
# AEROELASTIC OPTIMIZATION OF SOUNDING ROCKET FINS

THESIS

Joseph R. Simmons, III

AFIT/GSS/ENY/09-J02

**DEPARTMENT OF THE AIR FORCE**

**AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright Patterson Air Force Base, Ohio**

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GSS/ENY/09-J02

# AEROELASTIC OPTIMIZATION OF SOUNDING ROCKET FINS

THESIS

Presented to the Faculty of the

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 (Space Systems)

Joseph R. Simmons, III, B.F.A.

June, 2009

AFIT/GSS/ENY/09-J02

# AEROELASTIC OPTIMIZATION OF SOUNDING ROCKET FINS

Joseph R. Simmons, III, B.F.A.

Approved:

| | |
|---|---|
| Dr. Jonathan Black (Chairman) | 9 Jun 2009<br>Date |
| Lt Col Eric Swenson (Member) | 2 June 09<br>Date |
| Dr. Donald Kunz (Member) | 9 June 2009<br>Date |

AFIT/GSS/ENY/09-J02

*Abstract*

This research effort develops a multidisciplinary design tool to optimize sounding rocket fin geometries that minimize the mass of the fins while maintaining aerodynamic performance. This research grew out of a design problem experienced by the US Air Force Academy's FalconLAUNCH program. The FalconLAUNCH program is a senior design capstone project during which Air Force Academy cadets design, build and fly a sounding rocket over the course of an academic year. In the Spring of 2007, the FalconLAUNCH V vehicle experienced a catastrophic failure when three of its four fins sheared off due to flutter. When the following year's team developed the fins for FalconLAUNCH VI, the design requirement that the fins not experience flutter led to substantially more massive fins. The FalconLAUNCH team needs a design tool that can balance the competing needs for minimal mass sounding rocket components and aerodynamic performance. The tool developed during this research is designed to find an optimal solution for the fin geometry based on the competing needs of minimizing the fins' mass and ensuring the fins will not experience flutter. The design tool then provides for verification of the design throughout the designed flight profile.

*Acknowledgements*

I wish to thank Phoenix Integration, ZONA Technology, and Siemens PLM for providing copies of their software to the Air Force Institute of Technology for use in its research efforts including this one.

I also wish to thank the Dayton Area Graduate Studies Institute for their financial support, and Dr. Canfield, Barry Hellman, Ned Lindsley for their mentorship and advice during the course of this research.

I also want to thank my committee for their encouragement and support in this research. Specifically, I want to thank Dr. Black for helping shape the direction of this research, Lt Col Swenson for helping me with all things FEMAP, and Dr. Kunz helping me better understand flutter. I could not have done this without them.

I also want to thank the US Air Force Academy for allowing me to participate in my own way in the FalconLAUNCH program.

Finally, I want to thank my friends and family that encouraged me to apply to AFIT, and put up with my long periods of work as I conducted this research.

Joseph R. Simmons, III

*Table of Contents*

*List of Figures*

*List of Tables*

*List of Symbols*

Symbol Page

Symbol Page

# AEROELASTIC OPTIMIZATION OF SOUNDING ROCKET FINS

## *1. Introduction*

### *1.1 Background*

The US Air Force Academy (USAFA) Department of Astronautics uses its FalconLAUNCH program as one of its senior design projects. During this two-semester sequence, cadets with the assistance of USAFA faculty and staff design, build, and fly a sounding rocket. The technical goal of the program is for the Academy to develop a sounding rocket capable carrying a five pound payload to the edge of space, reaching an altitude of 100 kilometers. The educational goal is for the cadets to get real hands on experience in space flight systems engineering, or as the Academy puts it, "to learn space by doing space."

Starting in Academic Year 2002-2003, the Academy has designed six rockets, five of which have flown, and it is currently working on its seventh vehicle. FalconLAUNCH I flew as a subsonic proof of concept vehicle reaching an altitude of 30,000 feet. With this successful flight, FalconLAUNCH II attempted to progress to supersonic flight, but experienced uneven thrust that resulted in the loss of the vehicle. The following year, FalconLAUNCH III corrected the thrust problem and successfully flew to Mach 1.4 and an altitude of 18,000 feet. FalconLAUNCH IV was intended to fly significantly higher, models indicated it could reach as high as 134,000 feet, but a structural failure ended the flight just three seconds after launch [19].

In Academic Year 2006-2007, the Academy built FalconLAUNCH V, Figure 1. The primary goal of that year's vehicle was to correct the failure that occurred the previous year and complete the FalconLAUNCH IV mission. The FalconLaunch V flight took place at the Wallops Launch Facility in Virginia, and succeeded in overcoming the failure that ended the FalconLAUNCH IV flight. Unfortunately, FalconLAUNCH V experienced a catastrophic failure at 11,000 feet, sending it tumbling into the Atlantic Ocean.

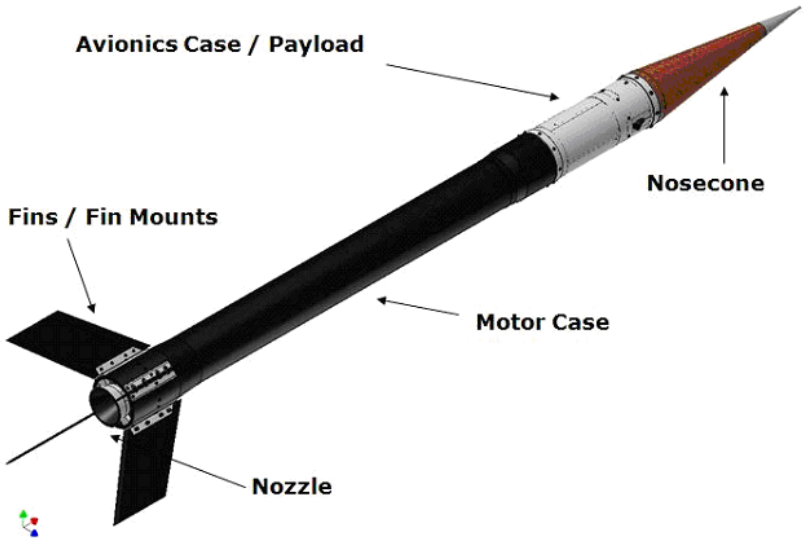



Figure 1 FalconLAUNCH V Structural Detail, credit USAFA

The FalconLAUNCH team recovered the vehicle and discovered that three of the four carbon-laminate stabilizing fins sheared off of the vehicle, as can be seen in Figure 2. The team then conducted a post-flight investigation to determine why the fins sheared off of the rocket. In addition to three of the fins having sheared off, the FalconLAUNCH team observed that the remaining fin had become delaminated, indicating that it probably experienced severe oscillations. These severe oscillations lead to structural failure indicating fin flutter was a likely cause, which prompted the team to review the calculations for the flutter velocity of the fins. Their calculations were based on Equation (18) in NACA Technical Note 4197 [28]. The form of this equation as used can be seen in Equation (1), see Figure 3 for the locations of the primary fin design parameters used in this equation. The FalconLAUNCH team found that the spreadsheet used to calculate the flutter velocity contained an error, causing the calculated flutter velocity to be much higher than the actual.

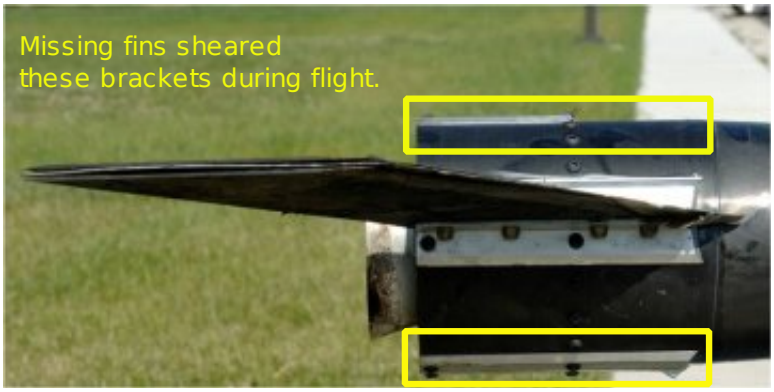



Figure 2 FalconLAUNCH V fins after failure, credit USAFA

$$V_f = a\sqrt{\frac{G_E}{\frac{39.3A^3}{(\frac{t}{c})^3(A+2)}(\frac{\lambda+1}{2})(\frac{p}{p_0})}} \tag{1}$$

$V_f$, Flutter Velocity

$a$, Speed of Sound

$G_E$, Effective Shear Modulus

$A$, Panel Aspect Ratio

$t$, Fin Thickness

$c$, Chord

$\lambda$, Taper Ratio (Ratio of Tip Chord to Root Chord)

$p$, Air Pressure at Altitude

$p_0$, Air Pressure at Sea Level

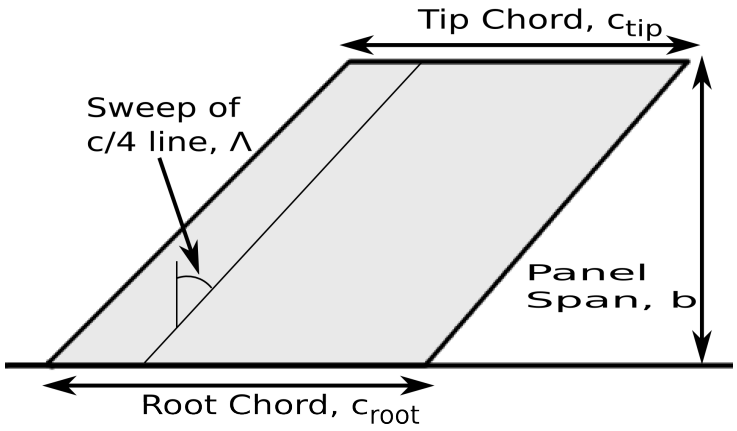
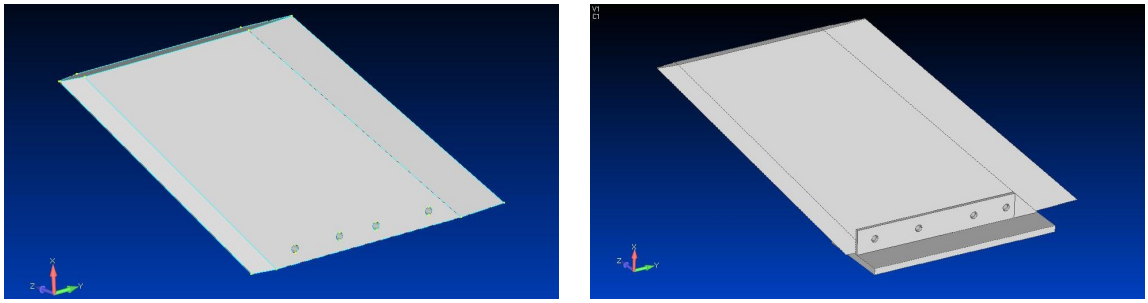



Figure 3 Fin Design Parameters

Using the corrected spreadsheet, the FalconLAUNCH team designed new fins for FalconLAUNCH VI. These fins are made from aluminum to simplify construction and to prevent delamination. They are also significantly thicker (0.40 inches on FalconLAUNCH VI vs 0.25 inches on FalconLAUNCH V) and shorter (semi-span of 8.0 inches on FalconLAUNCH VI vs 9.5 inches on FalconLAUNCH V). See Figure 4 for comparison between the two designs. An unrelated shipping problem prevented FalconLAUNCH VI from flying in Academic Year 2007-2008 and there was not a flight opportunity in Academic Year 2008-2009, so no flight data has been gathered on this design.



(a) FalconLAUNCH VI (b) FalconLAUNCH V

Figure 4 FalconLaunch VI and V Fin Geometries

### *1.2 Thesis Overview*

The sizable increase in mass of the FalconLAUNCH VI fin design over FalconLAUNCH V just to prevent flutter has led to a need for a design tool that can give

future teams the ability to design lightweight fins that will not flutter. This design tool needs to be available early in the design process to allow the FalconLAUNCH team to make accurate predictions about the weight and in turn the performance of the vehicle during the initial sizing process. What is required is a conceptual level design tool that can optimize the fin geometry to minimize the fin's mass while preventing flutter. The development and testing of such a tool is the goal of this research. The results of this research are broken into five chapters:

- Chapter 1: Introduction presented the background of the FalconLAUNCH program and demonstrated the need for a design tool to optimize sounding rocket fins with respect to flutter and their mass.
- Chapter 2: Literature Review will present the theoretical background for the solutions to flutter and rocket stability used in this research.
- Chapter 3: Methodology will present the development of the design tool to minimize the mass of a sounding rocket fin while preventing flutter.
- Chapter 4: Analysis and Results will show that the design tool is effective in finding an optimal fin design.
- Chapter 5: Conclusions and Recommendations will summarize the work presented in the thesis and make recommendations for future research.

# 2. Literature Review

## 2.1 Chapter Overview

This chapter covers material and previous research essential to the development of a design tool to minimize a sounding rocket fin's mass while preventing flutter. The chapter begins with a review of the design parameters that describe the geometry of a fin, and the role fins play in ensuring the stable flight of a rocket. Next, the chapter covers the definition and history of flutter. Following this, the chapter covers a number of techniques for predicting flutter on rocket fins. The chapter continues with a discussion of two examples of automated design optimization tools. The chapter concludes with a review of the fin geometry from a selection of successful sounding rockets/missiles.

## 2.2 Rocket Fins and Stability

Rocket fins are basically short wings placed on the rocket to provide passive stabilization. Like wings, fins have a number of primary and derived design parameters. The primary design parameters are the root and tip chords ($c_{root}$ and $c_{tip}$), the semi-span ($b$, the span of a single fin versus the entire span), the sweep angle ($\Lambda$), and the thickness ($t$). See Figure 5 for the relation of these parameters to the shape of a wing (and by extension a fin).

The derived design parameters include the Panel Aspect Ratio ($AR$, see Equation (2)), the Taper Ratio ($\lambda$, see Equation (3)), and the Panel Area ($S$, Equation (4)). [3]

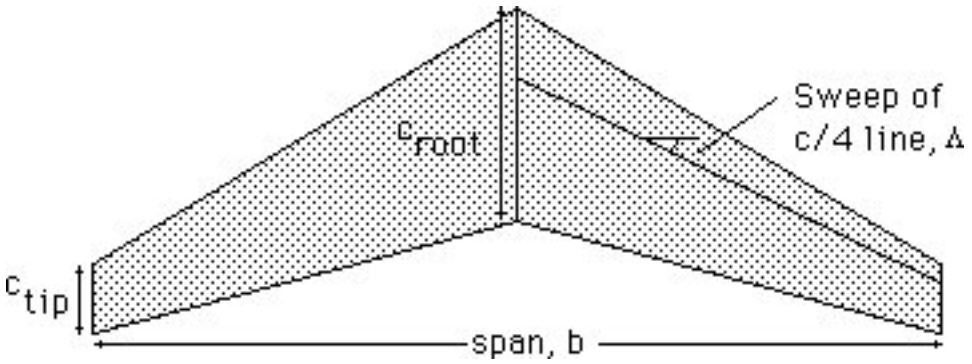



Figure 5 Wing Design Parameters [3]

$$AR = \frac{b}{c_{average}} \tag{2}$$

$AR$, Aspect Ratio

$b$, Span

$c_{average}$, Average Chord

$$\lambda = \frac{c_{tip}}{c_{root}} \tag{3}$$

$\lambda$, Taper Ratio

$c_{tip}$, Tip Chord

$c_{root}$, Root Chord

$$S = b \times c_{average} \tag{4}$$

$S$, Planform Area

$b$, Span

$c_{average}$, Average Chord

In passively controlled vehicles like sounding rockets, the fins provide the stability necessary for a successful flight. They provide this stability by moving the Center of Pressure ($C_p$) aft of the Center of Gravity ($C_g$). As long as this condition is maintained, small perturbations in the direction of flight are restored by the aerodynamic forces on the fins. Small perturbations cause an increased angle of attack on some of the fins, which causes the fins subjected to this increased angle of attack to experience higher lift. Since the rocket is a free flying body this increase in lift creates an unbalanced moment about the $C_g$. This moment causes the rocket to rotate away from the perturbation, or back toward its stable line of flight. As the rocket approaches its original line of flight, the lift on the fins decreases and the moment decreases until it becomes zero as the vehicle finishes restoring its line of flight. See Figure 6

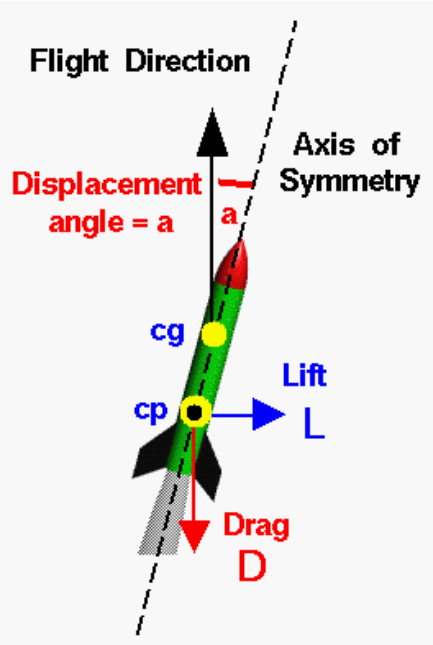



Figure 6 Rocket Stability, credit NASA [13]

Though a number of factors contribute to the location of the $C_p$, the fin's panel area has the greatest influence by far. [16] In 1967, James Barrowman derived a set of tools to predict the $C_p$ for a rocket based on the nose, body, and fin geometries and the Mach number. These tools included a set of equations, the Barrowman Equations, for calculating the $C_p$ at subsonic speeds, and a computer program, "Fin", to iteratively solve for the $C_p$ at supersonic speeds. [17] Later, he amended the Barrowman Equations to cover the transonic region. [33] It is worth noting that since the location of the $C_p$ is dependent on the Mach number, its location changes during flight. Since it is critical that the stability condition be maintained throughout the flight, designers must calculate the range of locations for the $C_p$ based on the flight plan.

### *2.3 Flutter*

Flutter is an aeroelastic phenomenon that is defined by the *NASA Space Vehicle Design Criteria* guide as "a self-excited oscillation of a vehicle surface or component caused and maintained by the aerodynamic, inertia, and elastic forces in the system." [1] Due to the multidisciplinary nature of flutter, a number of factors influence the onset of flutter

including Mach number, dynamic pressure, structural stiffness, total mass, mass distribution, system dynamics, and free play. Once flutter begins, it almost always leads to catastrophic failure of the excited component. [1]

Flutter was first studied in aircraft as early as 1916, but it was not until the work of Theodorsen, in 1934 [35], that an exact solution for predicting flutter was found. Despite this solution, flutter-related accidents were not uncommon, and various modes of flutter were discovered including wing flutter, control surface flutter, and flutter on tail surfaces. Both flight testing and predicting flutter became important in the development of aircraft. By the end of World War II, transonic aircraft and rockets necessitated further advances in the prediction and testing of flutter. [20] In the United State much of this research was conducted by NASA and the US Department of Defense, the results of which will be discussed in detail in the following section.

Two criteria are currently used to describe the likely onset of flutter, flutter velocity $V_f$ and flutter dynamic pressure $Q_f$. The flutter velocity is the velocity at given flight conditions (Mach number, altitude, etc.) beyond which a small disturbance is very likely to initiate flutter in the component under inspection. Similarly, flutter dynamic pressure is the dynamic pressure at given flight conditions beyond which a small disturbance is very likely to initiate flutter in the component under inspection. [23]

Though rockets can encounter many types of flutter beyond those encountered by aircraft, the most common type of flutter encountered by rockets is classical two-dimensional, or bending-torsion, flutter of the fins, which is also seen in wings on aircraft. [28] In this case, flutter occurs when the first bending and torsion structural resonance frequencies of the fins occur at the same frequency. [26] Similar to aircraft, once flutter occurs on rocket fins, it is most likely to lead to loss of the fin and ultimately the rocket due to the unbounded growth in the oscillations from insufficient damping. For this reason, NASA established guidelines early on that require a minimum 15% factor of safety between the vehicle's velocity and the flutter velocity and a 32% factor of safety between the vehicle's dynamic pressure and the flutter dynamic pressure across all portions of a vehicle's flight. It is important to note that predicted flutter values are not sufficient to meet these crite-

ria; NASA insists that at least one flight be carried out with instrumentation to verify the calculated results. [1]

### *2.4 Predicting Flutter on Rocket Fins*

In the web page "Fin Flutter" [9], the author uses the material from the NACA Report TN 4197 to derive an equation for the flutter velocity of a rocket fin. Starting with Equation (5), which is Equation (18) in TN 4197 [28], taking the square root of both sides yields Equation (6). Then taking $a$, the speed of sound, from the left side to the right side leaves Equation (7), shown in Chapter 1 as Equation (1). One can see from Equation (7) that the flutter velocity of a fin based on its geometry, shear modulus, and altitude [9]. One potential use of Equation (7) is to graph the predicted rocket velocity and the fin flutter velocity versus time and ensure there is a minimum 15% buffer between the two curves. As long as fin geometry is within the assumptions of TN 4197, specifically that the fin is a solid, thin airfoil, this method should provide one with reasonable assurance that flutter will not occur.

$$(\frac{V_f}{a})^2 = \frac{G_E}{\frac{39.3A^3}{(\frac{t}{c})^3(A+2)}(\frac{\lambda+1}{2})(\frac{p}{p_0})} \tag{5}$$

$$\frac{V_f}{a} = \sqrt{\frac{G_E}{\frac{39.3A^3}{(\frac{t}{c})^3(A+2)}(\frac{\lambda+1}{2})(\frac{p}{p_0})}} \tag{6}$$

$$V_f = a\sqrt{\frac{G_E}{\frac{39.3A^3}{(\frac{t}{c})^3(A+2)}(\frac{\lambda+1}{2})(\frac{p}{p_0})}} \tag{7}$$

$V_f$, Flutter Velocity

$a$, Speed of Sound

$G_E$, Effective Shear Modulus

$A$, Panel Aspect Ratio

$t$, Fin Thickness

$c$, Chord

$\lambda$, Taper Ratio (Ratio of Tip Chord to Root Chord)

$p$, Air Pressure at Altitude

$p_0$, Air Pressure at Sea Level

In this same NACA report, Martin proposes using the relationship between geometry elements to create a partial flutter factor X, as defined in Equation (8). Martin then combined this with altitude of launch (in the form of a pressure ratio $p/p_0$) and an additional geometric element ($(\lambda + 1)/2$) to create a total flutter factor as seen Equation (9) [28].

$$X = \frac{39.3A^3}{(\frac{t}{c})^3(A+2)} \tag{8}$$

$$TotalFlutterFactor = \frac{p}{p_0}\left(\frac{\lambda+1}{2}\right)X \tag{9}$$

Martin then goes on to compare success and failure of wind tunnel models and fins from sounding rockets and missiles to create an empirical relationship between Equation (9) and the shear modulus of the material of which the fin is made to predict if the fin will encounter flutter during its flight. Martin documented this relationship in Figure 7. Points below the diagonal line represent fin designs that will not flutter (the further below the line, the greater the factor of safety), points above the line represent fin designs that will likely flutter, and points along the line represent marginal designs. [28]

Martin gives an example of how to use the chart. Assume a fin's geometry has been selected but not the material used in its construction. If the fin's geometry yields a partial flutter factor X of $1.25 \times 10^6$ psi, and it is launched from sea level (making $p/p_0 = 1$), and it is untapered (making $(\lambda + 1)/2 = 1$), the total flutter factor on the y-axis is $1.25 \times 10^6$ psi. Reading across Figure 7 one can predict the likelihood the fin will experience flutter. If the fin were made of magnesium (denoted by the red point), it would be at high risk of flutter. If the fin were made of aluminum (denoted by the yellow point), it would be a marginal design. Lastly, if the fin were made of steel (denoted by the green point), it would be very unlikely to flutter. [28]

