
(a) Revision 1.



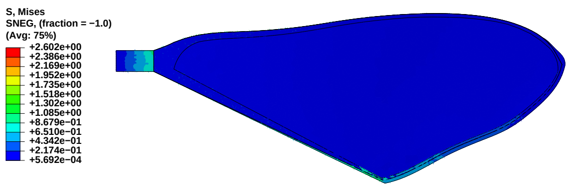
(b) Revision 2.



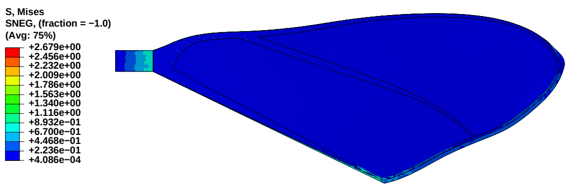
(c) Revision 3.



(d) Revision 4.



(e) Revision 5.



(f) Revision 6.

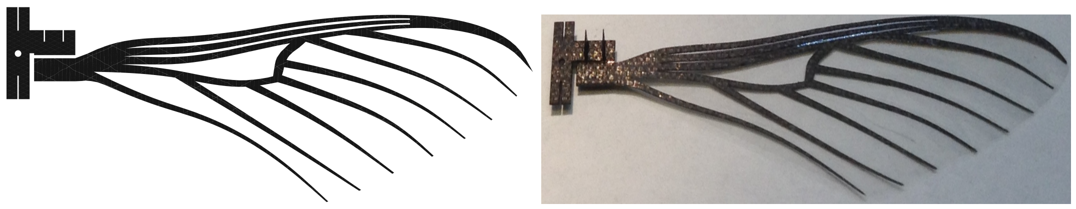
Figure 4.2: Finite Element Analysis Revisions 1 Through 6.

4.2 Manufacturing

The first step in analyzing the results is to consider the manufacturability of the designs. Changes to the wing design might reduce repeatability of the wing in production. Delamination issues were identified during research conducted by DeLuca [12], and it is important to identify the likelihood of delamination for different designs.

The Revision 0 wing design remained unchanged from the original wing of previous research [12, 28], and was manufactured without incident. The wing is shown in Figure 4.3. Limited delamination did occur at the tips of the veins, but not with any repeatable or predictable pattern, and overall very little difference from prior wings was noted.

The first wing revision was based upon reducing the number of veins in the chordwise direction. An additional spanwise member was added along the trailing edge to maintain



(a) SolidWorks. (b) Production.

Figure 4.3: Revision 0 Wing. The wing size in Figures 4.3a and 4.3b are the same, any apparent difference is due to the angle of the photograph.

some stiffness and prevent delamination. However, production of the wing was marred with difficulties. The trailing edge member was found to be too thin, and never survived production intact. The spanwise member would tear or crack when attempting to liberate the carbon fiber wing venation from the carbon fiber and Teflon backing. The trailing edge was widened by 0.44 mm, and production continued with no further issues. The widened trailing edge provided suitable attachment for the Mylar® and no delamination issues were experienced with the revised wing. The red circles in Figure 4.4 shows the location of common failure in the Revision 1 wing design. Figure 4.5 shows the successful production of the Revision 1 wing.

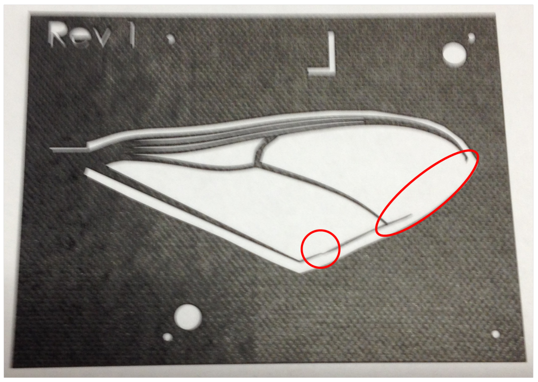


Figure 4.4: Revision 1 Beta Wing Failure Location.

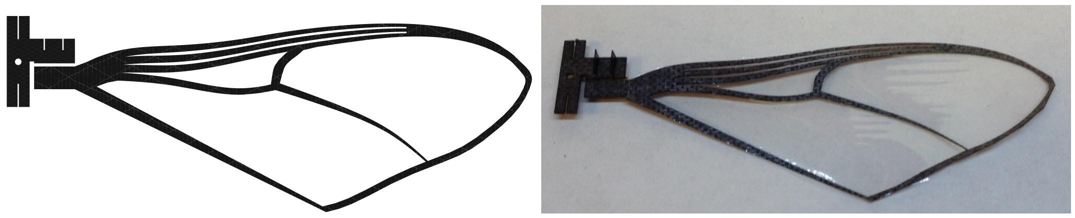

(a) SolidWorks. (b) Production.

Figure 4.5: Revision 1 Wing.

Wing Revision 2 is similar in construction to Revision 1 with the trailing edge member removed. Revision 2 was produced without incident and did not experience any delamination issues. The fact that delamination did not occur was slightly surprising as there was less physical area for the Mylar® to bond with the carbon fiber. One possible explanation is the reduced number of veins allowed a better seal with those remaining veins leading to a stronger bond. Figure 4.6 shows the second wing revision in SolidWorks and actual production.

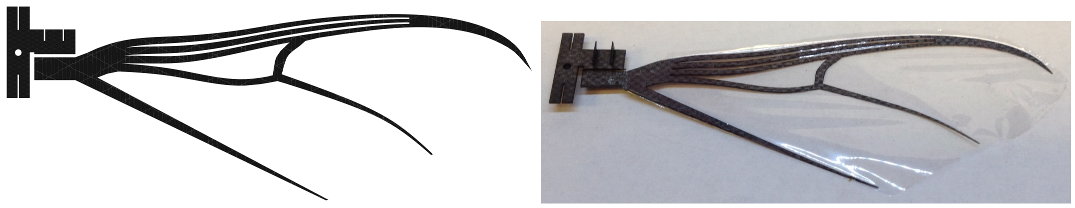
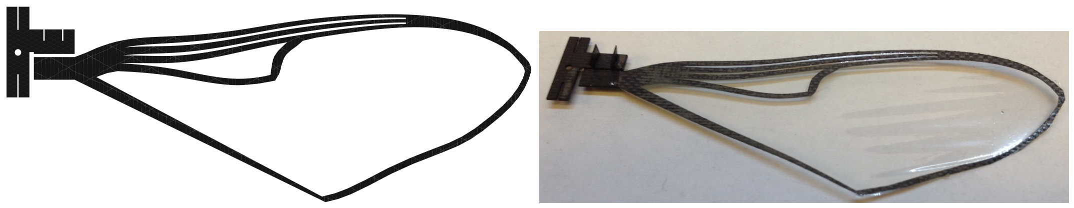

(a) SolidWorks. (b) Production.

Figure 4.6: Revision 2 Wing.

Revision 3 is based upon Revision 1 with all of the interior veins removed so that only the outlined wing venation remains. The trailing edge member of Revision 3 Beta failed in the same manner as Revision 1 and was widened by 0.44 mm. The resulting wing Revision 3 manufactured without issue, and did not experience any delamination problems during research. Figure 4.7 shows Revision 3.1.



(a) SolidWorks. (b) Production.

Figure 4.7: Revision 3 Wing.

Revision 4 sought to reduce wing mass further by removing the arculus vein from Revision 3. The arculus highlighted by the red oval in Figure 4.8. The same trailing edge issues were identified in Revision 4 Beta, and remedied in Revision 4 by widening the trailing edge. No further breakage was experienced, and no delamination occurred. Figure 4.9 illustrates the design of Revision 4.

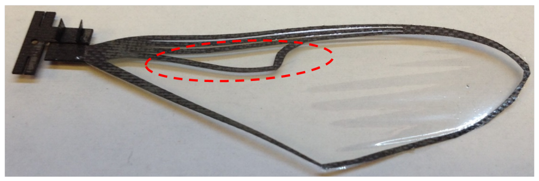
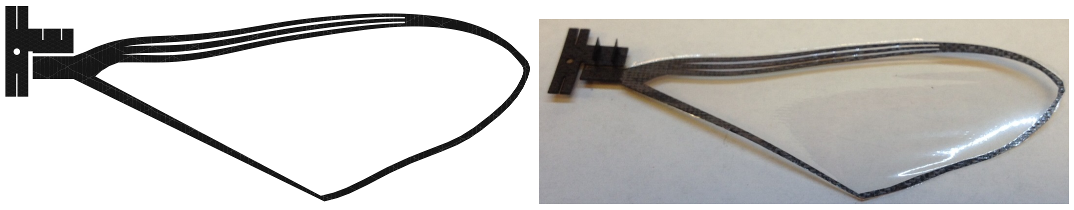


Figure 4.8: Location of Arculus Vein.



(a) SolidWorks. (b) Production.

Figure 4.9: Revision 4 Wing.

Revision 5 was a minimum mass design, removing all venation except the outer most structure connecting into an oval shape. No delamination was experienced in Revision

5 Beta, but the trailing edge failed in the same manner as previous designs. Revision 5 widened the trailing edge, which was then able to support production. The leading edge was thin, and had to be handled with care to prevent cracking. In one of the production models, the leading edge failed during manufacturing by snapping in half at the midpoint. This was not a repeatable failure, and as long as the wings were handled with care, the design did not pose any systemic issue. Revision 5 is shown in Figure 4.10.

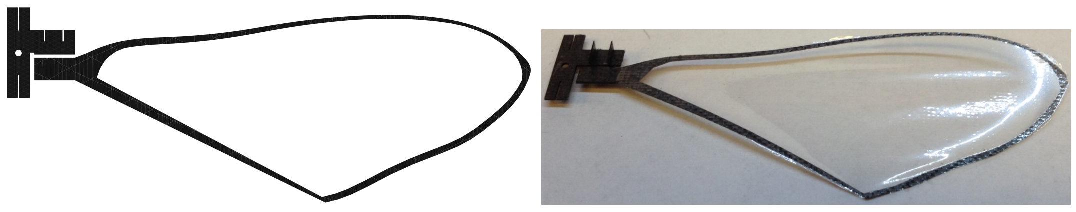

(a) SolidWorks. (b) Production.

Figure 4.10: Revision 5 Wing.

Wing Revision 6 is a blend between Revision 2 and Revision 5. Three veins were retained with the connected trailing edge. Again, Revision 6 Beta production failed on the trailing edge and that member had to be widened. Revision 6 is displayed in Figure 4.11.

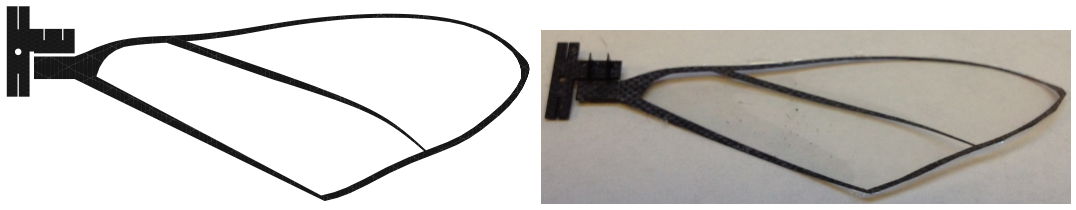
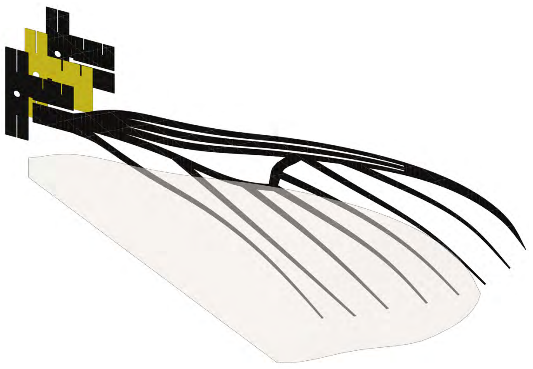

(a) SolidWorks. (b) Production.

Figure 4.11: Revision 6 Wing.

4.3 Wing Mass

The assemblies are created in Solidworks as shown in Figure 4.12. For each different wing revision, the carbon wing layer is replaced; however, the other three layers remain unchanged. The end result is that only the venation pattern is modified.


(a) Exploded View.

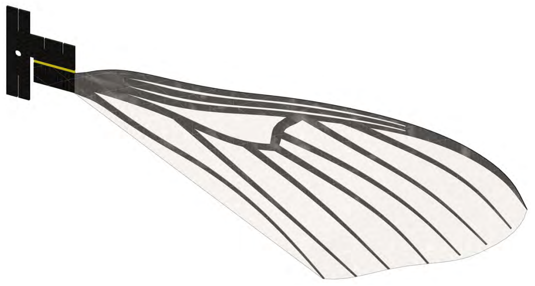

(b) Collapsed View.

Figure 4.12: Solidworks Assembled Wing Layers.

Table 4.1 shows the densities used by Solidworks in calculating the mass of the wing assemblies. The carbon fiber [24], Kapton® [15], and Mylar® [16] were obtained from their respective manufacturer websites. The assemblies are extruded to match the thickness of the actual engineered wing shown in Table 4.2.

Table 4.1: Material Properties.

Material	Density (g/cm^3)
Carbon Fiber	2.14
Kapton®	1.42
Mylar®	1.38

Table 4.3 shows the mass of each assembly calculated by Solidworks compared with the average masses of the actual production wings. The mass was measured on a Ohaus

Table 4.2: Extrusion Thickness for Individual Layers.

Layer	Thickness (mm)
Wing CF	0.1050
Mylar®	0.0025
Covering CF	0.1050
Kapton®	0.0250

Voyager Pro 214CN with a resolution of 0.1 mg. All of the new wing designs have a reduced mass from the Revision 0 in both the predicted mass and actual mass. The actual mass tends to be approximately two milligrams less than the predicted mass. The high excursion is Revision 4, which over predicted by 3.3 mg. The low excursion is Revision 5, which over predicted by only 0.1 mg. The discrepancy between the predicted mass and actual mass is due to inaccurate modeling in Solidworks. There is a layer of Pyralux® [14] sheet adhesive on the back of the 0-90-0 carbon fiber layup. The Pyralux® layer is modeled as CF in the Solidworks model to simplify construction. The density differences between the two materials cause part of the discrepancy between the actual and predicted masses. Additionally, the precision of the cut made by the PLU affects the mass of the production wings. Future wing designers can safely use the rule of thumb given in Equation 4.1 to determine the actual mass from the Solidworks prediction. Applying Equation 4.1 to the predicted masses given in Table 4.3, reproduces the actual masses with a standard deviation of 0.9 mg. Figure 4.13 presents the data of Table 4.3 in bar form with the scale given from 40 to 65 mg. The black outlines represent the predicted mass in mg, the green fill represents the average actual mass in mg, and the red error bars represent one standard deviation of the sample group for each wing revision.

$$M_{\text{actual}} = 0.968 \cdot M_{\text{predicted}} \tag{4.1}$$

Table 4.3: Predicted and Actual Mass of Different Wing Assemblies in mg.

Wing	Predicted	Actual (Avg.)	Std. Dev.	Sample Size
Original	63.6	61.5	1.1	3
Revision 1	59.6	57.5	0.3	5
Revision 2	54.2	52.5	1.0	4
Revision 3	57.4	55.3	1.6	5
Revision 4	53.1	49.8	2.2	4
Revision 5	42.0	41.9	0.7	3
Revision 6	46.4	45.4	0.7	4

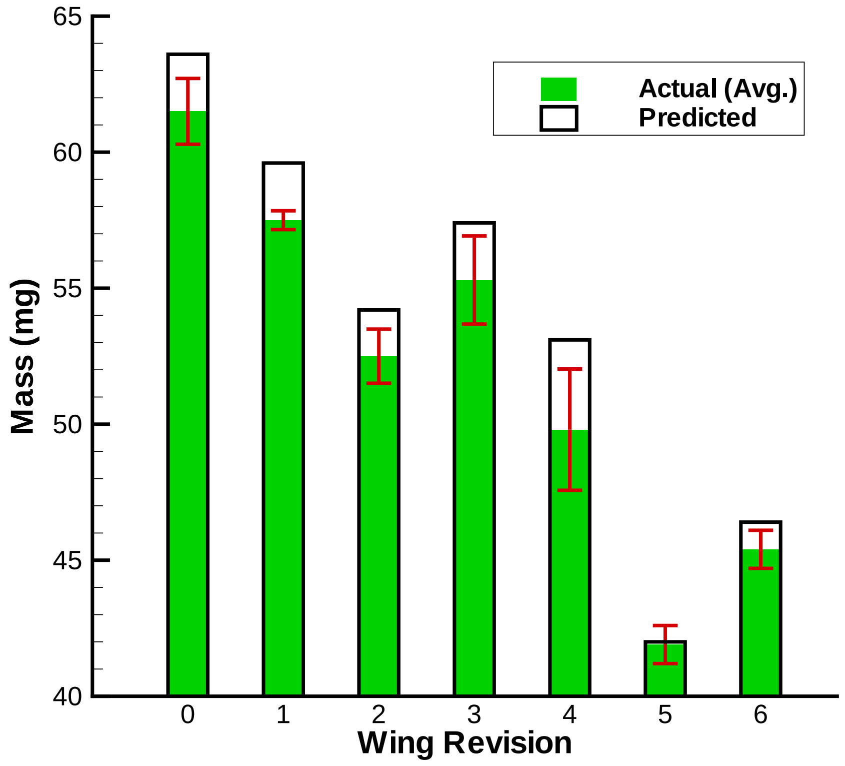


Figure 4.13: Predicted vs. Actual Mass. Error Bars Signify One Standard Deviation.

4.4 Natural Frequency

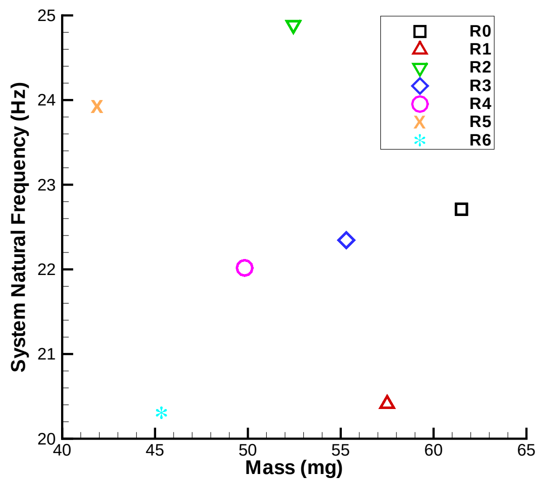
If stiffness were fixed, one might anticipate a strong correlation between mass and system natural frequency from Equation 3.1. Decreasing the mass would cause the natural frequency to increase. Figure 4.14a shows the natural frequency of the wings compared with their respective mass. If a strong correlation existed, the points would lie in a line from the top left to bottom right. It is clearly seen that the points are spaced throughout the domain, and a fixed stiffness in Equation 3.1 is not a good approximation when changing wing mass and design because it does not account for other interactions occurring. The data presented are the average of the successfully produced samples for each design revision.

Revision 0, which has the highest mass, has an average natural frequency of 22.7 Hz. Revision 6, which is 16 mg lighter, has a natural frequency of 20.3 Hz. Revision 2 is 9 mg lighter than Revision 0 and has a natural frequency of 24.9 Hz. Table 4.4 shows the mass of each design with its respective natural frequency in order of revision.

