ABSTRACT

BURGUN, ROBERT SCOTT, The Design, Analysis, Construction and Testing of an Uninhabited Aero Vehicle Platform. (Under the direction of Dr. Charles E. Hall, Jr.)

An uninhabited aero vehicle platform design is presented. This encompasses the landing gear system and the structures of the vehicle. The landing gear system consisted of the design, construction and testing of the main and nose gears. The testing of the landing gear resulted in a valid system that could then be integrated into the vehicle. The vehicle structures are composed of various configurations of composite sandwiches. Extensive material testing was conducted to experimentally produce the physical properties of the materials. These properties and techniques can be utilized by other vehicle designs. The structural design was refined and ultimately verified within a finite element analysis program, ANSYS. This analysis implemented a composite shell element that utilized all of the material properties gained from the material testing. This work resulted in an analyzed and constructed vehicle. Ultimately the vehicle was load tested to verify the analytical results.

THE DESIGN, ANALYSIS, CONSTRUCTION AND TESTING OF AN UNINHABITED AERO VEHICLE PLATFORM

By

ROBERT SCOTT BURGUN

A Thesis submitted to the Graduate Faculty of
North Carolina State University
In partial fulfillment of the
Requirements for the Degree of
Master of Science

AEROSPACE ENGINEERING

Raleigh, NC

2003

APPROVED BY:

Dr. Charles E Hall, Jr. Advisory Committee Chairman	Dr. Eric Klang Advisory Committee Member

Dr. James Selgrade
Advisory Committee Member

BIOGRAPHY

Robert Scott Burgun was born in Islip, New York in 1970 to Michael and Diane Burgun. He grew up with his older brother Michael and older sister Wendy. He moved to south Florida while he was in grade school. While he was in Florida he became interested in remotely controlled aircraft and electronics. He graduated from J. P. Tarravella High School in Coral Springs, Florida in 1989. He graduated from North Carolina State University in 1995 with a Bachelors of Science degree in Electrical Engineering. He worked for four year before returning to North Carolina State University in the spring of 1999 to pursue a degree in Aerospace Engineering. He graduated with a Bachelors of Science degree in Aerospace Engineering in 2001.

His graduate career started in the fall of 2001 in the Mechanical and Aerospace Department at North Carolina State University. In the spring of 2002, he was awarded a research assistantship to work on an industry sponsored project of a uninhabited aero vehicle under the direction of Dr. Charles E. Hall, Jr. Robert plans to graduate in December of 2003 with a Master of Science degree in Aerospace Engineering with a minor in Mathematics.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Charles E. Hall, Jr. for the opportunity to work on a fulfilling project. I would also like to thank Dr. Eric Klang and Dr. James Selgrade for being on my graduate committee. My thanks are given to Mr. “Skip” Richardson and Mr. Mike Breedlove for the machining work performed for this project.

The Integration and Demonstration Branch of the Air Force Research Laboratory support this research project. The AFRL contract is F33615-01-C-3117, with 1st LT Leslie Snodgrass as program manager. The support of Mr. Marvin “Skip” Gridley and Mr. Henry “Hank” Baust of AFRL is also gratefully acknowledged. This document was Cleared by ASC/PA on 12/23/2003 as Document Number ASC 03-3268.

The material support and discussions from Lockheed Matrin, Ft. Worth TX of Mr. Derek R. Bye, Dr. Daniel N. Miller, and Mr. Phil Truax in the overall vehicle design, propulsion system and requirements of micro-fluidic vortex generators are also gratefully acknowledged. Tao Systems, Williamsburg VA, Dr. Garimella R. Sarma and Dr. Siva M. Mangalam have provided many useful discussions concerning the operational requirements of the sensing package for the inlet.

I would like to especially thank my parents and family for their support of my aerospace engineering endeavor.

The experience of this project would not have been as fulfilling without the support of my colleague and propulsion system integrator, Wallis V. Collie. I would like to acknowledge an undergraduate student, David Roberts, for all of his assistance during the construction and material testing of the UAV.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Roman Symbols

b	Base of cross-section
c.g.	Center of gravity
C	Carbon fiber
C_L	Lift coefficient
C_p	Pressure coefficient
E	Modulus of elasticity
g	Gravity
G	Fiberglass
G	Shear modulus
H	Height
I	Moment of inertia
l	Length of bar
L	Lift
N	Reaction factor
P	Load
P	Pressure
q	Dynamic pressure
S	Reference area
V	Volume ratio
W	Weight of the UAV

Greek Symbols

d	Deflection
?	Dampener efficiency
?	Density
$?_b$	Tip back angle
$?_g$	Angle of the landing gear projected on the ground
$?_{over}$	Tip over angle
?	Poisson's ratio

Subscripts

x,y	Face sheet
z	Thru the core
8	Free stream

1. INTRODUCTION

The objective of this project was to design; manufacture, ground and flight test an uninhabited aero vehicle (UAV). The design phase of the UAV focuses on the landing gear and structural designs and their testing. The customer requirements of the UAV gave insight into the complexity of the design of the UAV.

The airframe of the UAV is composed of composite materials. Composites have been utilized in remotely piloted vehicles for the capstone coarse at North Carolina State University since 1989. Those remotely piloted vehicles have been designed, constructed and flight tested. During the ground testing of those vehicles it was found that the numerical analysis results did not match the experimental results. The deflections of the wings during load tests were off by a factor of two. The analysis results were consistently predicted lower than the experimental results. The strains that were incurred during load tests compared well to the analytical results. Composite properties vary greatly between different types of weaves and laminations. As a result, comprehensive material testing was conducted for this UAV.

1.1. OVERVIEW

The design of the UAV commenced with Concurrent Engineering. This aspect of the design process revealed that the main components of the UAV design focused on the landing gear, materials and the airframe. The landing gear was completely designed, analyzed, manufactured and tested for the vehicle. The landing gear design needed to fit into the UAV. The analysis covered landing and side load iterations. Upon completion of the analysis of the landing gear the structural design of the UAV started. The clearance dimensions of each gear and their reaction forces were utilized in the design layout of the structures of the UAV. The testing encompassed drop and side load tests of the main gear and drop and steering tests of the nose gear.

Before the structural design could be initiated, the composite materials and the control surface actuation needed development. The composite materials were tested to derive the actual laminated material properties. These properties were analyzed within ANSYS, finite element analysis software, to verify the composite properties. The control surface actuation was designed around the aerodynamic loads that are present on the control surfaces. The control surface design actuation included the design of the hinges and the control surface actuation. The design was tested on a mock-up of the control surface before it was implemented on the UAV.