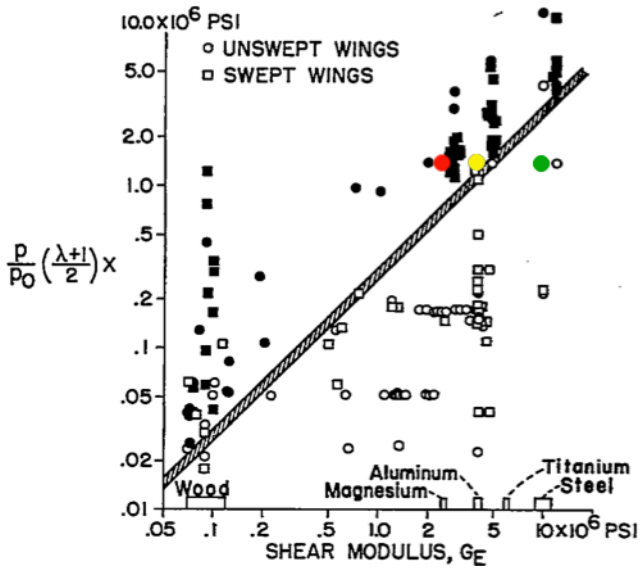



Figure 7 NACA TN 4197 Figure 3.- Composite chart for bending-torsion flutter

In 1962, researchers at Johns Hopkins University, as part of a research program on supersonic flutter prediction for thin plates (such as sounding rocket fins) for the US Bureau of Naval Weapons, developed an equation for a then common flutter parameter $\frac{V_f}{b\omega_\alpha}$, which can be seen in Equation (10). The researchers derived this equation from a simplified case of the two-degree-of-freedom flutter equations, where the only force acting on the system is lift due to supersonic flight, $N\alpha$, placed at the elastic axis or axis of rotation, see Equations (11) and (12). These equations form a system of linear equations that signify flutter when the determinant equals zero. The solution to this case, after simplification results in the Johns Hopkins flutter prediction seen in Equation (10). Note, due to the assumption of super sonic lift, this equation is only valid for supersonic flight conditions. [24]

$$\frac{V_f}{b\omega_\alpha} = \sqrt{\frac{\mu\bar{r}^2\sqrt{M^2-1}}{\bar{x}b} * \frac{[1-(\frac{\omega_h}{\omega_\alpha})^2]^2 + 4(\frac{\bar{x}}{\bar{r}})^2(\frac{\omega_h}{\omega_\alpha})^2}{2[1+(\frac{\omega_h}{\omega_\alpha})^2]}} \tag{10}$$

$V_f$, Flutter Velocity (ft/sec)

$b$, Fin Semi-chord (ft)

$\omega_\alpha$, Uncoupled Torsional Frequency (rad/s)

$\omega_h$, Uncoupled Bending Frequency (rad/s)

$\mu$, Mass Ratio Parameter = $m/\rho b^2$

$m$, Mass per Unit Span (lb*s$^2$/in$^2$)

$\rho$, Air Density (lb/in$^3$)

$\bar{r}$, Radius of Gyration with respect to the Elastic Axis or Axis of Rotation (ft)

$M$, Main stream Mach Number

$\bar{x}$, Distance from the Center of Gravity to the Elastic Axis or Axis of Rotation (ft)

$$(k_{hT} - I_h\omega^2)\bar{h} - S\omega^2\bar{\alpha} = -N\bar{\alpha} \tag{11}$$

$$-S\omega^2\bar{h} + (k_{\alpha T} - I_\alpha\omega^2)\bar{\alpha} = 0 \tag{12}$$

$k_{hT}$, equals $k_h + iG_h$

$k_h$, Generalized bending stiffness

$G_h$, Damping factor in bending

$I_h$, Generalized inertia in bending

$\omega$, Frequency

$\bar{h}$, equals $he^{-i\omega t}$

$h$, Linear displacement of mid-chord, positive downward

$\bar{\alpha}$, equals $\alpha e^{-i\omega t}$

$\alpha$, Angular displacement, positive stalling

$N$, Lift force per unit angle

$S$, Generalized mass unbalance

$k_{\alpha T}$, equals $k_\alpha + iG_\alpha$

$k_\alpha$, Generalized torsional stiffness

$G_\alpha$, Damping factor in bending

$I_\alpha$, Generalized inertia in twisting

The researchers compared the results of Equation (10) to a classical flutter solution presented in NACA Report 846. NACA Report 846 applied Possio theory for small disturbances in two-dimensional supersonic flows to the problems of flutter, divergence, and aileron reversal. The results of this work were presented in a series of tables and graphs. [21] The Johns Hopkins researchers selected six fin geometries and calculated the flutter parameter for them at Mach 2 and Mach 5 using both their simplified equation and the graphs provided in NACA Report 846. When they compared the two calculations, they found Equation (10) had reasonable agreement, generally within $\pm 6\%$, with the solutions from NACA Report 846. [24]

Beyond analytical solutions like the equations above, there are a number of software packages that can be used to predict flutter. Some of these are specifically designed to predict flutter of rocket fins, for example AeroRocket's AeroFinSIm$^{\text{TM}}$, while others are designed to predict flutter for a wide range of configurations, for example ZONA Technology's ZAERO$^{\text{TM}}$. AeroFinSim provides an option to solve for flutter velocity using either an analytic method or the U-G method, an iterative solution that is based on the K method. The inputs to this analysis are the fin geometry, construction material, and the rocket's maximum altitude. The computed result is predicted flutter velocity. [18]

ZAERO uses a modified P-K method Zona Technology calls the g method to solve for the flutter velocity, which transforms the flutter calculation into an eigenvalue problem. As with the U-G method, this is an iterative method, best handled on a computer. Since ZAERO is not specifically designed to solve for flutter on a specific type of surface, users have to construct a separate model of the structure before they can use ZAERO to find the flutter velocity. This structural model takes the form of a finite element model constructed in one of the many popular finite element analysis tools available such as a version of Nastran. The inputs to the analysis are the the output from the finite element model's normal modes analysis, and altitude and velocity data (for instance a range of altitudes

for a given velocity or vice versa). The user must configure a transform function between ZAERO's built-in aerodynamic model and the Finite Element Model. [5]

Flutter can be predicted using scale wind tunnel tests. To run scale wind tunnel tests, a scale model of the fin must be created that is dimensionally scaled so that the size of the fin is scaled to fit within the wind tunnel, aerodynamically scaled so that the aerodynamic response of the fin is scaled for the environment of the wind tunnel, and structurally scaled so that the dynamic structural response is scaled to match the size of the test fin. The dimensional scaling is a straight forward process and is driven by the size of the wind tunnel being used. Aerodynamic scaling is accomplished as with other wind tunnel test by establishing similarity parameters. Finally, structural scaling is accomplished by establishing influence coefficients. These coefficients can be calculated based on the fin's design, but should then be verified by testing the vibration responses of the actual fin against the calculated values. [27]

Once the scaling parameters have been calculated, a physical model can be constructed. The model should be checked as it is constructed to ensure it meets all of the scaling parameters, and adjusted as necessary. Once built, the model fin is ready for use in the wind tunnel. Care must be taken to avoid letting the fin experience flutter, both to preserve the model if possible and to prevent the model from damaging the wind tunnel. [27]

### *2.5 Design Optimization*

Design optimization is a process of refining a design in order to reach a desired level of performance for one or more features [2]. Implementing this process by hand can be time consuming, which has led to the development of automated optimization tools such as MSC.Nastran's Design Sensitivity and Optimization$^{TM}$package and Phoenix Integration's Optimization Toolkit$^{TM}$for ModelCenter. The following review of these tools and their use in optimizing geometries is intended to facilitate the selection of an automated optimization tool for use in the development of the sounding rocket fin design tool.

MSC.Nastran's Design Sensitivity and Optimization package allows users to conduct a design optimization on Nastran models. As with other automated optimization tools, users start with an initial design. Users then relate design parameters to a model of the initial design and conduct a sensitivity analysis to determine which variables have significant impact on the desired optimization. Finally, users create an optimization study with one or more goals, constraints, and a list of design variables to adjust during the optimization. The one case where this process must be modified in MSC.Nastran is when the optimization is modifying the geometry. When modifying geometry in MSC.Nastran, the user must create shape basis vectors that are used to modified sets of grid locations based on the design variables related to the shape of the model under optimization. The more complex the model under optimization, the longer the basis vectors need to be. Creating these vectors requires manual input by the user or the development of a second model to act as a boundary for the geometry optimization. [2]

The Optimization Toolkit from Phoenix Integration acts as enhanced trade study tools within ModelCenter. In order to use the Optimization Toolkit, users must first create a ModelCenter model of the initial design by integrating one or more analysis codes, such as an Excel$^{TM}$spreadsheet or a MATLAB$^{TM}$script. Integrating analysis codes into ModelCenter involves selecting which parameters of the analysis to treat as design variables and responses and then using one of the ModelCenter "PlugIns" or "Wrappers" to bring that analysis into the ModelCenter framework. As with MSC.Nastran, users then run a sensitivity analysis using one or more trade study tools (such as a Parametric Study or a Design of Experiments). Users then use one of the optimizers included in the Optimization Toolkit using a user interface that is very similar to the trade study tools. Unlike MSC.Nastran, there are no special processes to use when optimizing geometry beyond integrating an analysis code that represents the geometry. [8]

The sounding rocket fin design tool developed in this research will be built on Phoenix Integration's ModelCenter. This decision is based on ModelCenter's consistent treatment of geometric and non-geometric data, and the author's familiarity with ModelCenter.

### *2.6 Successful Fin Geometries*

Figures 8, 9, 10, and 11 show fins from four successful designs: the AIM-9B Sidewinder missile in Figure 8, the Cajun upper stage of the Nike-Cajun sounding rocket in Figure 9, the Pogo Hi sounding rocket in Figure 10, and the second stage of the Aerobee 170 sounding rocket in Figure 11. Two of these designs, the Sidewinder and the Cajun, are of a different family of fins than discussed previously, the clipped-delta family. Unlike fins derived from standard wing design parameters, the clipped-delta fins are derived from delta wing design parameters, namely root chord and span, with an additional parameter, the tip chord. Note, delta fins, like delta wings would have a tip chord of zero, as they come to a point at the tip. Clipped-delta fins all exhibit a straight trailing edge and a swept leading edge (like delta wings), with the sweep being determined by the span and root chord instead of being an independent design variable. Experimental tests show that the flutter characteristics of these types of fins are highly coupled to their aspect ratio, with lower aspect ratios having higher flutter velocities. [22] It is interesting to note that not only do the two clipped-delta designs in Figures 8 and 9 have low aspect ratio, but so do the other two designs in Figures 10 and Figure 11, implying that the correlation between aspect ratio and flutter velocity may apply to sounding rocket fins more generally. This is further supported by Equation (7), in which the flutter velocity is inversely proportional to the aspect ratio.

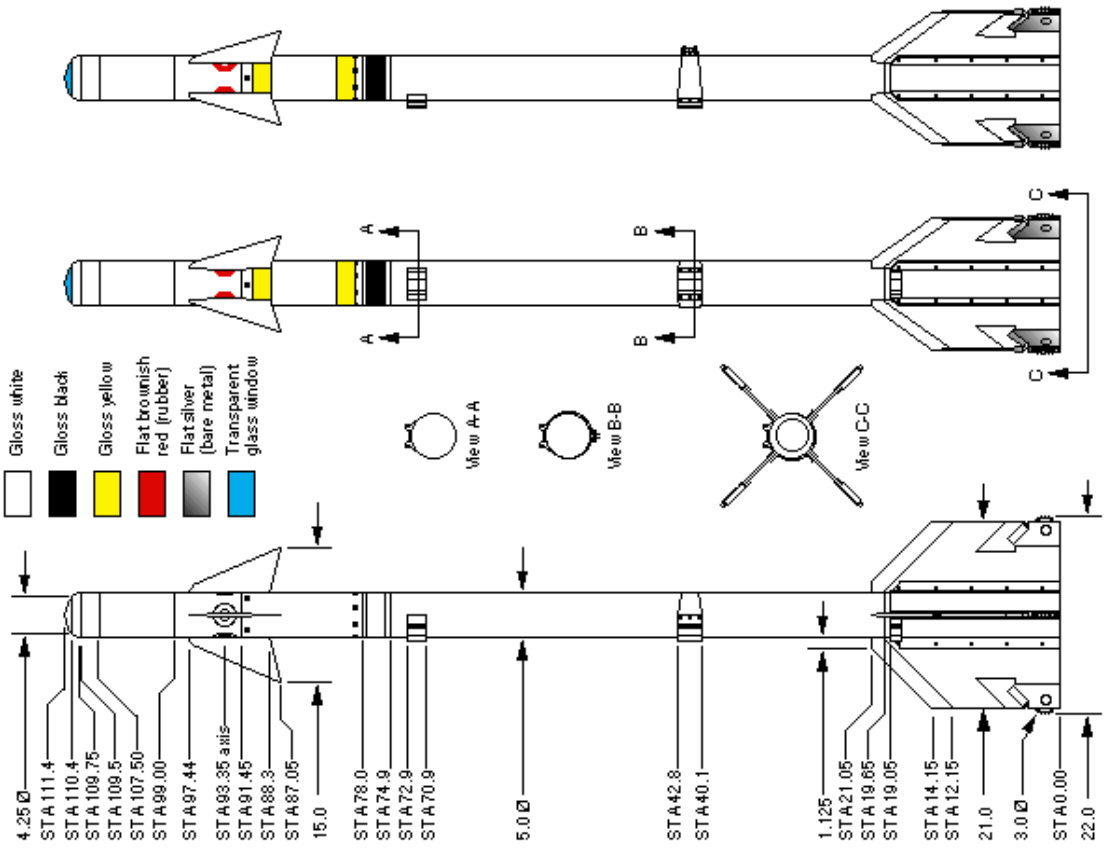



Figure 8 AIM-9B Sidewinder Diagram a clipped-delta design, dimensions in inches [15]

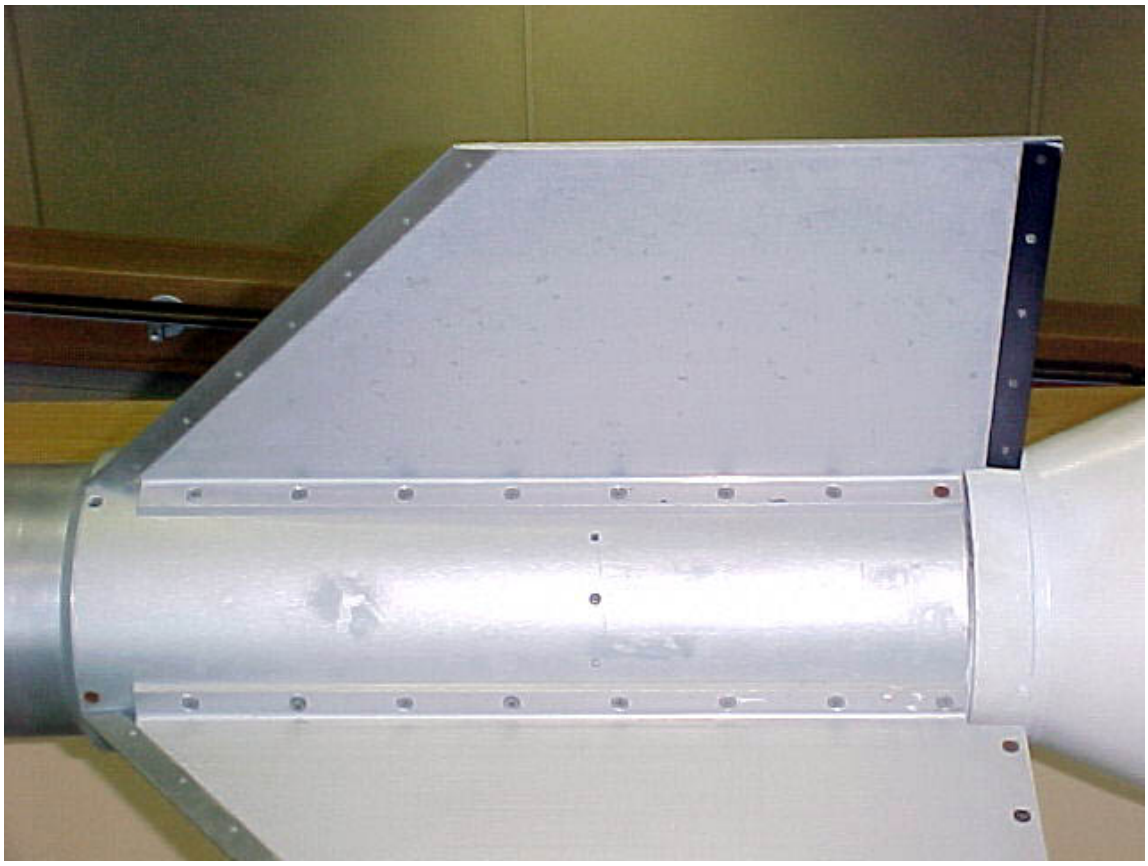


Figure 9 Cajun Sounding Rocket Fins a clipped-delta design [10]

Figure 10 Pogo Hi Sounding Rocket a delta design

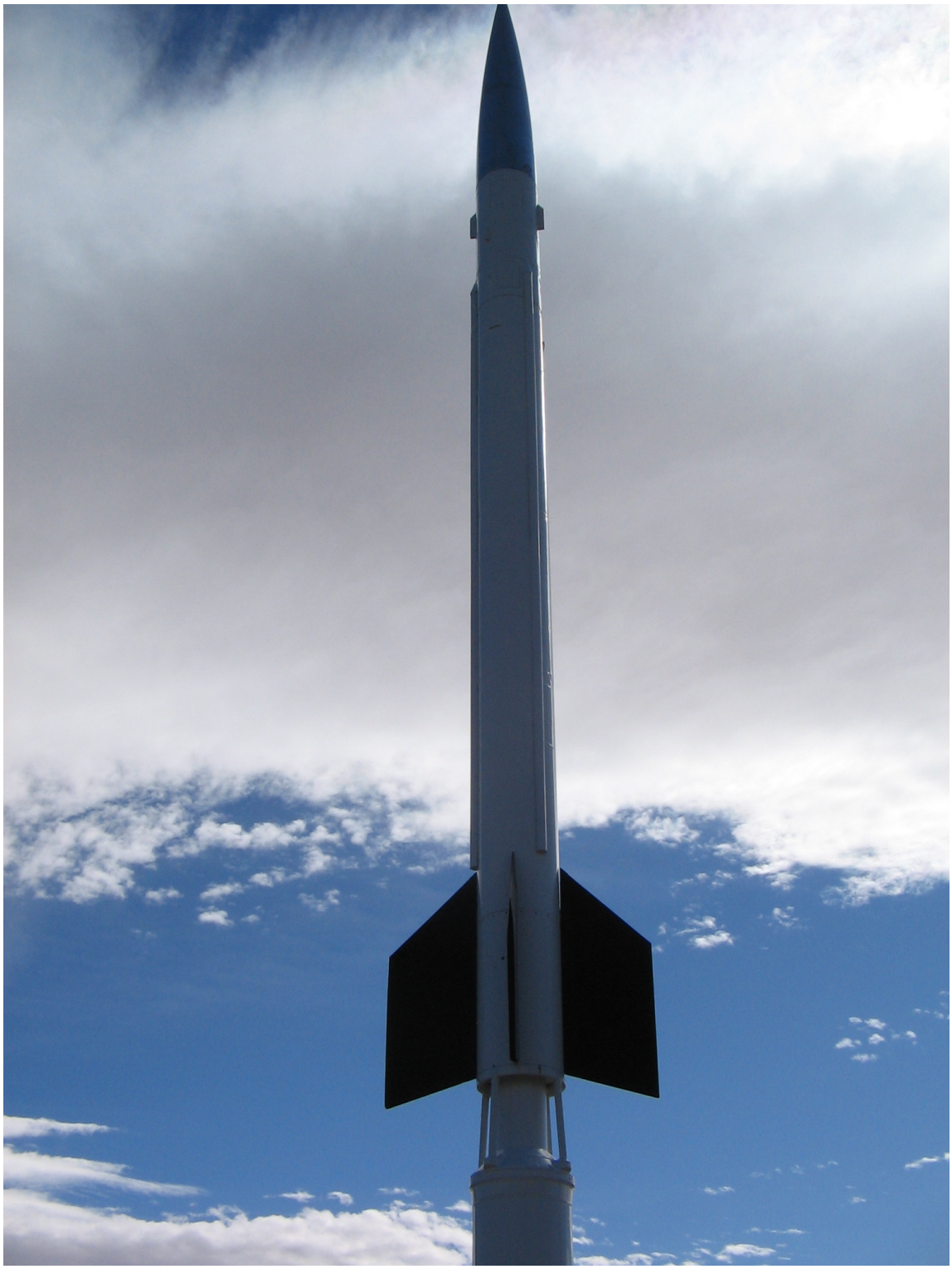


Figure 11 Second Stage of the Aerobee 170 Sounding Rocket a traditional design

## *2.7 Summary*

This chapter has shown several key issues related to sounding rocket fins, stability, and flutter that can be applied to this research effort. First, in order to maintain stability, the optimized fins need to maintain a minimum panel area. Second, a number of flutter calculation techniques are available for use in the design tool, including Equation (7), Equation (9), Equation (10), and developing a ZAERO$^{TM}$ model of the fin. The design tool will make use of one or more of these techniques to enforce the required minimum 15% factor of safety between the vehicle's velocity and the flutter velocity. Third, two examples of automated design optimization tools were compared. The design tool will use Phoenix Integration's ModelCenter for the design optimization process for its consistent treatment of design variables, especially those relating to geometry. Lastly, the selection of successful fin geometries implies that optimal designs will tend toward low aspect ratios.

# 3. Methodology

## 3.1 Chapter Overview

This chapter covers the development of the design tool. The chapter begins with an overview of the software used in the design tool and the software used to verify the designs it creates. Next the chapter covers the development of the design tool, which is an iterative, two-stage process, seen in Figure 12. Note, the development process will explore two versions of each of the stages. The first stage determines the fin design parameters that will give the minimal mass and still prevent flutter. The second stage validates the optimized fin geometry by comparing the flutter velocity to the required 15% factor of safety beyond the rocket's predicted velocity throughout its flight profile. The process is repeated until the design meets the flutter requirements.

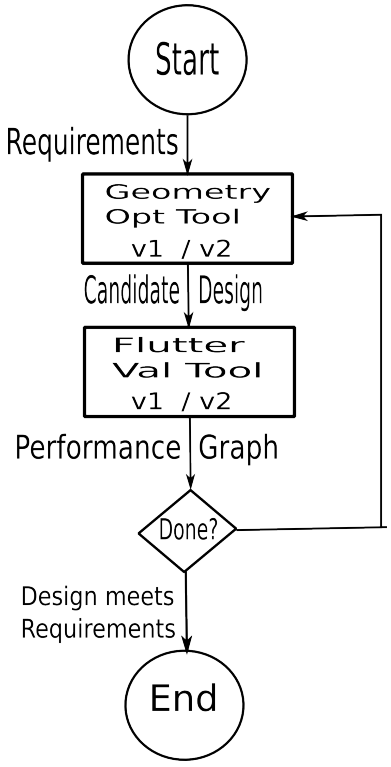



Figure 12 Stages of the Sounding Rocket Fin Design Tool

*3.2 Software used in this research*

*3.2.1 AeroLab v1.3.4.* Aerolab is an application to calculate the drag, lift, and stability for rockets for use by amateur rocket builders in predicting the performance of their rockets. It is written by Hans Olaf Toft of the Danish Amateur Rocketry Club. To use Aerolab, a user first specifies the key design parameters for the rocket (including specifying nose cone, body, and fin parameters such as root and tip chords, span, and thickness) and its flight profile in a wizard style dialog box accessed by pressing the Compose button. The results are then displayed in the main window as seen in Figure 13. The user then presses the Calculate button to run the the complete set of analyses. Once run, the plot window, seen in Figure 14, opens and the user can select from a variety of plotting results, including drag coefficients, lift coefficients, and centers of pressure versus Mach number. [36]

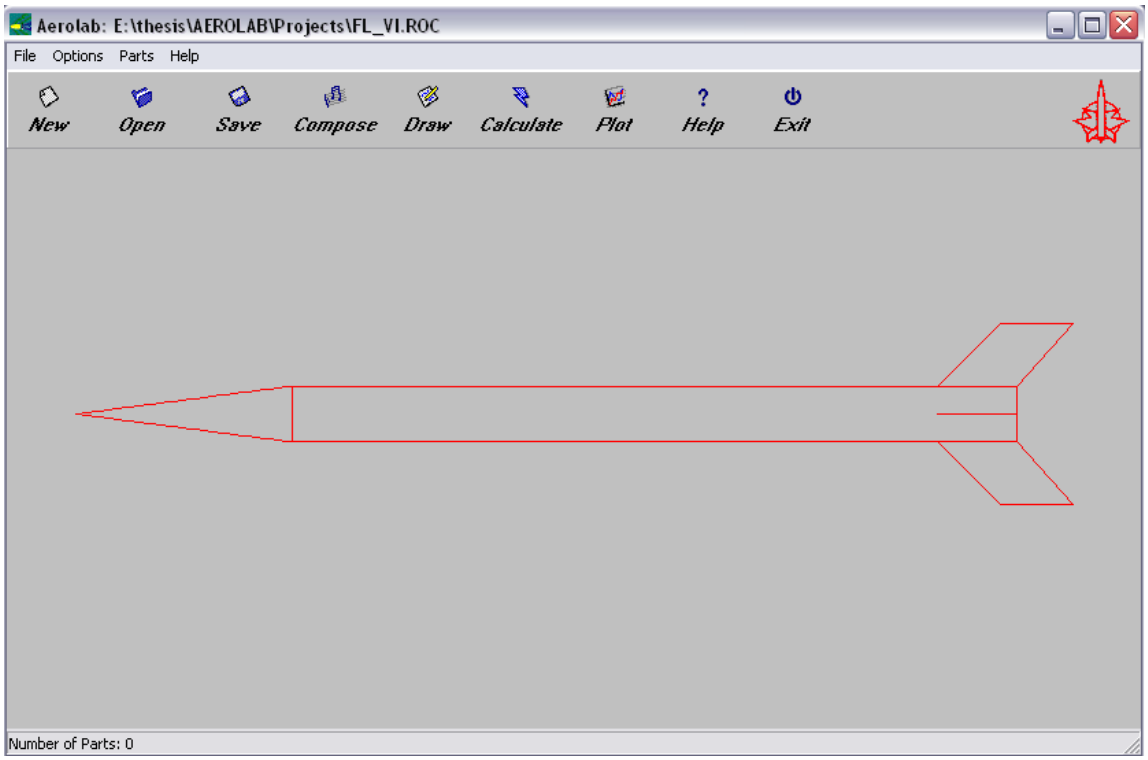



Figure 13 Aerolab Main Window

AeroLab will be used to verify that the optimized fins provide the required Center of Pressure location to maintain stable flight.