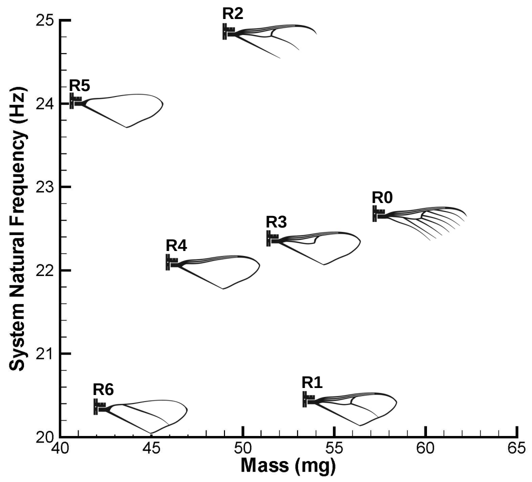
Clearly the stiffness of the wing varies between revisions. A method to quantify the relative stiffness of each wing is needed to assist in identifying design trends. The Relative Deduced Stiffness (RDS) is derived by rearranging the stiffness to mass ratio and normalizing by the RDS of Revision 0. The results are given in Equation 4.2, where $k_{\mathrm{R0}} = 31,690\ \mathrm{Hz}^2 \cdot \mathrm{mg}$ is the RDS of Revision 0 prior to normalization and $k_{\mathrm{De},i}$ is the normalized Relative Deduced Stiffness of subsequent revisions. The subscript i represents an individual Revision, i.e. 1 through 6. Table 4.4 lists each wing design with the respective measured mass (m_i), measured natural frequency ($\omega_{n,i}$), and normalized RDS.

$$k_{\mathrm{De},i} \propto \frac{\omega_{n,i}^2 m_i}{k_{\mathrm{R0}}} \tag{4.2}$$

Figure 4.14b shows the same data as Figure 4.14a with the actual wing designs overlaid on top of their design parameters. Those designs with the trailing edge member are concentrated in the middle and lower end of the frequency range, with the exception of Revision 5. The arculus appears to have no direct effect on natural frequency as those are spread throughout the domain.



(a) By Reference



(b) By Design

Figure 4.14: Average System Natural Frequency (Hz) vs. Average Mass (mg).

Table 4.4: Mass, System Natural Frequency, and Relative Deduced Stiffness.

Revision	Mass (mg)	Natural Frequency (Hz)	Deduced Stiffness
0	61.5	22.7	1.00
1	57.5	20.4	0.76
2	52.5	24.9	1.03
3	55.3	22.3	0.87
4	49.8	22.0	0.76
5	41.9	23.9	0.76
6	45.3	20.3	0.59

If one assumes the same wing planform and neglects aeroelastic effects, then increased vertical force would be expected from an increased natural frequency. In fact, some of the designs did produce increased vertical force with decreased natural frequency. Figure 4.15 illustrates the relationship between natural frequency and vertical force, note the axes do not start at zero. The individual data points in Figure 4.15 represent amplitudes tested, i.e. on the green line for Revision 2, the left most data point is for the test where A = 0.30, the 2nd point is for A = 0.35, the 3rd for A = 0.40, and the final point represents A = 0.45. Revision

2 produced the most vertical force at A = 0.45 amplitude and sits at the highest natural frequency of those present. Revision 3 has a similar natural frequency to the Revision 0 but produces more vertical force at the same amplitudes. The only identifiable trend is that at the higher amplitudes (A =0.40 and 0.45), all of the revised wings outperform Revision 0 in these experiments. However, the high speed video demonstrated that broad conclusions about the wing performance cannot be drawn.

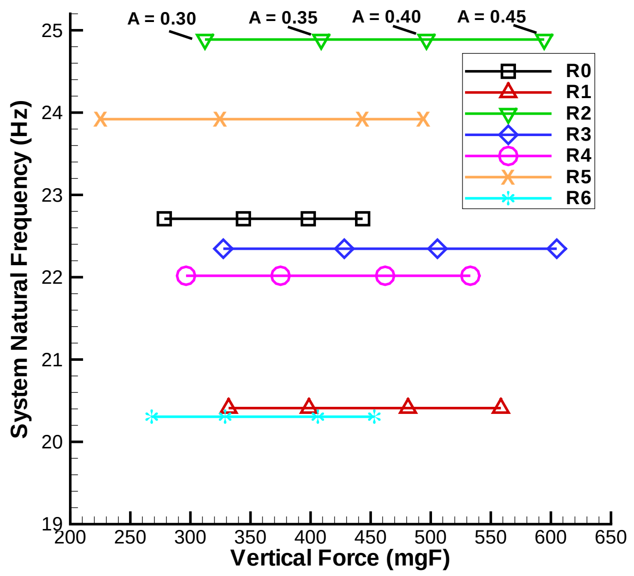


Figure 4.15: Average System Natural Frequency (Hz) vs. Average Vertical Force (mgF) of Individual Wing Revisions, Flapping at Resonance, 60° Angle Stop.

Figure 4.16 illustrates the lack direct correlation between natural frequency and vertical force produced by the wing. Figure 4.16 is presented to isolate natural frequency as the dependent variable and vertical force as the independent variable. Each data point represents the natural frequency and vertical force of a specific wing design revision at a specific amplitude. If an increased natural frequency were to cause an increase in vertical force produced, the data points would lie in a line from lower left to the upper right. Since this is not the case, no identifiable correlation exists between natural frequency alone and vertical force. There is a much more significant correlation between amplitude and vertical force that can be inferred from the plot. Figure 4.16 presents data for the maximum and minimum amplitudes tested, the intermediate amplitudes contained similar values. It is also important to note that the axis do not meet at zero.

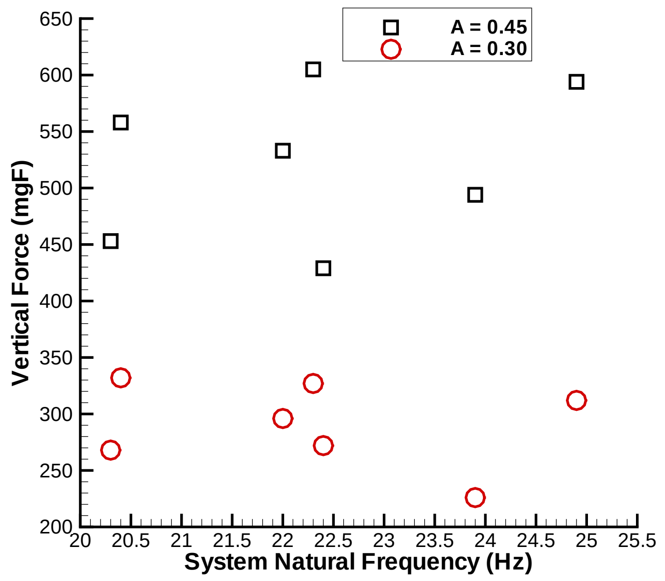


Figure 4.16: Average Vertical Force (mgF) vs. Average System Natural Frequency (Hz) of All Wing Revisions Combined, Flapping at Resonance, 60° Angle Stop.

4.5 Amplitude

The effects of changing amplitude[6] through a specified range of A = 0.30 - 0.45 were predictable. For most designs an increase in vertical force was consistent with a linear relationship to an increase in drive amplitude. Revision 5 is the only design that does not follow this relationship throughout the different amplitude ranges. The vertical force increased linearly from A = 0.30 - 0.40, however, from 0.40 - 0.45 the slope of the vertical force curve decreased. Figure 4.17 shows the averaged values for Revision 0 and the six subsequent design revisions; note that the axes start at values other than zero. The

[6]Recall that changing amplitude does not directly specify a change in flapping angle.

individual data points represent different amplitudes. Table 4.5 shows the results for each of the design revisions.

Revision 0 had the lowest overall vertical force production at A = 0.45. Revision 3 performed the best at all amplitudes, with the exception of A = 0.30 where Revision 1 was slightly better. Revision 2 also produced a significant vertical force at all amplitudes. Revisions 5 and 6 performed poorly at lower amplitudes (A = 0.30 and 0.35). Revision 6 continues that trend of low force production throughout the remaining higher amplitudes.

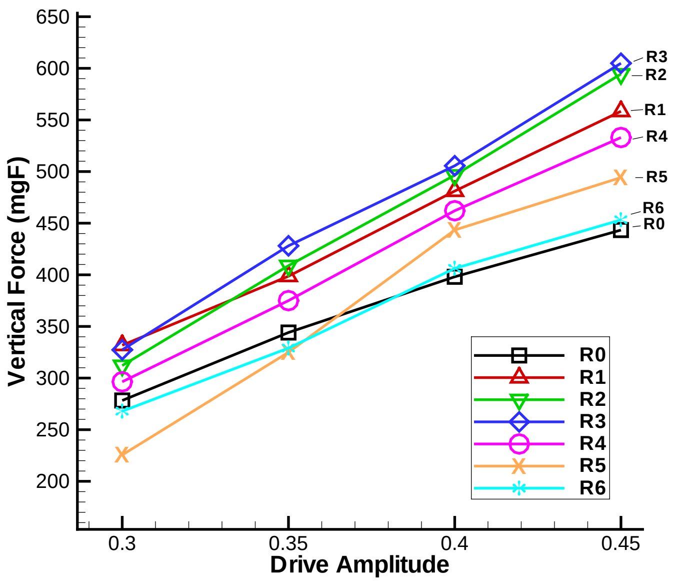


Figure 4.17: Average Vertical Force (mgF) vs. Drive Amplitude, Flapping at Resonance, 60° Angle Stop.

Table 4.5: Average Vertical Force (mgF) at Four Drive Amplitudes for all Seven Designs.

	Amplitude (A)			
Revision	0.30	0.35	0.40	0.45
0	278.3	344.1	398.1	443.4
1	331.8	398.7	481.1	558.4
2	312.1	408.8	496.5	594.4
3	327.4	428.1	505.6	604.9
4	296.4	375.0	462.1	533.0
5	225.6	324.9	443.4	494.2
6	267.9	329.0	406.2	453.2

4.6 Mass

The mass did not directly correlate with vertical force production in any predictable manner, which can be seen in Figure 4.18. Note that in the figure, the axes do not meet at zero and the data points are representative of the different tested amplitudes (A = 0.30,0.35, 0.40, 0.45). Drawing specific conclusions would be accomplished more easily with a design that was located in the lower right corner of the plot, giving high vertical force production at a low mass. Revisions 2 and 3 achieved high vertical force production at all amplitudes with a 9 mg and 6.2 mg mass reduction respectively when compared to the mass of Revision 0.

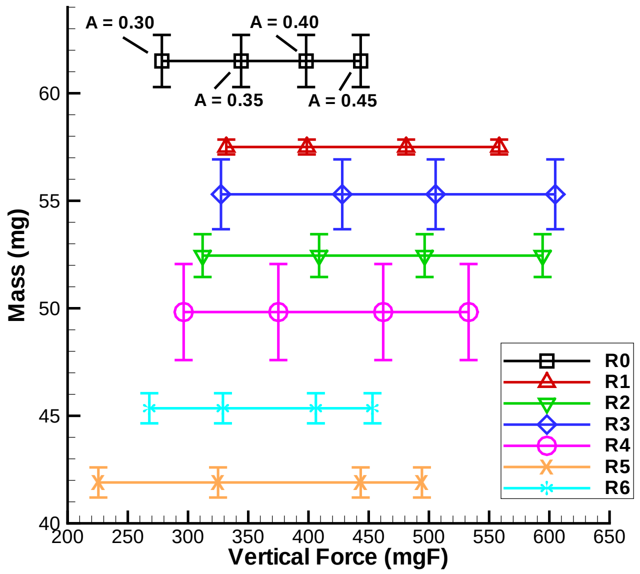
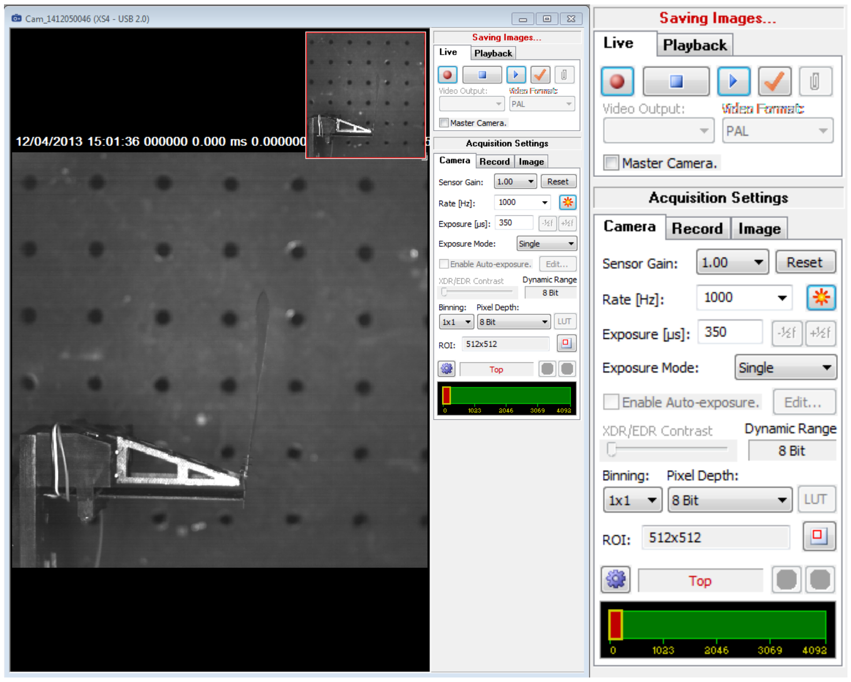


Figure 4.18: Average Mass (mg) vs. Average Vertical Force (mgF) at Four Amplitudes.

4.7 Video Capture

Video footage was taken with two IDT Vision X-Stream XS-4 High Speed Digital Camera [20] to identify characteristics of each design not obvious through force measurements. IDT Motion Studio Software [19] was used to trigger the cameras to record on their respective internal storage. The trigger signal is sent to Motion Studio from MATLAB® via the NI Box for both cameras. Following successful recording, the still images are downloaded from the internal memory to the local hard drive. The images are processed using Motion Studio by compiling into Audio Video Interleave (AVI) format. In addition, each frame was saved individually as a Portable Network Graphics (PNG) image file. This file archiving method allowed for video to be viewed an analyzed along with high quality images for inclusion here. The camera settings are shown in Figure 4.19. The most important settings to obtaining quality images are the Sensor Gain (1.0), the Rate (1000 Hz), and the Exposure Time (350 µs). Additionally, placing a plane white sheet of paper underneath the wing helped to provide better contrast between the wing and surroundings.



12/04/2013 15:01:36 000000 0.000 ms 0.000000

Video Format: PAL

Sensor Gain: 1.00

Rate [Hz]: 1000

Exposure [µs]: 350

Exposure Mode: Single

Dynamic Range: 8 Bit

Binning: 1x1

Pixel Depth: 8 Bit

ROI: 512x512

(a) Full (b) Settings Only

Figure 4.19: Motion Studio Camera Settings.

Figure 4.20 shows the still images of Revision 0. The motion is smooth; however, the deflection is larger to the left (downstroke) compared with deflection on the upstroke. This bias can be corrected for by running the *AutoTune* procedure[7] developed by DeLuca [12]. The images also reveal an interesting anomaly that is not present in most of the other designs. The supination does not appear to be fully complete, and the wing presents a flatter than normal profile during the upstroke. This may cause a significant increase in air resistance and possible reduction in lift as the wing travels toward the top of the stroke. The most likely cause of the binding is glue that has fouled the rotation joint. Unfortunately, the unsatisfactory flapping motion of Revision 0 was discovered late during the research, and a new set of tests for the Revision 0 wing could not be completed. The images in Figure 4.21 show the Revision 1 wing throughout the flapping cycle. The flapping motion is smooth,

[7]*AutoTune* is a MATLAB® script that automatically adjusts the bias so that the wing flaps symmetrically about the neutral axis.

however, the upstroke is faster than the downstroke. The increase in upstroke speed causes a decreased flapping period, possibly altering the force produced during each cycle. This revision does exhibit bias on the upstroke. Revision 2, shown in Figure 4.22, exhibits longitudinal bending during the transition from upstroke to pronation. The stroke is biased towards the right. The images in Figure 4.23 show a large amount of bias to the left during the downstroke of Revision 3. The wing almost touches the frame on the downstroke. What cannot be seen from the still images, but evident in the video, is the downstroke occurs more rapidly compared to the upstroke. The high speed video was captured at 1000 Hz. The average downstroke is 17 frames and the average upstroke is 21.3 frames which results in an angular stroke rate of 6700 and 5340 degrees / second respectively. Wing Revision 4 also exhibits a longitudinal bending, however, this bending occurs during the transition from downstroke to supination. Figure 4.24 shows the Revision 4 wing stroke and a large amount of right bias is present. Additionally, the wing did not supinate fully and it is likely that glue has invaded the rotation joint in a manner similar to that of Revision 0. Figure 4.25 shows wing Revision 5 throughout one cycle. This wing exhibits longitudinal bending in both the supination and pronation phases of the stroke. Revision 5 also displays a slight bias to the left, similar to Revision 3, this wing almost touches the frame on the downstroke. Revision 6 is shown in Figure 4.26 and the stroke is a smooth motion. There is a very slight amount of bending in the transition from downstroke to supination. This wing is biased on the upstroke, however the bias effect is of less magnitude than present in by Revisions 3.

Using the two images with the largest deflection for each design, an analysis of the stroke angle (Φ, defined in Figure 4.27) was able to be accomplished. Positive stroke angle is defined as occurring during the upstroke, that is in the positive-z direction. Negative

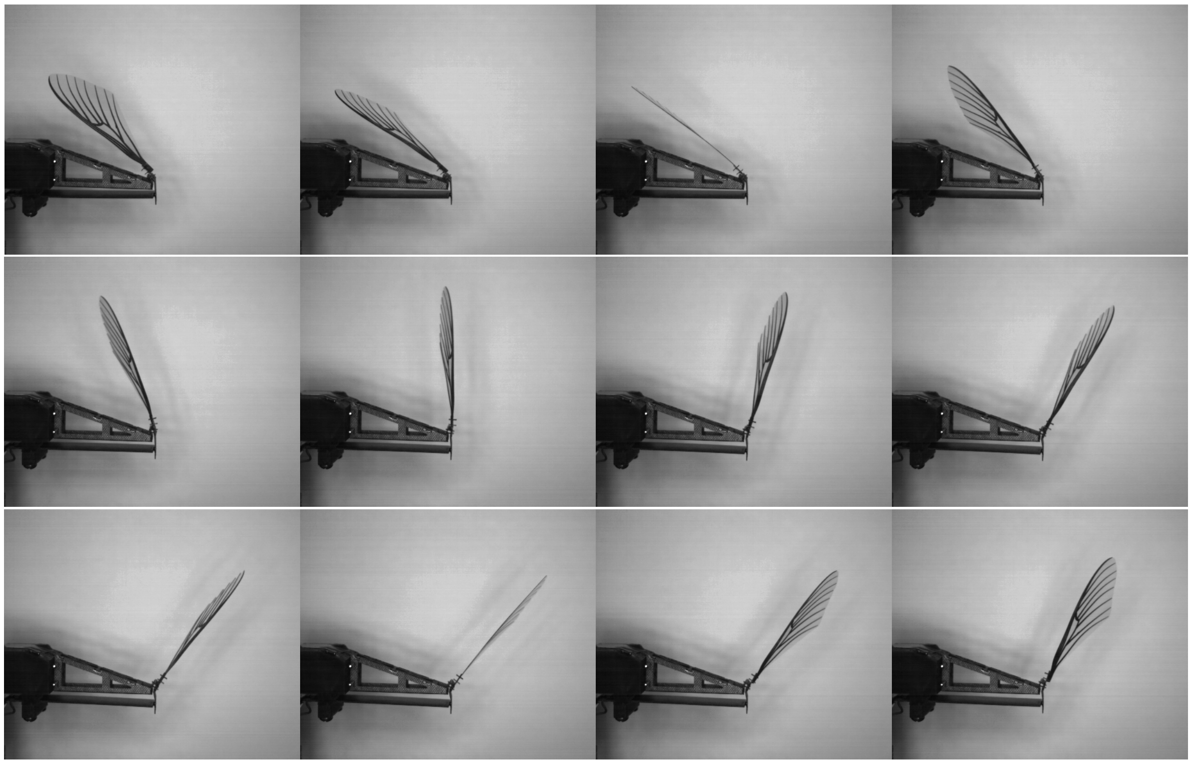


Figure 4.20: Video Stills for Revision 0 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.

Φ angles occur during the downstroke. This convention follows that used in previous research [12]. Table 4.6 shows the breakdown of stroke angles for each of the different designs.