After the landing gear design, material testing and control surface design was completed the airframe design, analysis, construction and testing commenced. The structural design of the vehicle incorporates advanced composites to handle the aerodynamic and landing loads. The internal components of the vehicle are mounted to the internal structures. The structural design of the UAV was analyzed within ANSYS. Once the airframe was analyzed, construction of the vehicle commenced. The construction encompassed access hatches, vehicle construction and system installation. After the entire airframe was completed it was subjected to static load testing. The load testing was performed to validate the computational analysis and prove that the UAV is structurally safe for flight conditions.

1.2. CUSTOMER REQUIREMENTS

The customer requirements define the end goals of the project. The requirements encompass the performance, payload, landing gear, airframe configuration, structural loads and the propulsion system of the vehicle. These requirements interact with each other. The propulsion system directly affects the performance of the UAV and the landing gear affects the airframe configuration. Each requirement required investigation before the design of the UAV could begin.

The performance goals of the vehicle range from its handling qualities to the landing and aerodynamic structural loads. The vehicle a design that can be flight tested under different static margins. The static margin needs to be configurable from a stable pilot controlled 8% to a computer assisted neutral static margin. The vehicle must controllable under flight conditions as well as on the ground. The vehicle must takeoff and land unassisted from a runway. Individual flight times range from twenty to thirty minutes for acquiring research data.

The landing gear, airframe configuration and structural loads all interact with each other. The landing gear configuration is a tri-cycle arrangement that is required to be retractable. They also need to absorb a 4g impact load with a vertical sink rate of 10 feet per second. These loads are then transferred into the structures of the vehicle. The vehicle also needs to be designed to handle maximum aerodynamic loads of +/- 6g with a safety factor of 1.5. The airframe configuration was also required to simulate the original vehicle, which makes the vehicle modular.

The vehicles payload includes avionics, flight control systems, control valves and other miscellaneous components necessary for flight. The avionics cover the flight computer, transducers, controllers and necessary actuators. The avionics will control all of the in flight data collection as well as the actuation and sensing duties. The flight control systems include the landing gear, steering and vehicle control surface systems. The control valves on the vehicle are used for enabling flow control in the inlet.

The propulsion system of the UAV consists of a turbojet power plant, inlet, fuel system and aircraft cooling/insulating. Lockheed Martin Aeronautics designed the inlet that this aircraft will test in flight conditions. The inlet is a highly compact serpentine design with flow control devices. The compressor section of the turbojet was tapped for bleed air for the inlet flow control. This flow needs to be cooled via a heat exchanger before reaching the turbojet inlet. Cooling of the avionics and the interior of the aircraft was accomplished by using two

NACA ducts located on the lower surface of the vehicle. The interior of the vehicle needs cooling because of the excessive heat produced by the turbojet power plant.

1.3. VEHICLE DESCRIPTON

The aircraft design for the uninhabited aerial vehicle (UAV) research test bed is a 47% scale version of the *SMART* REVCON air vehicle. The *SMART* REVCON vehicle was jointly designed by NASA and Lockheed Martin, Figure 1.1. This vehicle was designed to serve as a demonstrator for emerging UAV technologies. North Carolina State University (NC State) 47% scale version is called *VIPER*, **V**ersatile and **I**nnovative **P**latform for **E**ngineering **R**esearch.

Figure 1.1: NASA Langley/Lockheed Martin REVCON Air Vehicle.

The primary purpose of the *VIPER* UAV is to aid in the development of subsonic ultra-compact inlet duct systems for future tactical aircraft. The UAV allows the ultra-compact duct and flow control to be tested under realistic dynamic flow conditions. The UAV also permits an assessment of the integration issues related to the flow control systems and an evaluation of the performance of the complete aircraft. The design and testing of the micro-turbojet propulsion system is discussed in Collie.[1] Ultra-compact inlet ducts have large offsets between the inlet and exit areas, as well as a high exit-to-inlet area ratio[2]. To

overcome the design limitations a new technology, active inlet flow control, is being developed.[3] Although ground test experiments[4] and computational fluid dynamics methods[5] are being intensively applied to verify this emerging technology, these tests and design methods have not been able to fully simulate flight conditions.

The inlet technology will be tested in the *VIPER* UAV. The UAV needed to be large enough to make the integration of the inlet possible. The *VIPER* UAV's overall wingspan is eight feet, just under nine feet in length and two and a half feet in height. The UAV's airframe incorporates modularity as in the full-size vehicle. The NC State *VIPER* UAV can be seen in Figure 1.2. The center section of the UAV contains the retractable landing gear, turbojet, compact serpentine inlet, fuel and avionics. A micro-turbojet in the 150 pound thrust class turbine by Aviation Microjet Technology (AMT) powers the UAV. The *VIPER* UAV contains an avionics package that implements PC104 boards running real time LINUX. The avionics package is used for real-time flight control systems, data acquisition, and data storage[6].

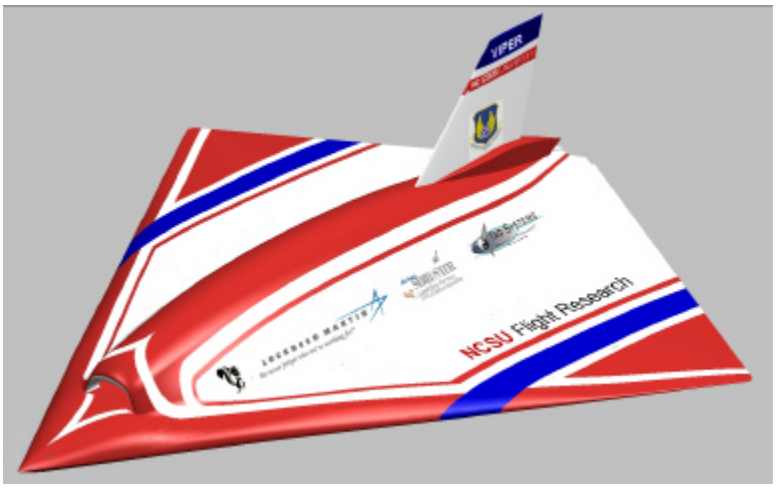


Figure 1.2: NC State *VIPER* UAV.

1.4. CONCURRENT ENGINEERING

The *VIPER* UAV's design started with concurrent engineering. Concurrent Engineering (CE) is a form of total quality management that aids in the design of the UAV. CE helps detail the

relevant factors, risks, tasks and their interrelationships. The topics for CE included aerodynamics, manufacturing, mission specifications, performance, propulsion, landing gear, structures and others. CE organizes ideas, highlights areas of concern and drives the design process from the least to most important aspect of the UAV.