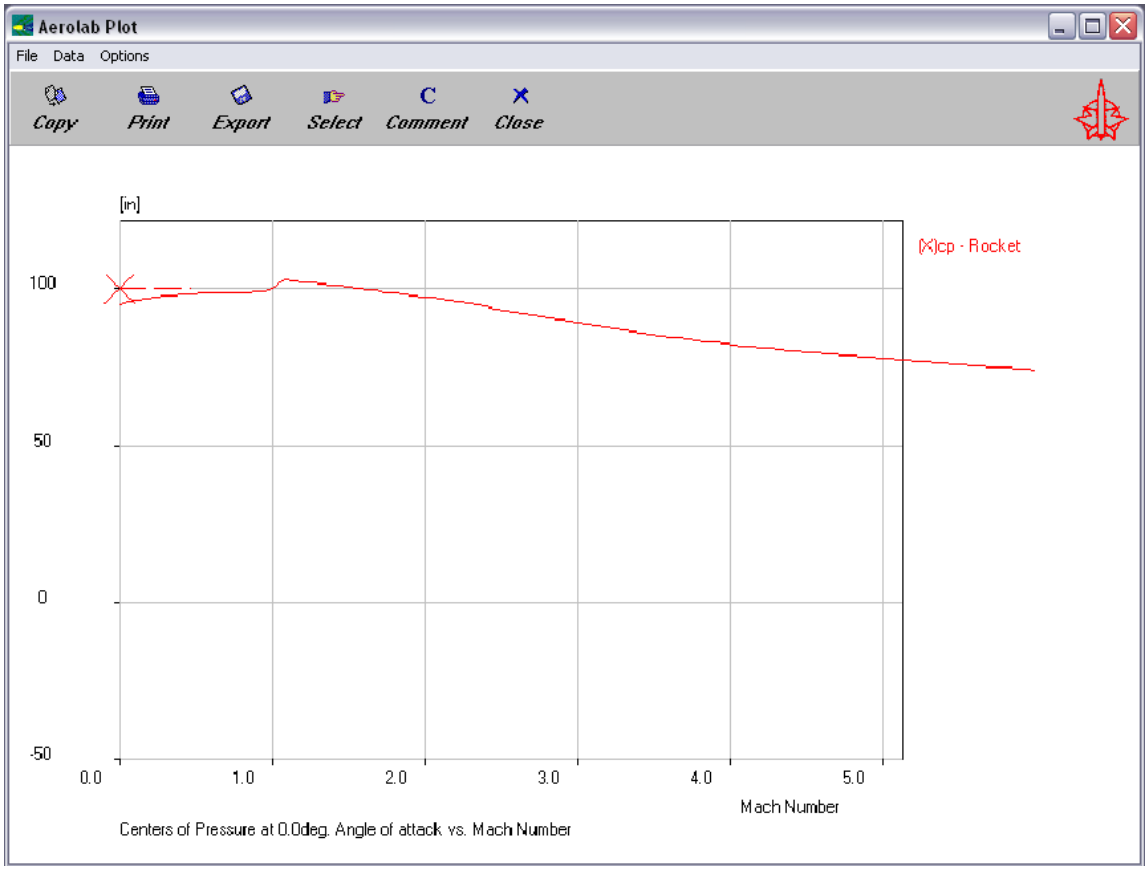



Figure 14 Aerolab $C_p$ Plot, in inches from nose

*3.2.2 FEMAP v9.31.* FEMAP is a finite element pre and post-processor that comes bundled with NX Nastran, produced by Siemens PLM Software. FEMAP can be used to create finite element models on which a variety of analyses can be performed, including thermal, stress/strain, and modal analyses. FEMAP can also import geometries from CAD files. From the CAD model, a finite element discretization can be created, before configuring material properties, boundary conditions, and any necessary loads for the finite element analysis. The user can then have FEMAP run the analysis using either the bundled copy of NX Nastran or another finite analysis package of their choice by writing out an appropriate input file. Finally, the results of the analysis are imported back into FEMAP where they can be viewed graphically. Figure 15 shows the main interface of FEMAP. [6]

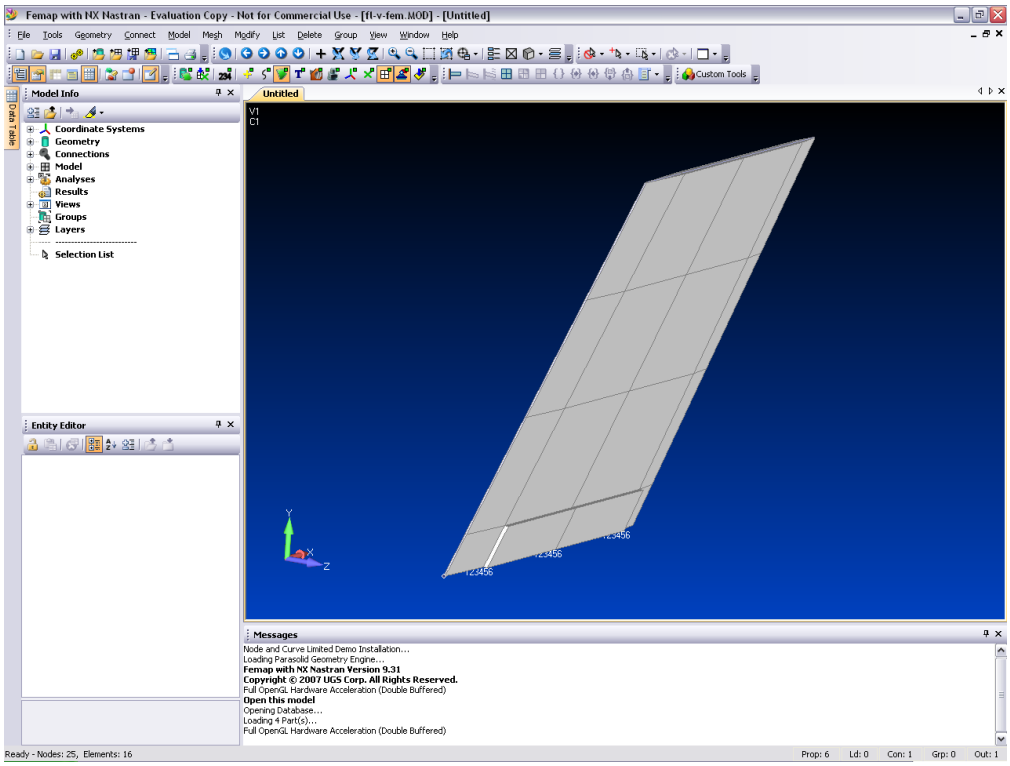



Figure 15 FEMAP Main Interface

FEMAP will be used to run Nastran analyses which calculate structural values for the optimized fin designs such as resonant frequencies, location of the center of gravity, and the moments of inertia.

*3.2.3 Mathcad v14.* Mathcad™ is a symbolic math documentation and solving application by Parametric Technology Corporation. Mathcad allows users to enter math formulas and equations in a natural style making them easier to read and validate. Mathcad can also produce the results of the formulas for given values of input variables and symbolically solve algebraic equations, integrals, and derivatives. Figure 16 shows the Mathcad interface with a worksheet that solves for flutter velocity using Equation (7). [7]

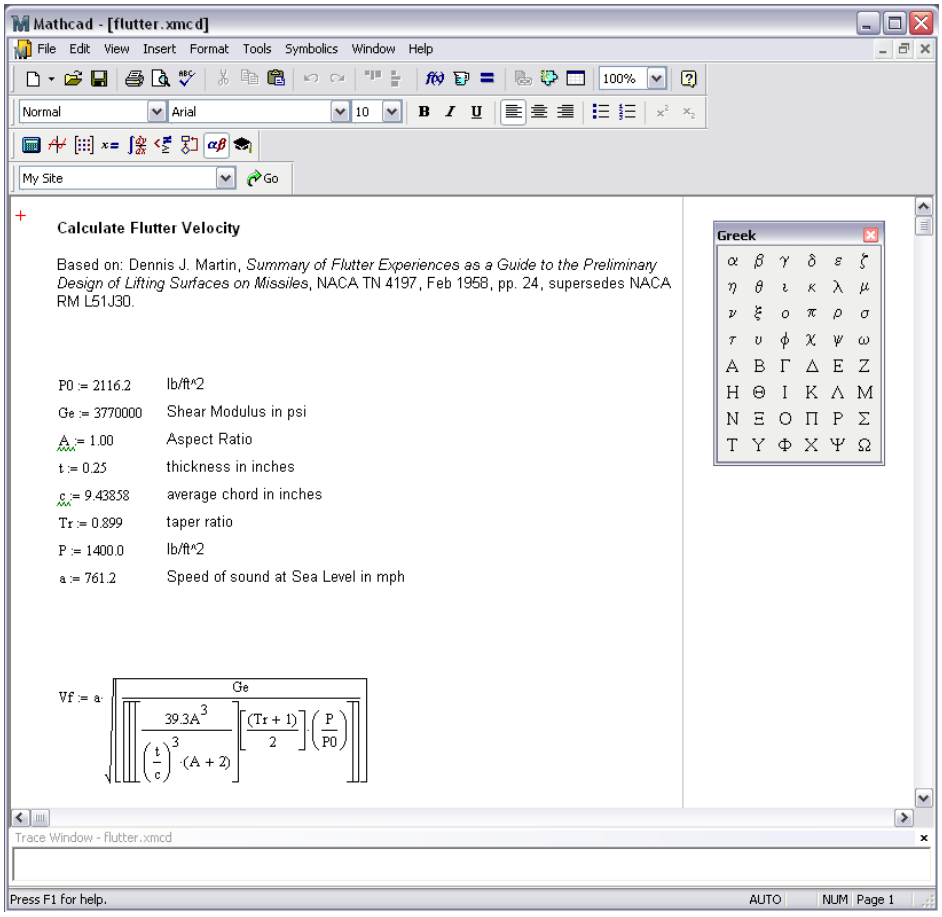



Figure 16 Mathcad Main Window

Mathcad will be used to implement and document the flutter calculations used in the design tool (such as Equation (7), Equation (9), Equation (10)). Mathcad's equation rendering will allow for easier validation and debugging of the implementations of these equations, which will speed up the development of the design tool.

*3.2.4 MATLAB R2007b.* MATLAB™ is an interpreted programming language that is designed for technical computing. The name "MATLAB" stands for "matrix laboratory", a reference to MATLAB's built in library of operations and functions for manipulating and using matrices in calculations. These features make MATLAB particularly well suited for use in solving systems of linear equations and other linear algebra operations.

MATLAB will be used to develop a post-processing tool to derive the radius of gyration relative to the elastic axis, $\bar{r}$, and distance from the elastic axis to the center of gravity, $\bar{x}$, for use in Equation (10) from the output of FEMAP finite element models of the optimized fin designs. [4]

*3.2.5 ModelCenter v8.0.* Phoenix Integration's ModelCenter™ is a multidisciplinary modeling environment that can be used to study the trade space of a design and optimize that design. ModelCenter can combine analyses developed in a variety of tools including MATLAB, Mathcad, Excel, command line executables, and more, into a single system level model as seen in Figure 17. Users can then use ModelCenter's built in trade study and visualization tools to conduct sensitivity studies to gain insight into what aspects of the design are the key drivers influencing the desired results. ModelCenter also provides a rich set of built-in optimization tools that can then be used to find values for the key drivers that optimize the design for a given goal (such as minimizing the mass of a fin) while ensuring constraints are not violated (such as maintaining the predicted velocity 15% below the flutter velocity). These optimization tools include a gradient optimizer, a genetic optimizer called Darwin™, and an optimizer that utilizes surrogate models of the design space during optimization called DesignExplorer™. [8]

ModelCenter will be used as the framework for the design tool. ModelCenter models will be built for each of the two stages, which will then be exercised by the built in trade study and optimization tools.

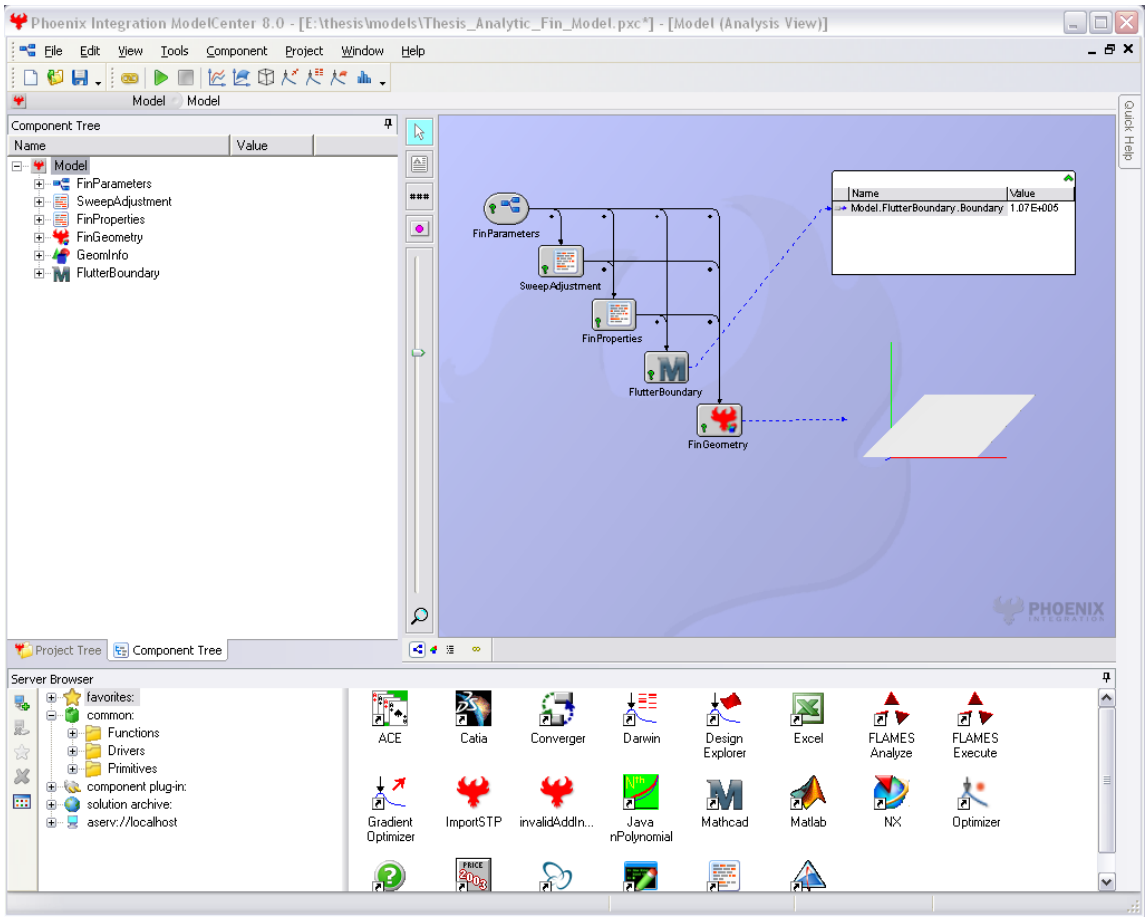



Figure 17 ModelCenter Main Window

*3.2.6 ZAERO v8.3.* ZAERO$^{TM}$ is an aeroelastic solver developed by ZONA Technology, Inc. It includes aerodynamic models that cover the full range of Mach numbers from subsonic to transonic to hypersonic. It can solve a number of aeroelastic calculations including fluter, static aeroelastic analyses, gust loading, and aeroservoelastic analyses. ZAERO does not include a finite element solver, so structural details concerning the design being studied must be generated in an outside finite element solver such as NASTRAN. ZAERO is run in a batch mode by creating an input file that specifies the analysis to run, and then executing the ZAERO program with this input file specified as an input. The results of the analysis are then placed in a newly created output file. [5]

### *3.3 Fin Geometry Optimization Tool*

The Fin Geometry Optimization Tool is the first stage of the overall design tool. The function of the Fin Geometry Optimization Tool is to find values for the fin design parameters (root chord, tip chord, semi-span, thickness, and sweep angle) that minimize the mass of the fin while ensuring that the Center of Pressure is located correctly for stable flight, and that fin flutter will not occur. Since Aerolab does not support batch processing, it cannot be integrated into a ModelCenter model, which means for enforcing the stability requirement. As explained in Section 2.2, a fin's panel area has the greatest influence on the location of the Center of Pressure, so the constraint on the location of the Center of Pressure will be enforced by requiring valid designs have a minimum panel area. To ensure this simplification does not lead to unstable designs, it will be verified by running a Center of Pressure analysis in Aerolab on all candidate designs.

Two different implementations of the constraint related to flutter will be considered. Both will be based on Martin's equations [28]. The first will calculate the flutter velocity of the fin at a specified altitude using Equation (7) ensuring it meets a specified minimum value. The second will calculate the total flutter factor from Equation (9) ensuring it does not exceed a specified a maximum value based on Figure 7.

The optimization process will be implemented by running a DesignExplorer $^{TM}$optimization. DesignExplorer is a Boeing-based optimization tool, included in the Phoenix Integration Optimization Toolkit, that incorporates surrogate models that can smooth potentially noisy design spaces and save computation time by limiting the number of runs executed on the full model [8]. The design variables to be optimized will be determined by running a design of experiments to determine which variables have a significant impact on the flutter velocity.

*3.3.1 Version 1: Flutter Velocity as Flutter Constraint.* The ModelCenter model for version 1 of the Fin Geometry Optimization Tool includes three parts. The first is a VBScript component that takes the fin design parameters (root chord, tip chord, semi-span, thickness, and sweep angle) defined in Figure 3 and calculates the derived values that are used as inputs in Equation (7) (aspect ratio, average chord, and taper ratio).

The next part of the model is a Mathcad worksheet that calculates the flutter velocity at an altitude specified by an atmospheric pressure using Equation (7). The last part uses a Phoenix Integration provided code to render the geometry of the current fin design for visual feedback to the user. The ModelCenter visual representation of the model can be seen in Figure 18.

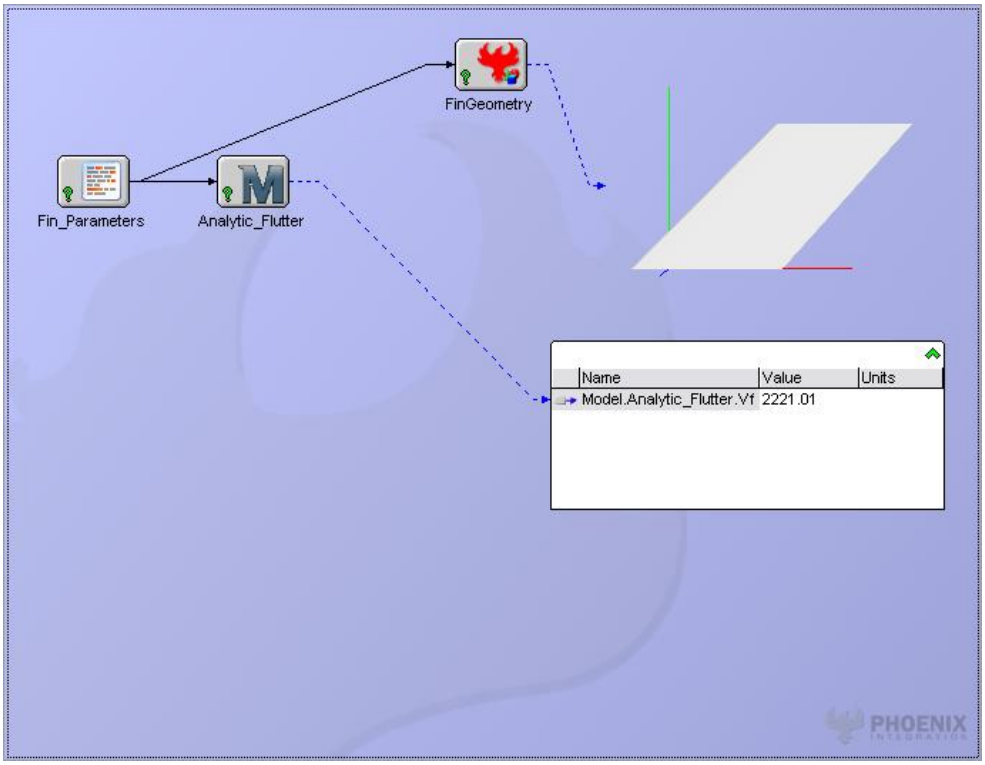



Figure 18 ModelCenter Fin Flutter Model

*3.3.2 Version 2: NACA Report TN 4197 Flutter Criteria as Flutter Constraint.* It is worth noting that to use version 1 of the Fin Geometry Optimization Tool, the user must select a single altitude and velocity pair to specify the flutter constraint. Since flutter varies with altitude, it is not obvious what pair the user should select to ensure a useful result. What is needed is a flutter constraint that can reflect the entire flight profile of the sounding rocket. This is exactly how Martin intended Equation (9) be used. The ModelCenter model for version 2 of the Fin Geometry Optimization Tool is similar to version 1 with one exception. The Mathcad worksheet in version 2 calculates the total flutter factor from Equation (9) instead of the flutter velocity, which will be compared to

a value from Figure 7 selected to ensure flutter will not be encountered. See Section 2.4 for details concerning how to select an appropriate value.

### *3.4 Flutter Factor of Safety Validation Tool*

After finding one or more potentially optimal fin geometries using the Fin Geometry Optimization Tool, it is important to validate the design across the entire predicted flight profile. This is the purpose of the Flutter Factor of Safety Validation Tool. The validation tool will calculate the flutter velocity for the altitudes and Mach numbers of the predicted flight profile, and then allow the user to compare the flutter velocity to the 15% factor of safety required by NASA guidelines. Two methods of calculating the flutter velocity will be investigated. The first will use a ZAERO model of the optimized fin design. This will require the development of a matching finite element model. The second method will use Equation (10), which will require finding the location of the elastic axis of the fin in order to calculate the radius of gyration relative to the elastic axis, $\bar{r}$, and the distance from the center of gravity from the elastic axis, $\bar{x}$. The elastic axis can be found using the results of the finite element model used in the first method.

With both flutter calculations requiring a finite element model of the optimized fin design, the development of the Flutter Factor of Safety Validation Tool will start with developing a finite element model of the fin. Since the Flutter Factor of Safety Validation Tool needs to calculate the flutter velocity based on the fin design parameters used in the Fin Geometry Optimization Tool, this finite element model needs to be configurable based on the fin design parameters, that is it needs to be parameterized. Once the parameterized finite element model is complete, the two flutter calculations will be formulated and then integrated into a ModelCenter model for use as the Flutter Factor of Safety Validation Tool.

*3.4.1 Parameterized Finite Element Model.* The objective of this analysis was to create a parameterized Finite Element Model of a sounding rocket fin (based on the Falcon-LAUNCH V geometry) to use in the Flutter Factor of Safety Validation Tool. Parametric modeling, used in most modern Computer Aided Drafting, is an approach to modeling that

Table 1 Measured Frequencies of Fin and Bracket [32]

| Mode | Frequency, Hz | Mode Shapes |
|---|---|---|
| 1 | 62 | First Bending |
| 2 | 212 | First Torsion |
| 3 | 417 | Second Bending |
| 4 | 580 | Second Torsion |

involves controlling the geometric properties through the use of "design variables." [31] Applying this approach to a Finite Element Model of a sounding rocket fin involves developing a model that can be reconfigured by an automated process driven by the values of the design variables used to describe the fin in the Fin Geometry Optimization Tool. To develop the parameterized Finite Element Model, a series of Finite Element Models were built, starting with a model that used a detailed solid element mesh, then one that used a detailed plate element mesh, and finally a model that used a simplified plate element mesh. The goal was to reduce the number of elements to a small enough set that a script could be easily written to modify the FEM's grid points directly based on the fin design parameters while maintaining the necessary degree of accuracy in the model's results.