Figure 4.27 shows the superimposition of the two images with their respective stroke angles overlaid. None of the wings are flapping symmetrically, however, some are more biased than others. Notably, Revision 3 has an almost 20° bias on the downstroke and Revision 4 is the reverse. Using the *AutoTune* routine developed by DeLuca [12] would help to remedy the bias. *AutoTune* was not used during the research because the asymmetry of the stroke angles was discovered too late, only during post-processing after all testing was completed.

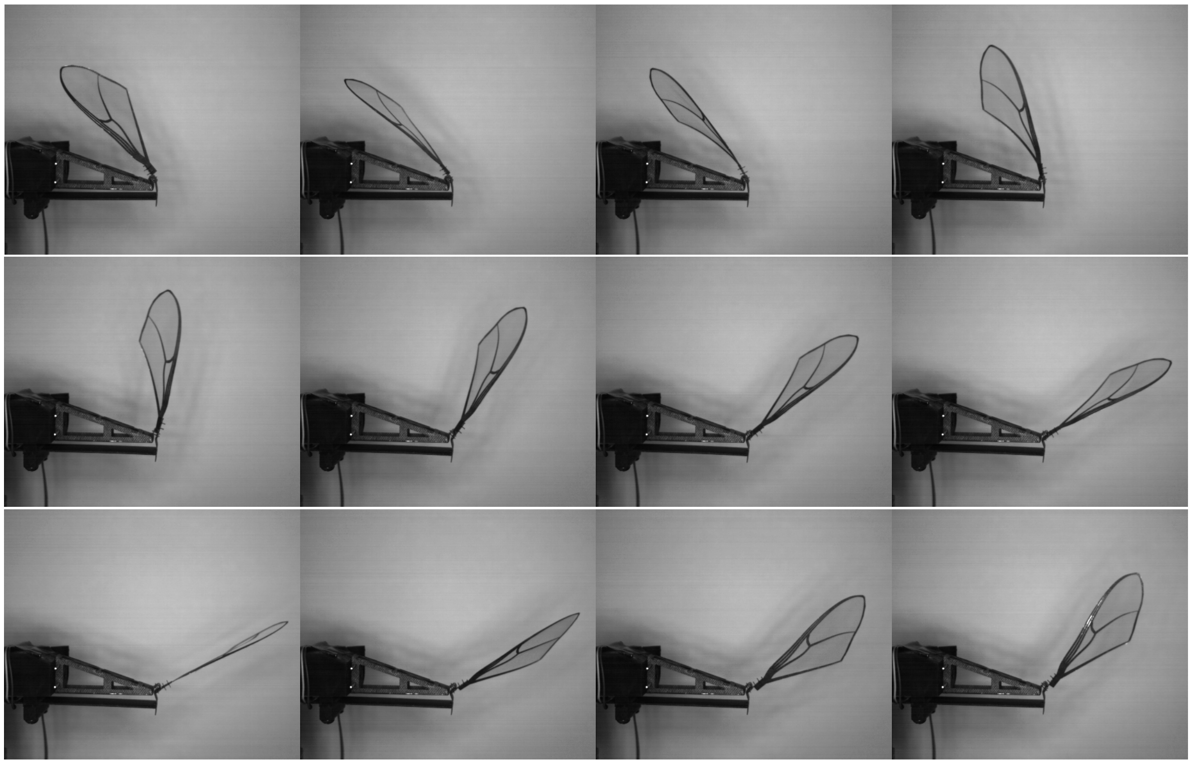


Figure 4.21: Video Stills for Revision 1 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.

Table 4.6: Stroke Angle (Φ) for Seven Different Designs, Flapping at Resonance, 60° Angle Stop.

Revision	Upstroke	Downstroke	Total
0	38.7°	-56.6°	95.3°
1	64.3°	-49.4°	113.7°
2	65.7°	-53.6°	119.3°
3	47.6°	-66.3°	113.9°
4	53.3°	-35.2°	88.5°
5	50.6°	-54.8°	105.5°
6	50.6°	-34.9°	85.5°

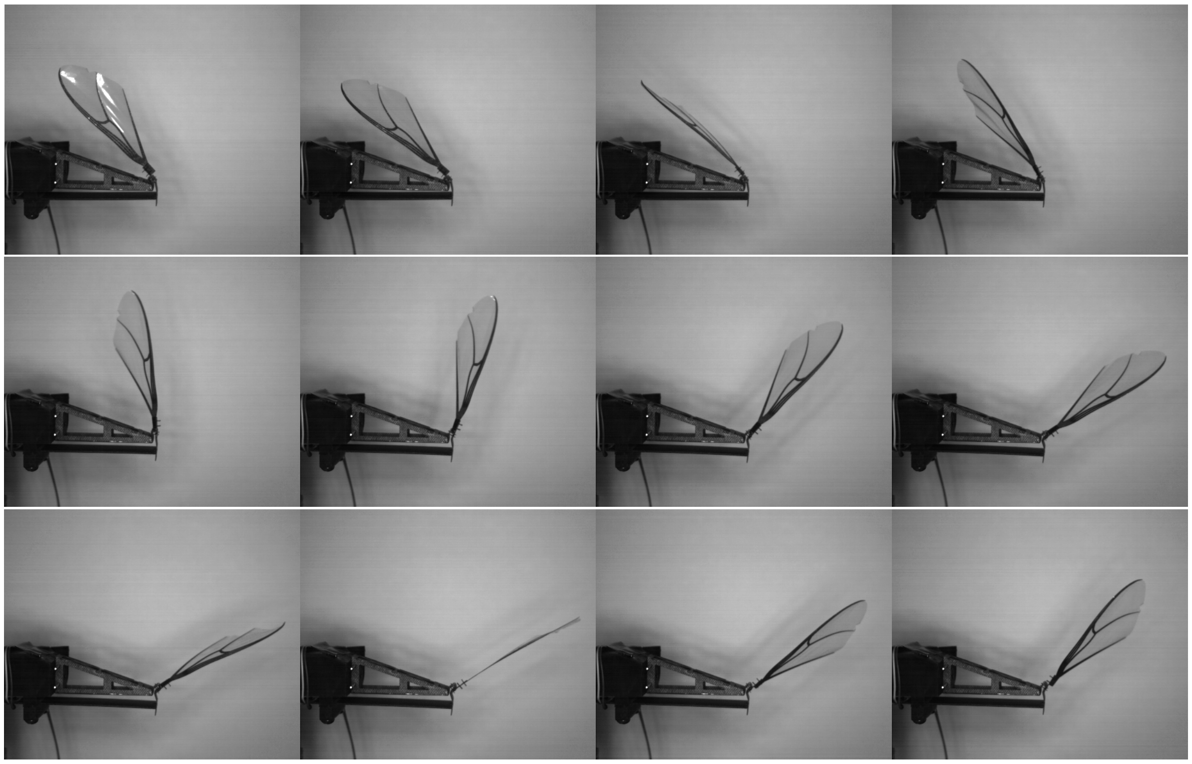


Figure 4.22: Video Stills for Revision 2 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.

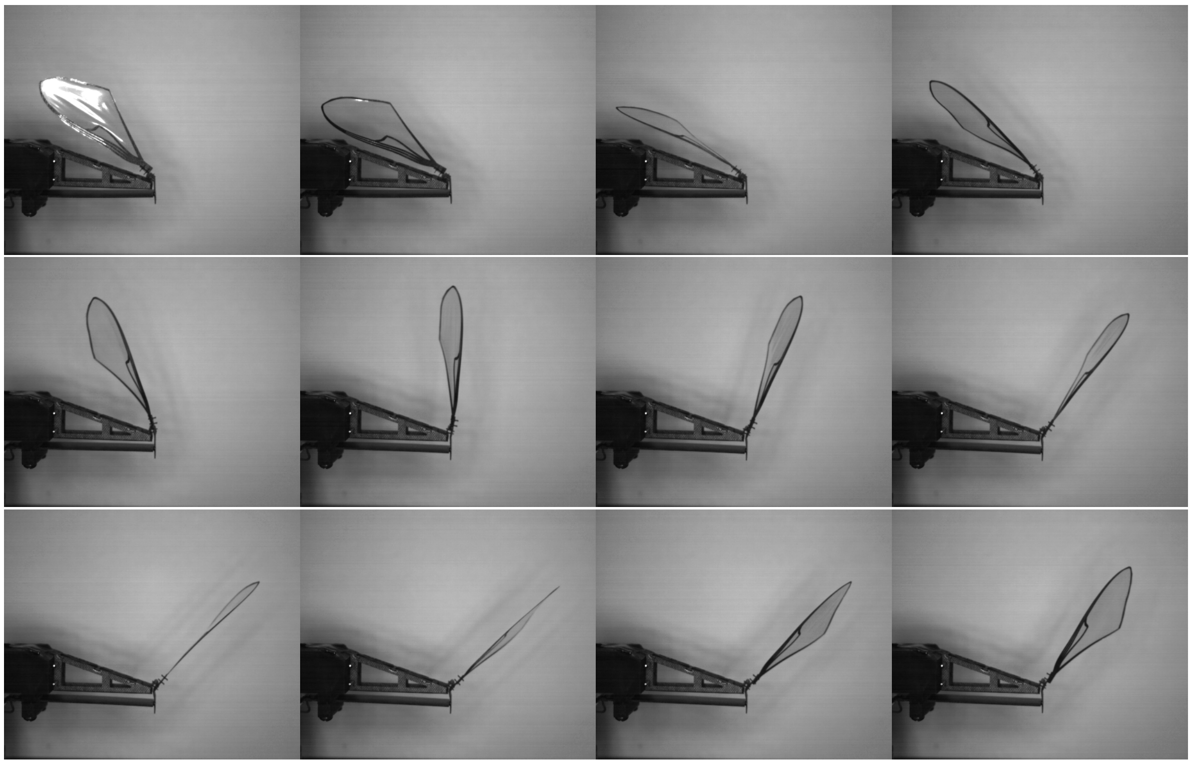


Figure 4.23: Video Stills for Revision 3 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.

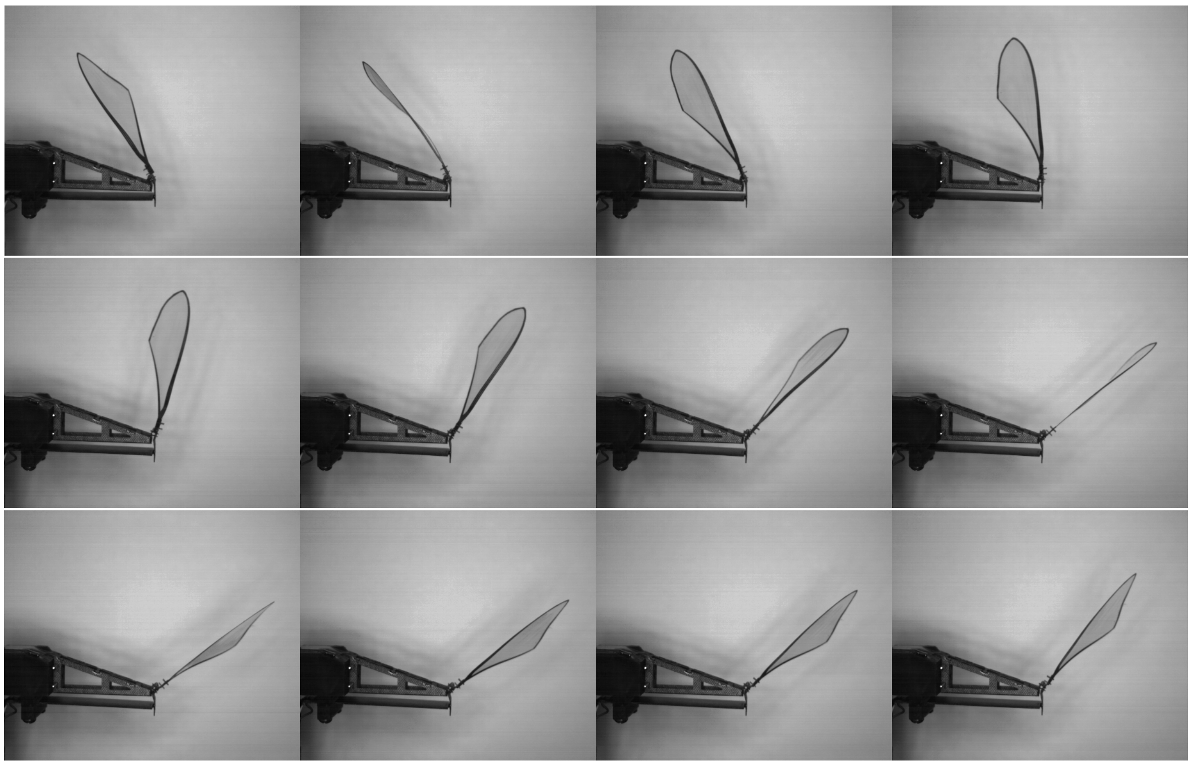


Figure 4.24: Video Stills for Revision 4 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.

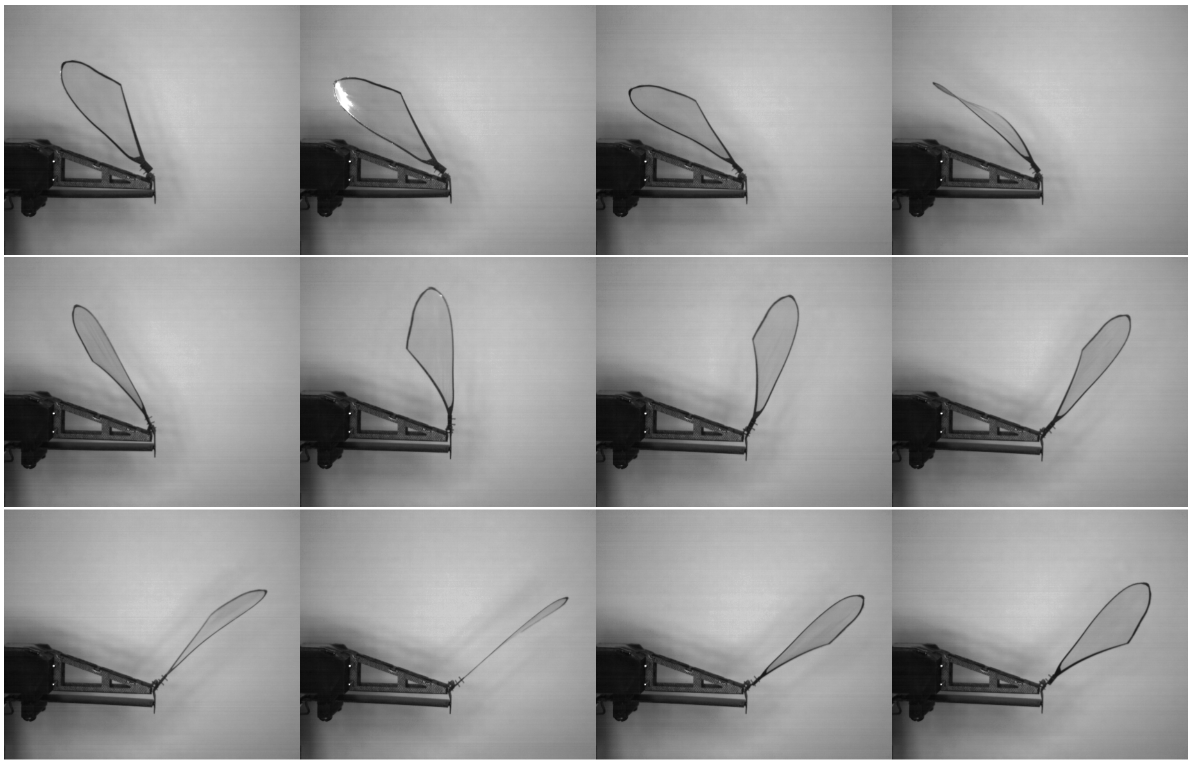


Figure 4.25: Video Stills for Revision 5 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.

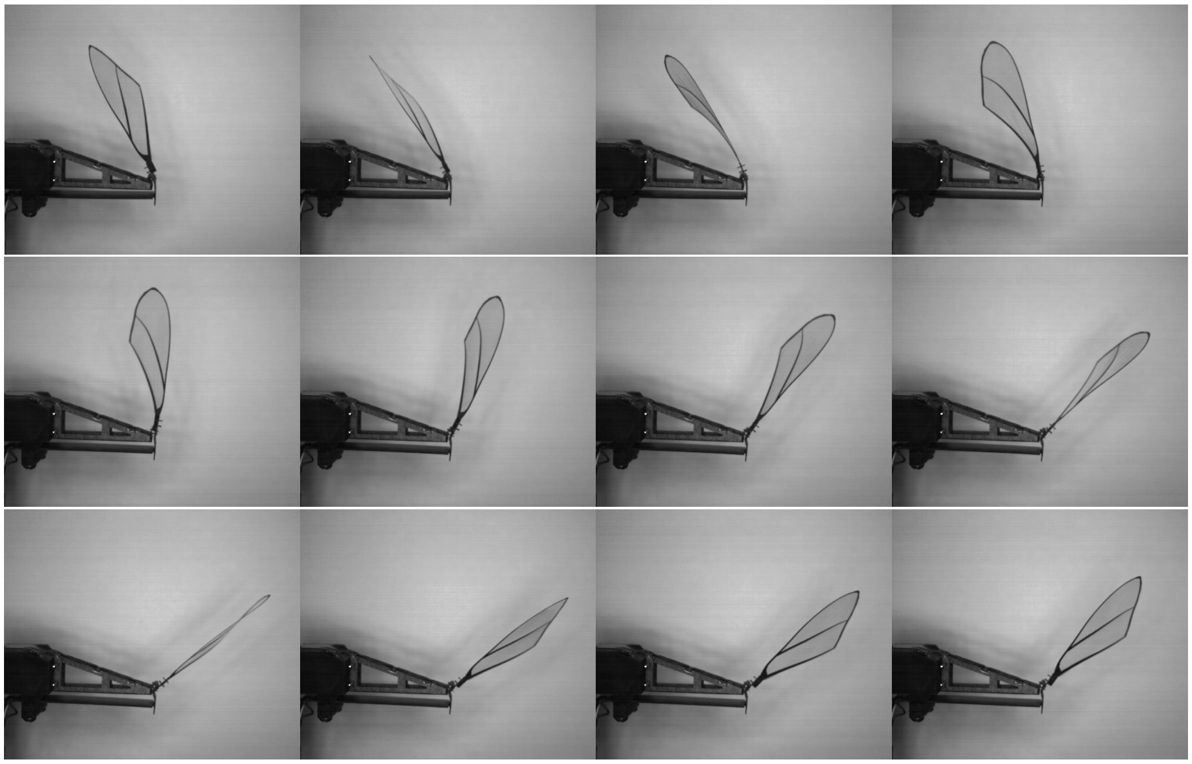
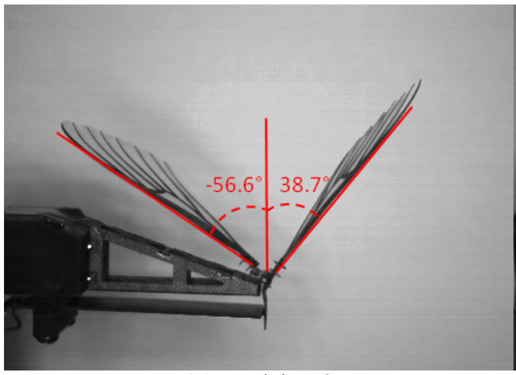
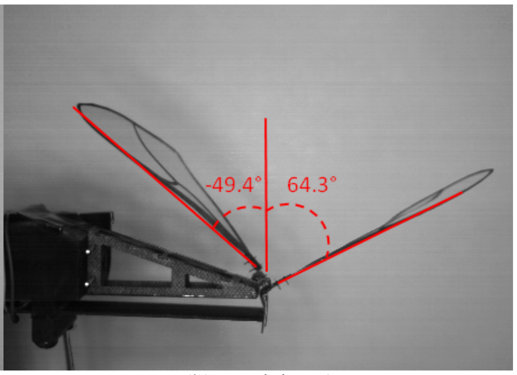


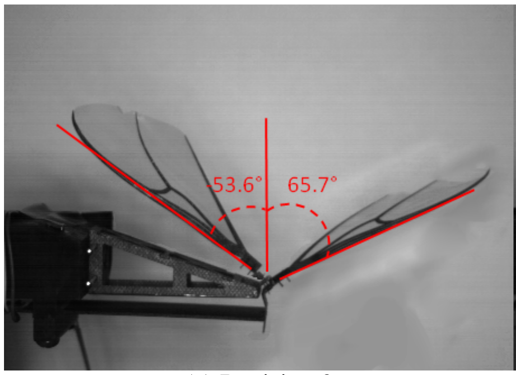
Figure 4.26: Video Stills for Revision 6 Wing at A = 0.45, Flapping at Resonance, 60° Angle Stop.