The steps of Concurrent Engineering are Affinity Diagrams, Tree Diagrams, Interrelationship Diagram, Prioritization Matrix and Quality Functional Deployment (QFD) Matrices. The Affinity and Tree Diagrams are used to organize all of the possible areas of the UAV under major headings. From the organized areas, the interaction and importance of each area is developed in the remaining stages of CE. The Prioritization Matrix weighs the interaction between all of the areas with each other. The QFD matrices are the final process of CE. These matrices depict the importance and relationship between the different stages of the entire project. The matrices start with the mission requirements versus the aircraft flight characteristics. Then the aircraft flight characteristics versus the aircraft system integration and end up with the maintenance versus the flight-testing of the vehicle. All of the steps of the CE process are located in Appendix 10.1.

During the CE process of the *VIPER* UAV, the propulsion system, vehicle structures, landing gear system, static margin shift and weight were very important. These aspects of the UAV dominated the Prioritization Matrix. The Interrelationship Diagram also emphasized the interaction between each topic. For example, if the propulsion system design were altered a cascade of changes would have to be made in the landing gear system, structural design, center of gravity and the weight of the UAV. Therefore, detailed designs of these systems were needed before the structural design of the UAV could start.

2. LANDING GEAR DESIGN AND ANALYSIS

2.1. LANDING CONCURRENT ENGINEERING

The design of the landing gear system started with concurrent engineering specifically for the landing gear. This iteration of concurrent engineering was not as formal as the entire UAV's concurrent engineering. The brainstorming, tree diagrams, interrelationship diagram and the prioritization matrix were conducted. Even though the landing gear concurrent engineering was a subset of the entire *VIPER* UAV, it had to have interaction with the vehicle itself. This meant that during the design of the landing gear the integration into the UAV was important.

The concurrent engineering for the landing gear showed that the landing loads, placement, weight, varying static margin and the manufacturing were critical aspects. It showed that repositioning or modification to the landing gear for different static margins would complicate the landing gear system. Any repositioning of the landing gear would also reduce the amount of usable aircraft volume for research devices. The placement of the landing gear dictates the available volume for the landing gear to occupy. The gear placement also ties in with the static margin shift of the UAV. The weight and strength of each landing gear design needed optimization to keep the total weight of the UAV down.

The retraction, interaction between the landing gear and other systems of the UAV were secondary driving forces of the design. The retraction mechanism needs to be strong, lightweight and compact. The construction, maintenance and installation of each gear would be simplified if they could be designed as complete individual units. Each unit contains all of the pivots for retraction, the retraction mechanism and the steering mechanism for the nose gear or the brake system for the main gear.

2.2. LANDING GEAR DESIGN

The necessity of large changes in static margin pushed the nose gear as far forward as possible. The *VIPER* UAV needs to be stable and maneuverable on the ground under all possible static margins. The stability on the ground requires that the UAV will not encounter dangerous situations during ground maneuvers. The maneuverability aspect correlates how effective the steering is on the ground.

At low static margins, adequate weight is on the nose wheel for maneuverability on the ground. High static margins add more weight to the nose and the distribution is adequate for rotation and maneuverability on the ground. The weight distribution for the extremes of the static margins is seen in Table 2.1.

Table 2.1: Weight Distributions for Varying Static Margins

Static Margin (%)	Weight Distribution (%)	
	Nose	Main
8	20	80
0	7.1	92.9

There are two critical angles that are formed from the layout of the landing gear system. These angles are the tip back and rollover angles. They determine the stability of the UAV during ground maneuvers. The tip back angle mainly corresponds to landing conditions. The roll over angle equates to the tendency of the vehicle to tip on its side during ground maneuvers. Both of these angles need attention for the vehicle to be safe and controllable.

The tip back angle is used to determine if the vehicle would fall backwards. This occurs mostly under landing conditions. A vehicle is unsafe if it tips back and cannot rotate back down. During ground maneuvers it is important to minimize the amount of time that the nose wheel is off the ground. To determine the tip back angle, the footprint geometry of the landing gear, Table 2.2, needs to be known.

Table 2.2: Landing Gear Footprint Geometry

Nose Wheel Location	7.00
Main Wheel Location	56.55
CG location at 8% SM	46.64
CG location at 0% SM	52.83
Height from Ground to CG	14.00
Main Gear Tire to Centerline	10.00

The tip back angle is calculated as shown in Figure 2.1. The tip back angle is the angle between the vertical wheel line and the line from the bottom of the main wheel passing through the center of gravity. Since the *VIPER* UAV will be flown at varying static margins, the extremes were calculated. The values for the extremes were calculated to be 35.3° for 8% static margin and 14.9° for the neutral static margin. The tip back angle should be greater than ~15 so that the vehicle will not sit on the tail section. The angle between the ground and the line that the main wheel makes to the tail section, tail strike angle, is 14.9° for the *VIPER* UAV therefore the tip back angles are reasonable.

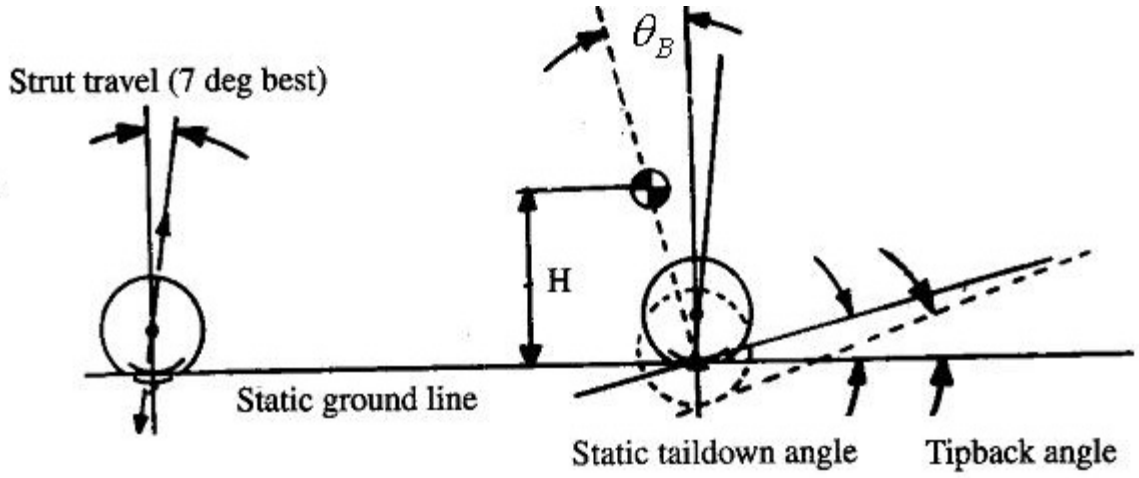


Figure 2.1: Landing Gear Tip Back Angle[7]

The roll over angle, Figure 2.2, is more critical to the maneuverability of the UAV on the ground than the tip back angle is. When the roll over angle gets too high the vehicle can roll onto its side while maneuvering on the ground. The rollover angle is defined by the angle between the vertical height, from the ground to the c.g., and the perpendicular distance from the c.g. to the line between the main and nose wheel. The maximum value associated with

large transport/cargo vehicle is 63°. Since the *VIPER* UAV will be flown at varying static margins, the extremes were calculated. The values for the extremes were calculated to be 63.3° for 8% static margin and 59.8° for the neutral static margin. These values turned out to be on the edge of the limit. The values were deemed acceptable since the UAV will be controlled via a transmitter and it will not be traveling fast.