*3.4.1.1 Experimental Baseline.* Before attempting to build the FEM, an experimental baseline for verification of the FEM was required. In January 2009, Simmons, Deleon, et al published a conference paper [32] that included the first four mode shapes and frequencies of the USAFA FalconLAUNCH V fin. These modes are of particular interest in bending-torsion flutter problems like those experienced by rocket fins. The test setup used in the paper is shown below in Figure 19. It consisted of the test hardware secured to the optical table, a modally tuned hammer connected to the computer as input and the Polytec Scanning Laser Velocimeter testing system. The Polytec testing system included software for analysis of the data as well as control of the 3-D laser used for measurements. The fin tested in the paper consisted of the airfoil bolted to a base section. The results from the paper are presented in Table 1 [32].

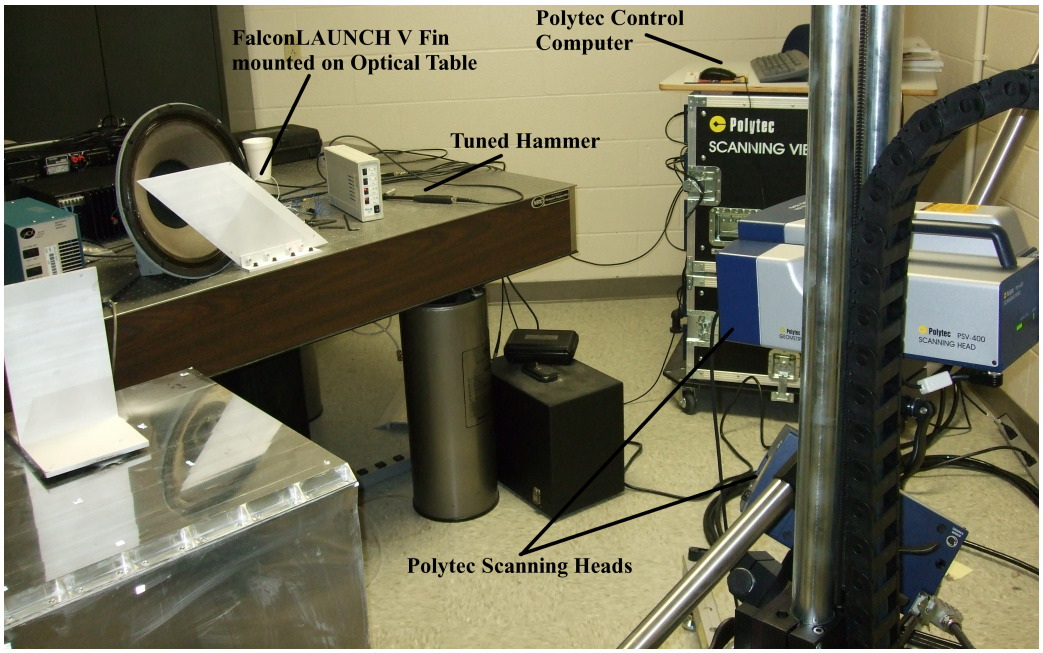



Figure 19 Lab Setup

*3.4.1.2 Model Simplifications.* All of the finite element models modeled the fin and the mounting bracket used to attach the fin to the launch vehicle. Three simplifications were used in the production of the finite element models.

1. Removal of the bolts and bolt holes

   Figure 20 illustrates there are 4 bolt holes for connecting the fin to the mounting bracket. Based on review of the post flight analysis (see Figure 2), it seems the failure occurred in the bracket structure, and not the bolts as the missing fins took the vertical parts of the brackets with them. This indicates that the bolts connecting the fin and bracket were not an area of interest, so that interface was simplified to remove the holes and bolts, and was modeled by merging coincident nodes creating one solid piece.

2. Removal of the leading edge taper on the bracket

   There is a slight taper in the leading edge of the base of the bracket. This is not a structurally significant feature, and would lead to difficulty in the meshing process, so it was removed from the finite element models.

3. Use of a flat base for the bracket

   The mounting brackets that flew on the FalconLAUNCH vehicle had rounded bases to match the curvature of the rocket case. The laboratory experiment used a flat base to facilitate mounting the fin on the test stand. The finite element models used the flat base from the laboratory experiments to maintain consistency with the experiments.

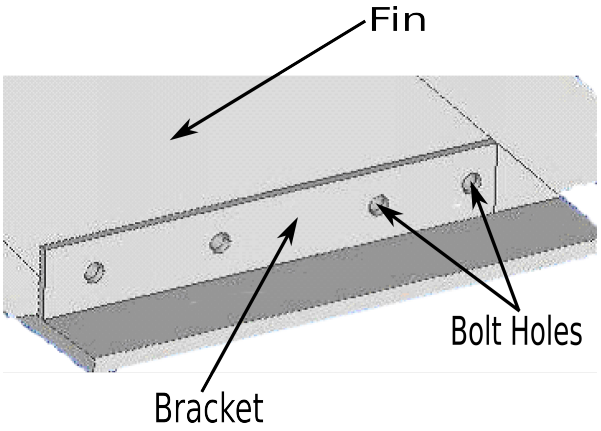



Figure 20 Detail of fin/bracket interface

The boundary conditions for the FEM model were limited to the base plate bottom surface being fixed. This simulated its attachment to the test bench. In reality the rocket body would flex depending on aerodynamic loading. Due to the nature of the fin failure during flight, this simulation seemed appropriate, and it allowed the model to be directly compared to the experimental results.

The laboratory experiment found the first four natural frequencies of the fin and bracket assembly listed in Table 1. These were compared to a NASTRAN Normal Modes/Eigenvalue analysis returns the first N modes (by default N=10).

*3.4.1.3 Solid Element mesh.* The first FEM that was created was a solid model. It was built in three steps. Step one was to mesh the fin. To do this, the fin geometry was imported and then sliced into three sections: the fin body, the leading edge and the trailing edge. Next the bolt holes were suppressed to prevent their being meshed. The size for the fin's elements was set to the width of the fin body to keep the number

manageable. The leading and trailing edges were set to have forty elements evenly spaced in the vertical direction, as were the number of elements in the vertical direction on the fin body. This was done to ensure the nodes would line up between the edges of the fin and the fin body. The number of horizontal elements on the leading edge was set to three and the number of horizontal elements on the trailing edge was set to two. Finally these three pieces of geometry were meshed using hex solids and coincident nodes were merged.

Step two was to import the bracket geometry into the fin model and mesh it. Before meshing the bracket, its geometry was aligned with the fin's mesh so the components could be easily connected after the bracket was meshed. The bracket geometry was sliced into five sections: the base, the rectangular portions of the two vertical brackets and the triangular portions of the vertical brackets. As was done for the fin, the bolt holes were suppressed to keep them from being meshed. The size for the bracket's elements was set to the width of the vertical supports to ensure that the base and the vertical supports would line up when meshed. Each piece of geometry was then meshed and coincident nodes were merged.

Step three was to attach the fin to the bracket and apply the constraints. The two meshes were combined by merging coincident nodes. When issuing the merge coincident nodes command, the user can specify the maximum spacing between nodes that will still be merged. The merge coincident nodes command was issued iteratively, setting the maximum spacing to larger values on each iteration. This process was followed until the merged meshes became deformed, after which, the step that produced the deformations was undone creating a complete and rigid connection between the bracket and the fin. Finally, a fixed constraint was applied to the bottom surface of the brackets geometry. See Figure 21 for the completed mesh on the fin and bracket with constraints applied.

Though fairly straight forward to create, it is clear from Figure 21 that this mesh is for too complex for use as a parameterized model. There are simply too many elements to alter for each design change, and if the design were to change too much, it may even require additional elements. For instance, a significantly thicker fin would likely require another set of elements to be added. This model is clearly not the correct one to use as the basis of a parameterized model.

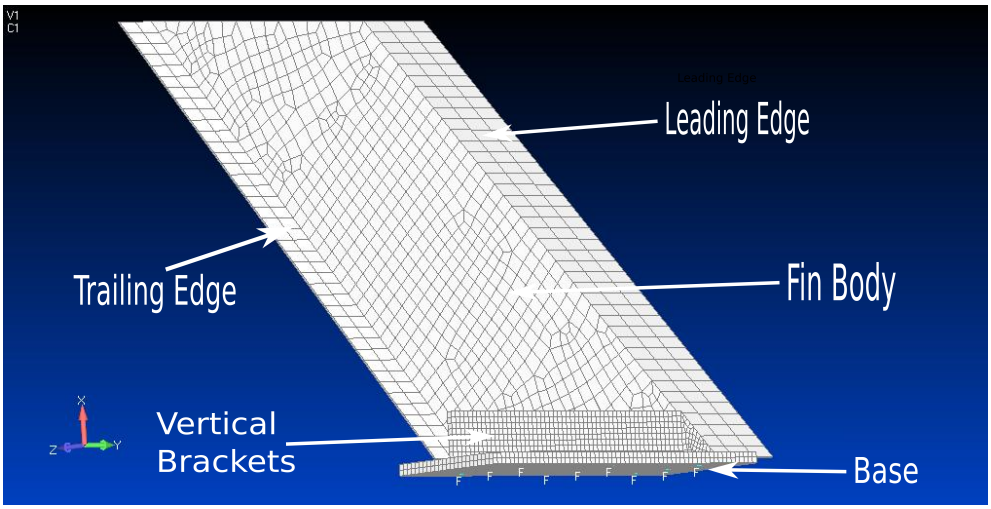



Figure 21 Solid mesh of fin and bracket with constraints

*3.4.1.4 Plate Element mesh.* In an attempt to simplify the FEM, a plate element model was created next. Using the supplied CAD geometry of the fin, midplanes were made. Once the midplanes were created, the plate elements were created on the midplanes. The number of horizontal elements in the brackets were set to match the number of horizontal elements on the base to ensure that the nodes would coincide in the merged model.

The fin was then meshed with plate elements. The fin had a leading edge and trailing edge with variable thickness, which were modeled separately to form the body of the fin. As with the brackets and the base, the number of elements in the leading and trailing edge were forced to match the elements in the body of the fin to ensure that the nodes would coincide in the merged model.

The fin, base and bracket were then combined into one model. All coincident nodes in the base and brackets were merged, as were all coincident nodes in the parts of the fin. The fin and bracket were connected using rigid elements that fix all of the degrees of freedom between the connected nodes. Two rows of these links were created with fin nodes independent and the bracket nodes dependent. After running a modal analysis in NASTRAN, the model was found to be stiffer than the laboratory results. The number of

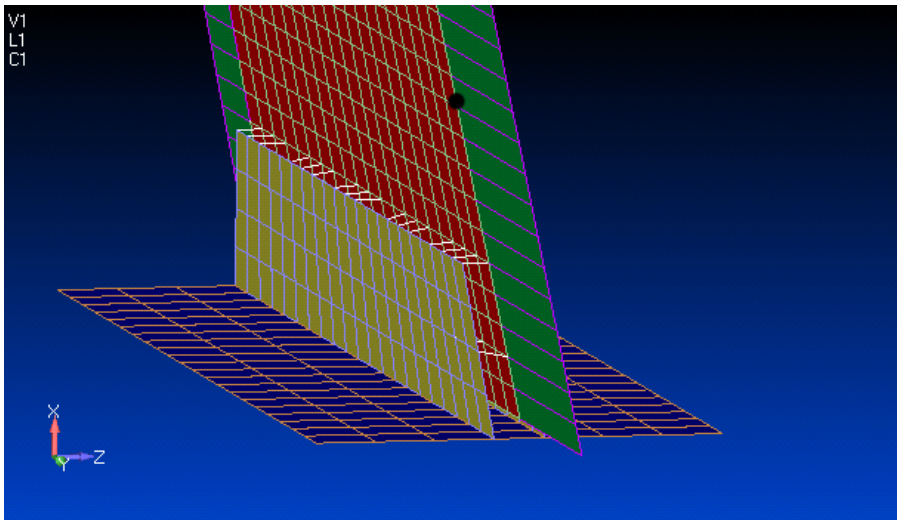



Figure 22 Plate Finite Element model with rigid links shown

rigid links was then altered until an arrangement that produced a strong correlation with experimental data was found.

Figure 22 shows the rigid links that connect the fin with the brackets. The parts are colored based on property cards and the rigid links are shown in white. The base plate is clearly visible in this figure.

The use of plate elements did simplify the model in some ways. The fin's thickness is now just a model parameter, it can be easily changed by changing the thickness of the plate elements used in the fin. However, changing the other design parameters would still be challenging due to the shear number (in the 100s) of elements present. This is still not the correct model for use as the basis of a parameterized model.

*3.4.1.5 Reduced Complexity Plate Element mesh.* As assumed, plate elements can accurately model the fin. But what is needed is a model with far fewer elements that can be easily parameterized. To that end, the next finite element model developed used a reduced complexity plate element mesh. This involved the use of both fewer plate elements and the following additional simplifying assumptions.

1. Removal of the Base Plate

   The first new simplifying assumption is that the base plate of the bracket does not

significantly contribute to the normal modes of the fin. It will be removed from the reduced complexity model.

2. The second is that the bracket-fin interface is essentially the same as a single piece of aluminum of equivalent thickness to the entire assembly. The reduced complexity model will model the bracket-fin interface as a single set of plates instead of the three sets from the previous model.

With these assumptions in mind, a new plate mesh was created. The geometry was sliced in the region where the bracket interfaces with the fin to create a separate set of plates to represent the bracket-fin interface which would be assigned a thickness based on the thickness of the fin and bracket assembly. The geometry was also sliced as before to provide for the leading and trailing edges of the fin with their own property cards to account for the taper in these surfaces.

The geometry was then meshed with far fewer elements than the previous plate mesh (25 nodes vs 100s). This was done to ensure that modifications to the model's nodes could be visually inspected to ensure accurate representation of fin geometry parameters. Finally, fixed constraints were added to the base nodes to represent the mounting in the experimental setup. The completed model can be seen in Figure 23.

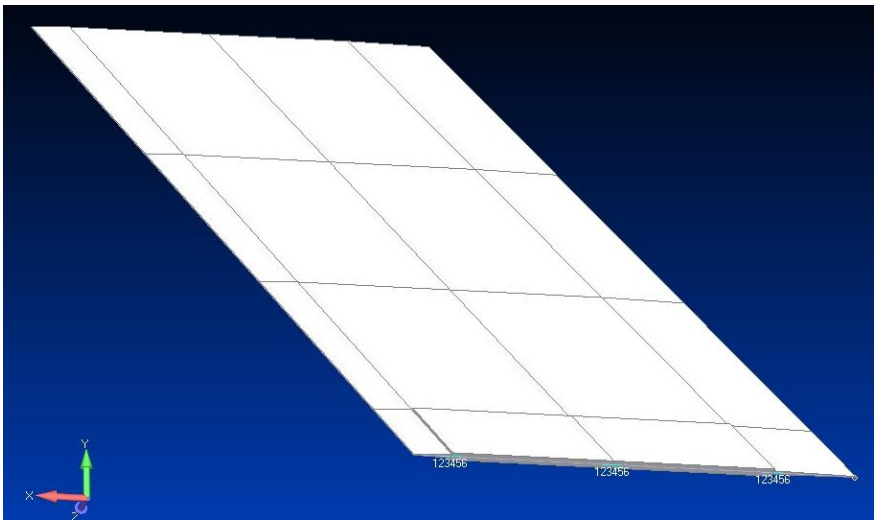



Figure 23 Reduced Plate Element model with constraints

This reduced model's 25 nodes makes it a far better candidate to be modified by a small script than the previous models, making it a strong candidate to be the basis of parameterized finite element model. However, before it can be used for this it needs to be compared to the other models and the lab results to verify its frequency response of the first two modes accurately models the FalconLAUNCH fins.

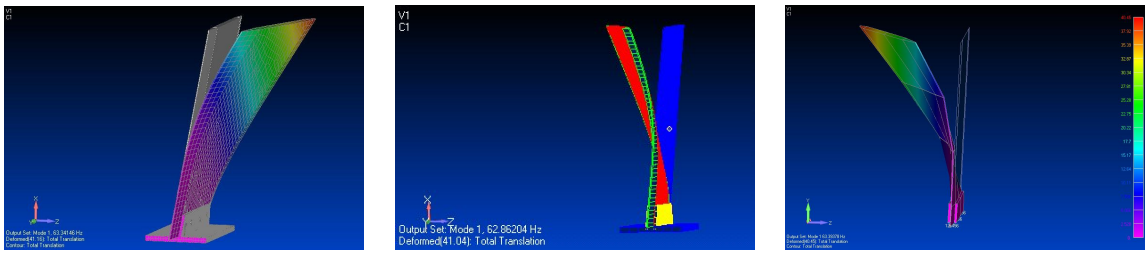
*3.4.1.6 Correlation of Finite Element Models.* A NASTRAN Normal Modes/Eigenvalues was run on each of the three finite element models to calculate the natural frequencies and mode shapes of the FalconLAUNCH V fin in order to compare them to each other and experimental results in order to verify their responses. The following figures show the deformed geometry for the first four modes of the finite element models.

All four modes show a high degree of correlation in the shape of the modes among the three models. These mode shapes also agree with predictions for cantilevered plates. [29]

The first four natural frequencies for each of the finite element models were calculated, too. The experimental and FEM results, after tuning the number and placement of the rigid links in the plate model, are presented in table 2. As was seen in the mode shapes for the first four modes, there is a high degree of correlation between the solid and plate models in the first four natural frequencies. The solid mesh and plate mesh models also compare very favorably with the results obtained in the lab (within 2%). The reduced plate mesh compares favorably with the first two modes (within 4%), though not nearly as favorably with the last two modes (up to 10% difference).

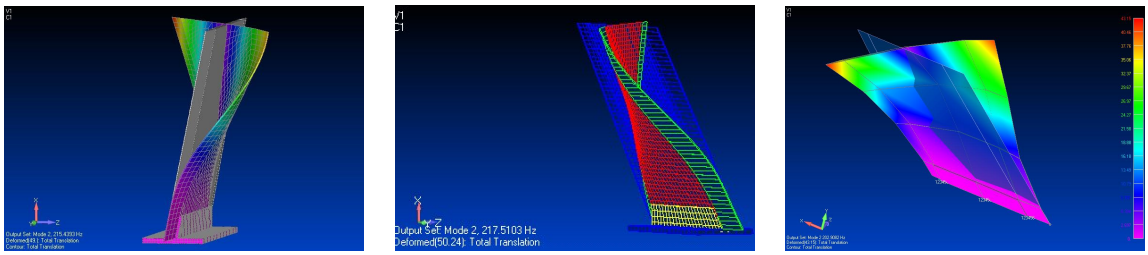
Table 2 First four natural frequencies from the FE models.

| Mode | Exp Results, Hz | Solid Model, Hz | Plate Model, Hz | Reduced Plate Model, Hz |
|---|---|---|---|---|
| 1 | 62 | 63.3 | 62.9 | 63.4 |
| 2 | 212 | 215.4 | 217.5 | 202.9 |
| 3 | 417 | 410.8 | 405.7 | 375.6 |
| 4 | 580 | 587.9 | 585.9 | 529.8 |



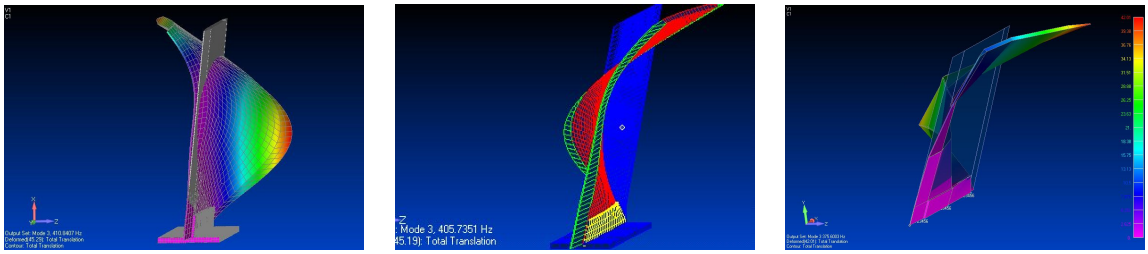

(a) Solid Mesh (b) Plate Mesh (c) Reduced Plate Mesh

Figure 24 Mode 1 deformed geometry



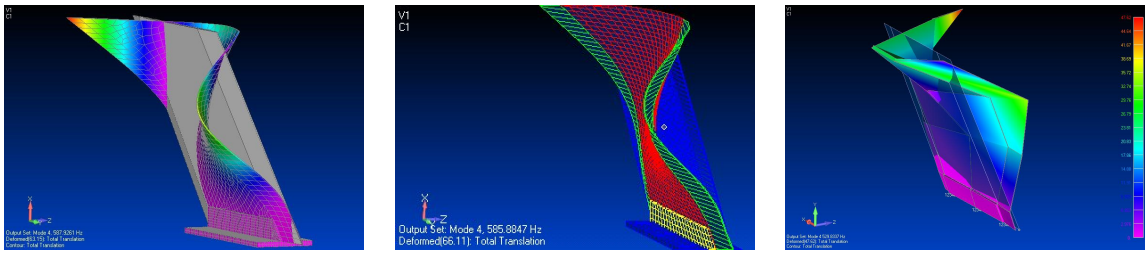

(a) Solid Mesh (b) Plate Mesh (c) Reduced Plate Mesh

Figure 25 Mode 2 deformed geometry




(a) Solid Mesh (b) Plate Mesh (c) Reduced Plate Mesh

Figure 26 Mode 3 deformed geometry




(a) Solid Mesh (b) Plate Mesh (c) Plate Mesh

Figure 27 Mode 4 deformed geometry

The mass of the actual hardware used in testing was reported in the conference paper by Simmons and Deleon [32]. FEMAP used the property and material cards to calculate the mass of the model which was then converted to lbf. The results are presented below in Table 3. The discrepancy between the measured value and the finite element models is due to the exclusion of the bolts in the FEM model which were found to weigh 0.12 lbf. This leaves less than 0.05 lbf difference between the measured values and those of the FEMs.

Table 3 Actual and Calculated Mass.

| Measurement | Mass | % Error |
|---|---|---|
| Lab, lbf | 2.85 | |
| Solid Model, lbf | 2.696 | 5.4% |
| Plate Model, lbf | 2.718 | 4.6% |
| Reduced Plate Model*, lbf | 2.699 | 5.3% |

* The calculated mass of the fin from the Reduced Plate Model does not include the mass of the base plate. The reported mass has been corrected by adding the computed mass of the base plate.

#### *3.4.1.7 Testing Parameterized Behavior of the Reduced Complexity Plate model.*

A pair of trade studies were run on both plate models to verify the utility of the reduced complexity model as a design tool. The first plate model was modified by editing the property cards in the FEM preprocessor (note, the brackets were left in their original position throughout the first trade study), while the second was modified by using a ModelCenter Parametric Study on the ModelCenter model described in Section 4.3.2 . The first trade study varied the fin thickness from the FalconLAUNCH V design of 0.25 in to the FalconLAUNCH VI design of 0.40 in. As expected, Figure 28 shows that as the thickness increases, the first and second natural frequencies separate. This is noteworthy, since a greater separation of the first and second natural frequencies tends to increase the flutter speed and in turn helps to prevent the onset of flutter.

The second trade study varied the fin's span by removing one row of elements from the tip of the fin for each run of the trade study. This technique was applied to the 0.40 in thick fin from the FalconLAUNCH V span of 9.5 in to 8.0 in, the span of the FalconLAUNCH VI fin. As was seen in the previous trade study, there is a trend toward

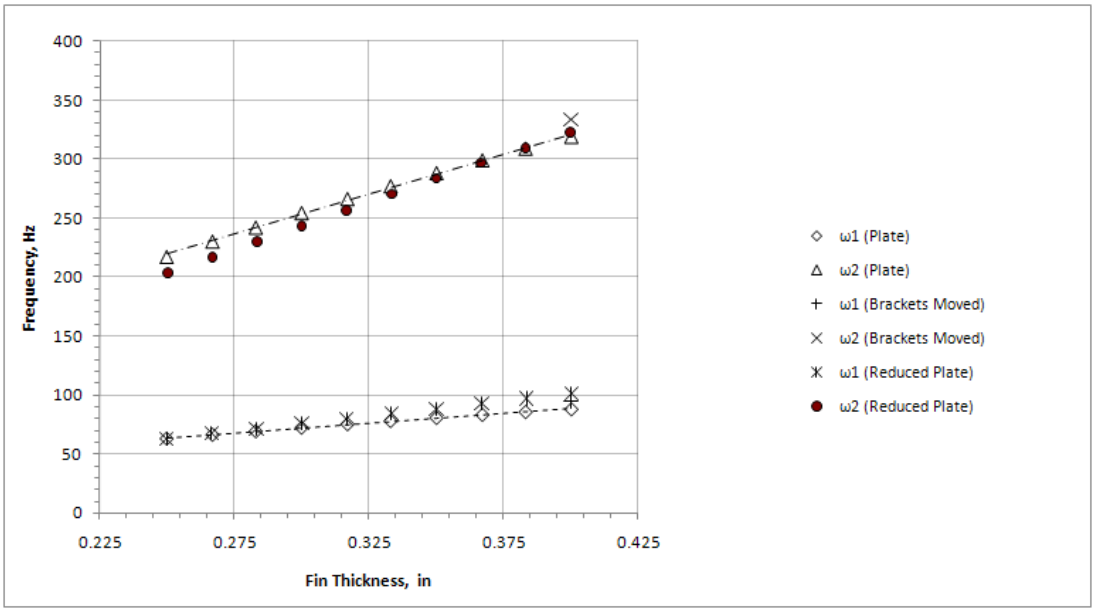



Figure 28 Frequency vs Fin Thickness (span of 9.5 in)

separation, this time as the fin is shortened. This separation was less pronounced than in the case of making the fin thicker as can be seen in Figure 29. This is likely due to the far greater percentage change in the values of the design variable in the first study than the in the second. Finally, both graphs show a strong correlation between the two plate models, indicating the reduced complexity model is suitable for use as the input to the flutter models.