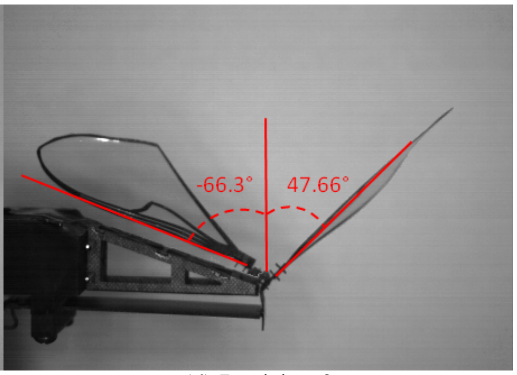
(a) Revision 0



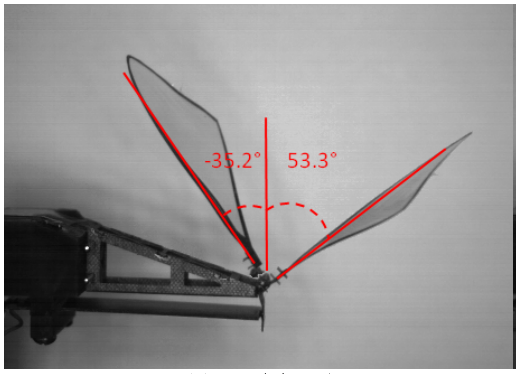
(b) Revision 1



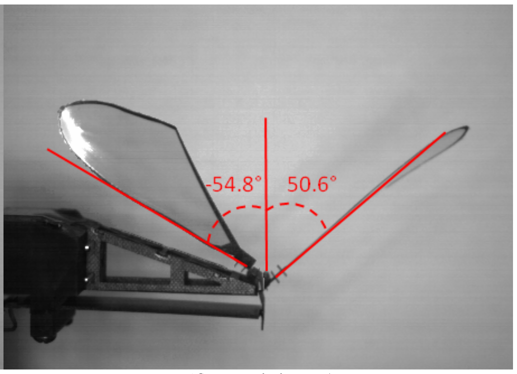
(c) Revision 2



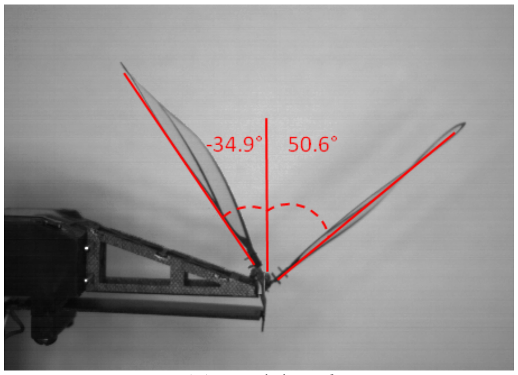
(d) Revision 3



(e) Revision 4



(f) Revision 5



(g) Revision 6

Figure 4.27: Stroke Angles (Φ) for Seven Different Designs, Flapping at Resonance, 60° Angle Stop, A = 0.45.

4.8 Vertical Force

Figure 4.28 serves to give a baseline comparison between present and previous research. Noticeably, the present research vertical force generation from the Revision 0 wing design falls short of that produced by DeLuca [12]. The previously discussed video capture of Revision 0 provided insight into possible reasons for the discrepancy. The lack of full supination present in Revision 0 caused a significant decreased vertical force production from previous research with the same wing shape [12]. Additionally, the age of the PZT used in testing could have affected the force production. DeLuca's research identified a degradation of PZT efficiency with continued use, which led to the use of a new PZT for each wing tested in that research. In the current research, only one PZT was used for each test sequence, this would not negatively impact Revision 0 results as that design was always tested first, however, it may have caused a decrease in the vertical force produced by subsequently tested wings. It is not known if the PZT degrades while not in use, an effective "shelf-life" may exist which would decrease lift produced by Revision 0. For those reasons, direct comparison of vertical force with Revision 0 of the present research is not valid since it is not consistent with the previously demonstrated capabilities of the wing design. Instead, comparisons must be made with those capabilities demonstrated by DeLuca [12]. Additionally, relative comparisons between the new designs of Revisions 1-6 are valid.

Table 4.7 shows the maximum vertical force achieved by each wing revision and the amplitude at which that occurred. For all of the wings tested, the maximum vertical force was achieved at the maximum amplitude tested (A = 0.45). The maximum vertical force values for each wing are the average of the full data set collected for that specific revision.

The RDS may be used to predict the changes in vertical force generated by the wing. Increasing the RDS causes an increase in the vertical force produced by the wing. Figure 4.29 shows a linear fit curve based on the data given in Table 4.7. In this figure, the vertical

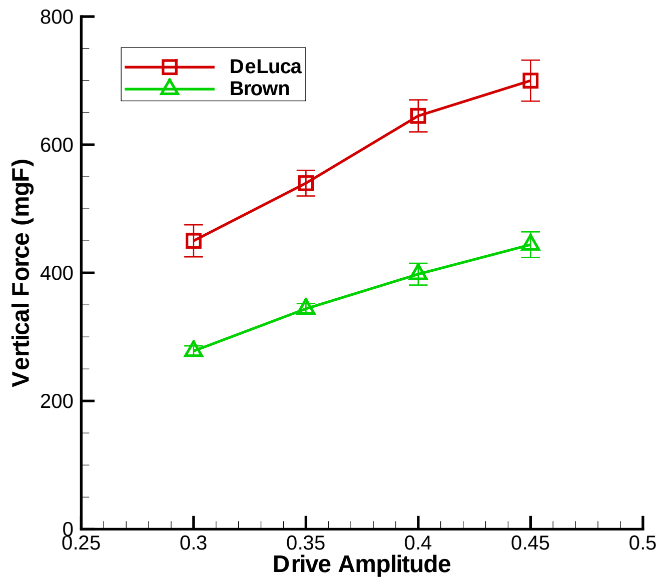


Figure 4.28: Comparison of Average Vertical Force Generation (mgF) for Original Wing Design at Varying Amplitude. Compares Results of DeLuca [12] and Present Research.

Table 4.7: Maximum Average Vertical Force Produced by Each Revision and their Relative Deduced Stiffness

Revision	Amplitude	Vertical Force (mgF)	Deduced Stiffness
0	0.45	443.4	1.00
1	0.45	558.4	0.76
2	0.45	594.4	1.03
3	0.45	604.9	0.87
4	0.45	533.0	0.76
5	0.45	494.2	0.76
6	0.45	453.2	0.59

force is given from 400 to 650 mgF, and the RDS from 0.5 to 1.1. Equation 4.3 gives the mathematical approximation for the linear fit that allows prediction of maximum vertical force generation based on the individual RDS. The Revision 0 wing has been excluded from this comparison since Figure 4.28 demonstrates its inaccurate representation of possible vertical force generation for that design.

$$\text{Vertical Force} = 266.04 + 345.29 \cdot \text{RDS} \tag{4.3}$$

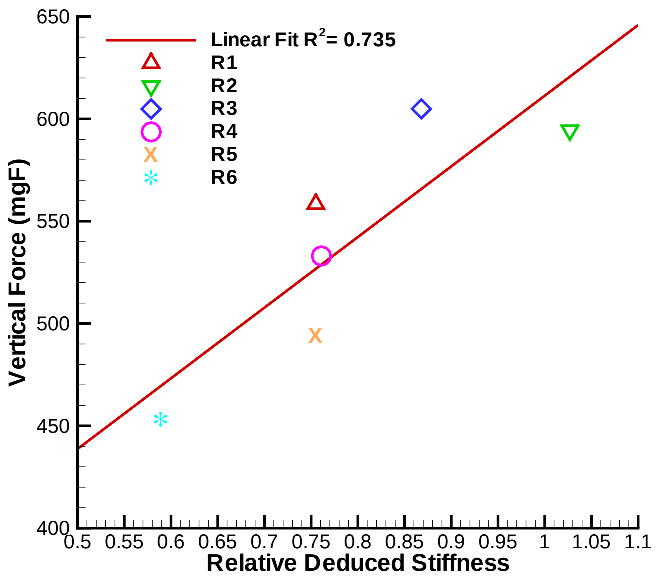


Figure 4.29: Comparison of Maximum Average Vertical Force (mgF) to Relative Deduced Stiffness of All Wings Excluding Revision 0. Flapping at Resonance, 60° Angle Stop.

Figure 4.30 shows the data of Table 4.7 in bar chart format. The vertical force is presented from 350 to 800 mgF. Additionally, the vertical force from previous research for the original design has been included for comparison [12]. The error bars signify one standard deviation. The green bars indicate the average of the actual vertical force produced by each design revision. The black boxes indicate the maximum possible vertical force generation predicted by the linear fit of Equation 4.3. There are no predictions made for the original wing of the previous research and Revision 0 of the present research since. The linear fit predicts the maximum vertical force generation accurately enough (σ = 30 mgF) to be used as an initial design tool.

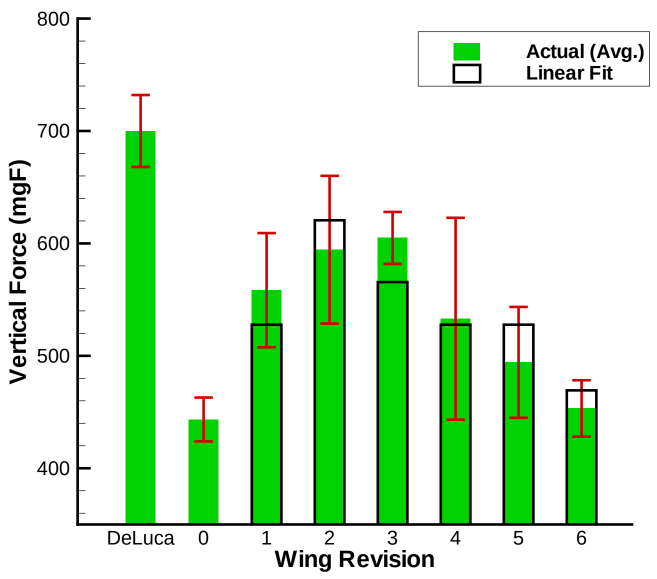
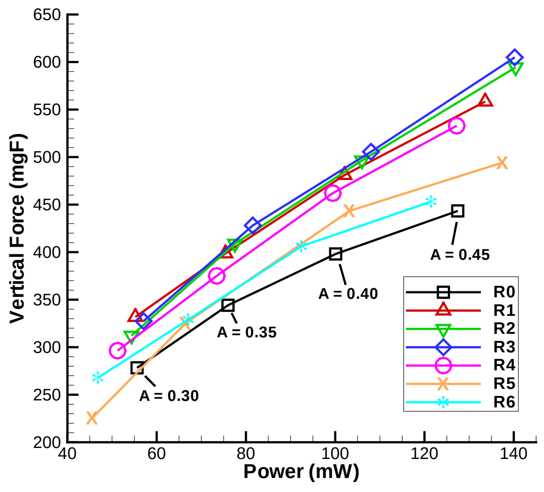


Figure 4.30: Comparison of Maximum Average and Predicted Vertical Force (mgF) by Revision. Flapping at Resonance, 60° Angle Stop.

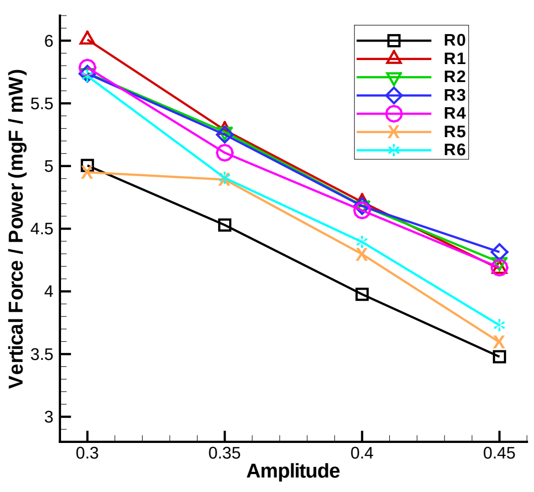
Revision 0 produces the least vertical force of all the designs. Revision 2 produces 15% less vertical force than the original wing design used in previous research. Revision 3 produces 14% less compared to the original wing. The lowest force production of the new designs is provided by Revision 6 which produces 35% less force than the original wing.

4.9 Power Consumption

The code in Appendix A saves power as an output. The power calculation is described briefly by Lindholm and Cobb [22] and in great detail by DeLuca [12, Section 4.8]. Figure 4.31a shows the power consumption for different vertical force values produced at four amplitudes (A = 0.30, 0.35, 0.40, 0.45). As amplitude increases, both the power consumed and vertical force produced increase. It is more instructive to look at Figure 4.31b which shows the vertical force to power ratio as a function of amplitude. A higher vertical force to power ratio indicates better performance where some wings are able to generate a significant amount of vertical force with lower power consumption. Revision 0 consumes the most power at all amplitudes for the least amount of vertical force produced, again this is due to the wing binding causing a decreased vertical force generation. At all amplitudes Revisions 1, 2, 3, and 4 are tightly grouped, producing a large amount of force for the least amount of power for this configuration.



(a) Vertical Force Versus Power.



(b) Vertical Force/Power Versus Amplitude.

Figure 4.31: Average Vertical Force (mgF) and Average Power (mW), Flapping at Resonance, 60° Angle Stop.

In reference to previous work, Figure 4.32 shows the vertical force produced for a given power consumption using the original wing design. The series include DeLuca [12] and the present research. The data points from DeLuca represent measured stroke angles (Φ), while the present research data points are plotted by amplitude (A). While Φ and A are not equal, they allow a relative comparison between the two data sets. Again, because the vertical force production was not the same, the current research uses more power to generate a given vertical force, an undesired result.

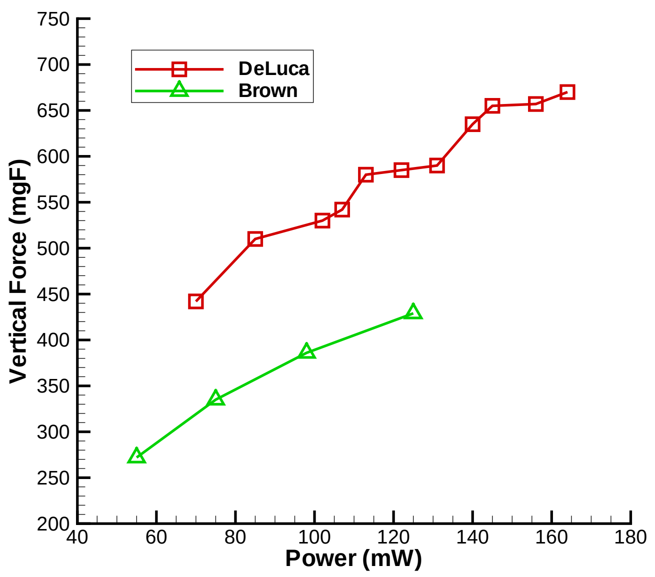


Figure 4.32: Average Vertical Force (mgF) Generated for a Given Average Power (mW) on Original Wing Design, Flapping at Resonance, 60° Angle Stop. Compares Results of DeLuca [12] and Present Research.

4.10 Chapter Summary

The results of the experimental research are presented. The FEA results indicated that the designs were not overly sensitive to removing mass from the middle of the structure, however, it would not be advisable to do the same at the attachment point. The manufacturing process for the wings was mostly uneventful, with the exception of needing to widen the trailing edge member on several designs. The mass of the wings was reduced successfully and a method for future designers to accurately predict the mass of a production design was presented. The RDS was discussed which allows for prediction of maximum vertical force generation for a specific design with accuracy suitable for the initial design stages. The best of the revised wings generated 14% less vertical force that the original design, but did so with a reduction in mass of 15% from that same design. Video capture was used to capture any remaining trends in the different wing designs. Chapter V will summarize in detail the conclusions from this research and present some recommended areas of future research.

V. Conclusions and Recommendations for Future Work

5.1 Research Conclusions

Several new processes were developed throughout this research effort. Most notably a new method of wing attachment was developed leading to increased testing repeatability in addition to consistency between tests. The wings were attached to the base with CrystalBond, while IC was used to attach the PZT to the base. The reversal of bonding agents used in prior research allowed for easy removal of the wings, while the base and PZT geometry remained consistent between tests. This new process answered the call for further research championed by Sladek [29].

A GUI was developed to allow the user to input test parameters consistently and without difficulty. The GUI allows for saving of settings files ensuring that the parameters used from one test to the next are the same. The saving of settings is not a new idea, however, the GUI implementation simplifies data entry and provides more awareness to the user.

The main emphasis of this research was to identify the plausibility of alternates to direct biomimetic wing structures for use in the AFIT FWMAV. In order to demonstrate that the current wing design is not the most desirable for our application, an experimental study of vertical force and mass was undertaken. The results show a lighter wing can execute the same motion without significant negative aeroelastic effects while generating substantial lift suitable for sustained flight. The aeroelastic effects (longitudinal bending, etc.) that were present did not cause low vertical force production in most of the designs.

Specifically, Revisions 2 and 3 provide consistently high vertical force-to-mass, and reduced the wing mass to 52.5 and 55.3 mg respectively. Compared with Revision 0, these two designs achieved approximately 6-9 mg mass savings per wing. No direct correlation was found between vertical force and natural frequency alone, however, the stiffness of the wing is an important factor which was identified in some of the video capture files where undesired non-linear wing behavior was seen.

Another interesting finding of the research was reduction of the vertical force-to-power ratio while still maintaining the same overall vertical force generation. Revisions 2 and 3 were consistently near the top of the vertical force-to-power curve. Notably, both of the wings had an arculus, but the other attributes of their respective designs varied substantially. Both offered significant improvements over the Revision 0 wing tested in this research, though not much insight can be gained from that comparison.

While a decrease of 9 mg mass is a modest reduction, when viewed in the context of the small scale of the FWMAV it comprises nearly a 15% reduction in mass per wing, which gives more margin for flight control, battery, and sensors. Additionally, the power consumption is reduced due to the decreased effort required to drive the wing, allowing for more flight time or a smaller battery. The 9 mg mass reduction per wing is a 1.16% mass reduction of the overall FWMAV based upon the average mass of the Hawkmoth identified by O'Hara [28].

The trailing edge member of Revisions 1,3,4,5, and 6 provides realization of an additional goal. The delamination experienced during DeLuca's [12] research did not occur on those wing revisions with the additional member. The total increase in surface area is minimal; however, the location of that surface provides a sturdy attachment point for the Mylar® wing membrane to the carbon fiber venation. The deleterious effects of delamination were not experienced on the wing designs with the added trailing edge member.

Most importantly, the Relative Deduced Stiffness identified in Equation 4.2 gives great insight into the design of a "good" wing. Those wings with values closest to or exceeding the original design performed the best consistently throughout testing. Increasing the deduced stiffness, as seen in Figure 4.29, produced an increase in vertical force resulting in a resonant frequency increase. Those with a RDS significantly lower than the original design did not fare well throughout the range of testing, and were susceptible to significant aeroelastic effects as evidenced by the video capture sequences shown in Chapter IV.

5.2 Future Opportunities

This research was an initial look into the reduction of wing mass. There are many opportunities to improve and extend this research. A significant first step would be to extend the test range for amplitude to ensure that the vertical force consistently increases with amplitude. Using the *AutoTune* functionality developed by DeLuca [12] would be beneficial in ensuring that the wings flapped symmetrically about the neutral axis.