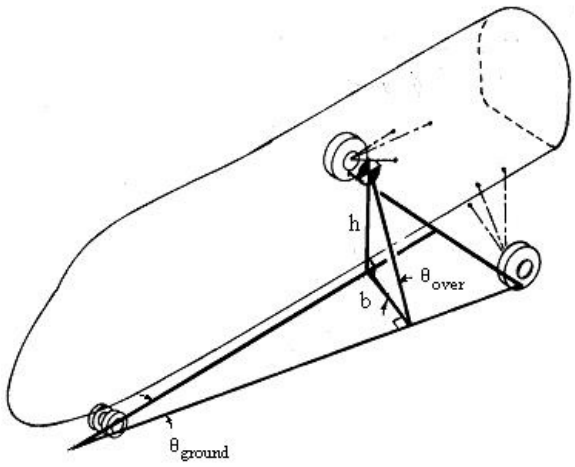


Figure 2.2: Landing Gear Configuration Roll Over

The UAV's geometry was a critical aspect that shaped the landing gear design. The nose and main gear strut designs are compact so that they can be retracted into the available volume. The volume constraints also shaped the design and the complexity of the retraction mechanism.

2.3. MAIN GEAR

The main landing gear is located close to the turbojet, fuel tank, exhaust pipe and wing connection locations. The main gear needs to handle high impact loads and it needs to be retracted into the available volume. The upper surface of the UAV slopes down to the transition of the wing above the main landing gear.

2.3.1. Design

From design requirements of the landing loads the main tires travel needed to be calculated.

Lift at landing is estimated by:

$$L_{landing} \approx 0.67 \cdot W$$

The shock strut efficiency is 0.85, which correlates to pneumatic shock. Pneumatic shocks will be used for the main landing gear dampening.

The reaction factor (N_{gear}) is at the extreme range for Air Force requirements.

$$Stroke \approx \frac{V_{vertical}^{2}}{2 \cdot g \cdot \left(N_{gear} - \left(\frac{W - L_{landing}}{W} \right) \right)}$$

$$Stroke = 6.07 \text{ inches}$$

A lightweight dampener was needed to arrest the impact loads seen by the main gear. A pneumatic shock and spring combination was the choice. The shock is the Cane-Creek AD-12, which is a mountain bike shock. The shock is light since it does not utilize oil as the dampening fluid and heavy springs. The pneumatic shock can be tailored for different landing conditions. By adjusting the air pressure inside the shock the spring rate is changed. The bounce/rebound can be adjusted by changing the external screws for the internal valves.

The first step in designing the main gear was to create a kinematics design. The kinematics design was used to place the shock to achieve the necessary stroke of the wheel. The shock has a maximum travel of two inches. The wheel needs a travel of six inches. A series of iterations were completed until a design that can achieve the necessary stroke was determined. The design iterations also focused on the design being compact, Figure 2.3. At

the maximum stroke of the wheel the knee, the joint between the lower bar and the upper bar, has one-half of an inch clearance to the ground.

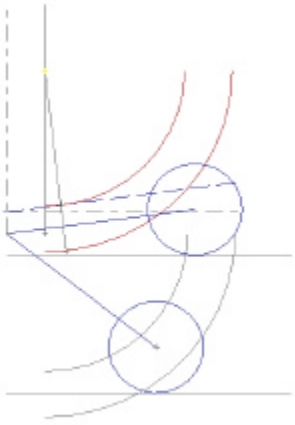


Figure 2.3: Kinematics Design of the Main Gear.

After the kinematics design was completed, the design process moved on to fitting the main landing gear into the UAV. The available volume at the location of the main landing gear necessitated a compact design. The rotation point was located behind the vertical bar to get the main gear into the UAV. Then the vertical bar's layout could be completed. The vertical bar extends over the top of the shock rearward to the rotation point. With the retraction point behind the shock, the retracted main gear now fits within the UAV with adequate clearances between the main landing gear and the skin of the UAV, Figure 2.4.

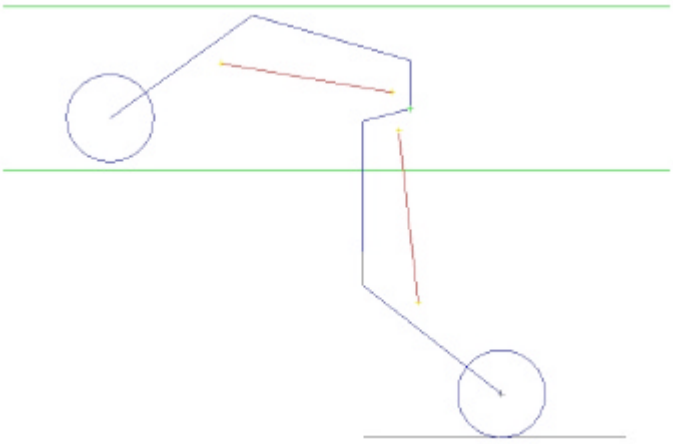


Figure 2.4: Retracted Layout of the Main Landing Gear

The overall structural design of the bars for the main landing gear is an I-beams. The I-beam design was chosen for its excellent strength versus weight characteristics. The web of the I-beam was placed in the direction of motion so that it can absorb and transfer the landing loads. The vertical bar required stiffening to reduce the twisting incurred from high side loading. To reduce the twisting cross bracing was added to the webbing.

The lower bar holds the tire and brake assembly. The high landing loads created a few obstacles. The high landing loads seen by the tire require a strong design for mounting the axel. A fork design was implemented to handle the loads for several reasons. The fork design is lighter than a cantilevered design, but it does create mounting complications. The tire is located on the centerline of the strut with the fork design, which means that the loads are transmitted evenly through each fork. The difficulty with the fork design is that the brake system and tire need to be installed as a complete unit. After the tire is mounted the brake system has to be bolted to the side of the fork. The fork is mirrored between the two sides to keep the brake system on the inside of the vehicle.

The brake system consists of a modified version of the brake system that was previously used on the F/A 18 E/F Remotely Piloted Vehicle (RPV)[8]. The brake system is a drum brake setup where the drum is the inner surface of the rim. A piston return spring was added to the design to aid in the release of the brakes after they are deactivated.