In summary, the reduced complexity model is simple enough to be easily parameterized, has similar mode shapes and natural frequencies to the more complex models and experimental results, responds similarly to the higher fidelity plate model when changes in geometry are applied, and closely matches the mass of the laboratory model. This is the appropriate model for use in the Flutter Factor of Safety Validation Tool.

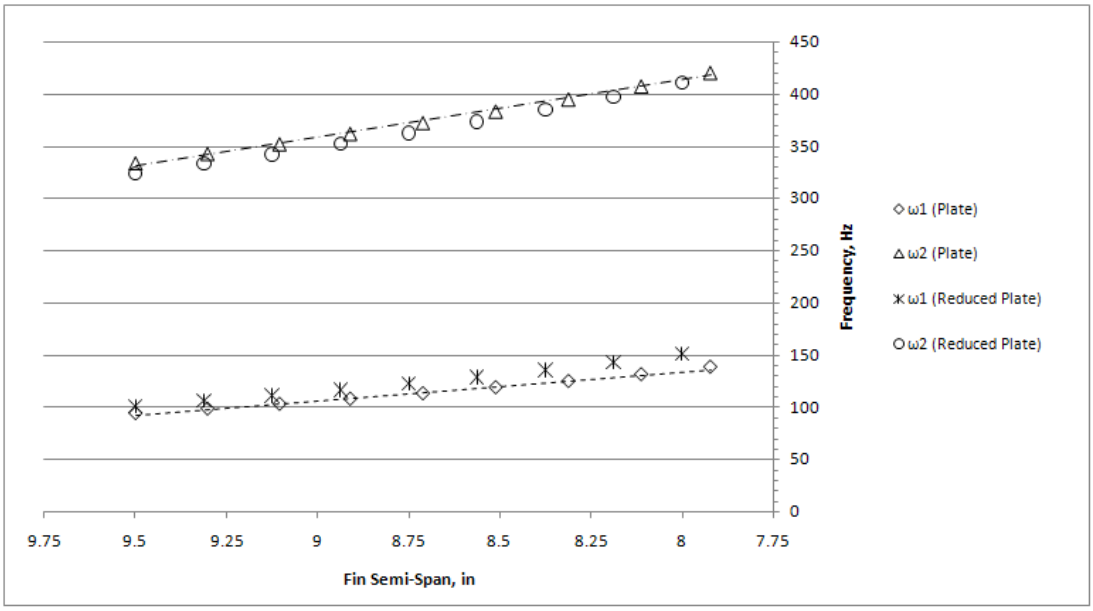



Figure 29 Frequency vs Fin Span (thickness of 0.40 in)

*3.4.2 Version 1: ZAERO Flutter Model.* The first version of the Flutter Factor of Safety Validation Tool was intended to use ZAERO for flutter calculations. When the first test aerodynamic models were created, several problems with using ZAERO in the intended environment were encountered. These included sensitivity in the aerodynamic model when there were changes the structural model, including geometry, and difficulties setting the problem up when the only element of the model was the fin. These difficulties led to the use of ZAERO being dropped from this research. See Appendix A for more details

*3.4.3 Version 2: Equation (10) Flutter Model.* The second version of the Flutter Factor of Safety Validation Tool uses Equation (10) to solve for the flutter velocity. This equation can only be used to solve for the flutter velocity during supersonic flight. Therefore, using just this equation in the Flutter Factor of Safety Validation Tool assumes that flutter will not occur during the subsonic and transsonic portions of the flight. Reviewing the flight profile of FalconLAUNCH V, seen in Figure 30, supports this assumption. Note how the fins' failure did not occur until a velocity of nearly Mach 4, well above the subsonic and transsonic regimes.

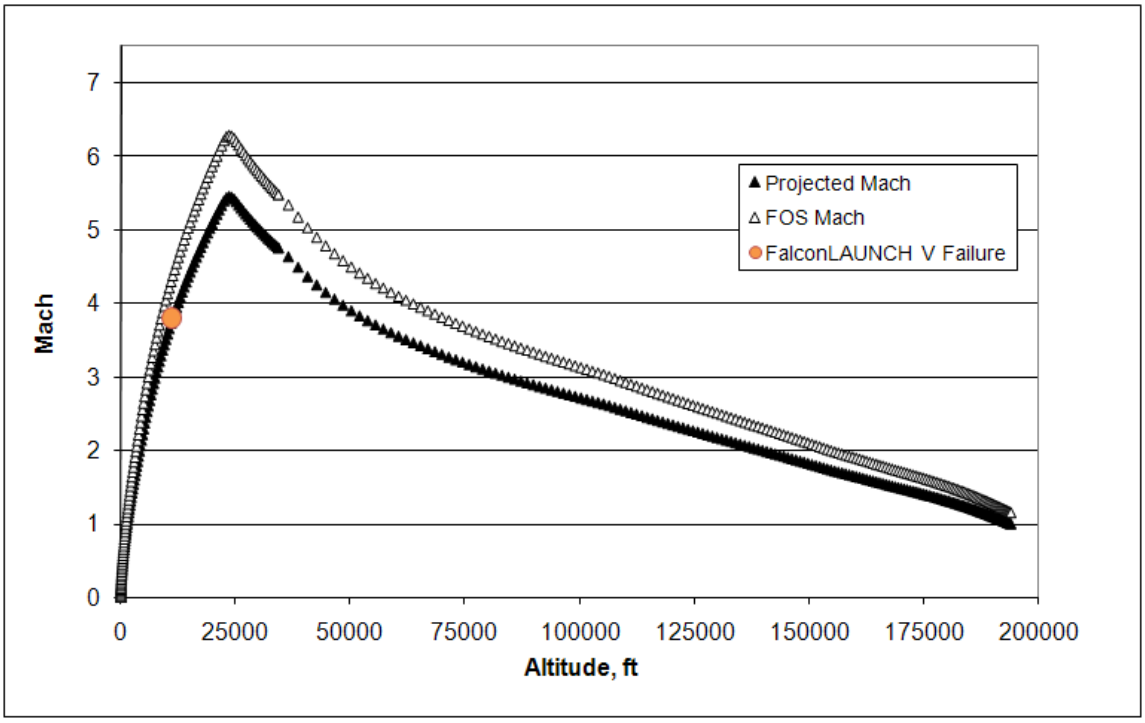



Figure 30 FalconLAUNCH V Flight Profile, flight failure occurred at orange point

This second version of the Flutter Factor of Safety Validation Tool is built using ModelCenter and is made of twelve components. Generally speaking, these components convert the design parameters used in the optimization model to values that can be used in a finite element model, used to obtain the natural bending and torsion frequencies, and in the Johns Hopkins flutter calculation. This section will identify and discuss the function of each of the twelve components in the model, seen in Figure 31, starting from the upper left and working toward the lower right.

Component 1: Fin Parameters

The *FinParameters* component is a ModelCenter Assembly component used to group the top level design parameters for the fin. As such, it does not provide calculated results, and just has the following variables:

1. Root Chord, in inches
2. Tip Chord, in inches
3. Sweep Angle, in degrees

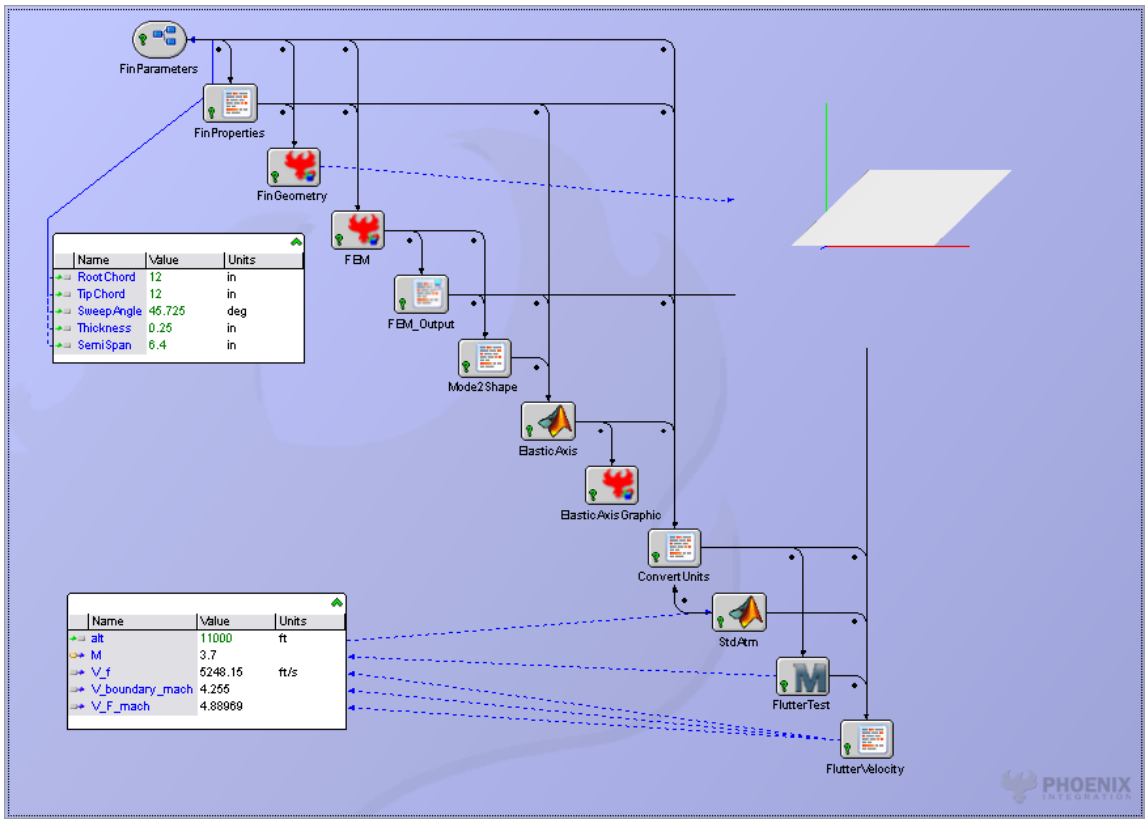



Figure 31 Flutter Factor of Safety Validation Tool, Version 2

4. Thickness (of the fin body), in inches
5. Semi Span (span of just one fin), in inches

Component 2: Fin Properties

The *FinProperties* component takes the Fin Parameters, and creates the values needed by the finite element analysis and the fin geometry components (see Appendix B for more details). This component has the same inputs as the variables in the *FinParameters* component. The outputs include:

1. Span (of the combined fins) $= 2 * SemiSpan$, in inches
2. Average Chord $= (RootChord + TipChord)/2$, in inches
3. Aspect Ratio $= Span/AverageChord$, unitless
4. Taper Ratio $= TipChord/RootChord$, unitless

5. RTC (Ratio of Thickness to Chord) = $Thickness/AverageChord$, unitless
6. Panel Aspect Ratio (of the fin) = $SemiSpan/AverageChord$, unitless
7. Panel Area = $SemiSpan * AverageChord$, in inches$^2$
8. Approximate Volume (of the fin) = $PanelArea * Thickness$, in inches$^3$
9. X Coordinates (of grid points in FEM)
10. Y Coordinates (of grid points in FEM)

Component 3: Fin Geometry

The *FinGeometry* component uses a Phoenix Integration distributed code to render the shape of the fin based on values from the Fin Parameters and Fin Properties components. The rendered view is displayed in the model and updated real time (Figure 32), allowing the analyst to verify the input parameters match the design that is being tested.

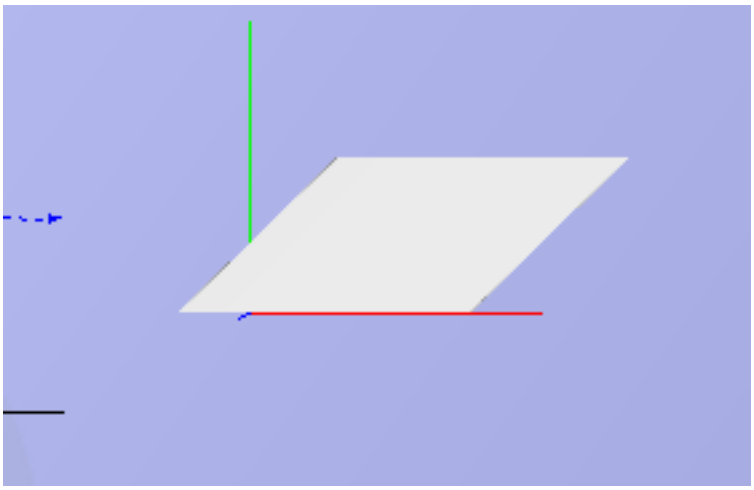


Figure 32 Embedded view of Fin Geometry Component

Component 4: Finite Element Model (FEM)

The *FEM* component is a ModelCenter VB Script Component that takes the calculated X and Y coordinates from the Fin Properties component and uses FEMAPs COM API to relocate the grid points in the FalconLAUNCH fin plate finite element model, and to adjust the plate thicknesses. It then runs a normal modes analysis, and exports the results file which contains the model frequencies and moments of inertia and the punch file which contains the mode shape coordinates for use by other components in the model.

To alter the grid locations, the FEM component first uses a lookup array to find the Node ID for the current position, and then gets a Component Object Model (COM) reference to that node from FEMAP (feNode::Get). It then sets the nodes x and y values to the corresponding location, and finally pushes the update to the finite element model (feNode::Put). This process is done for each of the 25 grid points in the plate model.

The plate thicknesses are updated similarly. The script first gets a COM reference to the properties cards (feProp::Get). It then sets the entry corresponding to the thickness (pval(0)) as specified in ModelCenter (note: bracket plates are set to twice the fin thickness). The script then pushes the updates to the finite element model (feProp::Put).

Finally the script runs the Analysis (feAnalysisSet::Analyze with an argument of 1 to mean run the first analysis in the model), and sets up a busy loop to wait on the analysis to complete (the Analyze call is non-blocking, and configuring call back functions in an ModelCenter script component was beyond the scope of this work). Once the wait is complete, the script reads the primary analysis results file (model.f06) which contains the natural frequencies, the mass, and the moments of inertia, and the punch file (model.pch) which contains the mode shape data from the disk into the components output variables for later use.

Component 5: FEM Output The *FEM_Output* component is an AnalysisServer QuickWrap component that reads the primary analysis results file (model.f06) and the punch file (model.pch) and parses out the following variables for use by other components:

1. Mode 1 Natural Frequency, in Hz and rad/s
2. Mode 2 Natural Frequency, in Hz and rad/s
3. Mode 3 Natural Frequency, in Hz
4. Mode 4 Natural Frequency, in Hz
5. X and Y locations of the CG, in inches (CG Z location is 0 since the fin is symmetric about the Z axis)
6. Iyy, in lb*$s^2$*in
7. Mass, in lb*$s^2$/in

8. Data about the Mode 2 mode shape in a series of arrays

   (a) Grid number

   (b) X coordinate

   (c) Y coordinate

   (d) Z coordinate

Component 6: Mode 2 Shape

The *Mode2Shape* component is a ModelCenter VB Script Component that iterates through the list of grid points and coordinates of the Mode 2 mode shape from the FEM Output component and reorders the data into the order of the grid points in the Validation Model.

Component 7: Elastic Axis

The *ElasticAxis* component is a Matlab Plugin that calls a Matlab script which calculates the location of the elastic axis of the fin, and then uses this to calculate the distance from the elastic axis to the center of gravity of the fin and the radius of gyration of the fin about the elastic axis as required by Equation (10). To locate the elastic axis, the component uses the grid point locations of the undeformed fin and of the mode shape of mode 2. From these two sets of grid points, the script iterates through the 3 lines of points that are not part of the bracket and solves for the intersection of the undeformed and second mode shapes lines. The script then solves for the equation of the line running through these three points to get the location of the elastic axis. The script then solves for the distance from the point of the center of gravity to the line of the elastic axis using Equation (13). It also solves for the radius of gyration Equation (14). Finally, this component renders the undeformed geometry, the second mode shape, elastic axis, and the center of gravity to a jpeg image file for review (see Figure 33).

$$\bar{x} = \frac{|a * cgX + b * cgY + c|}{\sqrt{a^2 + b^2}} \tag{13}$$

[11]

$\bar{x}$, Distance from the Center of Gravity to the Elastic Axis or Axis of Rotation

$a$, coefficient of x for standard form of the equation of a line

$b$, coefficient of y for standard form of the equation of a line

$c$, constant for standard form of the equation of a line

$cgX$, x coordinate of the CG

$cgY$,y coordinate of the CG

$$k = \sqrt{\frac{I_{yy}}{m} + \bar{x}^2} \tag{14}$$

[25]

$k$, Radius of Gyration relative to the Elastic Axis

$I_{yy}$, Principle Moment of Inertia about the y-axis

$m$, Mass

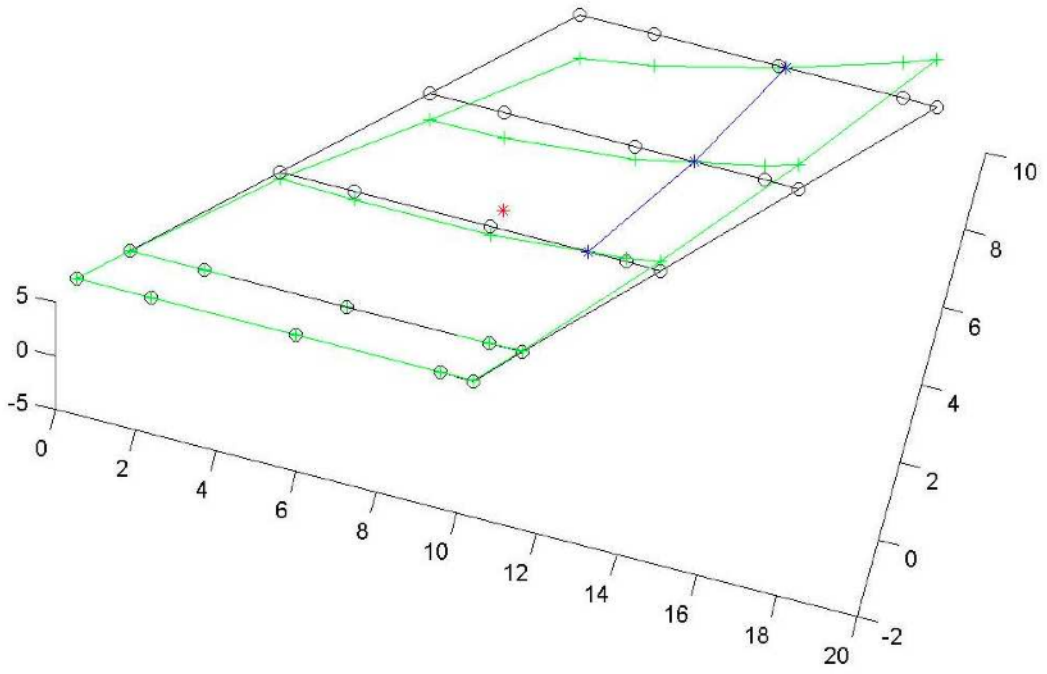



Figure 33 Visual representation of the Elastic Axis for the FalconLAUNCH V Fin

Component 8: Elastic Axis Graphic

This component simply waits for the Elastic Axis component to run, and then imports the image file from the Elastic Axis component into ModelCenter for review.

Component 9: Convert Units

The *ConvertUnits* component is a ModelCenter VB Script Component that transforms the units of various inputs from the design units of inches into the analysis units of feet used in the Flutter Test component. The inputs are:

1. Mass, in (lb*s$^2$/in)
2. Radius of gyration about the elastic axis, in inches
3. Chord, in inches
4. Mode 1 Natural Frequency, in rad/s
5. Mode 2 Natural Frequency, in rad/s
6. Distance from center of gravity to elastic axis, in inches
7. Air Density at target altitude, in lbm/ft$^3$

The outputs are:

1. Mass ratio parameter ($MassperUnitSpan/(4*Density*Semi-Chord^2$, where Mass per Unit Span is in lb*s$^2$/in$^2$ and Density is in lb/in$^3$)
2. Radius of gyration about the elastic axis, in feet
3. Semi-chord, in feet
4. Ratio of Mode 1 to Mode 2 natural frequencies, unitless
5. Distance from center of gravity to elastic axis, in feet

Component 10: Standard Atmosphere

The *StdAtm* component is a Matlab Plugin that calls a Matlab script that calculates the 1976 US Standard Atmosphere written by Richard Rieber [30]. This code provides the density used in the Convert Units component. It was tested against another copy of the 1976 US Standard Atmosphere [12].

Component 11: Flutter Test

The *FlutterTest* component is a MathCad Plugin that calls a MathCad worksheet that implements Equation (10), which generates a flutter parameter $V_f/(b\omega_\alpha)$. The goal is to use

this component to generate the flutter parameter, in order to calculate the flutter velocity from the definition of the parameter and compare the results for the FalconLAUNCH V and VI geometries and the three optimized results to the actual flight profiles.

Component 12: Flutter Velocity

The *FlutterVelocity* component is a ModelCenter VB Script Component that transforms the flutter parameter from the Flutter Test component into the flutter Mach number. This value can then be graphed versus altitude to compare it to the predicted flight Mach number to check for the required 15% factor of safety.

### *3.5 Summary*

This chapter has shown the development and details of the Fin Geometry Optimization and Flutter Factor of Safety Validation Tools. Two versions of the Fin Geometry Optimization Tool were developed, using two variations of the flutter equation, Equations (9) and (7), from NACA TN 4197. The Flutter Factor of Safety Validation Tool developed in this chapter uses Equation (10) from the Johns Hopkins University research in 1962. The use of these tools will be tested on the FalconLAUNCH V and FalconLAUNCH VI fins to determine the utility and applicability of these tools in optimizing sounding rocket fins.

# 4. Analysis and Results

## 4.1 Chapter Overview

This chapter covers the testing and use of the design tool developed in Chapter 3. This includes the Fin Geometry Optimization and Flutter Factor of Safety Validation Tools. The chapter begins by using the two versions of the Fin Geometry Optimization Tool to optimize the FalconLAUNCH VI fin geometry. It then tests and uses the Flutter Factor of Safety Validation Tool to validate the optimized fin geometries for the required 15% factor of safety beyond the rocket's predicted velocity throughout its flight profile. The chapter concludes by validating the optimized fin geometries for the required Center of Pressure throughout rocket's flight profile.

## 4.2 Optimizing the FalconLAUNCH VI Fin Geometry

This section demonstrates the use of both versions of the Fin Geometry Optimization Tool. Before running any optimization studies, a Design of Experiments is developed and run to determine which design parameters have strong influence on the flutter velocity and which do not. The section then runs an optimization study each of the Fin Geometry Optimization Tool versions.

*4.2.1 Optimization Problem Statement.* Recall that the goal of the Fin Geometry Optimization Tool is to find a fin design that minimizes the mass while preventing flutter and ensuring the rocket maintains stable flight. Since all of the candidate designs are for solid plate fins and mass is proportional to volume in homogenous solids, the goal can be restated as finding a design that minimizes the volume. Therefore, the optimization problem can be defined as:
Minimize

$$vol = c_{average} * t * b \tag{15}$$

$vol$, Fin Volume

$c_{average}$, Average Chord

$t$, Fin Thickness

$b$, Fin Span

subject to the following constraints:

Minimum planform area

$$S \geq S_{min} \tag{16}$$

$S$, Planform Area

$S_{min}$, User specified minimum Planform Area

Standard fin shape

$$\lambda \leq 1 \tag{17}$$

$\lambda$, Taper Ratio

Version 1 flutter constraint

$$V_f \geq V_{f-min} \tag{18}$$

$V_f$, Flutter Velocity

$V_{f-min}$, User specified minimum Flutter Velocity

Version 2 flutter constraint

$$F \leq F_{min} \tag{19}$$

$F$, Total Flutter Factor, as defined in Equation (9)

$F_{min}$, User specified minimum Total Flutter Factor

The design variables are notionally the fin design parameters, root chord, tip chord, sweep angle (no effect on volume, but may affect flutter constraints), semi-span, and the thickness. The following section investigates the influence of these parameters on the flutter velocity to refine the list of design variables and gain insight into their impact on the design.