Particle Image Velocimetry would be an excellent extension to the research. This would provide insight into the flow field around the different wing designs. By characterizing the flow, certain traits of the flow field around a “good” design could be identified and used to modify the wings.

Additionally, this research constrained the shape and planform area of the wing to that of the original design. Extending the research to examine larger or smaller wings would provide beneficial information and help to identify the most successful wing to be used in the production FWMAV.

Appendix A: MATLAB® Test Code

A.1 Single Wing Flapping Frequency Response Function Bias Drive

This code was initially written by Lt Col. DeLuca. It was adapted by Capt. Garrison Lindholm for speed and addition of a simulated test signal. Finally, during the present research, the code was modified to accept input from the user selected through the GUI. This code generates a chirp input to drive the PZT and measure the displacement produced. The frequency response function is then calculated.

```
% SWF_FRF_BD.m
% Written By: LtCol DeLuca
% Modified By: Garrison Lindholm
% 10 Dec 2012
%
% Inputs:
%     plotFlag: 1 == true (plot FRF), 0 == false
%     simFlag:  1 == true (simulation), 0 == false (hardware test)
%     maxVAC:              max amplitude for pzt life(volts)
%     amp:                % of max amplitude
%     avg:                number of averages to use
%     overlap:            % overlap
%     SampleSeconds:      how many seconds to sampe for (seconds)
%     TareSeconds:        how long of a Tare (seconds)
%     SampleRate:         Sample rate (Hz)
%     eraName:            'savename.mat' creates mat for use with era
%
% Outputs:
%     H1_Y1:              H1 FRF
%     H2_Y1:              H2 FRF
%     AVG_CX1_Y1:         Coherence
%     era mat file:       mat file to be used with era
%
% Summary: Script will perform will drive the pzt with Chirp input and
% measure the displacement.  It will then calculate the FRF using the
% random input and displacement measurements.  Most the FRF portion
% of the script was adapted from LtCol Deluca's bias drive based
% script.  Only changes were optimizations for speed, a simulated
% signal, and additional creation of ERA mat file.
%
% NI DAQ Configuration:
% Outputs:
% Channel   Cable
% 0         Bias Signal
% 1         Drive Signal
% Inputs:
% Channel   Cable
% 22        Displacement
% 21        ChanA Input Signal Voltage
```

```
%
% Modified By: LCDR Zach Brown
% 19 NOV 2013
% Made compatible with GUI by loading inputs from file

clear all
close all
clc

% Settings Load From GUI Inputs
load('Folder_Structure.mat');
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

%pathName = strcat('../',Fol.parm,'/');
loadFile = strcat(Fil.frf.path,'/',Fil.frf.full);
load(loadFile)
clear loadFile

% Settings loaded by GUI
% plotFlag: 1 is on, 0 is off
% simFlag: 1 is Simulation, 0 is Actual Run
% maxVAC, Bias
% amp: usually 0.05 works for testing
% avg: set # of averages (bins)to break the total
% data into 10 for testing
% overlap: Usually 0.50 works for testing
% simNoise: not used in actual testing
% SampleSeconds: Time in Seconds, usually 60 for testing
% TareSeconds: Time in Seconds, usually 1 for testing
% desiredCutFreq: Hz 60-70 usually for testing
% eraName

SampleRate    = desiredCutFreq*1.25*2; % Calculated to get desired freq
SampleNumber = round(SampleRate * SampleSeconds); % Total Samples
SampleTime = linspace(0,SampleSeconds,SampleNumber); %Time Vector
S.SampleSeconds = SampleSeconds;
S.SampleRate = SampleRate;
S.SampleNumber = SampleNumber;
S.SampleTime = SampleTime;
TareNumber  = round(TareSeconds*SampleRate); % Tare Samples
TareTime    = linspace(0,TareSeconds,TareNumber);
% Zero Voltage Output based on TareTime
TareData    = zeros(TareNumber,1);
% Account for Tare at Front and Back of Output Signal
TotalSeconds  = SampleSeconds + (TareSeconds*2);
% Account for Tare at Front and Back of Output Signal
TotalNumber = SampleNumber + (TareNumber*2);
TotalTime     = linspace(0,TotalSeconds,TotalNumber);
S.TareTime = TareTime;
S.TotalSeconds = TotalSeconds;
```

```matlab
S.TotalNumber = TotalNumber;
S.TotalTime = TotalTime;

%% Voltage output
ChirpTime = linspace(0,SampleSeconds/avg,SampleNumber/avg);
VR = ...
    amp.*(maxVAC/30)*chirp(ChirpTime,0,ChirpTime(end),(SampleRate/2))'...
     +(Bias/30);
VR = repmat(VR,avg,1);
VRTare = Bias/30*ones(length(TareData),1);
VR = [VRTare;VR;VRTare];
VR(end) = 0;

DC    = 2*Bias/30*ones(SampleNumber,1);
DCTare = 2*Bias/30*ones(length(TareData),1);
DC    = [DCTare;DC;DCTare];% Add TareData to "delay" flapping
DC(end) = 0;

%% Error Check on VR
if max(VR*30) > maxVAC+Bias+1
    errordlg('Amplitude Is Above Max Voltage!','Amplitude Error')
    fprintf('Test Stopped - Amplitude Error\n')
    return
end
if min(VR*30) < -maxVAC+Bias-1
    errordlg('Amplitude Is Below Min Voltage!','Amplitude Error')
    fprintf('Test Stopped - Amplitude Error\n')
    return
end

%% Either Simulation or Hardware test
switch simFlag
    case 1 % Simulation
        w1 = 25*2*pi;
        zeta1 = 0.3;
        w2 = 100*2*pi;
        zeta2 = 0.5;
        G1 = tf(0.01*w1^2,[1 2*zeta1*w1 w1^2]);
        G2 = tf(0.01*w2^2,[1 2*zeta2*w2 w2^2]);
        G3 = G1+G2;
        Simout= lsim(G3,VR,TotalTime,0,'zoh');
        dis    = Simout(TareNumber+1:TareNumber+SampleNumber);
        dis = dis+max(dis)*simNoise*rand(SampleNumber,1);
        X1 = VR(TareNumber+1:TareNumber+SampleNumber);
        Y1 = dis;

        % total number of data points collected = length of the data ...
            array
        N = size(X1,1);
        del = 1/SampleRate; % (1/hz)=sec --> time step between samples
        T = N*del;         % (sec) sets the period of the data set
        ws = 2*pi/del;     % (rad/s) sampling freq
```

```
% create a time vector from 0-->T, the length of N
% to plot the time history data
T_vec = linspace(0,N,N);

% sets the bin size as a function of # of avgs and % overlap
p = floor(N/(1+(avg-1)*(1-overlap)));
% sets p to the nyquist equivalent and it
% to the lowest integer value
p_n = floor(p/2);
% sets frequency of p to 80% to remove the aliasing from overlap
p_8 = floor(0.8*p_n);
% sets the frequency span of the data set = sample rate
fs = 1/del;
% sets the nyquist sampling frequency to
% 1/2 the sampling frequency
fn = (fs/2);
% sets the fn used in the calcs to 80% of
% the nyquist to remove the aliaising
fn_8 = floor(.8*fn);
% sets the frequency span and spacing for the reduced data set
w=0:fn_8/(p_8-1):fn_8;
% calcs the next 2^N power since the data set is not 2^n
NFFT = 2^nextpow2(p);
% sets hanning window = size of the bin set size
h_win=hann(p);

k=0; % set inital pointer index = 0
X1_mag = zeros(avg,p);
Y1_mag = zeros(avg,p);
X1_win = zeros(avg,p);
Y1_win = zeros(avg,p);
aPSD_X1 = zeros(avg,p);
aPSD_Y1 = zeros(avg,p);
cPSDX1_Y1 = zeros(avg,p);
cPSDY1_X1 = zeros(avg,p);
for i=1:avg
    X1_mag(i,:) = X1(k+1:k+p);
    Y1_mag(i,:) = Y1(k+1:k+p);
    X1_win(i,:) = X1_mag(i,:).*h_win';
    Y1_win(i,:) = Y1_mag(i,:).*h_win';
    %Calculate the auto PSD's for each sensor (Sxx)
    aPSD_X1(i,:) = fft(X1_win(i,:)).*conj(fft(X1_win(i,:)));
    aPSD_Y1(i,:) = fft(Y1_win(i,:)).*conj(fft(Y1_win(i,:)));
    %Calculate the cross PSD's for each
    % sensor (Sxy) input-->output
    cPSDX1_Y1(i,:) = fft(Y1_win(i,:)).*conj(fft(X1_win(i,:)));
    %Calculate the cross PSD's for each
    % sensor (Syx) output-->input
    cPSDY1_X1(i,:) = fft(X1_win(i,:)).*conj(fft(Y1_win(i,:)));
    % advances pointer index to middle of the current bin
    k=floor(k+overlap*p);
end
```

```
        %Calculate the average Cross and Auto PSD's from the ten bins
        %Avg Auto PSDs
        AVG_aPSD_X1 = mean(aPSD_X1,1);
        AVG_aPSD_Y1 = mean(aPSD_Y1,1);

        %Dump points in the Averaged Bin beyond 80% Fn
        AVG_aPSD_X1(p_8+1:p) = [];
        AVG_aPSD_Y1(p_8+1:p) = [];

        %Avg Cross PSDs
        AVG_cPSDX1_Y1 = mean(cPSDX1_Y1,1);
        AVG_cPSDY1_X1 = mean(cPSDY1_X1,1);

        %Dump points in the Averaged Bin beyond 80% Fn
        AVG_cPSDX1_Y1(p_8+1:p) = [];
        AVG_cPSDY1_X1(p_8+1:p) = [];

        %Calculate the Avg FRF's, H1=Sxy/Sxx, H2=Syy/Syx
        %where Sxx=input rand and the output is displacement
        H1_Y1 = AVG_cPSDX1_Y1./AVG_aPSD_X1;
        H2_Y1 = AVG_aPSD_Y1./AVG_cPSDY1_X1;

        %Calculate the %Coherence ratio (|Sxy^2|/(Syy * Sxx)
        AVG_CX1_Y1=(abs(AVG_cPSDY1_X1).^2./(AVG_aPSD_Y1.*AVG_aPSD_X1));

    case 0 % Hardware
        %% Get NI USB-6229 Data
        NI=daqhwinfo('nidaq');

        %% Create Analog Device object
        ao = analogoutput('nidaq',NI.InstalledBoardIds{1});
        ai = analoginput('nidaq',NI.InstalledBoardIds{1});

        %% Set Channels
        addchannel(ao,[0 1]); % Analog Out Channel 0 is Bias, 1 is Drive
        % Analog In(data back to computer)Channels 19
        addchannel(ai,[20 22]);
        %                           Collected Array Column
        %   20  = ChanB Input Signal Voltage  1
        %   22  = Displacement                2
        %

        %% Set AO Sample Options
        ao.SampleRate = SampleRate;

        %% Set AI Sample Options
        ai.SampleRate = SampleRate;
        ai.TriggerRepeat = 0;
        ai.SamplesPerTrigger = TotalNumber;

        %% Set TriggerType to Manual for fastest triggering
        set([ai ao],'TriggerType','Manual')

        %% Queue the output on 2 channels
        putdata(ao,[DC VR]);

        %% Start and Trigger the Output
        start([ai ao]);   % Start Analog Output
        trigger([ai ao]); % Start Analog Input

        %% Capture Analog Input Data from the buffer
        data_ai = getdata(ai);

        %%  Distance Data
        % Raw Voltage
        dist_volts  = data_ai(:,2);
        %find mean during tare
        b           = mean(dist_volts(1:length(TareData)));
        %12.53 is slope from calibrated tool and remove mean
        dist_inches = (dist_volts - b)/12.53;
        dis         = dist_inches * 25.4; %25.4mm/in
        X1          = data_ai(:,1);
        Y1          = dis;

        % total number of data points collected = length of the data ...
            array
        N = size(X1,1);
        del = 1/SampleRate; % (1/hz)=sec --> time step between samples
        T = N*del;          % (sec) sets the period of the data set
        ws = 2*pi/del;      % (rad/s) sampling freq
        % create a time vector from 0-->T, the length of
        % N to plot the time history data
        T_vec = linspace(0,N,N);

        % sets the bin size as a function of # of avgs and % overlap
        p = floor(N/(1+(avg-1)*(1-overlap)));
        % sets p to the nyquist equivalent and it
        % to the lowest integer value
        p_n = floor(p/2);
        % sets frequency of p to 80% to remove the aliasing from overlap
        p_8 = floor(0.8*p_n);
        % sets the frequency span of the data set = sample rate
        fs = 1/del;
        % sets the nyquist sampling frequency
        % to 1/2 the sampling frequency
        fn = (fs/2);
        % sets the fn used in the calcs to
        % 80% of the nyquist to remove the aliaising
        fn_8 = floor(.8*fn);
        % sets the frequency span and spacing for the reduced data set
        w=0:fn_8/(p_8-1):fn_8;
        % calcs the next 2^N power since the data set is not 2^n
        NFFT = 2^nextpow2(p);
        % sets hanning window = size of the bin set size
        h_win=hann(p);

        k=0; % set inital pointer index = 0
        X1_mag = zeros(avg,p);
        Y1_mag = zeros(avg,p);
        X1_win = zeros(avg,p);
        Y1_win = zeros(avg,p);
        aPSD_X1 = zeros(avg,p);
        aPSD_Y1 = zeros(avg,p);
        cPSDX1_Y1 = zeros(avg,p);
        cPSDY1_X1 = zeros(avg,p);
        for i=1:avg
            X1_mag(i,:) = X1(k+1:k+p);
            Y1_mag(i,:) = Y1(k+1:k+p);
            X1_win(i,:) = X1_mag(i,:).*h_win';
            Y1_win(i,:) = Y1_mag(i,:).*h_win';
            %Calculate the auto PSD's for each sensor (Sxx)
            aPSD_X1(i,:) = fft(X1_win(i,:)).*conj(fft(X1_win(i,:)));
            aPSD_Y1(i,:) = fft(Y1_win(i,:)).*conj(fft(Y1_win(i,:)));
            %Calculate the cross PSD's for each
            % sensor (Sxy) input-->output
            cPSDX1_Y1(i,:) = fft(Y1_win(i,:)).*conj(fft(X1_win(i,:)));
            %Calculate the cross PSD's for each
            % sensor (Syx) output-->input
            cPSDY1_X1(i,:) = fft(X1_win(i,:)).*conj(fft(Y1_win(i,:)));
            % advances pointer index to middle of the current bin
            k=floor(k+overlap*p);
        end
        %Calculate the average Cross and Auto PSD's from the ten bins
        %Avg Auto PSDs
        AVG_aPSD_X1 = mean(aPSD_X1,1);
        AVG_aPSD_Y1 = mean(aPSD_Y1,1);

        %Dump points in the Averaged Bin beyond 80% Fn
        AVG_aPSD_X1(p_8+1:p) = [];
        AVG_aPSD_Y1(p_8+1:p) = [];

        %Avg Cross PSDs
        AVG_cPSDX1_Y1 = mean(cPSDX1_Y1,1);
        AVG_cPSDY1_X1 = mean(cPSDY1_X1,1);

        %Dump points in the Averaged Bin beyond 80% Fn
        AVG_cPSDX1_Y1(p_8+1:p) = [];
        AVG_cPSDY1_X1(p_8+1:p) = [];

        %Calculate the Avg FRF's, H1=Sxy/Sxx, H2=Syy/Syx
        %where Sxx=input rand and the output is displacement
        H1_Y1 = AVG_cPSDX1_Y1./AVG_aPSD_X1;
        H2_Y1 = AVG_aPSD_Y1./AVG_cPSDY1_X1;

        %Calculate the %Coherence ratio (|Sxy^2|/(Syy * Sxx)
        AVG_CX1_Y1=(abs(AVG_cPSDY1_X1).^2./(AVG_aPSD_Y1.*AVG_aPSD_X1));

        %% Clean and Remove ai & ao from memory
        delete(ai)
        delete(ao)
    otherwise
        disp('Unknown simulation flag parameter')
        return
end

if plotFlag == 1
    subplot(3,1,1)
    plot(w,20*log10(abs(H1_Y1)),w,20*log10(abs(H2_Y1)))
    grid on
    ylabel('Displacement (dB)')
    subplot(3,1,2)
    plot(w,rad2deg(angle(H1_Y1)),w,rad2deg(angle(H2_Y1)))
    grid on
    ylabel('Phase (deg)')
    subplot(3,1,3)
    plot(w,AVG_CX1_Y1)
    grid on
    ylabel('Coherence')
    xlabel('Frequency (Hz)')
    ylim([0 1])
end
if length(Fil.era.full) > 1
    FreqV = w';
    frf = transp(H1_Y1);

    saveFile = strcat(Fil.era.path, Fil.era.full);
    save(saveFile,'FreqV','frf');
    clear saveFile
end
uiwait(msgbox('Ready To Execute ERA?','Back to Flapper Executable'));
cd ..
Flapper_Front_End
```

A.2 Single Wing Flapping Test Bias Drive

This code was written by Capt. Garrison Lindholm and modified during the present research. The modifications only implemented GUI input of test parameters and saving of the results to a MATLAB® data file. The code performs the tests indicated by the user.

```
%% SWF_Test_BD.m
% Garrison Lindholm
% 10 Dec 2012
%
% Inputs:
```

```
%     Bias:     Bias Voltage (volts)
%     w:        frequency (rad/sec)
%     eta:      bias parameter
%     A:        Amplitude Right wing (% of max (300-0 Volts AC)
%     tau:      Stroke reversal time shift Right
%     M1pR:     Magnitude 1st harmonic
%     M2pR:     Magnitude 2nd harmonic
%     beta1pR:  Phase 1st harmonic
%     beta2pR:  Phase 2nd harmonic
%     samples:  Number of samples per test point
%     simFlag:  1 == true (simulation), 0 == false (hardware test)
%     testFlag: 2 == BABM, 1 == Frequency, 0 == Single test cases
%
% Outputs:
%     data: Data structure with inputs, outputs, units, and information
%           pertaining to the tests
%
% Summary: Script will perform the designed set of experiments
% on a single wing flapper with a bias drive.  Various experiments
% can be preformed by varying BABM parameters, frequency, or single
% test case.  The data will be stored in a array of structures,
% 1 structure per test.
%
% NI DAQ Configuration:
% Outputs:
% Channel   Cable
% Analog Out Channel 0-3
% 0 = ChanA (DC Bias)
% 1 = ChanB (Sine Signal Rt Wing)
% 3 = Video Trigger (5V TTL )
% Inputs:
% Channel   Cable
% 0         ChanA Voltage
% 1         ChanA Current
% 2         ChanB Voltage
% 3         ChanB Current
% 6         Nano17 Channel 1
% 7         Nano17 Channel 2
% 16        Nano17 Channel 3
% 17        Nano17 Channel 4
% 18        Nano17 Channel 5
% 19        Nano17 Channel 6
% 22        Displacement
%
% Modified By: LCDR Zach Brown
% Date: 9 NOV 13
% Made compatible with GUI by loading inputs from file
% BABM Test Matrix Format col: 1 = A, 2 = tau, 3 = eta

close all; clear all; clc;
tic
load('Folder_Structure.mat');
```

```

clear Fil
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

%Load Parameters from Input File
loadFile = strcat(Fil.BD.path,Fil.BD.full);
load(loadFile)
clear loadFile

loadFile = strcat(Fil.SC.path,Fil.SC.full);
load(loadFile)
clear loadFile


switch testFlag
    case 0 %Single Test
        w = wn*2*pi;
    case 1 %Range of Frequencies
        w = (w1:step:w2)'.*2.*pi;
    case 2
    %If it is a BABM Test Load the Test Matrix col:
    % 1 = A, 2 = tau, 3 = eta
        loadFile = strcat(Fil.TM.path,Fil.TM.full);
        load(loadFile);
        w = wn*2*pi;
        eta  = 0.0; %BABM will Overwrite
        AR   = 0.0; %BABM will Overwrite
        tauR = 0.0; %BABM will Overwrite
end

BABMs.AR = AR;
BABMs.AL = AR;
BABMs.tauR = tauR;
BABMs.tauL = tauR;
BABMs.eta = eta;
BABMs.Bias = Bias;
M1pR = abs(H1);
M2pR = abs(H2);
beta1pR = angle(H1);
beta2pR = angle(H2);
BABMs.const.M1pR = M1pR;
BABMs.const.M2pR = M2pR;
BABMs.const.M1pL = M1pR;
BABMs.const.M2pL = M2pR;
BABMs.const.beta1pR = beta1pR;
BABMs.const.beta2pR = beta2pR;
BABMs.const.beta1pL = beta1pR;
BABMs.const.beta2pL = beta2pR;