Figure 2.5: Drum Brakes for the *VIPER* UAV

The final aspect of the main landing gear design was to add a device to reduce the torque at the retraction point under landing loads. This device was deemed necessary in the early stages of the main landing gear development. The device that was implemented to remove the torque was a drag link. The drag Link is also used as the down lock. The drag link consists of a telescoping bar that attaches to the knee on the landing gear and to a forward bulkhead in the UAV. The fully extended main designed landing gear with the drag link is seen in Figure 2.6.

Figure 2.6: Main Landing Gear Complete Design

2.3.2. Structural Analysis

The structural analysis of the main landing gear was comprised of three different load cases. These load cases were the uncompressed shock, fully compressed shock and maximum side loads. Each analysis was conducted in Unigraphics™ within the structural analysis engine. The material that was used for the main landing gear was 7075 Aluminum, which is a strong aluminum for highly loaded structural members. The maximum value for stress that this material can handle before it starts deforming is 73200psi.

The uncompressed shock analysis, Figure 2.7, was conducted for the maximum impact loads. The load case for this analysis corresponds to a 4 g impact at a neutral static margin. The

boundary conditions imposed on the structure were located at the retraction point and the main pivot. The maximum stress was found to be 40,000psi occurring at the neck of the upper bar. The maximum value in this plot corresponds to the pivot between the lower and upper bars. Restricting the torque between the lower and upper bars caused this.

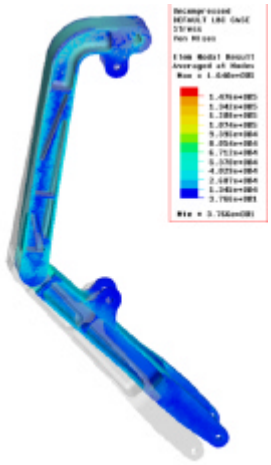


Figure 2.7: Main Gear Analysis - Uncompressed Shock

The compressed shock analysis further verified that the structural design would handle this case, Figure 2.8. The load case for this analysis again corresponds to a 4 g impact at a neutral static margin. The boundary conditions imposed on the structure were located at the main pivot. The maximum stress was found to be 47,000psi occurring at the neck of the upper bar. This stress value was expected to be higher than the uncompressed shock because the axle is further away from the pivot, which produces a larger moment arm. The maximum value in this plot also corresponds to the pivot between the lower and upper bars. This is caused by trying to achieve the worst-case scenario.

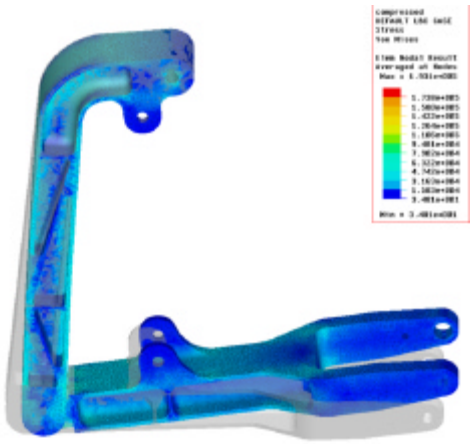


Figure 2.8: Main Gear Analysis - Fully Compressed Shock

The side load analysis completed the structural analysis of the main gear design, Figure 2.9. The load case for this analysis corresponds to the maximum side load that the tire can hold before it looses traction. This value turned out to be 34 pounds of side force. The boundary conditions imposed on the structure were located at the main pivot. The maximum stress was found to be 43,500psi occurring at the neck of the upper bar.

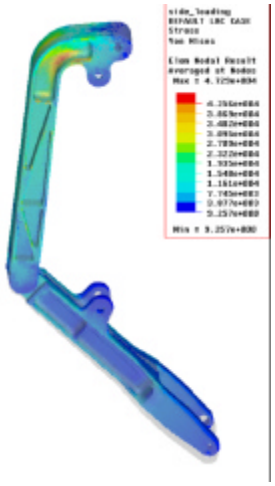
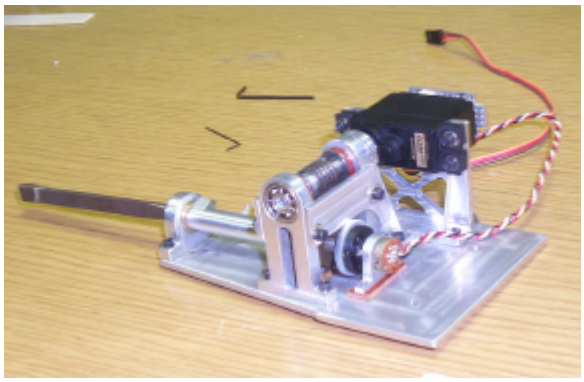


Figure 2.9: Main Gear Analysis - Side Loading

2.3.3. Retraction

The retraction mechanism that was implemented on the *VIPER* UAV is a worm drive setup. The mechanism is a modified version from the F/A 18 E/F RPV leading edge slat mechanism[8]. The servo was repositioned to provide a smaller mounting plate. Additional shaft supports were added to the open side of the worm gear. The attachment of the travel monitor, variable resistor, is now connected directly to the rotating shaft. The original slat mechanism and the modified worm gear retraction mechanism are seen in Figure 2.10.



Slat Mechanism **Modified Retraction Mechanism**

Figure 2.10: Worm Gear Retraction Mechanisms

2.4. NOSE GEAR

The nose landing gear was located as close to the nose of the vehicle as possible. When the nose gear is retracted the turbojet inlet and the cooling ducts are in close proximity. The nose gear needs to handle impact loads, be maneuverable on the ground and be retracted into the available volume. The volume at the nose of the UAV is limited since it is a slim nosed vehicle.

2.4.1. Design

The design of the nose gear started with a kinematics design. Since the nose gear was positioned close to the nose of *VIPER*, the strut had to retract rearward. The strut of the nose gear is canted forward seven degrees, which promotes maneuverability on the ground. The nose wheel axle is located slightly behind the centerline of the strut to reduce the risk of nose wheel oscillations. The over center lock is the main aspect of the rearward retracting nose gear design. The upper surface of the UAV is the main constraint on the design of the nose gear because the turbojet inlet is located on the upper surface behind the nose of the UAV. The over center lock comprised the kinematics design, Figure 2.11. The over center bar has to lock the gear in the down position as well as rotate to almost vertical. When the lock bar is vertical it holds the weight of the strut without transferring high loads to the retraction mechanism.

Figure 2.11: Kinematics Design of the Nose Landing Gear

After the kinematics design was completed the structural design commenced. A cage design was implemented to promote a complete removable unit for the nose gear. This design houses all of the pivots for the nose gear, which include the pivot of the strut and the lock bar. The retraction mechanism can also be mounted to the cage. The only aspect of the nose gear that was not attached to the cage was the steering servo because of the limited volume.