*4.2.2 Investigating Influence of Design Parameters on Flutter Velocity.* Before setting up an optimization, it is often helpful to identify the key design variables. This allows the modeler to focus the optimization on only those variables that have a significant impact on the design, speeding up the optimization study and preventing unnecessary and

Table 4 Design of Experiments

| Parameter | FalconLAUNCH V | FalconLAUNCH VI | DOE Low | DOE High |
|---|---|---|---|---|
| Root Chord, in | 9.9 | 10.0 | 6.0 | 12.0 |
| Tip Chord, in | 8.9 | 9.2 | 6.0 | 12.0 |
| Sweep Angle, deg | 44.25 | 44.25 | 0 | 60 |
| Semi-span, in | 9.5 | 8.0 | 4.5 | 15.0 |
| Thickness, in | 0.25 | 0.4 | 0.2 | 0.5 |

inconsequential changes from being proposed by the optimizer. Based on the parameters chosen by Jerry Allen in his study of rocket and missile fins [14] and modified to include additional parameters needed to fully describe the FalconLAUNCH fin, the design parameters included in the Fin Geometry Optimization Tool are the root chord, the tip chord, the sweep angle, the semi-span and the thickness, shown in Figure 3. These parameters were used in a ModelCenter Design of Experiments trade study to look at their impacts on the flutter velocity. The inputs to the Design of Experiments can be see in Table 4.

After completing the Design of Experiments, Phoenix Integration's Variable Influence Profiler was used to generate the Main Effects plot see Figure 34. A main effects plot is used to display the influence of design variables on one of a model's response variables, in this case the flutter velocity. The longer the bar, the more significant the influence of the variable. The plot in Figure 34 shows the two key variables in determining the flutter velocity are the semi-span and then thickness. The plot also shows a coupling between the effects of the thickness and the semi-span (Thickness:SemiSpan). Of particular interest in minimizing the weight of the FalconLAUNCH VI fin is that the semi-span has a much greater impact on the flutter velocity than the thickness as indicated by its having nearly twice the length in the Main Effects plot. Since the changes made by the USAFA cadets to prevent flutter from FalconLAUNCH V to VI are dominated by the change in thickness, there should still be some room to reduce the weight without encountering flutter. It is worth noting that Martin's equations, (7) and (9), do not include sweep angle. This explains why the sweep angle is shown to have no effect at all on flutter velocity, though it does not indicate if there should be an effect or not. Similarly, no apparent effect from delta versus normal fins would be expected using Martin's equations since delta fins adjust the sweep angle so the trailing edge perpendicular to the rocket body.

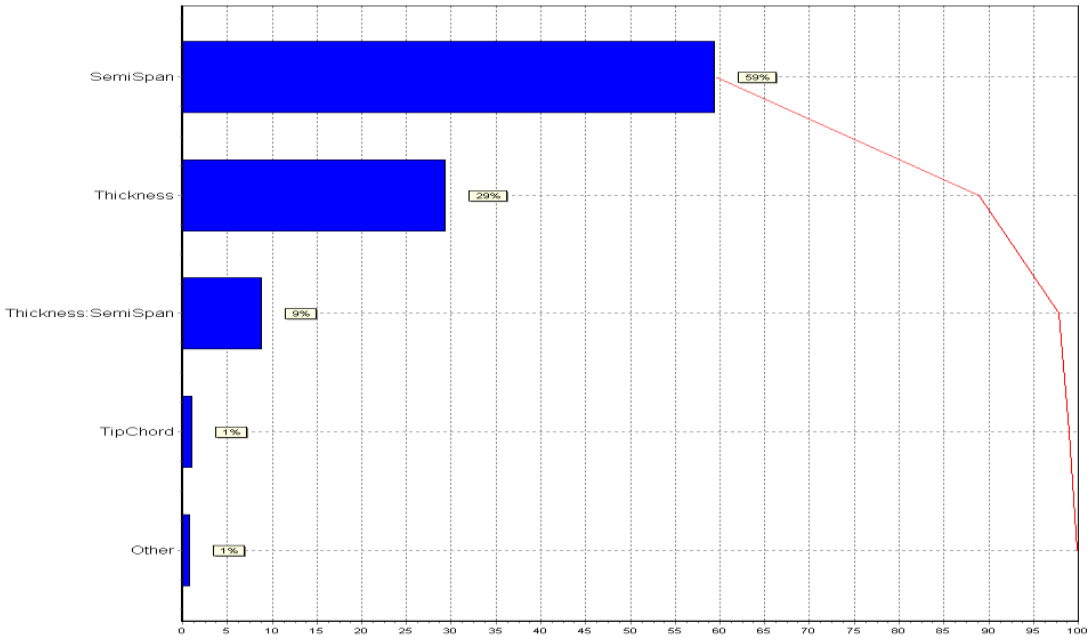



Figure 34 Main Effects Plot

*4.2.3 Optimization Study using Fin Geometry Optimization Tool Version 1.*

A ModelCenter optimization was then prepared using the DesignExplorer tool. Per the optimization problem statement the objective was set to minimize the fin's volume. The values for the constraints were based upon the FalconLAUNCH VI design. The minimum flutter velocity was set to 5568 mph at 11000 ft, the altitude at which the FalconLAUNCH V fins fluttered, with the intent of capturing the degree to which FalconLAUNCH VI design exceeded that of FalconLAUNCH V. The minimum panel area was set to 76.8 $in^2$ to ensure the optimized geometry would provide the same Center of Pressure as FalconLAUNCH VI. The last constraint, the taper ratio, was set not to exceed 1 to ensure the tip chord did not exceed the root chord. Finally the design variables were set to start from the initial design point, FalconLAUNCH VI, and given the same ranges as in the Design of Experiments in Section 4.2.2. See Table 5 for a complete listing of the design variable configuration.

After 66 iterations, DesignExplorer completed its optimization study and found a geometry that was over 15% lighter than the FalconLAUNCH VI design. Table 6 summarizes this design in comparison to both FalconLAUNCH V and VI. The difference in the

Table 5 Design Variable Configuration for Optimization

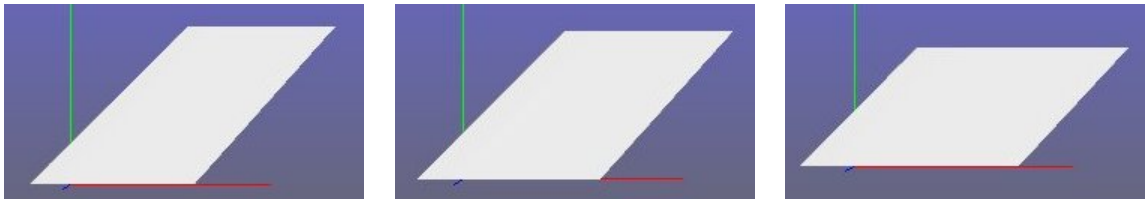
| Parameter | Start Value | Lower Bound | Upper Bound |
|---|---|---|---|
| Flutter Velocity, mph | - | 5568 | - |
| Panel Area, $in^2$ | - | 76.8 | - |
| Taper Ratio | - | - | 1 |
| Root Chord, in | 10.0 | 6.0 | 12.0 |
| Tip Chord, in | 9.2 | 6.0 | 12.0 |
| Semi-span, in | 8.0 | 4.5 | 12.0 |
| Thickness, in | 0.4 | 0.2 | 0.5 |

Goal: Minimize Volume

Table 6 Comparison of Designs

| Parameter | FalconLAUNCH V | FalconLAUNCH VI | Optimization #1 |
|---|---|---|---|
| Root Chord, in | 9.9 | 10.0 | 12.0 |
| Tip Chord, in | 8.9 | 9.2 | 11.6 |
| Semi-span, in | 9.5 | 8.0 | 6.5 |
| Thickness, in | 0.25 | 0.4 | 0.34 |
| **Volume, $in^3$** | **22.3** | **30.7** | **26.1** |
| **Flutter Velocity at 11,000 ft, mph** | **2203** | **5567** | **5585** |

optimized design is striking and can be clearly seen in Figure 35. Of particular note is the dramatically shorter semi-span, which reflects the dominance of the semi-span in the flutter calculation. This response is also predicted by the trade studies run in Section 3.4.1.7. In Figure 29, the separation between the first bending and first torsional frequencies increases as the span is reduced, which delays the point at which the two frequencies converge and cause the onset of flutter as discussed in Section 2.3 [26].



(a) FalconLAUNCH V (b) FalconLAUNCH VI (c) Optimization #1

Figure 35 FalconLaunch VI and V Fin Geometries

*4.2.4 Optimization Study using Fin Geometry Optimization Tool Version 2.* The only difference between the first and second versions of the Fin Geometry Optimization Tool is the flutter boundary condition. Version 1 uses a calculation of the flutter velocity at

Table 7 Design Variable Configuration for Optimization

| Parameter | Start Value | Lower Bound | Upper Bound |
|---|---|---|---|
| Total Flutter Factor, psi | - | - | $5x10^5$ |
| Panel Area, $in^2$ | - | 76.8 | - |
| Taper Ratio | - | - | 1 |
| Root Chord, in | 10.0 | 6.0 | 12.0 |
| Tip Chord, in | 9.2 | 6.0 | 12.0 |
| Semi-span, in | 8.0 | 4.5 | 12.0 |
| Thickness, in | 0.4 | 0.2 | 0.5 |

Goal: Minimize Volume

a specified altitude. This version uses the total flutter factor from Equation (9). The user then has to specify an upper bound for this parameter by reading a value from Figure 7 based on the shear modulus of the material of the fins and how conservative the design is intended to be (values that have more empirical reference points near them represent more conservative choices for the total flutter value). With this in mind, the design variables and most of the constraints used in this version of the Fin Geometry Optimization Tool are the same as those in Version 1.

A ModelCenter optimization was again prepared using the DesignExplorer tool. The objective was set to minimize the fin's volume, the total flutter factor was chosen from Figure 7 to be at the top of the range of successful designs for aluminum fins ($5x10^5$ psi) in an attempt to find an even lighter design than before. The remainder of the optimization settings remained the same as in the first optimization. See Table 7 for a complete listing of the design variable configuration.

After 87 iterations, DesignExplorer completed its optimization study and found a geometry that was over 47% lighter than the FalconLAUNCH VI design. Table 8 summarizes this design in comparison to both FalconLAUNCH VI and the first optimization study's results. The similarity in the optimized designs is striking and can be clearly seen in Figure 36. The key difference between the two optimized geometries is the fin thickness, with this second design being 38% thinner and in turn lighter, resulting in a correspondingly lower flutter velocity.

This difference in results comes from the different flutter constraints. In particular, the lower flutter velocity for Optimization #2 clearly indicates that its flutter constraint

Table 8 Comparison of Designs

| Parameter | FalconLAUNCH VI | Optimization #1 | Optimization #2 |
|---|---|---|---|
| Root Chord, in | 10.0 | 12.0 | 12.0 |
| Tip Chord, in | 9.2 | 11.6 | 12.0 |
| Semi-span, in | 8.0 | 6.5 | 6.4 |
| Thickness, in | 0.4 | 0.34 | 0.21 |
| **Volume, in$^3$** | **30.7** | **26.1** | **16.1** |
| **Flutter Velocity at 11,000 ft, mph** | **5567** | **5585** | **2768** |

is more aggressive, resulting in a less conservative solution than Optimization #1. What is not clear from these results is if either of the designs meets the minimum 15% factor of safety for flutter, and by how much over or under the designs are if at all. Section 4.3 will use the Flutter Factor of Safety Validation Tool to determine the answer to just this question.

(a) FalconLAUNCH VI (b) Optimization #1 (c) Optimization #2

Figure 36 Optimized FalconLAUNCH Fin Geometries

### *4.3 Validating Optimized Fin Designs*

The Flutter Factor of Safety Validation Tool is designed to discriminate between potential designs. After generating a candidate design with the Fin Geometry Optimization Tool, the Flutter Factor of Safety Validation Tool generates a plot of Mach vs altitude for three critical curves. The first is the predicted flight profile, which is one of the inputs to the tool, and generated by a separate analysis. The second is the the flutter factor of safety, which is the predicted flight profile plus 15%. The last is the flutter velocity, as predicted by Equation (10). If this velocity crosses under the predicted flight profile curve, then the likelihood of encountering flutter is high. If the predicted flutter velocity stays above the predicted flight profile but crosses under the flutter factor of safety curve,

then the design is marginal, and needs only minor adjustments to find a valid design. If the flutter velocity stays above the factor of safety curve, then the design is valid. One important point, if the flutter velocity exceeds the factor of safety by a large margin, then the design is excessive and there is still unnecessary margin that can be safely eliminated from the design. By using this information, a user can make appropriate changes to the flutter constraint value in the Fin Geometry Optimization Tool on subsequent iterations of the design optimization process.

This section uses the Flutter Factor of Safety Validation Tool and Aerolab to guide the two optimized designs. Before applying the Flutter Factor of Safety Validation Tool, this section first validates the Flutter Factor of Safety Validation Tool by using it to compute the flutter velocity for FalconLAUNCH V and FalconLAUNCH VI, two designs that have some data to compare the tool's predictions against. The section concludes with the validation tests on the optimized designs.

*4.3.1 Validation of the Flutter Factor of Safety Validation Tool.* Using predicted flight data provided by the USAFA, the Flutter Factor of Safety Validation Tool was validated by using it to generate the flutter velocity for the flight profiles of FalconLAUNCH V and FalconLAUNCH VI. ModelCenter trade studies were run to compute the flutter velocity versus the predicted altitudes and velocitie of the flight profiles. Since the underlying equation for flutter velocity used in the Flutter Factor of Safety Tool was only confirmed for velocities up to Mach 5, the following figures distinguish flutter predictions for Mach numbers above and below Mach 5 (those points above Mach 5 are listed as "extrapolated" and are limited to no more than 50% above this threshold or Mach 7.5). These results should be further validated during detailed design stages using higher fidelity flutter calculations. Figure 37 shows the flutter velocity for FalconLAUNCH V versus altitude and plotted with the predicted velocity and the required 15% factor of safety. This graph shows that the predicted velocity exceeds the flutter velocity very early in the flight and that a significant separation is established by 11,000 feet, the altitude at which FalconLAUNCH V experienced flutter, indicating that the FalconLAUNCH V design would have been likely to experience flutter by 11,000 feet or lower.

Figure 38 shows the same data for FalconLAUNCH VI. In contrast with Figure 37, this graph shows that the flutter velocity greatly exceeds the predicted velocity, which would be in keeping with the expectation for the design of FalconLAUNCH VI, which was intended to prevent fin flutter at the expense of added mass. This increase in the flutter velocity also matches the increased separation of the first and second natural frequencies that was seen in the trade studies run in Section 3.4.1.7 for the increase in thickness and decrease in span of the FalconLAUNCH VI fin over FalconLAUNCH V.

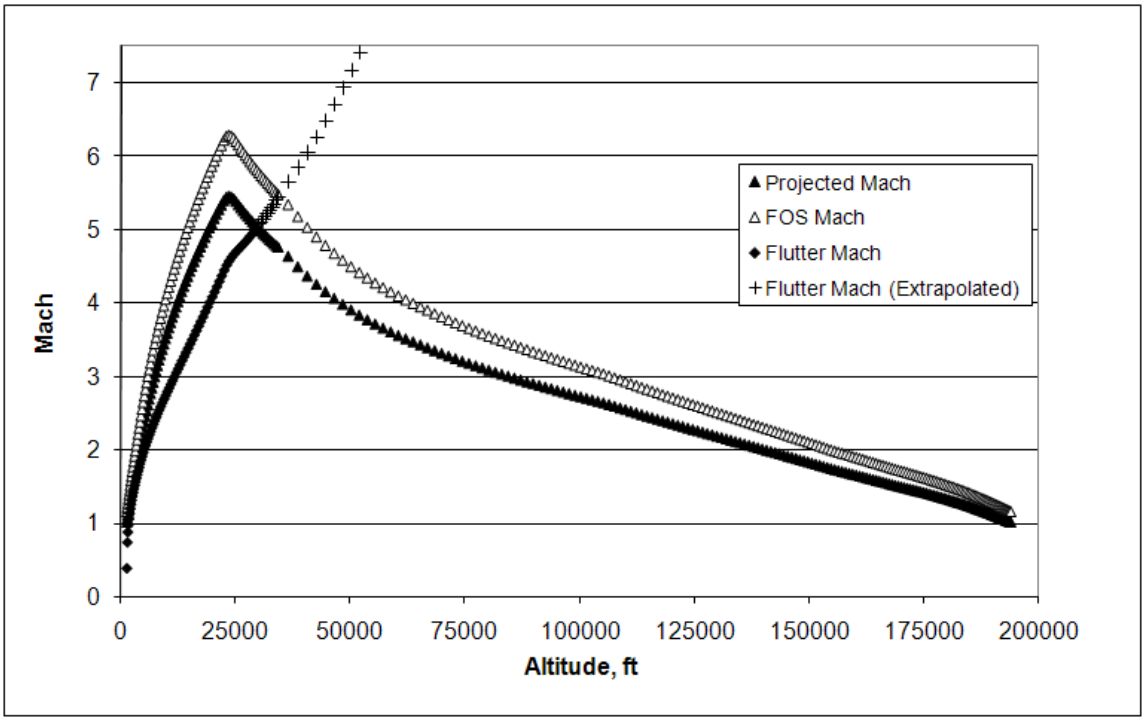



Figure 37 FalconLAUNCH V Flutter Prediction

These results demonstrate that the Flutter Factor of Safety Validation Tool is accurately predicting flutter behavior. The graph for FalconLAUNCH V clearly indicates that it should have failed, and that the failure should have occurred near the altitude that the vehicle lost three of its fins. Similarly, the graph for FalconLAUNCH VI clearly shows that not only is the design unlikely to experience flutter, but that the goal of ensuring that flutter will not occur can be seen by the large separation between the flutter velocity and the flutter factor of safety.

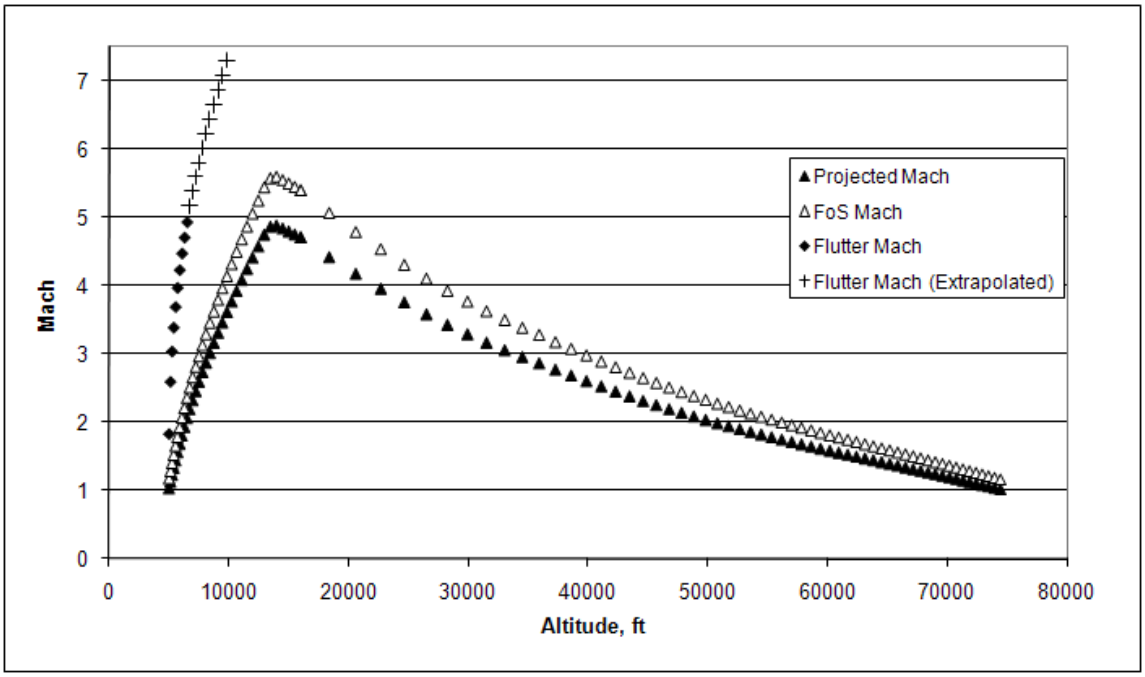



Figure 38 FalconLAUNCH VI Flutter Prediction

*4.3.2 Flutter Validation of Optimized Fin Designs.* With this testing complete, Flutter Factor of Safety Validation Tool can be used to test the results of the Fin Geometry Optimization Tool. Using the FalconLAUNCH VI flight profile, ModelCenter trade studies were run for the results of Optimization #1 and Optimization #2. Figure 39 shows the results for Optimization #1, which are very similar to the results for the FalconLAUNCH VI fin. This is not surprising since the flutter boundary conditions used in the optimization was set to match that of FalconLAUNCH VI, by setting the minimum flutter velocity at 11,000 ft to be the predicted flutter velocity of FalconLAUNCH VI at that altitude. Like FalconLAUNCH VI, this fin should not experience flutter, but it exceeds the factor of safety enough to indicate there is room for further refinement.

Figure 40 shows the results for Optimization #2. These results show that this optimization is slightly aggressive. The flutter velocity begins between the predicted velocity and the 15% factor of safety, before dropping just below the predicted velocity at its peak which happens at 15,000 ft. This inadequate margin between the predicted velocity and the flutter velocity makes this a marginal design.

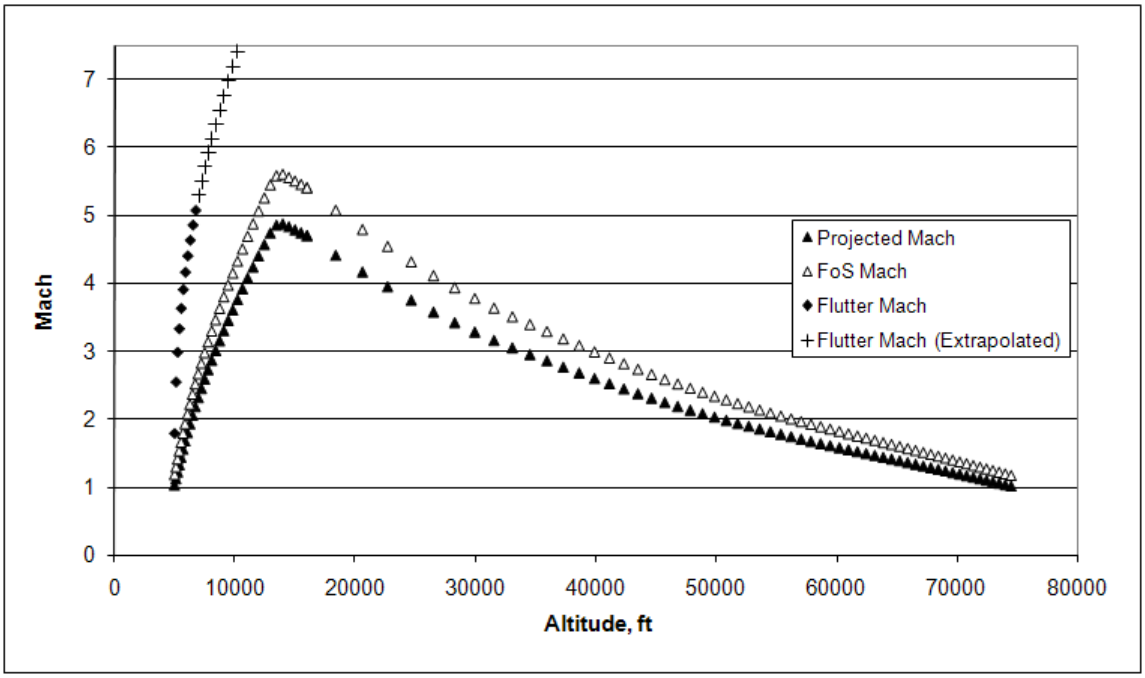



Figure 39 Optimization #1 Flutter Prediction

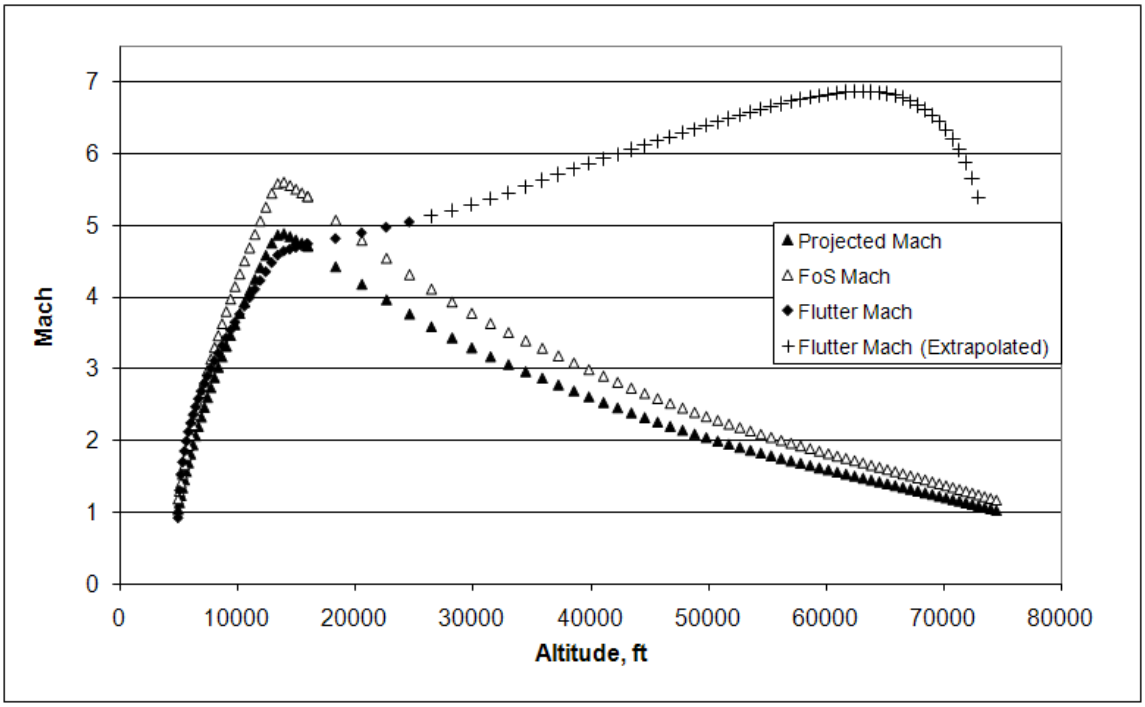



Figure 40 Optimization #2 Flutter Prediction

Since Optimization #2 is only just outside of an acceptable design, there may be a modification that can be made that will make the design an acceptable one. Recalling that this fin is only 0.21 inches thick, which makes it thinner than FalconLAUNCH V, the thickness was increased to 0.25 inches to match FalconLAUNCH V, and this new design was validated with the Flutter Factor of Safety Validation Tool. Figure 41 shows the results for this modification to Optimization #2. Note how the flutter velocity follows just outside of the factor of safety all the way until the peak velocity, offering just slightly over the minimum 15% required by NASA. This additional margin in flutter velocity cost less than 20% additional mass when compared to Optimization #2 and still saves over 37% mass when compared to FalconLAUNCH VI.