```

```
usingFF_compensator = 0; %% set to 1 if using FF compensation
if usingFF_compensator == 1
    load SWF_300_2_coef.mat
    coefL = coeffvalues(SWF_300_2_A);
    coefL = coefL(1);
end

%% Run Test
cd ..
cd(Fol.supp)
switch testFlag
    case 0 %single test case
        if usingFF_compensator == 1
            Lift_0 = feval(SWF_300_2_Tau,0);
            Lift_Tau = feval(SWF_300_2_Tau,tauR);
            Lift_N = Lift_0 - Lift_Tau;
            A_N = Lift_N/coefL;
            AR = AR+A_N;
        end
        fprintf('Frequency = %g Hz\n',w/2/pi)
        fprintf('Amplitude = %g%%\n',AR*100)
        fprintf('Split-Cycle = %g \n',tauR)
        fprintf('Bias = %g \n',eta)
        fprintf('--------------------\n\n')
        data.Test(1) = SWFt_BD(maxVAC,Bias,w,BABMs,samples,simFlag);
    case 1 % varying frequency case
        for i = 1:length(w)
            fprintf('Frequency = %g Hz\n',w(i)/2/pi)
            if usingFF_compensator == 1

            end
            data.Test(i) = ...
                SWFt_BD(maxVAC,Bias,w(i),BABMs,samples,simFlag);
        end
    case 2 % varying BABM parameters case
        for i = 1:length(TM)
            BABMs.AR   = TM(i,1);
            BABMs.AL   = TM(i,1);
            BABMs.tauR = TM(i,2);
            BABMs.tauL = TM(i,2);
            BABMs.eta  = TM(i,3);
            if usingFF_compensator == 1
                Lift_0 = feval(SWF_300_2_Tau,0);
                Lift_Tau = feval(SWF_300_2_Tau,abs(BABMs.tauR));
                Lift_N = Lift_0 - Lift_Tau;
                BABMs.A_N = Lift_N/coefL;
                BABMs.AR = BABMs.AR+BABMs.A_N;
            end
            fprintf('--------------------\n')
            fprintf('Test Case = %g\n',i)
            fprintf('Amplitude = %g%%\n',BABMs.AR*100)
            fprintf('Split-Cycle = %g \n',BABMs.tauR)
            fprintf('Bias = %g \n',BABMs.eta)
            data.Test(i) = SWFt_BD(maxVAC,Bias,w,BABMs,samples,simFlag);
            fprintf('Avg forceX = %g grams force\n',...
                data.Test(i).ave.forceX *...
                data.Test(i).Units.NewtonsToGramsF);
            fprintf('--------------------\n\n')
            fileNameForClipboard = sprintf('%s_Amp=%3.2f',...
                Fil.res.partial, BABMs.AR);
            clipboard('copy', fileNameForClipboard);
            helpText = sprintf('Save the video file for the ...
               completed test case, %s_Amp=%3.2f',...
                Fil.res.partial, BABMs.AR);
            uiwait(helpdlg(helpText))
        end
    otherwise
        disp('Unknown Test Case')
end
clearvars -except data Fol
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

saveFile = strcat(Fil.res.path,Fil.res.full);
save(saveFile,'-struct','data')
clear saveFile

cd ..
toc
```

A.3 Find Split Cycle Parameters

This code was written by Capt. Garrison Lindholm to determine the frequency response of the system.

```
%% Find SC Parameters
function Find_SC_Parameters(wn)

load('Folder_Structure.mat');

clear Fil
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

loadFile = strcat(Fil.fit.path, Fil.fit.full);
load(loadFile)
clear loadFile

Fil.SC.full = strcat('SWF_',Fil.era.partial,'_SC.mat');
Fil.SC.path = strcat(Fol.work,'/');

```

```
saveFile = strcat(Fol.work,'/','File_Structure.mat');
save(saveFile,'Fil')
clear saveFile


saveFile = strcat(Fil.SC.path, Fil.SC.full);
G = ss(aera,bera,cera,dera,Ts);
H1 = freqresp(G,wn,'Hz');
H2 = freqresp(G,2*wn,'Hz');
save(saveFile,'H1','H2')

loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

cd ..
Flapper_Front_End
```

Appendix B: MATLAB® GUI Code

B.1 Main GUI Program

The idea for this program was based on work by DeLuca [12] and O'Hara [28]. Instead of an adaptation the code was rewritten from the ground up to facilitate better interaction with the Test Code and perform the functions needed by this research. This program delivers the main menu from which all other choices and inputs are accessed.

```

 function Flapper_Front_End
%**********************************************************************
% Front End GUI for Flapper
% Initial Idea From Code By: Lt Col Deluca and Maj Ohara
% Written By: LCDR Zach Brown
% 9 NOV 2013
% Front End to the GUI.  All slections and calls are made from
% this program
%**********************************************************************

close all; clear all; clc;
 %% Folder Structure
cd 01.Main_Files
if exist('Folder_Structure.mat','file') == 2
    load('Folder_Structure.mat');
    cd ..
else
    cd ..
    Fol.base = pwd;
    Fol.main = '01.Main_Files';
    Fol.era  = '01.Main_Files/Ezera_71';
    Fol.supp = '02.Supporting_Files';
    Fol.plot = '03.Plotting_Files';
    Fol.parm = '04.Parameters_Files';
    Fol.out  = 'Results';
    Fol.work = strcat(Fol.base,'/',Fol.parm);

    pathName = strcat(Fol.main,'/');
    fname = 'Folder_Structure';
    SaveFile = strcat(pathName, fname);
    save(SaveFile,'Fol')
    clear pathName fname SaveFile
end

cd(Fol.work)
if exist('File_Structure.mat','file') == 2
    clear Fil
    load('File_Structure.mat');

```

```matlab
    if isfield(Fil,'res') == 0
        Fil.res.full = strcat('Results_',datestr(now,...
            'ddmmmyy_HHMM'),'.mat');
        Fil.res.path = strcat(Fol.work,'/');
        saveFile = strcat(Fol.work,'/','File_Structure');
        save(saveFile,'Fil')
        clear saveFile
    end
else
    idx = find(Fol.base == '\',1,'last');
    Fil.TM.path = strcat(Fol.base(1:idx),Fol.out,'/');
    Fil.TM.full = 'TS_SweepA-30-45.mat';
    Fil.res.full = ...
       strcat('Results_',datestr(now,'ddmmmyy_HHMM'),'.mat');
    Fil.res.path = strcat(Fol.work,'/');
    saveFile = strcat(Fol.work,'/','File_Structure');
    save(saveFile,'Fil')
    clear saveFile idx
end
cd(Fol.base)

%% Create Popup Menu To Choose Functions
S.fh = figure('units','pixels',...
              'position',[1100 500 400 400],...
              'menubar','none',...
              'name','Main Menu',...
              'numbertitle','off',...
              'resize','off');

S.Auto.grp = uibuttongroup('visible','on','units','pixels','pos',...
    [105 355 140 35]);
% Create two radio buttons in the button group.
S.Auto.rdo2 = uicontrol('Style','radiobutton','String','Manual',...
    'pos',[70 2 60 30],'parent',S.Auto.grp,'HandleVisibility','off');
S.Auto.rdo1 = uicontrol('Style','radiobutton','String','Auto',...
    'pos',[10 2 60 30],'parent',S.Auto.grp,'HandleVisibility','off');

S.Name.grp = uibuttongroup('visible','on','units','pixels','pos',...
    [105 277 140 20]);
% Create two radio buttons in the button group.
S.Name.rdo1 = uicontrol('Style','radiobutton','String','A-Name',...
    'pos',[10 2 60 15],'parent',S.Name.grp,'HandleVisibility','off');
S.Name.rdo2 = uicontrol('Style','radiobutton','String','M-Name',...
    'pos',[70 2 60 15],'parent',S.Name.grp,'HandleVisibility','off');


S.Work.txt1 = uicontrol('Style', 'text',...
    'String', 'Curren Working Dir',...
    'Fontweight', 'bold',...
    'pos', [25 300 75 50]);

S.Work.txt2 = uicontrol('Style', 'text',...
    'String', Fol.work,...
    'pos', [105 300 200 50]);

S.Work.btn = uicontrol('Style', 'pushbutton',...
    'String', 'Change',...
    'Fontweight', 'bold',...
    'pos', [310 300 65 50]);

S.Res.txt1 = uicontrol('Style', 'text',...
    'String', 'Results Saved Here',...
    'Fontweight', 'bold',...
    'pos', [25 235 75 40]);

S.Res.txt2 = uicontrol('Style', 'text',...
    'String', Fil.res.full,...
    'visible', 'off',...
    'pos', [105 235 200 40]);

S.Res.btn = uicontrol('Style', 'pushbutton',...
    'String', 'Change',...
    'Fontweight', 'bold',...
    'visible', 'off',...
    'pos', [310 235 65 40]);

S.Res.txt3 = uicontrol('Style', 'text',...
    'String', 'Results_',...
    'Fontweight', 'bold',...
    'pos', [105 245 75 20]);

S.Res.ed1 = uicontrol('Style', 'edit',...
    'String', 'R0A',...
    'Fontweight', 'bold',...
    'pos', [185 245 50 20]);

S.Res.txt4 = uicontrol('Style', 'text',...
    'String', strcat('_',datestr(now,'ddmmmyy_HHMM'),'.mat'),...
    'Fontweight', 'bold',...
    'pos', [240 245 120 20]);

S.RUN = uicontrol('Style', 'pushbutton',...
    'String', 'Run',...
    'visible', 'off',...
    'pos', [25 200 350 30]);

S.FRF = uicontrol('Style', 'pushbutton',...
    'String', 'Step 1 - Frequency Response Function - Single Wing',...
    'visible', 'on',...
    'pos', [25 200 350 30]);

S.ERA = uicontrol('Style', 'pushbutton',...
    'String', 'Step 2 - ERA - Find Curve Fit',...
    'visible', 'on',...
    'pos', [25 150 350 30]);

S.SCP = uicontrol('Style', 'pushbutton',...
    'String', 'Step 3 - Find SC Parameters',...
    'visible', 'on',...
    'pos', [25 100 350 30]);

S.BABM = uicontrol('Style', 'pushbutton',...
    'String', 'Step 4 - Test',...
    'visible', 'on',...
    'pos', [25  50 350 30]);



%% Set Call Functions
set(S.Auto.grp,'SelectionChangeFcn',{@selcbk,S});
set(S.Name.grp,'SelectionChangeFcn',{@selcbk2,S});
set([S.Work.btn,S.Res.btn,S.RUN,S.FRF,S.ERA,S.SCP,S.BABM],...
    'Call', {@openProg,S,Fol,Fil});
function selcbk2(source,eventdata,varargin)
% selectedTest = get(get(source,'SelectedObject'),'String');
selectedTest = get(source,'SelectedObject');
[S] = varargin{[1,1]}; %Get calling structure.
switch selectedTest
    case S.Name.rdo1
       autoVis = 'on';
       manVis = 'off';
    case S.Name.rdo2
        autoVis = 'off';
        manVis = 'on';
end
set(S.Res.txt2,'visible',manVis);
set(S.Res.btn,'visible',manVis);
set(S.Res.ed1, 'visible', autoVis);
set(S.Res.txt3, 'visible', autoVis);
set(S.Res.txt4, 'visible', autoVis);

function selcbk(source,eventdata,varargin)
selectedTest = get(source,'SelectedObject');
[S] = varargin{[1,1]}; %Get calling structure.
switch selectedTest
    case S.Auto.rdo1
       autoVis = 'on';
       manVis = 'off';
    case S.Auto.rdo2
        autoVis = 'off';
        manVis = 'on';
end
set(S.RUN,'visible',autoVis);
set(S.FRF,'visible',manVis);
set(S.ERA,'visible',manVis);
set(S.SCP,'visible',manVis);
set(S.BABM,'visible',manVis);

function [] = openProg(varargin)
    % Get calling handle and structures.
    [h,S,Fol,Fil] = varargin{[1,3,4,5]};
    switch h
        case S.Work.btn
            dirName = uigetdir(Fol.work,...
                'Choose a Working Directory');
            if (dirName ≠ 0)
                Fol.work = dirName;
                clear dirName
            end
            set(S.Work.txt2, 'String',Fol.work);
            pathName = strcat(Fol.main,'/');
            fname = 'Folder_Structure';
            saveFile = strcat(pathName, fname);
            save(saveFile,'Fol')
            clear pathName fname saveFile
            Flapper_Front_End
        case S.Res.btn
            filSpec = strcat(Fil.res.path,Fil.res.full);
            [filename,pathname] = uiputfile(filSpec,'Data Filename');
            if (filename ≠ 0)
                Fil.res.full = filename;
                Fil.res.path = pathname;
            end
            set(S.Res.txt2,'String',Fil.res.full);
            saveFile = strcat(Fol.work,'/','File_Structure.mat');
            save(saveFile,'Fil')
            clear saveFile
            cd(Fol.work)
            clear Fil
            load('File_Structure.mat');
            cd(Fol.base)
            Flapper_Front_End
        case S.RUN
            selectedNaming = get(S.Name.grp,'SelectedObject');
            switch selectedNaming
                case S.Name.rdo1
                    Fil.res.partial = get(S.Res.ed1,'String');
                    Fil.res.full = strcat(get(S.Res.txt3,'String'),...
                        get(S.Res.ed1,'String'),get(S.Res.txt4,'String'));
                    saveFile = ...
                       strcat(Fol.work,'/','File_Structure.mat');
                    save(saveFile,'Fil')
                    clear saveFile
                case S.Name.rdo2
            end

            cd(Fol.main);

            SWF_FRF_GUI();
            uiwait;
            disp 'SWF FRF Complete'

            cd(Fol.era)
            ezera;
            uiwait;
            uiwait;
            disp 'ERA Complete'

            close all;

            clear Fil
            loadFile = strcat(Fol.work,'/','File_Structure.mat');
            load(loadFile);
            clear loadFile

            loadFile = strcat(Fil.frf.path, Fil.frf.full);
            load(loadFile,'wnz');
            clear loadFile

            cd(Fol.main);
            Find_SC_Parameters(wnz(1,2));

            clear Fil
            loadFile = strcat(Fol.work,'/','File_Structure.mat');
            load(loadFile);
            clear loadFile
            disp 'Find SC Parameters Complete'

            close all;
            cd(Fol.main);
            SWF_Test_GUI;
            uiwait;
            disp 'Test Complete'
        case S.FRF %FRF
            selectedNaming = get(S.Name.grp,'SelectedObject');
            switch selectedNaming
                case S.Name.rdo1
                    Fil.res.partial = get(S.Res.ed1,'String');
                    Fil.res.full = strcat(get(S.Res.txt3,'String'),...
                        get(S.Res.ed1,'String'),get(S.Res.txt4,'String'));
                    saveFile = ...
                       strcat(Fol.work,'/','File_Structure.mat');
                    save(saveFile,'Fil')
                    clear saveFile
                case S.Name.rdo2
            end

            clear Fil
            loadFile = strcat(Fol.work,'/','File_Structure.mat');
            load(loadFile);
            clear loadFile

            cd(Fol.main);
            SWF_FRF_GUI;
        case S.ERA %ERA
            %close all;
            delete(S.fh);
            cd(Fol.era);
            ezera;

        case S.SCP %Find SC Parameters
            close all;

            clear Fil
            loadFile = strcat(Fol.work,'/','File_Structure.mat');
            load(loadFile);
            clear loadFile

            loadFile = strcat(Fil.frf.path, Fil.frf.full);
            load(loadFile,'wnz');
            clear loadFile

            cd(Fol.main);
            Find_SC_Parameters(wnz(1,2));

            clear Fil
            loadFile = strcat(Fol.work,'/','File_Structure.mat');
            load(loadFile);
            clear loadFile
        case S.BABM %Test
            close all;
            cd(Fol.main);
            SWF_Test_GUI;
    end
```