The cage design is a lightweight truss structure with the pivots located at the cross members, Figure 2.12.

Figure 2.12: Nose Gear Cage Design

The absorption of impact loads was accomplished by utilizing a spring within the strut. The spring absorbs the impact, but it does produce oscillations. A series of o-rings were located at all of the joints to help increase the dampening. The o-rings allow the air inside the strut to escape slowly which acts as a shock.

The nose gear steering was complicated because of the retraction. The steering was accomplished by actuating the lower part of the strut. The steering rod utilizes a keyed shaft and a keyed rod. The keyed rod and shaft are able to turn while moving up and down with the strut. The shaft is attached to the wheel yoke and is located in the center of the strut. The rod is attached to the top of the strut and it is allowed to turn freely, Figure 2.13. A pull-pull system is used to actuate the nose wheel. The complete design of the nose landing gear system is seen in Figure 2.14.

Figure 2.13: Nose Gear Steering Mechanism

Figure 2.14: Nose Landing Gear Design

2.4.2. Structural Analysis

The main structural members of the nose landing gear system were structurally analyzed. The analysis focused on the strut and the side of the cage design. The material that these parts were manufactured from was 6061-T6 Aluminum. This material is approximately half as strong as the 7075 Aluminum. The nose gear does not require the higher strength properties that the main gear required. The maximum value for stress that this material can handle before it starts deforming is 39900psi.

The structural analysis of the strut consisted of the over center bars and the upper strut with all of the pivots, Figure 2.15. The load case for this analysis corresponds to a 4 g impact at the maximum static margin of eight percent. The boundary conditions were imposed to simulate actual landing conditions. The structure was prevented to move at the main pivots and the over center actuation pivots. The maximum stress was found to be 4,600psi occurring at support structure for the main pivots. This value is well below the ultimate strength of the material, but a static analysis was conducted and higher concentrations might be present under dynamic loading.

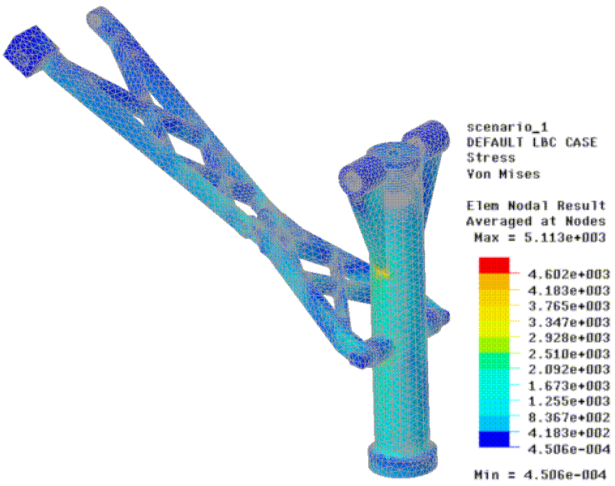


Figure 2.15: Nose Gear Strut and Lock Bar Analysis

The structural analysis of the cage side panel was necessary to validate the structural design of the nose landing gear, Figure 2.16. The load case for this analysis corresponds to a 4 g impact at the maximum static margin of eight percent present at the end of the over center bars. The cage was prevented to move at the mounting location to the *VIPER* UAV mounting plate. The maximum stress was found to be 8,250psi occurring at the structure that forces the load path from the over center pivot mount to the cage frame.

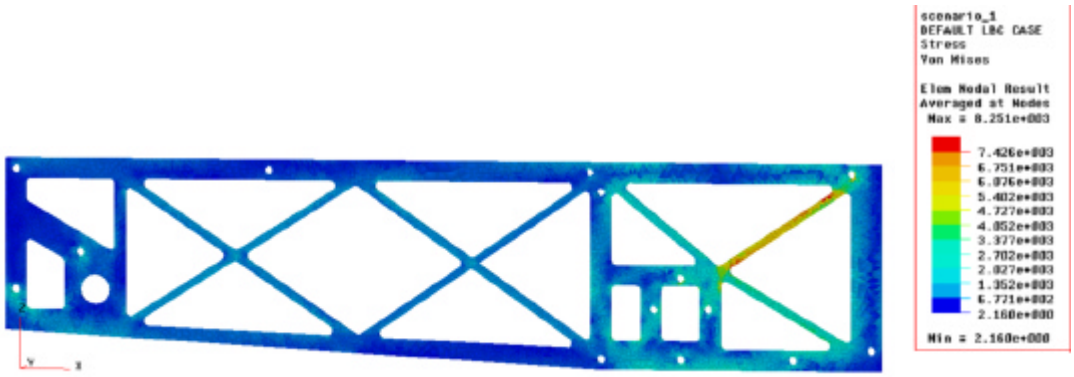


Figure 2.16: Nose Gear Cage Analysis

2.4.3. Retraction

The retraction mechanism for the nose gear is the same mechanism that is utilized for the main landing gear. The mount plate for the mechanism hangs off the right side of the cage with support bracing. The retraction mechanism actuates the rear mount of the over center bar.

2.5. LANDING GEAR CONSTRUCTION

The landing gear was manufactured for the most part in house. At the time of release of the landing gear a CNC machine was not available to our departments machinists. The complicated parts that would have been very difficult to machine by hand were taken to the precision machine shop, in nuclear engineering, to CNC those parts. As the parts were received from manufacturing they were checked for accuracy. This was done to make sure that the parts would not cause problems once each gear system was assembled.

2.5.1. Main Landing Gear

The main landing gear system was assembled as the parts were completed. The brake system was assembled before any component integration commenced. The upper and lower bars were assembled with the shock before they were joined to the retraction mechanism. The retraction mechanism was partially assembled on the main retraction shaft on the upper bar. Then the remaining parts of the retraction mechanism were attached to the mounting plate. When the main gear was a complete unit then the tire and brakes were installed. The last component that was attached to the main gear was the drag link.

The main gears needed to be assembled and tested before they could be installed into the UAV. The retraction mechanism was checked for the amount of torque available. The servo that drives the worm was changed to a more powerful unit to provide enough torque for the retraction of the gear. The original servo produced 88 ounce-inches of torque at the servo, which resulted in 984 ounce-inches at the retraction shaft. The new more powerful servo produces 168 ounce-inches of torque at the servo and 1879 ounce-inches of torque on the retraction shaft. This servo produces more than the required torque of 1250 ounce-inches, which corresponds to the amount of torque that was calculated to overcome the aerodynamic loads at 1.5 times the landing velocity. This means that the new servo has some reserve torque. During the assembly of retraction mechanism, it was found that the tolerance between the gears was too small therefore shims were added to produce the correct tolerance between the gears.