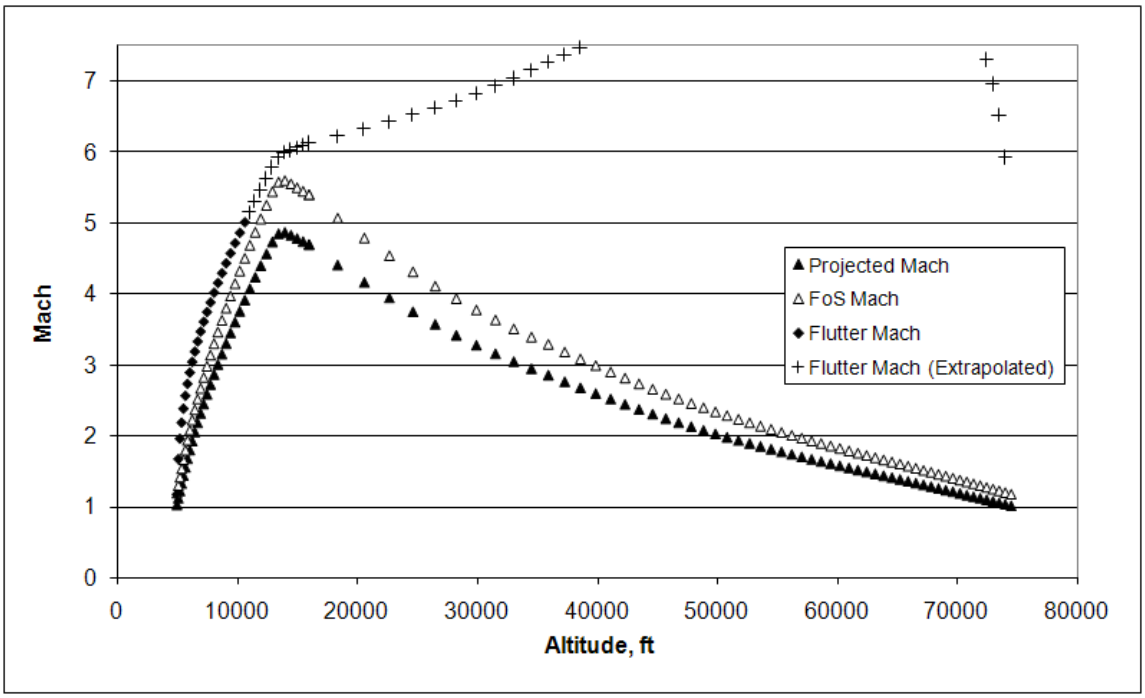



Figure 41 Optimization #2a Flutter Prediction

What is interesting about this design process is how much closer to a useful design Optimization #2 was than Optimization #1. By starting with a flutter constraint that is based on empirical results for a large number of vehicles instead of a flutter velocity selected from either a previous design or some other source, version 2 of the Fin Geometry Optimization Tool gave better results faster, which is a powerful combination in engineering

design. In this case, Optimization #2 was so close to being valid that another iteration through the Fin Geometry Optimization Tool was not necessary. Had there been more of a crossing by the flutter velocity under the curve of the predicted velocity, then the flutter criteria value could have been lowered some and another complete iteration could have been run generating a new candidate design and to evaluate.

*4.3.3 Stability Validation of Optimized Fin Designs.* Since the Fin Geometry Optimization Tool used the simplification that a minimum panel area would ensure the required stability, the optimized designs had be validated using an outside tool to determine if they would provide adequate stability. The tool selected was Aerolab. Aerolab was used to compare the location of Center of Pressure (measured in inches from the nose) of the original FalconLAUNCH VI fins and the fins from Optimization #1 and #2a over the predicted range of Mach numbers for the flight profile of FalconLAUNCH VI. Figure 42 shows the results of the Center of Pressure analysis from Aerolab. All three curves are closely matched. The largest difference is between the FalconLAUNCH VI design and the optimized designs from launch to just over Mach 1, and this is only a 3 inch difference forwards, which is less than the diameter of the rocket. The accepted rule of thumb in passively stabilized rocket design is to keep the Center of Pressure between one and two rocket diameters, or calibers, behind the Center of Gravity [34] regardless of the vehicle's length. This indicates that both the Optimization #1 and #2a designs should provide similar stability to the original FalconLAUNCH VI design. Note, this does not address the fact that the Center of Gravity would move forward with the lighter fins from the optimizations, which further supports the conclusion that the Center of Pressure will be located aft of the Center of Gravity, and that the new fins will provide similar stability to the original fins. Taken together, these results confirm the Fin Geometry Optimization Tool's stability simplification of requiring a minimum panel area.

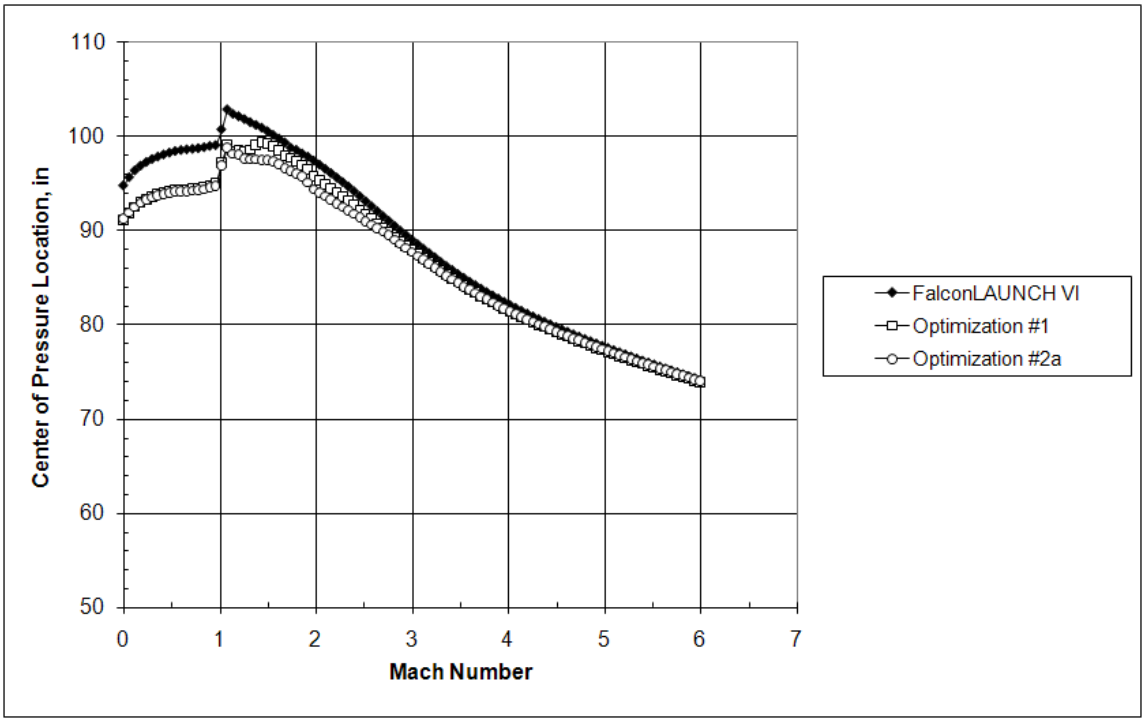



Figure 42 Center of Pressure (inches from nose) vs Mach Number

### *4.4 Summary*

This chapter has shown how the design tool developed in Chapter 3 can be used to optimize the geometry of sounding rocket fin. The examples in this chapter used the FalconLAUNCH VI fin as an initial design for the optimization. It was observed that version 2 of the Fin Geometry Optimization Tool yielded better results in the first iteration because of its use of flutter constraint based on a large number of historical designs. The chapter then proceeded to validate the Flutter Factor of Safety Validation Tool using data from FalconLAUNCH V and FalconLAUNCH VI, before using the Flutter Factor of Safety Validation Tool with the optimized designs. The first optimization was found to be too robust, and the second to be marginal. An acceptable design was found during the next iteration of the design tool by increasing the thickness of the second optimization. This final design, referred to as Optimization #2a is over 37% lighter than the FalconLAUNCH VI fins, and should not encounter flutter during the predicted flight profile for FalconLAUNCH

VI. Finally, the designs were validated for flight stability by using Aerolab to ensure that the stability simplification made in the Fin Geometry Optimization Tool was valid.

This example optimization process has demonstrated the utility of the design tool developed in Chapter 3. By using version 2 of the Fin Geometry Optimization Tool and the Flutter Factor of Safety Validation Tool an optimized fin geometry was designed and validated for the FalconLAUNCH VI flight profile. Further, the use of this tool is not limited to the FalconLAUNCH sounding rockets. Any sounding rocket that intended to use solid metal plate fins could be considered, once the predicted flight profile and required panel area have been calculated.

# 5. Conclusions and Recommendations

## 5.1 Thesis Summary

From this thesis research effort, a pair of multidisciplinary design tools to optimize the geometry of sounding rocket fins, minimizing the mass while maintaining aerodynamic performance were developed. The research began in response to the failure of the USAFA FalconLAUNCH V in 2007, which was caused by the fins of that sounding rocket shearing off due to flutter. The following year, the USAFA developed the next iteration of the FalconLAUNCH sounding rockets, FalconLAUNCH VI. The USAFA cadets modified the geometry of the fins for FalconLAUNCH VI in order to prevent flutter, which resulted in significantly more massive fins. This sizable increase in mass just to prevent flutter led to the need for a conceptual design tool that could give future teams the ability to design lightweight fins that would not flutter.

The research began by investigating key issues related to sounding rocket fins, stability, and flutter. First, in order to maintain stability, the fins provide stability by influencing the location of the Center of Pressure, and ensuring that it remains behind the Center of Gravity. To do this the fins need to maintain a minimum panel area since panel area is the dominant factor influencing the location of the Center of Pressure [16]. Second, a number of flutter calculation techniques were found, including Equations (7), (9), and (10), and developing a ZAERO model of the fin. Third, two examples of automated design optimization tools were compared. The design tool used Phoenix Integration's ModelCenter for the design optimization process for its consistent treatment of design variables, especially those relating to geometry. Lastly, a selection of successful fin geometries implied that optimal designs will tend toward low aspect ratios.

The research then covered the development and details of the design tools, the Fin Geometry Optimization and Flutter Factor of Safety Validation Tools. Two versions of the Fin Geometry Optimization Tool were developed and later compared, using two variations of the flutter equation, Equations (9) and (7), from NACA TN 4197. The Flutter Factor of Safety Validation Tool used Equation (10) from the Johns Hopkins University research in 1962.

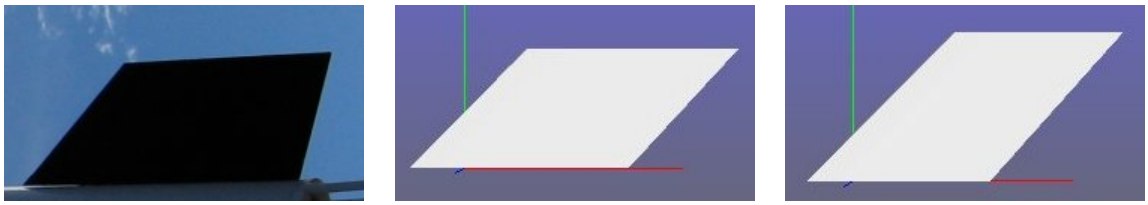
Following the development of the design tools, this research demonstrated their use by optimizing the FalconLAUNCH VI fins. It was observed that version 2 of the Fin Geometry Optimization Tool yielded better results in the first iteration because of its use of flutter constraint based on a large number of historical designs, making it the better choice of the first stage of the overall design optimization process. The research then proceeded to validate the Flutter Factor of Safety Validation Tool using data from FalconLAUNCH V and FalconLAUNCH VI, before using it on the optimized designs. When the first design was shown to be too robust and the second was shown to fail to meet the required factor of safety, a modification of the second design was considered. This design showed more than adequate factor of safety, while still saving a significant amount of mass over the FalconLAUNCH VI design. Finally, the optimized designs were validated using Aerolab, showing that they provided similar stability to the original FalconLAUNCH VI design.

### *5.2 Conclusions*

This research demonstrated that the geometry of sounding rocket fins plays a key role in determining the effectiveness of those fins. The geometry determines the planform area of the fins which determines the location of the Center of Pressure and in turn affects the stability of the vehicle. The geometry, specifically the semi-span and the thickness, also influences the flutter velocity. Finally, the geometry determines the fin's volume and in turn its mass. The geometry's influence over these key performance characteristics means the geometry can be optimized to minimize the mass while providing the required stability and preventing flutter.

The tools developed during this research were developed with this task in mind. They have been shown to accurately predict flutter for sounding rocket fins and to efficiently and easily provide optimized geometries for these fins early in the design process. For example, the Flutter Factor of Safety Validation tool showed that the FalconLAUNCH V fins were highly susceptible to flutter, and that by 11,000 feet in altitude, the onset of flutter was extremely likely. Had these tools been available at the time FalconLAUNCH V was being designed, they could have predicted the fin failure, and aided the USAFA cadets in developing a fin design that would not have failed during flight.

When the design tools developed in this research were applied to the FalconLAUNCH VI fins, they found a new fin geometry that saved over 37% mass over the original design. This design uses a geometry that maintains a just over 15% factor of safety in flutter, instead of a significantly over robust one, and provides the same stability for the rocket as the original design. Finally, the geometry found by using the Fin Geometry Optimizer shares more in common with the fins of previously successful sounding rocket fins than the geometry of FalconLAUNCH V or FalconLAUNCH VI, as can be seen Figure 43. The implication of the optimized fin geometries is that the conjecture in Section 2.6 is correct, and that lower aspect ratio fins have higher flutter velocities. Designers of high velocity sounding rockets like FalconLAUNCH should keep this in mind, regardless of the process they use for designing the fins for their vehicles.



(a) Aerobee 170 Fin Geometry (b) Optimized Fin Geometry (c) FalconLAUNCH VI Fin Geometry

Figure 43 Aerobee 170 and Optimized Fin Geometries vs FalconLAUNCH VI

### *5.3 Recommendations for future work*

The design tool developed in this research not only provides the intended utility, but can also act as the basis of of a more robust set of tools. Possible extensions include:

- Enhance the stability constraint by integrating an analytic center of pressure calculation into the optimization and validation tools
- Use the more robust flutter calculation based on Equation (10) from the Flutter Factor of Safety Validation Tool to re-run the Design of Experiments described in Table 4 to confirm the influence of the fin design parameters on flutter, specifically that the semi-span and thickness are the dominant factors influencing flutter and that sweep angle has little influence over flutter

- Extend the tools to include static aeroelastic effects by adding an aeroelastic divergence constraint to the optimization and validation tools
- Add alternate finite element models of fins to the validation tool to allow designers to consider more complex internal structures than simple solid metal fins
- Extend the flutter calculations to predict flutter for other geometries, including but not limited to geometries such as those being explored as part of the ExFit research at AFIT, which involve relatively large aerodynamic surfaces, such as rudders, mounted at the tips of fins and wings

## *Appendix A. ZAERO Flutter Model*

To prepare the flutter analysis in ZAERO, an aerodynamic model had to be fitted to the reduced plate finite element model, and a flutter velocity calculation configured. The first aerodynamic model had a comparable number of aerodynamic boxes to the number of nodes in the reduced plate model (36) arranged in a symmetric pattern. Figure 44 shows that the two grids are fitted well (the panels represent the aerodynamic boxes and the red crosses represent the nodes from the finite element model). Note how the nodes from the FEM are bounded by the aerodynamic grid. Of particular interest are the nodes along the perimeter, which lie on the edge of the aerodynamic grid, confirming the exact fit of the two models' perimeters. Further verification that the aerodynamic model is properly splined to the finite element model as can be seen in Figures 45 and 46. In these figures, the aerodynamic model can be seen following the structural model's first two normal modes shown in Figures 24 and 25.

These tests implied that the aerodynamic model was fit properly to the structural model. However, initial testing of the flutter model failed to yield the expected results. This initial testing used the FalconLAUNCH V fin geometry and had the altitude set to 11,000 ft, the altitude at which FalconLAUNCH V failed. The calculated flutter velocity was expected to be slightly below the velocity of the vehicle at its point of failure. The actual value generated by the ZAERO analysis was substantially lower.

An aerodynamic mesh refinement study was then run that showed that the number of aerodynamic boxes had been too low to accurately predict the flutter velocity. After adjusting the aerodynamic mesh, the flutter calculation for the FalconLAUNCH V fin at 11,000 ft was re-run. This new ZAERO model calculated the flutter velocity to be in the expected range. However, when the altitude of analysis was adjusted (to simulate using the ZAERO model in the Flutter Factor of Safety Validation Tool), the resulting flutter velocity did not change more than a few percent, which was very unexpected. The flutter velocity should have gone up dramatically as the altitude increased due to the decreasing density of the atmosphere.

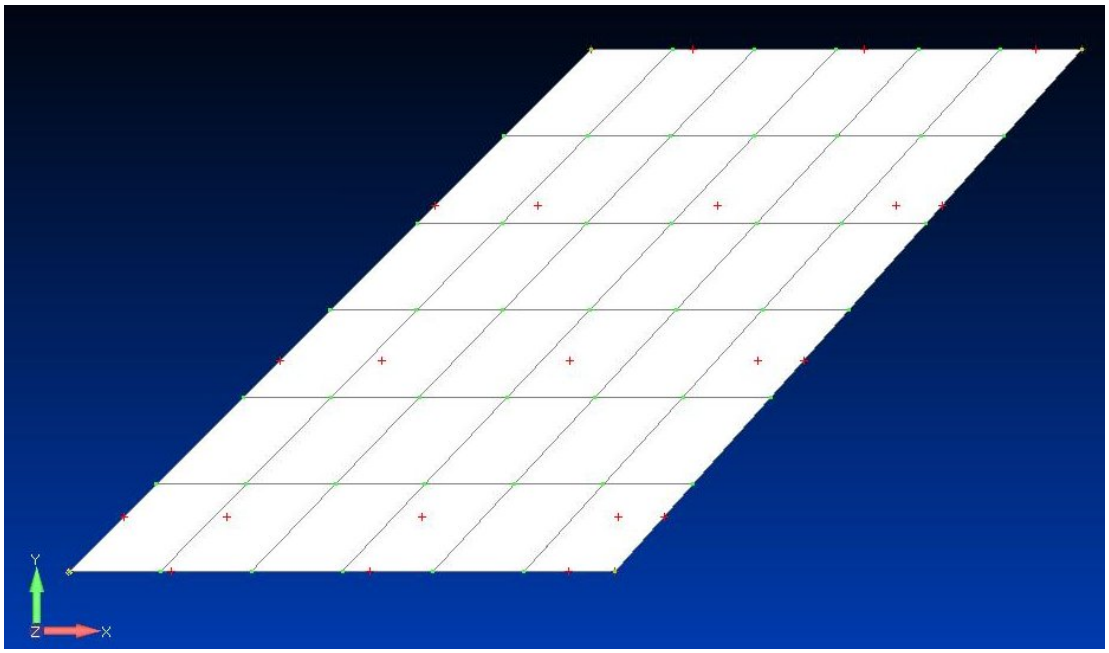
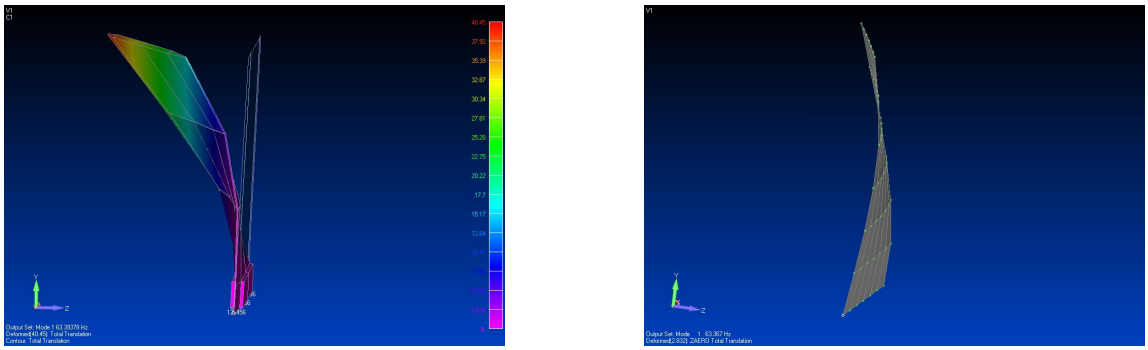


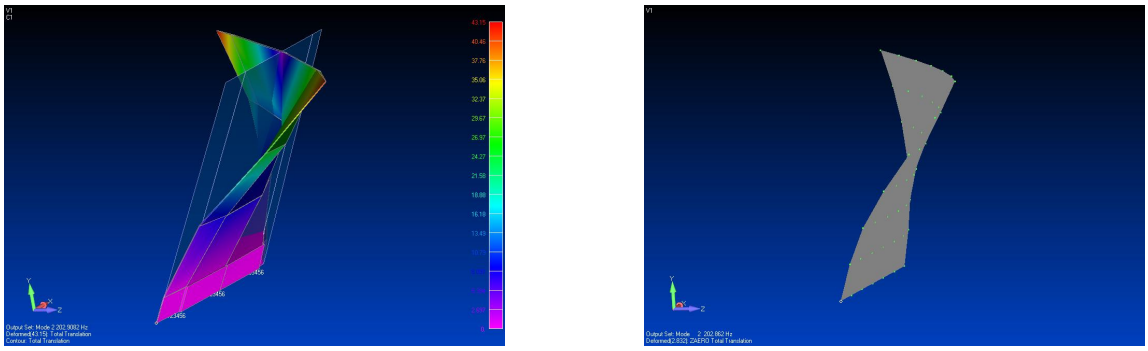
Figure 44 Aeroelastic Analysis Grid with Finite Element Grid overlay



(a) Reduced Plate Model Mode 1 (b) Aero Model Mode 1

Figure 45 Aerodynamic Grid Response to Structure's Normal Mode 1

In addition to the difficulties experienced with changes in altitude, changes in geometry also led to difficulties. For the ZAERO model being used, any changes in fin planform, that is changes in chords or span, required a new aerodynamic mesh refinement study in order to give accurate flutter calculations. This meant that programatically adjusting the ZAERO model would require not just updating geometry information in the model file, but also writing code to run a mesh refinement study which involves additional runs of



(a) Reduced Plate Model Mode 2

(b) Aero Model Mode 2

Figure 46 Aerodynamic Grid Response to Structure's Normal Mode 2

ZAERO, and then updating and testing the aerodynamic mesh, all of which often involves a user in the loop to ensure the model is configured correctly.