B.2 Frequency Response Function GUI Code

This code was written during the present research to take user inputs and store them into a file for use during the Frequency Response Function (FRF) identification portion of the code.

```
function [] = SWF_FRF_GUI()
close all; clear all; clc;
load('Folder_Structure.mat')
%% Set the Figure
S.fh = figure('units','pixels',...
              'position',[400 300 400 675],...
              'menubar','none',...
              'name','SWF_FRF_BD Options',...
              'numbertitle','off',...
```

```
               'resize','off');

%% Position Vector
LT = 25;
UP = [625 600 575 550 525 500 475 450 425 400 375 350 325 300 275 ...
   250 225 200 175 150 125 100 75 50 25];

%% AC Voltage Slider
S.lbl.VAC = uicontrol('Style','text',...
    'Position',[LT UP(2) 360 15],...
    'String','Enter Max AC Voltage (maxVAC) (1 - 250)');

VAC.slMin = 1;
VAC.slMax = 250;
VAC.slStep(1) = 1/(VAC.slMax-VAC.slMin);
VAC.slStep(2) = 1/(VAC.slMax-VAC.slMin)*10;
S.sl.VAC = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(3) 250 20],...
                 'min',VAC.slMin,'max',VAC.slMax,...
                 'SliderStep',[VAC.slStep(1) VAC.slStep(2)],...
                 'val',200);
S.ed.VAC = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(3) 100 20],...
                 'fontsize',16,...
                 'string','200');
set([S.ed.VAC,S.sl.VAC],'call',{@ed_call,S});  % Shared Callback.

%% Bias Slider
S.lbl.Bias = uicontrol('Style','text',...
    'Position',[LT UP(4) 360 15],...
    'String','Enter Bias (1 - 100)');
Bias.slMin = 1;
Bias.slMax = 100;
Bias.slStep(1) = 1/(Bias.slMax-Bias.slMin);
Bias.slStep(2) = 1/(Bias.slMax-Bias.slMin)*10;
S.sl.Bias = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(5) 250 20],...
                 'min',Bias.slMin,'max',Bias.slMax,...
                 'SliderStep',[Bias.slStep(1) Bias.slStep(2)],...
                 'val',100);
S.ed.Bias = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(5) 100 20],...
                 'fontsize',16,...
                 'string','100');
set([S.ed.Bias,S.sl.Bias],'call',{@ed_call,S});  % Shared Callback.

%% Amplitude Slider
S.lbl.Amp = uicontrol('Style','text',...
    'Position',[LT UP(6) 360 15],...
    'String','Enter Amplitude (.01 - 0.65)');
Amp.slMin = 0.01;
Amp.slMax = 0.65;
Amp.slStep(1) = 0.01/(Amp.slMax-Amp.slMin);
Amp.slStep(2) = 0.05/(Amp.slMax-Amp.slMin);
S.sl.Amp = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(7) 250 20],...
                 'min',Amp.slMin,'max',Amp.slMax,...
                 'SliderStep',[Amp.slStep(1) Amp.slStep(2)],...
                 'val',0.05);
S.ed.Amp = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(7) 100 20],...
                 'fontsize',16,...
                 'string','0.05');
set([S.ed.Amp,S.sl.Amp],'call',{@ed_call,S});  % Shared Callback.

%% Number of Averages Slider
S.lbl.Avg = uicontrol('Style','text',...
    'Position',[LT UP(8) 360 15],...
    'String','Set # of Averages (bins)to Break the Total Data Into ...
        (1-20)');
Avg.slMin = 1;
Avg.slMax = 20;
Avg.slStep(1) = 1/(Avg.slMax-Avg.slMin);
Avg.slStep(2) = 5/(Avg.slMax-Avg.slMin);
S.sl.Avg = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(9) 250 20],...
                 'min',Avg.slMin,'max',Avg.slMax,...
                 'SliderStep',[Avg.slStep(1) Avg.slStep(2)],...
                 'val',10);
S.ed.Avg = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(9) 100 20],...
                 'fontsize',16,...
                 'string','10');
set([S.ed.Avg,S.sl.Avg],'call',{@ed_call,S});  % Shared Callback.

%% Overlap Slider
S.lbl.Ovrlp = uicontrol('Style','text',...
    'Position',[LT UP(10) 360 15],...
    'String','Set % Overlap - ??');
Ovrlp.slMin = 0.1;
Ovrlp.slMax = 1;
Ovrlp.slStep(1) = .1/(Ovrlp.slMax-Ovrlp.slMin);
Ovrlp.slStep(2) = .3/(Ovrlp.slMax-Ovrlp.slMin);
S.sl.Ovrlp = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(11) 250 20],...
                 'min',Ovrlp.slMin,'max',Ovrlp.slMax,...
                 'SliderStep',[Ovrlp.slStep(1) Ovrlp.slStep(2)],...
                 'val',0.5);
S.ed.Ovrlp = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(11) 100 20],...
                 'fontsize',16,...
                 'string','0.5');
set([S.ed.Ovrlp,S.sl.Ovrlp],'call',{@ed_call,S});  % Shared Callback.

%% Sim Noise
S.lbl.Noise = uicontrol('Style','text',...
    'Position',[130 UP(1)+ 15 145 15],...
    'String','Sim Noise');
Noise.slMin = 0.01;
Noise.slMax = 0.1;
Noise.slStep(1) = .01/(Noise.slMax-Noise.slMin);
Noise.slStep(2) = .03/(Noise.slMax-Noise.slMin);
S.sl.Noise = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[130 UP(1) 145 15],...
                 'min',Noise.slMin,'max',Noise.slMax,...
                 'SliderStep',[Noise.slStep(1) Noise.slStep(2)],...
                 'val',0.05);
S.ed.Noise = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(1) 100 30],...
                 'fontsize',16,...
                 'string','0.05');
set([S.ed.Noise,S.sl.Noise],'call',{@ed_call,S});  % Shared Callback.
set([S.ed.Noise,S.sl.Noise,S.lbl.Noise],'visible','off')

%% Simulation Option
opt2 = 'Simulation';
S.check2=uicontrol('Style', 'checkbox',...
    'String', opt2,...
    'Position', [LT UP(1) 100 30],...
    'Value',0);
set([S.check2],'call',{@simOptions,S});  % Shared Callback.

%% Sample Time
S.lbl.SampLen = uicontrol('Style','text',...
    'Position',[LT UP(12) 360 15],...
    'String','Select Sample Time in Seconds (30-180)');
SampLen.slMin = 30;
SampLen.slMax = 180;
SampLen.slStep(1) = 1/(SampLen.slMax-SampLen.slMin);
SampLen.slStep(2) = 20/(SampLen.slMax-SampLen.slMin);
S.sl.SampLen = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(13) 250 20],...
                 'min',SampLen.slMin,'max',SampLen.slMax,...
                 'SliderStep',[SampLen.slStep(1) SampLen.slStep(2)],...
                 'val',60);
S.ed.SampLen = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(13) 100 20],...
                 'fontsize',16,...
                 'string','60');
set([S.ed.SampLen,S.sl.SampLen],'call',{@ed_call,S});  % Shared ...
    Callback.

%% Tare Time
S.lbl.TareLen = uicontrol('Style','text',...
    'Position',[LT UP(14) 360 15],...
    'String','Select Tare Time in Seconds (0-10)');
TareLen.slMin = 0;
TareLen.slMax = 10;
TareLen.slStep(1) = 1/(TareLen.slMax-TareLen.slMin);
TareLen.slStep(2) = 3/(TareLen.slMax-TareLen.slMin);
S.sl.TareLen = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(15) 250 20],...
                 'min',TareLen.slMin,'max',TareLen.slMax,...
                 'SliderStep',[TareLen.slStep(1) TareLen.slStep(2)],...
                 'val',1);
S.ed.TareLen = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(15) 100 20],...
                 'fontsize',16,...
                 'string','1');
set([S.ed.TareLen,S.sl.TareLen],'call',{@ed_call,S});  % Shared ...
    Callback.

%% Cut Frequency
S.lbl.CutHz = uicontrol('Style','text',...
    'Position',[LT UP(16) 360 15],...
    'String','Select Cut Frequency in Hz (50-80)');
CutHz.slMin = 50;
CutHz.slMax = 80;
CutHz.slStep(1) = 1/(CutHz.slMax-CutHz.slMin);
CutHz.slStep(2) = 5/(CutHz.slMax-CutHz.slMin);
S.sl.CutHz = uicontrol('style','slide',...
                 'unit','pix',...
                 'position',[LT UP(17) 250 20],...
                 'min',CutHz.slMin,'max',CutHz.slMax,...
                 'SliderStep',[CutHz.slStep(1) CutHz.slStep(2)],...
                 'val',70);
S.ed.CutHz = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[285 UP(17) 100 20],...
                 'fontsize',16,...
                 'string','70');
set([S.ed.CutHz,S.sl.CutHz],'call',{@ed_call,S});  % Shared Callback.
```

```

%% File Name
S.lbl.Fname.Instr = uicontrol('Style','text',...
    'Position',[LT UP(18) 360 15],...
    'String','Type Middle Part of Filename To Save - SWF_Your ...
        Stuff_ERA');
S.lbl.Fname.Beg = uicontrol('Style','text',...
    'Position',[LT UP(19) 75 20],...
    'fontsize',12,...
    'String','SWF_');

clear Fil
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

if isfield(Fil.res,'partial') == 1
    partialFname = Fil.res.partial;
else
    partialFname = 'Rev0';
end

S.ed.Fname = uicontrol('style','edit',...
                 'unit','pix',...
                 'position',[105 UP(19) 200 20],...
                 'fontsize',12,...
                 'fontweight','bold',...
                 'string',partialFname);

 S.lbl.Fname.End = uicontrol('Style','text',...
    'Position',[310 UP(19) 75 20],...
    'fontsize',12,...
    'String','_ERA');

%% Save Info Box
S.lbl.SaveLoc.Info = uicontrol('Style','text',...
    'Position',[LT UP(22) 350 15],...
    'Fontweight', 'bold',...
    'String','Current Working Directory:');
S.lbl.SaveLoc.Loc = uicontrol('Style','text',...
    'Position',[LT UP(24) 350 45],...
    'String',Fol.work);

%% Plot Option
opt1 = 'Plots On';
S.check1=uicontrol('Style', 'checkbox',...
    'String', opt1,...
    'Position', [LT UP(21) 100 30],...
    'Value',1);
%% Save Button
S.SaveButton = uicontrol('style','pushbutton',...
    'String', 'Push Here to Save and Flap',...
    'unit','pix',...
    'position',[130 UP(21) 250 30],...
    'fontweight','bold',...
    'fontsize',13,...
    'Callback', {@saveAndFlap,S});

%% Load Button
S.LoadButton = uicontrol('style','pushbutton',...
    'String', 'Push Here to Load Previous Settings',...
    'unit','pix',...
    'position',[LT UP(25) 350 20],...
    'fontweight','bold',...
    'fontsize',13,...
    'Callback', {@loadSettings,S});
function [] = simOptions(varargin)
    [h,S] = varargin{[1,3]}; % Get calling handle and structure.
    simCheck = get(S.check2,'Value');
    if simCheck == 1
        set([S.ed.Noise,S.sl.Noise,S.lbl.Noise],'visible','on')
    else
        set([S.ed.Noise,S.sl.Noise,S.lbl.Noise],'visible','off')
    end



function [] = saveAndFlap(varargin)
%% Get Values
    [h,S] = varargin{[1,3]};  % Get calling handle and structure.
    Res.plotFlag = get(S.check1,'value'); %1 is on, 0 is off
    %1 is Simulation, 0 is Actual Run
    Res.simFlag = get(S.check2,'Value');
    Res.maxVAC = get(S.sl.VAC,'value');
    Res.Bias = get(S.sl.Bias,'value');
    Res.amp = get(S.sl.Amp,'value');
    Res.avg = get(S.sl.Avg,'value');
    Res.overlap = get(S.sl.Ovrlp, 'value');
    Res.simNoise = get(S.sl.Noise, 'value');
    Res.SampleSeconds = get(S.sl.SampLen, 'value');
    Res.TareSeconds = get(S.sl.TareLen, 'value');
    Res.desiredCutFreq = get(S.sl.CutHz, 'value');

    load('Folder_Structure.mat');
    clear Fil
    loadFile = strcat(Fol.work,'/','File_Structure.mat');
    load(loadFile);
    clear loadFile
    Fil.era.partial = get(S.ed.Fname,'string');
    Fil.era.full = strcat('SWF_',Fil.era.partial,'_ERA.mat');
    Fil.era.path = strcat(Fol.work, '/');
    Fil.frf.full = 'SWF_FRF_Paramtrs.mat';
    Fil.frf.path = strcat(Fol.work,'/');

    saveFile = strcat(Fol.work,'/','File_Structure');
    save(saveFile,'Fil')
    clear saveFile

    saveFile = strcat(Fil.frf.path, Fil.frf.full);
    save(saveFile,'-struct','Res')
    clear saveFile
    SWF_FRF_BD
function [] = loadSettings(varargin)
%% Load Settings File
    [h,S] = varargin{[1,3]};  % Get calling handle and structure.
    load('Folder_Structure.mat');
    loadFile = strcat(Fol.work,'/','File_Structure.mat');
    load(loadFile);
    clear loadFile

    filSpec = strcat(Fol.work,'/','SWF_FRF_Paramtrs.mat');

    [filename,pathname] = uigetfile(filSpec,...
    'Choose a MATLAB Data file');

    if (filename ≠ 0)
        saveFile = [pathname filename];
        eval(['load ' saveFile ';']);
        clear saveFile

        Fil.frf.full = filename;
        Fil.frf.path = pathname;
        saveFile = strcat(Fol.work,'/','File_Structure.mat');
        save(saveFile,'Fil')

        clear filSpec filename pathname;

        set(S.check1,'value',plotFlag);
        set(S.check2,'value',simFlag);

        set(S.sl.VAC,'value',maxVAC);
        set(S.ed.VAC,'string',maxVAC);

        set(S.sl.Bias,'value',Bias);
        set(S.ed.Bias,'string',Bias);

        set(S.sl.Amp,'value',amp);
        set(S.ed.Amp,'string',amp);

        set(S.sl.Avg,'value',avg);
        set(S.ed.Avg,'string',avg);

        set(S.sl.Ovrlp,'value',overlap);
        set(S.ed.Ovrlp,'string',overlap);

        set(S.sl.Noise,'value',simNoise);
        set(S.ed.Noise,'string',simNoise);

        set(S.sl.SampLen,'value',SampleSeconds);
        set(S.ed.SampLen,'string',SampleSeconds);

        set(S.sl.TareLen,'value',TareSeconds);
        set(S.ed.TareLen,'string',TareSeconds);


        set(S.sl.CutHz,'value',desiredCutFreq);
        set(S.ed.CutHz,'string',desiredCutFreq);
    end


function [] = ed_call(varargin)
%% Slider Functions
% Callback for the edit box and slider.
[h,S] = varargin{[1,3]};  % Get calling handle and structure.

switch h  % Who called?
    case S.ed.VAC
        % Get the slider's info.
        L = get(S.sl.VAC,{'min','max','value'});
        E = str2double(get(h,'string'));  % Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            % E falls within range of slider.
            set(S.sl.VAC,'value',E)
        else
            % User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.VAC
        roundSlider = round(get(h,'value'));
        % Set edit to current slider.
        set(S.ed.VAC,'string',roundSlider)
        set(S.sl.VAC,'value',roundSlider)

    case S.ed.Bias
        % Get the slider's info.
        L = get(S.sl.Bias,{'min','max','value'});
        % Numerical edit string.
        E = str2double(get(h,'string'));
        if E ≥ L{1} && E ≤ L{2}
            % E falls within range of slider.
            set(S.sl.Bias,'value',E)
        else
            % User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.Bias
        roundSlider = round(get(h,'value'));
        %  Set edit to current slider.
        set(S.ed.Bias,'string',roundSlider)
        set(S.sl.Bias,'value',roundSlider)

    case S.ed.Amp
        % Get the slider's info.
        L = get(S.sl.Amp,{'min','max','value'});
        % Numerical edit string.
        E = str2double(get(h,'string'));
        if E ≥ L{1} && E ≤ L{2}
            % E falls within range of slider.
            set(S.sl.Amp,'value',E)
        else
            % User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.Amp
        %round(get(h,'value'));
        roundSlider = round(get(h,'value') * 100)/100;
        % Set edit to current slider.
        set(S.ed.Amp,'string',roundSlider)
        set(S.sl.Amp,'value',roundSlider)

    case S.ed.Avg
        % Get the slider's info.
        L = get(S.sl.Avg,{'min','max','value'});
        E = str2double(get(h,'string')); % Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            set(S.sl.Avg,'value',E)  %E falls within range of slider.
        else
            set(h,'string',L{3}) %User tried to set slider out of range.
        end
    case S.sl.Avg
        roundSlider = round(get(h,'value'));
        %Set edit to current slider.
        set(S.ed.Avg,'string',roundSlider)
        set(S.sl.Avg,'value',roundSlider)

    case S.ed.Ovrlp
        %Get the slider's info.
        L = get(S.sl.Ovrlp,{'min','max','value'});
        E = str2double(get(h,'string')); %Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            %E falls within range of slider.
            set(S.sl.Ovrlp,'value',E)
        else
            %User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.Ovrlp
        %round(get(h,'value'));
        roundSlider = round(get(h,'value') * 10)/10;
        % Set edit to current slider.
        set(S.ed.Ovrlp,'string',roundSlider)
        set(S.sl.Ovrlp,'value',roundSlider)

     case S.ed.Noise
        %Get the slider's info.
        L = get(S.sl.Noise,{'min','max','value'});
        E = str2double(get(h,'string')); %Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            %E falls within range of slider.
            set(S.sl.Noise,'value',E)
        else
            %User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.Noise
        %round(get(h,'value'));
        roundSlider = round(get(h,'value') * 100)/100;
        %Set edit to current slider.
        set(S.ed.Noise,'string',roundSlider)
        set(S.sl.Noise,'value',roundSlider)

    case S.ed.SampLen
        %Get the slider's info.
        L = get(S.sl.SampLen,{'min','max','value'});
        E = str2double(get(h,'string'));  %Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            %E falls within range of slider.
            set(S.sl.SampLen,'value',E)
        else
            %User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.SampLen
        roundSlider = round(get(h,'value'));
        %Set edit to current slider.
        set(S.ed.SampLen,'string',roundSlider)
        set(S.sl.SampLen,'value',roundSlider)

    case S.ed.TareLen
        %Get the slider's info.
        L = get(S.sl.TareLen,{'min','max','value'});
        E = str2double(get(h,'string')); %Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            %E falls within range of slider.
            set(S.sl.TareLen,'value',E)
        else
            %User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.TareLen
        roundSlider = round(get(h,'value'));
        %Set edit to current slider.
        set(S.ed.TareLen,'string',roundSlider)
        set(S.sl.TareLen,'value',roundSlider)

    case S.ed.CutHz
        %Get the slider's info.
        L = get(S.sl.CutHz,{'min','max','value'});
        E = str2double(get(h,'string'));  %Numerical edit string.
        if E ≥ L{1} && E ≤ L{2}
            %E falls within range of slider.
            set(S.sl.CutHz,'value',E)
        else
            %User tried to set slider out of range.
            set(h,'string',L{3})
        end
    case S.sl.CutHz
        roundSlider = round(get(h,'value'));
        %Set edit to current slider.
        set(S.ed.CutHz,'string',roundSlider)
        set(S.sl.CutHz,'value',roundSlider)

    otherwise
        % Do nothing.

end
```

B.3 Test GUI Code

This code was written during the present research to take user inputs and store them into a file for use during the testing portion of the code.

```
function [] = SWF_Test_GUI()
close all; clear all; clc;

%% Set the Figure
S.fh = figure('units','pixels',...
              'pos',[400 500 500 400],...
              'menubar','none',...
              'name','SWF_FRF_BD Options',...
              'numbertitle','off',...
              'resize','off');

%% Position Vector
LT = [25 80 150 205 265 320];
UP = [375 350 325 300 275 250 225 200 175 150 125 100 75 50 25];

%% Simulation Option
S.Sim.chk1=uicontrol('Style', 'checkbox',...
    'String', 'Simulation',...
    'pos', [LT(1) UP(1) 100 30],...
    'Value',0);