The complete main landing gear unit has been drop tested to validate the design, Figure 2.17. The drop tests were performed by an undergraduate student, Ben Lilly, under the supervision of Wallis Collie and myself. The shock was tailored to the landing loads. The air pressure inside the shock was adjusted to produce the necessary amount of spring force. The absorption and rebound rates were also tailored to the landing loads. The drop test results can be seen in Figure 2.18.

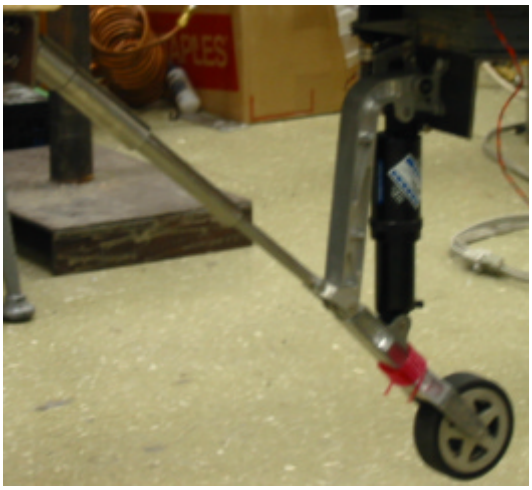


Figure 2.17: Main Landing Gear of the *VIPER* UAV

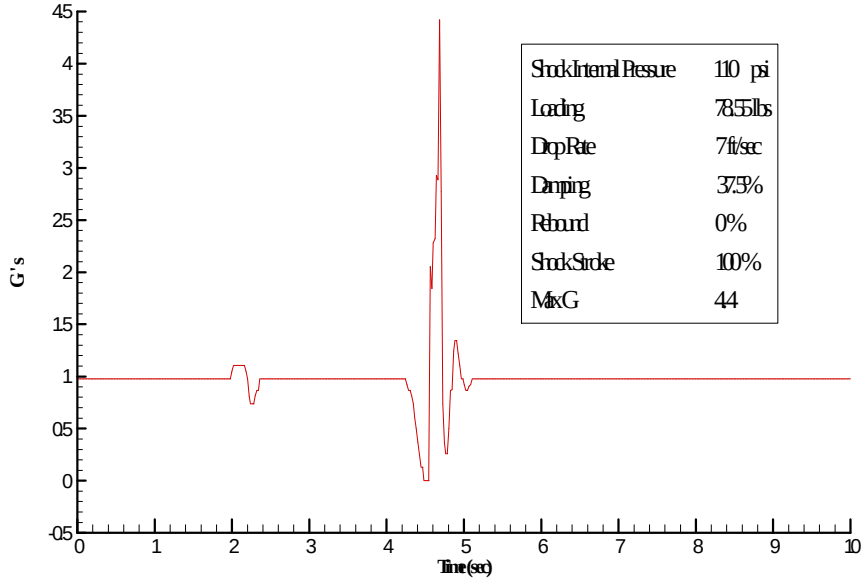


Figure 2.18: Drop Test Results for the Main Landing Gear

2.5.2. Nose Landing Gear

The nose gear assembly is complicated because of the design of the retraction scheme. The strut needed to be assembled before the cage because the pivots are attached to the strut before it is installed into the cage. After the strut is assembled the cage is assembled around

it. The over center lock bars can be installed after that. The retraction mechanism is easiest to assemble on the mounting plate itself first. Once the mechanism and the cage are assembled they can then be integrated together. The retraction pivot is keyed so that the shaft will only go in one way. The fully assembled nose gear can be seen in Figure 2.19.

Figure 2.19: Nose Landing Gear of the *VIPER* UAV

3. AIRFRAME DESIGN

3.1. AERODYNAMIC MODEL

Aerodynamics contributes directly to the structural design of the *VIPER* UAV. The load patterns and control surface moments are obtained from the aerodynamics model. Ultimately the pressure distribution of the aerodynamic analysis was applied to the skin of the UAV during the structural analysis phase. An aerodynamic model of the UAV was constructed and analyzed using CMARC, which is a low order inviscid panel method code. The aerodynamic model was constructed from the Unigraphics™ model of *VIPER*. The CMARC geometry of *VIPER* is shown in Figure 3.1.

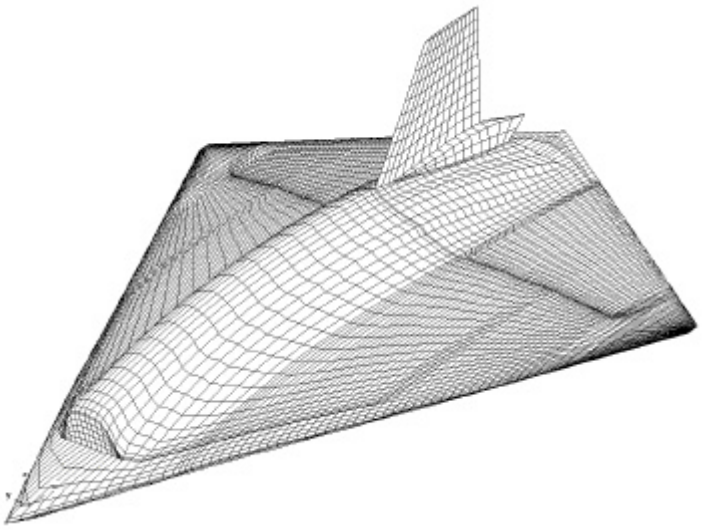


Figure 3.1: Aero Geometry of the *VIPER* UAV

The aerodynamic model was run at several angles of attack (AOA) to determine the validity of the analysis. The lift curve was generated from the various AOA runs and compared to a NASA/Langley wind tunnel model of the full scale UAV, Figure 3.2. The result was that the inviscid model adequately correlated with the wind tunnel data up to four degrees angle of

attack. After the four degrees AOA vortex lift becomes a large factor and CMARC cannot handle viscous effects.