In short, the ZAERO model developed during this research was simply not ready for automation, making it an ill suited for use in the Flutter Factor of Safety Tool.

## *Appendix B. Flutter Factor of Safety Validation Tool Source Code*

The following is a list of the components of the Flutter Factor of Safety Tool and any associated source code.

Component 1: Fin Parameters

The *FinParameters* component is a ModelCenter Assembly component used to group the top level design parameters for the fin. It has no source code.

Component 2: Fin Properties

The *FinProperties* component takes the Fin Parameters, and creates the values needed by the finite element analysis and the fin geometry components. The X and Y Coordinates are arrays of 25 values, giving the locations of the 25 grid points in the finite element model. They are calculated from the root chord, tip chord, semi span, and sweep angle. The process starts with the first row (which is the length of the root chord, and starts at the origin and goes down the x axis). After establishing the end points, the points on the inside of the leading and trailing edges are located, for instance, the point on the inside of the leading edge is located at the (leading edge distance, 0). Finally, the center of the fin body is located half way between the inner points.

Next, the points along the tip are located. To begin, the location of the line of the sweep angle is located. The sweep angle is the angle between the quarter chords (root and tip). To find the tip point, the Semi Span is divided by the tangent of the sweep angle (which is the x distance from root quarter-chord to the tip quarter chord) and then this is added to the location of the root quarter chord to get the x location of the tip quarter chord. Note, the y location is simply the semi span. From this point, the lead point of the tip can be found (tip quarter chord location  (1/4) * tip chord). And then the process used for the root line is used to generate the points of the tip line (substituting the tip chord for the root chord in this case).

Finally, the remaining three rows are calculated by finding the points along the lines connecting the end points found in the previous processes. The y locations for each of these rows are: the height of the bracket, one-third of the way from the bracket to the tip, and two-thirds of the way from the bracket to the tip.

```
XCoordinates.SetLength(25)
YCoordinates.SetLength(25)

Function CalculateX(y, x1, x2, y1, y2)
    CalculateX = x1 + ((x2-x1)*(y-y1)/(y2-y1))
End Function


sub run
  Span = 2 * SemiSpan
  AverageChord = (RootChord+TipChord)/2
  AspectRatio = SemiSpan/AverageChord
  PanelArea = SemiSpan*AverageChord
  ApproxVolume = PanelArea*Thickness
  PanelAspectRatio = SemiSpan/AverageChord
  TaperRatio = Tipchord/RootChord
  RTC=Thickness/AverageChord


  ' setup some useful values for calculating node locations
  LeadingEdge = 1.857
  TrailingEdge = 0.839
  Bracket = 1.0
  FinRow1 = ((SemiSpan - Bracket)/3) + Bracket
  FinRow2 = (2*(SemiSpan - Bracket)/3) + Bracket
  pi=4*Atn(1)

  ' First Row of Node Locations
  XCoordinates(0) = 0
  YCoordinates(0) = 0
```

```
XCoordinates(1) = LeadingEdge
YCoordinates(1) = 0
XCoordinates(2) = LeadingEdge + ((RootChord - LeadingEdge - TrailingEdge)/2)
YCoordinates(2) = 0
XCoordinates(3) = RootChord - TrailingEdge
YCoordinates(3) = 0
XCoordinates(4) = RootChord
YCoordinates(4) = 0

' Last Row of Node Locations
tipQuarterChordX = (Rootchord/4) + (SemiSpan / tan(pi*SweepAngle/180))
XCoordinates(20) = tipQuarterChordX - (Tipchord/4)
YCoordinates(20) = SemiSpan
XCoordinates(21) = XCoordinates(20) + LeadingEdge
YCoordinates(21) = SemiSpan
XCoordinates(22) = XCoordinates(20) + LeadingEdge + ((TipChord - LeadingEdge - _
TrailingEdge)/2)
YCoordinates(22) = SemiSpan
XCoordinates(23) = XCoordinates(20) + TipChord - TrailingEdge
YCoordinates(23) = SemiSpan
XCoordinates(24) = XCoordinates(20) + TipChord
YCoordinates(24) = SemiSpan

' 2nd Row of Node Locations
XCoordinates(5) = CalculateX(Bracket,XCoordinates(0),XCoordinates(20), _
YCoordinates(0),YCoordinates(20))
YCoordinates(5) = Bracket
XCoordinates(6) = CalculateX(Bracket,XCoordinates(1),XCoordinates(21), _
YCoordinates(1),YCoordinates(21))
YCoordinates(6) = Bracket
```

```
XCoordinates(7) = CalculateX(Bracket,XCoordinates(2),XCoordinates(22), _
YCoordinates(2),YCoordinates(22))
YCoordinates(7) = Bracket
XCoordinates(8) = CalculateX(Bracket,XCoordinates(3),XCoordinates(23), _
YCoordinates(3),YCoordinates(23))
YCoordinates(8) = Bracket
XCoordinates(9) = CalculateX(Bracket,XCoordinates(4),XCoordinates(24), _
YCoordinates(4),YCoordinates(24))
YCoordinates(9) = Bracket

' 3rd Row of Node Locations
XCoordinates(10) = CalculateX(FinRow1,XCoordinates(0),XCoordinates(20), _
YCoordinates(0),YCoordinates(20))
YCoordinates(10) = FinRow1
XCoordinates(11) = CalculateX(FinRow1,XCoordinates(1),XCoordinates(21), _
YCoordinates(1),YCoordinates(21))
YCoordinates(11) = FinRow1
XCoordinates(12) = CalculateX(FinRow1,XCoordinates(2),XCoordinates(22), _
YCoordinates(2),YCoordinates(22))
YCoordinates(12) = FinRow1
XCoordinates(13) = CalculateX(FinRow1,XCoordinates(3),XCoordinates(23), _
YCoordinates(3),YCoordinates(23))
YCoordinates(13) = FinRow1
XCoordinates(14) = CalculateX(FinRow1,XCoordinates(4),XCoordinates(24), _
YCoordinates(4),YCoordinates(24))
YCoordinates(14) = FinRow1

' 4th Row of Node Locations
XCoordinates(15) = CalculateX(FinRow2,XCoordinates(0),XCoordinates(20), _
YCoordinates(0),YCoordinates(20))
```

```
  YCoordinates(15) = FinRow2
  XCoordinates(16) = CalculateX(FinRow2,XCoordinates(1),XCoordinates(21), _
  YCoordinates(1),YCoordinates(21))
  YCoordinates(16) = FinRow2
  XCoordinates(17) = CalculateX(FinRow2,XCoordinates(2),XCoordinates(22), _
  YCoordinates(2),YCoordinates(22))
  YCoordinates(17) = FinRow2
  XCoordinates(18) = CalculateX(FinRow2,XCoordinates(3),XCoordinates(23), _
  YCoordinates(3),YCoordinates(23))
  YCoordinates(18) = FinRow2
  XCoordinates(19) = CalculateX(FinRow2,XCoordinates(4),XCoordinates(24), _
  YCoordinates(4),YCoordinates(24))
  YCoordinates(19) = FinRow2

end sub
```

Component 3: Fin Geometry

The *FinGeometry* component uses a Phoenix Integration distributed code to render the shape of the fin based on values from the Fin Parameters and Fin Properties components. It has no source code.

Component 4: Finite Element Model (FEM)

The *FEM* component is a ModelCenter VB Script Component that takes the calculated X and Y coordinates from the Fin Properties component and uses FEMAPs COM API to run a structural analysis of the specified fin design.

```
NodeIDS.SetLength(25)
XCoordinates.SetLength(25)
YCoordinates.SetLength(25)

sub run
```

```
' clean up any previous automated runs
On Error Resume Next
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
fso.DeleteFile app.ModelDirectory & "\\FEM\\fl-v-0*.*", True

' start Femap
Dim femap
Set femap = CreateObject("femap.model")

' load model
femap.feFileOpen False, app.ModelDirectory & "\\FEM\\fl-v-fem.MOD"

' insert Node Update code here
Dim nd
Set nd = femap.feNode
For index = 0 to 24
  nd.Get(NodeIDs(index))
  nd.x = XCoordinates(index)
  nd.y = YCoordinates(index)
  nd.Put(NodeIDs(index))
Next

' Update Property Cards
Dim pr
Set pr = femap.feProp
pr.Get(3)  ' Fin Plates
pr.pval(0) = Thickness
pr.Put(3)
```

```
  pr.Get(4)  ' Bracket Plates
  pr.pval(0) = 2 * Thickness
  pr.Put(4)

  pr.Get(5)  ' Leading Edge Plates
  pr.pval(0) = Thickness
  pr.pval(3) = Thickness
  pr.Put(5)

  pr.Get(6)  ' Trailing Edge Plates
  pr.pval(0) = Thickness
  pr.pval(3) = Thickness
  pr.Put(6)

  ' try to run an analysis set
  Dim analysis
  Set analysis = femap.feAnalysisSet
  analysis.Analyze 1

  ' wait on the analysis to complete, then load f06 file
  a = 0
  do while a < 10000000
    a = a + 1
  loop
  f06.fromFile app.ModelDirectory & "\\FEM\\fl-v-000.f06"
  pch.fromFile app.ModelDirectory & "\\FEM\\fl-v-000.pch"
end sub
```

Component 5: FEM Output The *FEM_Output* component is an AnalysisServer QuickWrap component that reads the primary analysis results file (model.f06) and the punch

file (model.pch) and parses them for variable values to be used by other components. It has no source code.

Component 6: Mode 2 Shape

The *Mode2Shape* component is a ModelCenter VB Script Component that iterates through the list of grid points and coordinates of the Mode 2 mode shape from the FEM Output component and reorders the data into the order of the grid points in the Validation Model.

```
mode2x.SetLength(25)
mode2y.SetLength(25)
mode2z.SetLength(25)

sub run

for index = 0 to 24

  grid = NodeIDs(index)
  entry = 0

  for target = 0 to 49
    if trim(Grids(target)) = CStr(grid) then
      entry = target
    end if
  next

  mode2x(index) = X(entry)
  mode2y(index) = Y(entry)
  mode2z(index) = Z(entry)
next

end sub
```

Component 7: Elastic Axis

The *ElasticAxis* component is a Matlab Plugin that calls a Matlab script which calculates the location of the elastic axis of the fin, and then uses this to calculate the distance from the elastic axis to the center of gravity of the fin and the radius of gyration of the fin about the elastic axis as required by Equation (10).

```
% code to find the elastic axis of the FalconLAUNCH fin

%declare variables for the mode shapes to be plotted and used to find
%the elastic axis
mode0x;% = zeros(1,25);
mode0y;% = zeros(1,25);
mode0z;% = zeros(1,25);

mode2x;% = zeros(1,25);
mode2y;% = zeros(1,25);
mode2z;% = zeros(1,25);

scalingFactor = 0.1;
mode2xsum = mode0x + scalingFactor*mode2x;
mode2ysum = mode0y + scalingFactor*mode2y;
mode2zsum = mode0z + scalingFactor*mode2z;

%calculate the elastic_axis
ea_x = zeros(1,3);
ea_y = zeros(1,3);
ea_z = zeros(1,3);

for i=2:4
    row = i+1;
    % find end points for line of deformed surface at this row that crosses
```

```
% the x-y plane
P4 = [0,0,0];
P5 = [0,0,0];
for j=1:4                                       % pos * neg is a neg
    if mode2zsum(5*i+j) * mode2zsum(5*i+j+1) < 0
        P4 = [mode2xsum(5*i+j),mode2ysum(5*i+j),mode2zsum(5*i+j)];
        P5 = [mode2xsum(5*i+j+1),mode2ysum(5*i+j+1),mode2zsum(5*i+j+1)];
        break
    end
end

% find equation of undeformed line
x = [mode0x(5*i+1), mode0x(5*i+5)];
z = [mode0z(5*i+1), mode0z(5*i+5)];
c = [1, 1];
eqn = [x(:) c(:)]\z(:);
m1 = eqn(1);
b1 = eqn(2);

% find equation for line of deformed surface at this row
x = [P4(1), P5(1)];
z = [P4(3), P5(3)];
eqn2 = [x(:) c(:)]\z(:);
m2 = eqn2(1);
b2 = eqn2(2);

% find intersection of these two lines and store in ea_*(i)
p = [-m1, 1; -m2, 1]\[b1; b2];
ea_x(i-1) = p(1);
ea_y(i-1) = P4(2);
```

```
    ea_z(i-1) = p(2);
end

% plot the mode shapes and elastic axis
f = figure('Visible', 'off');
plot3(mode0x, mode0y, mode0z, 'ok', ...
      mode0x(1:5), mode0y(1:5), mode0z(1:5), '-k', ...
      mode0x(6:10), mode0y(6:10), mode0z(6:10), '-k', ...
      mode0x(11:15), mode0y(11:15), mode0z(11:15), '-k', ...
      mode0x(16:20), mode0y(16:20), mode0z(16:20), '-k', ...
      mode0x(21:25), mode0y(21:25), mode0z(21:25), '-k', ...
      [mode0x(1), mode0x(6), mode0x(11), mode0x(16), mode0x(21)], ...
      [mode0y(1), mode0y(6), mode0y(11), mode0y(16), mode0y(21)], ...
      [mode0z(1), mode0z(6), mode0z(11), mode0z(16), mode0z(21)], '-k', ...
      [mode0x(5), mode0x(10), mode0x(15), mode0x(20), mode0x(25)], ...
      [mode0y(5), mode0y(10), mode0y(15), mode0y(20), mode0y(25)], ...
      [mode0z(5), mode0z(10), mode0z(15), mode0z(20), mode0z(25)], '-k',...
      mode2xsum, mode2ysum, mode2zsum, '+g', ...
      mode2xsum(1:5), mode2ysum(1:5), mode2zsum(1:5), '-g', ...
      mode2xsum(6:10), mode2ysum(6:10), mode2zsum(6:10), '-g', ...
      mode2xsum(11:15), mode2ysum(11:15), mode2zsum(11:15), '-g', ...
      mode2xsum(16:20), mode2ysum(16:20), mode2zsum(16:20), '-g', ...
      mode2xsum(21:25), mode2ysum(21:25), mode2zsum(21:25), '-g', ...
      [mode2xsum(1), mode2xsum(6), mode2xsum(11), mode2xsum(16), mode2xsum(21)], ...
      [mode2ysum(1), mode2ysum(6), mode2ysum(11), mode2ysum(16), mode2ysum(21)], ...
      [mode2zsum(1), mode2zsum(6), mode2zsum(11), mode2zsum(16), mode2zsum(21)], '-g', ...
      [mode2xsum(5), mode2xsum(10), mode2xsum(15), mode2xsum(20), mode2xsum(25)], ...
      [mode2ysum(5), mode2ysum(10), mode2ysum(15), mode2ysum(20), mode2ysum(25)], ...
      [mode2zsum(5), mode2zsum(10), mode2zsum(15), mode2zsum(20), mode2zsum(25)], '-g', ...
      ea_x, ea_y, ea_z, '-*b', ...
```

```
      cgX, cgY, 0, '*r');
set(gca,'DataAspectRatio',[1 1 1])
%view(10,55);
view(15,75);
saveas(f,'fin.jpg');


%find radius of gyration and cg relative to the elastic axis
%  start with eqn for elastic axis
c = [1, 1, 1];
ea = [ea_x(:) c(:)]\ea_y(:);

%  then find distance from elastic axis to cg
%  see http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html
a = -ea(1);
b = 1;
c = -ea(2);
xbar = abs(a*cgX + b*cgY + c)/sqrt(a^2 + b^2);

r = sqrt(Iyy/mass + xbar^2);
```

Component 8: Elastic Axis Graphic

This component simply waits for the Elastic Axis component to run, and then imports the image file from the Elastic Axis component into ModelCenter for review.

```
sub run
  image.isBinary = true
  image.fileExtension = "jpg"
  image.fromFile app.ModelDirectory & "\\matlab\\fin.jpg"
end sub
```

Component 9: Convert Units

The *ConvertUnits* component is a ModelCenter VB Script Component that transforms the units of various inputs from the design units of inches into the analysis units of feet used in the Flutter Test component.

```
sub run
x = xbar_inches/12
r = r_inches/12
b = chord_inches/12/2
w = Mode1/Mode2

' m is mass per unit length in lbf*s^2/in^2
m = mass / span

' density must by in lbm/in^3
rho = density / (12*12*12)

mu = m/(4*rho*b*b)
end sub
```

Component 10: Standard Atmosphere

The *StdAtm* component is a Matlab Plugin that calls a Matlab script that calculates the 1976 US Standard Atmosphere written by Richard Rieber [30]. Source code available from its author.

Component 11: Flutter Test

The *FlutterTest* component is a MathCad Plugin that calls a MathCad worksheet that implements Equation (10), which generates a flutter parameter $V_f/(b\omega_\alpha)$. It has no source code.

Component 12: Flutter Velocity

The *FlutterVelocity* component is a ModelCenter VB Script Component that transforms the flutter parameter from the Flutter Test component into the flutter Mach number.

```
sub run
flutter_m = V_sim * b * w / a
flutter_fos = flutter_m * 1.15
end sub
```

*Bibliography*


1. Nasa space vehicle design criteria: Flutter, buzz, and divergence. Technical Report NASA SP 8003, NASA, 1964.
2. *Design Sensitivity and Optimization User's Guide*. MSC Software Corporation, Santa Ana, CA, 2003.
3. *Applied Aerodynamics: A Digital Textbook*. Desktop Aeronautics, Inc., Stanford, CA, 2007. URL `http://www.desktopaero.com/appliedaero/preface/welcome.html`.
4. *MATLAB R2007b User Manual*. The Mathworks, Inc., Natick, MA, 2007.
5. *ZAERO User Manual*. Zona Technology, Scottsdale, AZ, 2007.
6. *FEMAP 9.31 User Manual*. Siemens PLM Software, Plano, TX, 2008.
7. *Mathcad 14 User Manual*. Parametric Technology Corporation, Needham, MA, 2008.
8. *ModelCenter 8.0 User Manual*. Phoenix Integration, Blacksburg, VA, 2009.
9. Fin flutter (web page), cited 01/31/2009. URL `http://www.info-central.org/design_finflutter.shtml`.
10. Cajunfin2, cited 02/12/2009. URL `http://www.ahpra.org/CajunFin2.jpg`.
11. Point-line distance–2-dimensional – from wolfram mathworld, cited 05/20/2009. URL `http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html`.
12. Standard atmosphere calculator, cited 05/20/2009. URL `http://www.digitaldutch.com/atmoscalc/`.
13. Rocket stability, cited 05/28/2009. URL `http://exploration.grc.nasa.gov/education/rocket/rktstab.html`.
14. Jerry M. Allen. Parametric fin-body and fin-plate database for a series of 12 missile fins. Technical Report 2001-210652, NASA, 2001.
15. Peter Alway. Missile scale data, cited 02/12/2009. URL `http://www.rocketryonline.com/jimball/alway/missile\_scale\_data.htm`.
16. Anonymous. *Military Handbook: Design of Aerodynamically Stabilized Free Rockets*, volume MIL-HDBK-762(MI). 1990.
17. James S. Barrowman. *The Practical Calculation of the Aerodynamic Characteristics of Slender Finned Vehicles*. (MS Thesis, The Catholic University of America), 1967.
18. John Cipolla. Aerofinsim description page (web page), cited 01/31/2009. URL `http://aerorocket.com/finsim.html`.
19. Marty France and Ralph Sandfry. Falconlaunch iv launch & recovery. Technical report, US Air Force Academy, 2007. URL `http://www.usafa.af.mil/df/dfas/Papers/20062007/FalconLAUNCH%20IV%20La% unch%20%26%20Recovery%20-%20France.doc`.

20. Y. C. Fung. *An Introduction to the Theory of Aeroelasticity*. Galcit Aeronautical Series. John Wiley & Sons, New York, 1955.

21. I. E. Garrick and S. I. Rubinow. Flutter and oscillating air-force calculations for an airfoil in a two-dimensional supersonic flow. Technical Report NACA Report No. 846, NACA, 1946.

22. Perry W. Hanson and Gilbert M. Levey. Experimental and calculated results of a flutter investigation of some very low aspect-ratio flat-plate surfaces at mach numbers from 0.62 to 3.00. Technical Report NASA TM X-53, NASA, 1962.

23. Dewey H. Hodges and Alvin G. Pierce. *An Introduction to Structural Dynamics and Aeroelasticity*. Galcit Aeronautical Series. John Wiley & Sons, New York, 2002.

24. J. P. Kearns. Flutter simulation. Technical Report DTIC AD0650981, Johns Hopkins University, 1962.

25. William Kent. *Mechanical Engineers´ Handbook*. Wiley engineering handbook series. Wiley, New York, 1950.

26. William T. Lauten and J. G. Barmby. Continuation of wing flutter investigation in the transonic range and presentation of a limited summary of flutter data. Technical Report NACA RM L9B25B, NACA, 1949.

27. Natori M., Onoda J., and Kitamura T. Flutter analysis of a sounding rocket fin. In *Space Technology and Science*, pages 305–310, 1984. adsnote: Provided by the SAO/NASA Astrophysics Data System.

28. Dennis J. Martin. Summary of flutter experiences as a guide to the preliminary design of lifting surfaces on missiles. Technical Report NACA TN 4197, NACA, 1958.

29. Walter D. Pilkey. *Formulas for Stress, Strain, and Structural Matrices*. John Wiley & Sons, New York, 1994.

30. Richard Rieber. Matlab central - file detail - standard atmosphere calculator, cited 05/20/2009. URL `http://www.mathworks.com/matlabcentral/fileexchange/8799`.

31. Randy H. Shih. *Parametric Modeling with Autodesk Inventor R6*. SDC Publications, Mission, KS, 2002.

32. J. Simmons, A. Deleon, et al. Aeroelastic analysis and optimization of falconlaunch sounding rocket fins. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, number AIAA-2009-515, 2009.

33. Pat Stakem. A letter to the technical editor. *Model Rocketeer*, February 1973. URL `http://www.geocities.com/rocketguy101/downloads/ssfincna.pdf`.

34. Harry G. Stine and Bill Stine. *Handbook of Model Rocketry*. John Wiley & Sons, New York, seventh edition edition, 2004.

35. T. Theodrsen. General theory of aerodynamic instability and the mechanisms of flutter. Technical Report NACA TR 496, NACA, 1934.

36. Hans O. Toft. *Aerolab 1.3 User Manual*. 2003.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704–0188*



| 1. REPORT DATE (DD–MM–YYYY) | 2. REPORT TYPE | 3. DATES COVERED *(From — To)* |
|---|---|---|
| 18-06-2009 | Master's Thesis | JAN 2008 – JUNE 2009 |

**4. TITLE AND SUBTITLE**

Aeroelastic Optimization of Sounding Rocket Fins

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

**6. AUTHOR(S)**

Joseph R. Simmons, III

5d. PROJECT NUMBER
#JON 343

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/ENY)
2950 Hobson Way
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GSS/ENY/09-J02

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Air Force Research Laboratory Air Vehicles Directorate
Attn Barry Hellman
2130 8th St, Bldg 45
WPAFB Dayton, OH 45433
937-255-6795 (DSN: 785-6795) Barry.Hellman@WPAFB.AF.MIL

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RBAA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

**12. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; Distribution Unlimited

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

This research effort develops a multidisciplinary design tool to optimize sounding rocket fin geometries that minimize the mass of the fins while maintaing aerodynamic performance. This research grew out of a design problem experienced by the US Air Force Academy's FalconLAUNCH program. The FalconLAUNCH program is a senior design capstone project during which Air Force Academy cadets design, build and fly a sounding rocket over the course of an academic year. In the Spring of 2007, the FalconLAUNCH V vehicle experienced a catastrophic failure when three of its four fins sheared off due to flutter. When the following year's team developed the fins for FalconLAUNCH VI, the design requirement that the fins not experience flutter led to substantially more massive fins. The FalconLAUNCH team needs a design tool that can balance the competing needs for minimal mass sounding rocket components and aerodynamic performance. The tool developed during this research is designed to find an optimal solution for the fin geometry based on the competing needs of minimizing the fins' mass and ensuring the fins will not experience flutter. The design tool then provides for verification of the design throughout the designed flight profile.

**15. SUBJECT TERMS**
Sounding Rocket Fins, Flutter, Optimization, Geometry Optimization, ModelCenter, Aeroelasticity, Finite Element Analysis

| 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. Jonathon Black |
| U | U | U | UU | 106 | 19b. TELEPHONE NUMBER (Include Area Code)<br>937-255-3636 x4578<br>e-mail: Jonathon.black@afit.edu |