```

```matlab
%% Test Type
S.TestType.lbl = uicontrol('Style','text',...
    'pos', [LT(1) UP(2) 450 15],...
    'String', 'Select the Type of Test');

% Create the button group.
S.TestType.grp = uibuttongroup('visible','off','units',...
    'pixels','pos',[LT(1) UP(4) 450 45]);

% Create three radio buttons in the button group.
S.TestType.rdo3 = uicontrol('Style','radiobutton','String','BABM',...
    'pos',[210 7 100 30],'parent',S.TestType.grp,...
    'HandleVisibility','off');
S.TestType.rdo1 = uicontrol('Style','radiobutton',...
    'String','Single Freq','pos',[10 7 100 30],'parent',...
    S.TestType.grp,'HandleVisibility','off');
S.TestType.rdo2 = uicontrol('Style','radiobutton',...
    'String','Freq Range','pos',[110 7 100 30],'parent',...
    S.TestType.grp,'HandleVisibility','off');



%% Different Test Options
%Create the Options For Different Tests
S.TestOpt.grp = uibuttongroup('visible','on','units','pixels',...
    'pos',[LT(1) UP(7) 450 70]);

%Single Test
S.TestOpt.Sing.lbl = uicontrol('Style','text','String','Test ...
   Frequency',...
    'pos',[10 45 75 15],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');
S.TestOpt.Sing.ed = uicontrol('Style','edit','String','22',...
    'pos',[90 37 50 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');

%Frequency Range
S.TestOpt.FreqRng.lbl1 = uicontrol('Style','text','String',...
    'Start Frequency','pos',[10 45 75 15],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');
S.TestOpt.FreqRng.ed1 = uicontrol('Style','edit','String','18',...
    'pos',[90 37 50 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');
S.TestOpt.FreqRng.lbl2 = uicontrol('Style','text','String',...
    'End Frequency','pos',[150 45 75 15],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');
S.TestOpt.FreqRng.ed2 = uicontrol('Style','edit','String','24',...
    'pos',[230 37 50 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');
S.TestOpt.FreqRng.lbl3 = uicontrol('Style','text','String','Step',...
    'pos',[290 45 25 15],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');
```

```
S.TestOpt.FreqRng.ed3 = uicontrol('Style','edit','String','2',...
    'pos',[320 37 50 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','off');

%BABM
load('Folder_Structure.mat');
clear Fil
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

if isfield(Fil,'TM')
    testMtx = Fil.TM.full;
else
    testMtx = 'TS_XXX.mat';
end

S.TestOpt.BABM.lbl1 = uicontrol('Style','text',...
    'String','Test Frequency',...
    'pos',[10 45 75 15],'parent',S.TestOpt.grp,...
    'HandleVisibility','on','visible','on');
S.TestOpt.BABM.ed1 = uicontrol('Style','edit','String','22',...
    'pos',[90 37 50 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','on','visible','on');
S.TestOpt.BABM.lbl2 = uicontrol('Style','text','String','Test ...
   Matrix',...
    'pos',[150 45 75 15],'parent',S.TestOpt.grp,...
    'HandleVisibility','on','visible','on');
S.TestOpt.BABM.ed2 = uicontrol('Style','edit','String',testMtx,...
    'pos',[230 37 150 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','off','visible','on');
S.TestOpt.BABM.btn1 = ...
   uicontrol('Style','pushbutton','String','Browse',...
    'pos',[385 37 50 30],'parent',S.TestOpt.grp,...
    'HandleVisibility','on','visible','on');
%% Non BABM Test Parameters
% Amplitude
S.Amp.lbl = uicontrol('Style','text',...
    'parent',S.TestOpt.grp,'HandleVisibility','on',...
    'pos', [10 15 75 15],...
    'visible', 'off',...
    'String', 'Amplitude');

S.Amp.ed = uicontrol('Style','edit',...
    'parent',S.TestOpt.grp,'HandleVisibility','on',...
    'pos', [90 7 50 30],...
    'visible', 'off',...
    'String', '0.35');

% Tau Option
S.Tau.lbl = uicontrol('Style','text',...
    'parent',S.TestOpt.grp,'HandleVisibility','on',...
    'pos', [150 15 75 15],...
    'visible', 'off',...
    'String', 'Tau');

S.Tau.ed = uicontrol('Style','edit',...
    'parent',S.TestOpt.grp,'HandleVisibility','on',...
    'pos', [230 7 50 30],...
    'visible', 'off',...
    'String', '0.0');

% Eta Option
S.Eta.lbl = uicontrol('Style','text',...
    'parent',S.TestOpt.grp,'HandleVisibility','on',...
    'pos', [290 15 25 15],...
    'visible', 'off',...
    'String', 'Eta');
S.Eta.ed = uicontrol('Style','edit',...
    'parent',S.TestOpt.grp,'HandleVisibility','on',...
    'pos', [320 7 50 30],...
    'visible', 'off',...
    'String', '0.0');

%% Options that are always Set
% Bias Option
S.Bias.lbl = uicontrol('Style','text',...
    'pos', [LT(1) UP(8) 50 20],...
    'String', 'Bias');

S.Bias.ed = uicontrol('Style','edit',...
    'pos', [LT(2) UP(8) 50 20],...
    'String', '100');

% Max VAC
S.MaxVAC.lbl = uicontrol('Style','text',...
    'pos', [LT(3) UP(8) 50 20],...
    'String', 'MaxVAC');

S.MaxVAC.ed = uicontrol('Style','edit',...
    'pos', [LT(4) UP(8) 50 20],...
    'String', '200');

% Number of Samples Option
S.Samp.lbl = uicontrol('Style','text',...
    'pos', [LT(5) UP(8) 50 20],...
    'String', 'Samples');

S.Samp.ed = uicontrol('Style','edit',...
    'pos', [LT(6) UP(8) 50 20],...
    'String', '3');

%% SC FileName Option
S.SCFname.lbl = uicontrol('Style','text',...
    'pos', [LT(1) UP(9) 150 15],...
    'String', 'SC Parameters Filename');

S.SCFname.ed = uicontrol('Style','edit',...
    'pos', [LT(1) UP(10) 350 28],...
    'String', Fil.SC.full);

S.SCFname.btn = uicontrol('Style','pushbutton','String','Browse',...
    'pos',[380 UP(10) 50 30],'HandleVisibility','on');

%% FF Compensator
% No options coded Yet
S.FFComp.lbl = uicontrol('Style','text',...
    'pos', [LT(1) UP(11) 450 15],...
    'String', 'FF Compensator Not Coded - Need to Add');

%% Save Button
S.Save.btn = uicontrol('style','pushbutton',...
    'String', 'Push Here to Save and Flap',...
    'unit','pix',...
    'position',[100 UP(13) 250 30],...
    'fontweight','bold',...
    'fontsize',13);

%% Set Callbacks for Radio Buttons & Other Buttons
set(S.TestType.grp,'SelectionChangeFcn',{@selcbk,S});
%set(S.TestType.grp,'SelectedObject',[]);  % No selection
set(S.TestType.grp,'Visible','on');
set(S.TestOpt.BABM.btn1, 'Call',{@loadfile,S,Fol,Fil});
set(S.SCFname.btn, 'Call',{@loadfile,S,Fol,Fil});
set(S.Save.btn,'Call', {@saveAndFlap,S,Fol});

%% Load File Function
function [] = loadfile(varargin)
%Get calling handle and structure.
[h,S,Fol,Fil] = varargin{[1,3,4,5]};

%name = strcat('SWF_',baseFileName,'_SC');
%filSpec = strcat(Fol.base,'/',Fol.parm,'/',name,'.mat');
%filSpec = strcat(Fol.work,'/',name,'.mat');
%clear name baseFileName Fol

switch h
    case S.TestOpt.BABM.btn1
        if isfield(Fil,'TM')
            filSpec = strcat(Fil.TM.path,Fil.TM.full);
        else
            filSpec = strcat(Fol.work,'/','TS_XXX.mat');
        end
    case S.SCFname.btn
        filSpec = strcat(Fil.SC.path,Fil.SC.full);
end
```

```
[filename,pathname] = uigetfile(filSpec,...
            'Choose a Test Sequence');

if (filename ≠ 0)
    switch h
        case S.TestOpt.BABM.btn1
            set(S.TestOpt.BABM.ed2,'String',filename);
            Fil.TM.path = pathname;
            Fil.TM.full = filename;
        case S.SCFname.btn
            set(S.SCFname.ed,'String',filename);
            Fil.SC.path = pathname;
            Fil.SC.full = filename;
    end
    clear filename pathname;
    saveFile = strcat(Fol.work,'/','File_Structure.mat');
    save(saveFile,'Fil')
    clear saveFile
end
%% Change Selection Call Back Function
function selcbk(source,eventdata,varargin)
% selectedTest = get(get(source,'SelectedObject'),'String');
selectedTest = get(source,'SelectedObject');
[S] = varargin{[1,1]}; %Get calling structure.
switch selectedTest
    case S.TestType.rdo1
        disp('1')
        singTestVis = 'on';
        freqRngVis = 'off';
        babmVis = 'off';
        oppBabmVis = 'on';

    case S.TestType.rdo2
        disp('2')
        singTestVis = 'off';
        freqRngVis = 'on';
        babmVis = 'off';
        oppBabmVis = 'on';

    case S.TestType.rdo3
        disp('3')
        singTestVis = 'off';
        freqRngVis = 'off';
        babmVis = 'on';
        oppBabmVis = 'off';
end
%Single Frequency Options
set(S.TestOpt.Sing.lbl,'visible',singTestVis);
set(S.TestOpt.Sing.ed,'visible',singTestVis);

%Frequency Range Options
set(S.TestOpt.FreqRng.lbl1,'visible',freqRngVis);
set(S.TestOpt.FreqRng.lbl2,'visible',freqRngVis);
set(S.TestOpt.FreqRng.lbl3,'visible',freqRngVis);
set(S.TestOpt.FreqRng.ed1,'visible',freqRngVis);
set(S.TestOpt.FreqRng.ed2,'visible',freqRngVis);
set(S.TestOpt.FreqRng.ed3,'visible',freqRngVis);

%BABM Options
set(S.TestOpt.BABM.lbl1,'visible',babmVis);
set(S.TestOpt.BABM.lbl2,'visible',babmVis);
set(S.TestOpt.BABM.ed1,'visible',babmVis);
set(S.TestOpt.BABM.ed2,'visible',babmVis);
set(S.TestOpt.BABM.btn1,'visible',babmVis);

%Options That are In BABM Test Matrix
set(S.Amp.lbl,'visible',oppBabmVis);
set(S.Amp.ed,'visible',oppBabmVis);

set(S.Tau.lbl,'visible',oppBabmVis);
set(S.Tau.ed,'visible',oppBabmVis);

set(S.Eta.lbl,'visible',oppBabmVis);
set(S.Eta.ed,'visible',oppBabmVis);

function [] = saveAndFlap(varargin)
% Get Values
    [h,S,Fol] = varargin{[1,3,4]};  % Get calling handle and structure.

    selectedTest = get(S.TestType.grp,'SelectedObject');
    switch selectedTest
        case S.TestType.rdo1 %Single
            Res.testFlag = 0;
            Res.wn = str2num(get(S.TestOpt.Sing.ed, 'String'));

            Res.AR = str2num(get(S.Amp.ed, 'String'));
            Res.tauR = str2num(get(S.Tau.ed, 'String'));
            Res.eta = str2num(get(S.Eta.ed, 'String'));

        case S.TestType.rdo2 %Freq Range
            Res.testFlag = 1;
            Res.w1 = str2num(get(S.TestOpt.FreqRng.ed1, 'String'));
            Res.w2 = str2num(get(S.TestOpt.FreqRng.ed2, 'String'));
            Res.step = str2num(get(S.TestOpt.FreqRng.ed3, 'String'));

            Res.AR = str2num(get(S.Amp.ed, 'String'));
            Res.tauR = str2num(get(S.Tau.ed, 'String'));
            Res.eta = str2num(get(S.Eta.ed, 'String'));

        case S.TestType.rdo3 %BABM
            Res.testFlag = 2;
            Res.wn = str2num(get(S.TestOpt.BABM.ed1, 'String'));
            % A, Tau, and Eta set in BABM Test Matrix
    end
```

```
%1 is Simulation, 0 is Actual Run
Res.simFlag = get(S.Sim.chk1,'Value');
Res.Bias = str2num(get(S.Bias.ed, 'String'));
Res.maxVAC = str2num(get(S.MaxVAC.ed, 'String'));
Res.samples = str2num(get(S.Samp.ed, 'String'));

clear Fil
loadFile = strcat(Fol.work,'/','File_Structure.mat');
load(loadFile);
clear loadFile

Fil.BD.path = strcat(Fol.work,'/');
Fil.BD.full = 'SWF_Test_BD.mat';

saveFile = strcat(Fol.work,'/','File_Structure');
save(saveFile,'Fil')
clear saveFile

saveFile = strcat(Fil.BD.path, Fil.BD.full);
save(saveFile,'-struct','Res')
clear saveFile

SWF_Test_BD
```

Bibliography

[1] AeroVironment. “Black Widow”. http://www.avinc.com/uas/adc/black_widow/. Accessed on December 5, 2013.

[2] AeroVironment. “AeroVironment Develops World’s First Fully Operational Life-Size Hummingbird-Like Unmanned Aircraft for DARPA”. http://www.avinc.com/resources/press_release/aerovironment_develops_worlds_first_fully_operational_life-size_hummingbird, February 17, 2011. Press Release.

[3] Alexander, David E and Steven Vogel. *Nature’s flyers: birds, insects, and the biomechanics of flight*. JHU Press, 2004.

[4] Anderson, Michael L. *Design and Control of Flapping Wing Micro Air Vehicles*. Dissertation, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433, September 2011.

[5] Anderson, Michael L., Christopher J. Perry, Brandon M. Hua, Dakota S. Olsen, Jason R. Parcus, Kenneth M. Pederson, and Daniel D. Jensen. “The Sticky-Pad Plane and Other Innovative Concepts for Perching UAVs”. *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Orlando, FL, January 2009.

[6] Anderson, Michael L., Nathanael J. Sladek, and Richard G. Cobb. “Design, Fabrication, and Testing of an Insect-Sized MAV Wing Flapping Mechanism”. *Proceedings of the 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Orlando, FL, January 2011.

[7] Carl, Justin R. *Power Requirements For Bi-Harmonic Amplitude and Bias Modulation Control of a Flapping Wing Micro Air Vehicle*. Thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433, March 2013.

[8] Cobb, Richard G. “State-Space Model Identification using the Eigensystem Realization Algorithm Toolbox for MATLAB”. Computer Software, Air Force Institute of Technology, November 2003. Version 1.1.

[9] Conn, Andrew, Stuart Burgess, Rick Hyde, and Chung Seng Ling. “From Natural Flyers to the Mechanical Realization of a Flapping Wing Micro Air Vehicle”. *Robotics and Biomimetics, 2006. ROBIO ’06. IEEE International Conference on*, 439–444. Institute of Electrical and Electronics Engineers, Kunming, China, December 2006.

[10] Cook, Robert D., David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons Inc., Hoboken NJ, fourth edition, 2002.

[11] (DARPA), Defense Advanced Research Projects Agency. "Nano Air Vehicle (NAV)". http://www.darpa.mil/Our_Work/DSO/Programs/Nano_Air_Vehicle_(NAV).aspx, (n.d.).

[12] DeLuca, Anthony M. *Aerodynamic Performance and Particle Image Velocimetery of Piezo Actuated Biomimetic Manduca Sexta Engineered Wings Towards the Design and Application of a Flapping Wing Flight Vehicle*. Dissertation, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433, December 2013.

[13] Dickinson, Michael H, Fritz-Olaf Lehmann, and Sanjay P Sane. "Wing rotation and the aerodynamic basis of insect flight". *Science*, 284(5422):1954–1960, 1999.

[14] DuPont Electronic Technologies. "Pyralux FR Sheet Adhesive Technical Information". http://www2.dupont.com/Pyralux/en_US/assets/downloads/pdf/FRadhesive_H-73235.pdf, May 2012.

[15] DuPont High Performance Films. "DuPont Kapton HN Polyimide Film Technical Data Sheet". http://www2.dupont.com/Kapton/en_US/assets/downloads/pdf/HN_datasheet.pdf, April 2001.

[16] DuPont Teijin Films. "Mylar Polyester Film Physical-Thermal Properties". http://usa.dupontteijinfilms.com/informationcenter/downloads/Physical_And_Thermal_Properties.pdf, June 2003.

[17] Ellington, Charles. "Insects Versus Birds: The Great Divide". *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006-35, 1–6. American Institute of Aeronautics and Astronautics, Reno, NV, January 2006.

[18] Grossman, Lev, Mark Thompson, Jeffrey Kluger, Alice Park, Bryan Walsh, Claire Suddath, Eric Dodds, Kayla Webley, Nate Rawlings, Feifei Sun, Cleo Brock-Abraham, and Nick Carbone. "The 50 Best Inventions". *Time Magazine*, November 28, 2011.

[19] IDT. "Motion Studio User Manual". http://idtvision.com/documents/mstudio_man_en.pdf, December 2013.

[20] IDT Vision. "X-Stream XS-4 Specifications". http://www.velocimetry.net/downloads/brochure_xs_4.pdf, July 2004.

[21] Lindholm, Garrison J. *Closed-Loop Control of Constrained Flapping Wing Micro Air Vehicles*. Dissertation, 2950 Hobson Way, WPAFB, OH 45433, 2014. AFIT-ENY-DS-14-M-02, submitted for publication.

[22] Lindholm, Garrison J. and Richard G. Cobb. "Closed-Loop Control of a Constrained Resonant-Flapping Micro Air Vehicle". submitted for publication, March 2014.

[23] McMichael, James M. and Michael S. Francis (Col. Ret.). *Toward a New Dimension in Flight*. Technical report, Defense Advanced Research Projects Agency, 675 North Randolph Street, Arlington, Va, 22203-3114, August 1997.

[24] Nippon Graphite Fiber Corporation. "High Modulus and High Strength Fibers YSH Fibers". http://www.ngfworld.com/dcms_media/other/skill_catalog06.pdf, February 2012.

[25] Norris, Aaron G. *Experimental Characterization of the Structural Dynamics and Aero-Structural Sensitivity of a Hawkmoth Wing Toward the Development of Design Rules for Flapping- Wing Micro Air Vehicles*. Dissertation, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433, March 2013.

[26] Norris, Aaron G., Anthony N. Palazotto, and Richard G. Cobb. "Experimental Structural Dynamic Characterization of the Hawkmoth (Manduca Sexta) Forewing". *International Journal of Micro Air Vehicles*, 5(1):39–54, March 2013.

[27] O'Hara, R P and A N Palazotto. "The morphological characterization of the forewing of the Manduca Sexta species for the application of biomimetic apping wing micro air vehicles". *Bioinspiration and Biomimetics*, 7(4):1–13, October 2012.

[28] O'Hara, Ryan P. *The Characterization of Material Properties and Structural Dynamics of the Manduca Sexta Forewing for Application to Flapping Wing Micro Air Vehicle Design*. Dissertation, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433, September 2012.

[29] Sladek, Nathanael J. *Flapping Wing Micro Air Vehicle Wing Manufacture and Force Testing*. Thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433, March 2011.

[30] Willmott, Alexander P. and Charles P. Ellington. "The Mechanics of Flight in the Hawkmoth Manduca Sexta I. Kinematics of Hovering and Forward Flight". *The Journal of Experimental Biology*, 200:2705–2722, 1997.

[31] Willmott, Alexander P. and Charles P. Ellington. "The Mechanics of Flight in the Hawkmoth Manduca Sexta II. Aerodynamic Consequences of Kinematic and Morphological Variation". *The Journal of Experimental Biology*, 200:2723–2745, 1997.

[32] Wilson, Sam. "Small Business Innovation Research (SBIR) Success Story". *DARPATech 2002 Symposium*. Defense Advanced Research Projects Agency (DARPA), Anaheim, CA, August 2002. http://archive.darpa.mil/DARPATech2002/presentations/tto_pdf/speeches/WILSONSB.pdf.

[33] Wood, R. J. “Liftoff of a 60mg flapping-wing MAV”. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 1889–1894. 2007.

Vita

Lieutenant Commander Zachary “Zach” Brown graduated from Westerville North High School in 1998. He earned his Bachelors of Science in Marine Engineering Systems from the United States Merchant Marine Academy in 2002. He was directly commissioned into the United States Navy as an Ensign.

LCDR Brown completed Naval Aviator Flight Training at NAS Pensacola, NAS Corpus Christi, earning the coveted *Wings of Gold* in 2004 at Vance AFB. He proceeded to Tinker AFB to complete training in the E-6B Mercury a variant of the Boeing 707-300 airframe. He flew over 1500 hours operationally in the E-6B.

LCDR Brown then proceeded to Fleet Air Reconnaissance Squadron Seven (VQ-7) to serve as a flight instructor training future E-6B pilots. He earned his Airline Transport Pilot License along with a type rating in the Boeing 737. While transitioning to VQ-7, LCDR Brown deployed to Qatar and flew over 20 missions in support of Operation Iraqi Freedom.

Prior to reporting to AFIT, LCDR Brown served as the Assistant Navigator on the USS John C. Stennis. During his tour he ensured the safe navigation of the aircraft carrier in over 116,700 miles of steaming during 366 days underway. USS John C. Stennis flew numerous sorties in support of Operation Iraqi Freedom and Operation Enduring Freedom during their deployment to 5th fleet.

After graduation, LCDR Brown will return to Fleet Air Reconnaissance Squadron Three (VQ-3) as a department head.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 27–03–2014	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Oct 2012–Mar 2014

4. TITLE AND SUBTITLE Experimental Characterization of Wings for a Hawkmoth-Sized Micro Air Vehicle	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Brown, Zachary R., Lieutenant Commander, USN	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENY-14-M-10
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Mr. Michael J. Sytsma michael.sytsma@us.af.mil Air Force Research Laboratory Munitions Aerodynamics Sciences Branch 101 West Eglin Blvd. Suite 348 Eglin AFB, FL 32542-6810	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RWWV
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Considerable prior research has been conducted on many aspects of Hawkmoth-sized, piezo-driven FWMAVs, but the majority utilized a common structural wing to conform with a biomimetic design. In this research, six alternate wing designs with the same planform and size, but different structures were built and explored. FEA code was used to determine the location of maximum stress, and then mass was removed from minimally stressed areas, under the premise that equal force production with a lighter wing would improve the MAV design. The main metric for this research was vertical force generation per mass; high speed video provided complementary insight. The angle stop setting was fixed at 60° based on prior studies, and tests were executed by mounting the mechanism on an ATI Nano-17 Titanium force transducer. As part of this effort, several manufacturing processes enhancing repeatability and efficiency during testing and assembly were developed. The new method for wing and PZT attachment allow for constant drive linkage geometry, and non-destructive wing replacement.
The alternate designs created a 3.8 mg to 19.4 mg (6% to 31% of original wing and 0.5% to 2.5% of a typical Hawkmoth) mass reduction, and generated vertical forces approaching the original design. Combining the mass and natural frequency into a RDS, an effective wing design can be predicted. The best design produced 14% less vertical force than the original design, however, it resulted in a 15% mass reduction from the original wing. High speed video suggested small additional changes to the wing motion could improve performance.

15. SUBJECT TERMS

Flapping Wing Micro Air Vehicle, MAV, AFIT, Wing Characterization

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Mark F. Reeder, AFIT/ENY
U	U	U	UU	149	**19b. TELEPHONE NUMBER** *(include area code)* (937) 785-3636 x4530 mark.reeder@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18