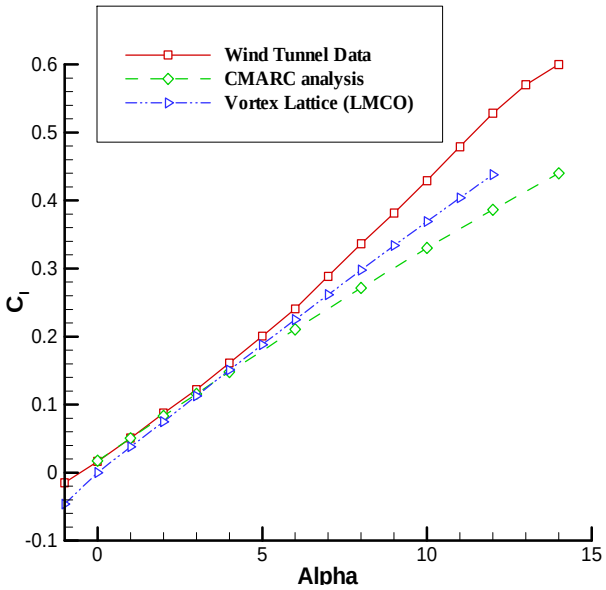


Figure 3.2: Lift Curve of *VIPER* UAV Versus Wind Tunnel Data

The pressure distribution that was utilized during the design and analysis phase corresponded to four degrees AOA, Figure 3.3. The reasons for using four degrees AOA were that it occurs at the end of the linear region of the lift curve, and that the pressure distribution is the furthest outboard on the wings. Using the outboard loaded pressure distribution ensures that the design of the wings will handle the maximum conditions.

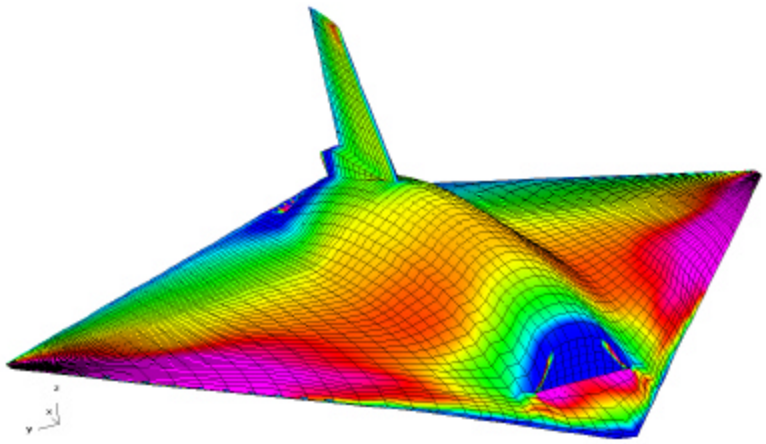


Figure 3.3: *VIPER* UAV Pressure Distribution at Four Degrees AOA

3.2. MATERIALS

The materials that were available for the strength of the VIPER UAV were Fiberglass, Kevlar® and Carbon fiber. Each of the materials exhibits their own strengths and weaknesses. Fiberglass is a lightweight material and it is stronger than Kevlar® when loaded in the tensile direction. Fiberglass also exhibits excellent bending properties, but under large bending it starts to crack. Kevlar® is slightly weaker than fiberglass in the tensile directions. Its bending properties are slightly stronger, and it does not incur cracks under large bending. Carbon fiber is stronger than both Kevlar® and fiberglass in tensile direction yet it is the most brittle in the bending properties.

The available core materials for the construction of the *VIPER* UAV were Korex® and Rohacell®. Korex® is a honeycomb material that is made from Kevlar® pulp. The honeycomb structure is very resistant to collapsing in the direction of the honeycomb extrusion. In the direction across the extrusion, the structure flexes easily. Rohacell® is a closed cell structural foam. This foam resists deformation from bending loads. Further analysis is needed to provide some insight into which core material is more suited for the UAV.

The *VIPER* UAV was designed to utilize composites in the sandwich configuration for their excellent strength versus weight. A typical sandwich configuration is composed of the outer layers, called face sheets, and inner layers, called the core, Figure 3.4. The structures of the UAV are comprised of the skins, internal structures and the hardpoints. Each of the components varied from location to location on the UAV depending on the requirements of the particular area.

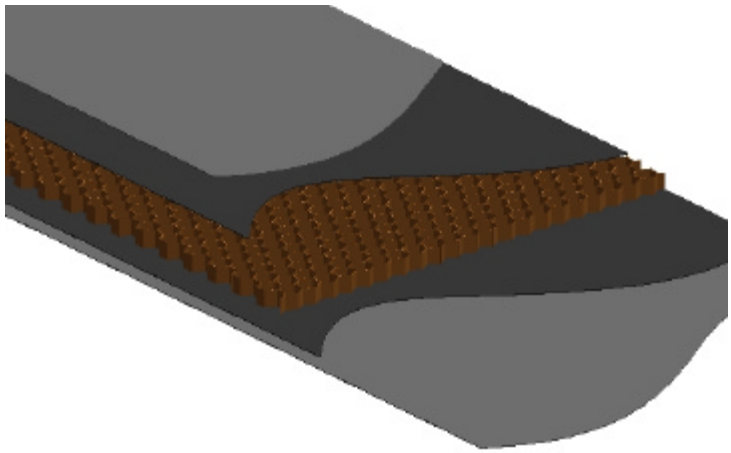


Figure 3.4: Typical Sandwich Layout

The majority of the UAV is composed of sandwich materials. The skin and internal structures on the UAV have to be lightweight and strong to cope with the flight loads. The composition of the skin varies from the hatches to the vertical tail to the rest of the vehicle because of the loads that it has to carry. The hatches need to be relatively stiff since they are usually placed on relatively flat areas. They are designed with the use of carbon fiber layers with Korex® and Rohacell® as the core. The Rohacell® is used for the edges of the hatches to give a durable edge.

The use of carbon fiber for the skin of past UAV's at NC State has posed interference problems with the avionics system. The receiver for the control system is located in the vertical and the antenna of the receiver is partially located inside the vertical of the UAV. Therefore the tail structures consist of conventional aircraft plywood. The skin of the vertical for *VIPER* will not consist of carbon fiber. It was decided to use different sizes of fiberglass for the vertical skin instead of Kevlar®. The main reason for utilizing fiberglass instead of Kevlar® was because Kevlar® is difficult to work with.

The last main concern for materials is the mounting locations. When an object is bolted to the structures of a sandwich material the core material needs to be replaced in that area to accommodate the bolt and the transfer of loads. The material that replaces the core material is

called a hardpoint. Aluminum and layered carbon fiber were the prospective candidates for hardpoint materials. Each material would be able to handle the transfer of loads from the bolt to the composites. The driving force of which material to use for the hardpoint was how good of a bond it would have to the face sheets. The solid carbon fiber hardpoint would have superior bonding properties since it is itself a composite material.

3.3. AIRFRAME DESIGN

The design of *VIPER* UAV started with placing the key systems and their associated structures. Then the focus was towards creating a load path between the adjoining structures. The final aspect of the airframe design was placing the access hatches.

The key systems that needed to be placed in certain locations of the UAV are the turbojet, inlet system, exhaust pipe, landing gear, control surface system and the avionics. These systems were placed into the UAV and then structures were placed where the mounting locations are, Figure 3.5.

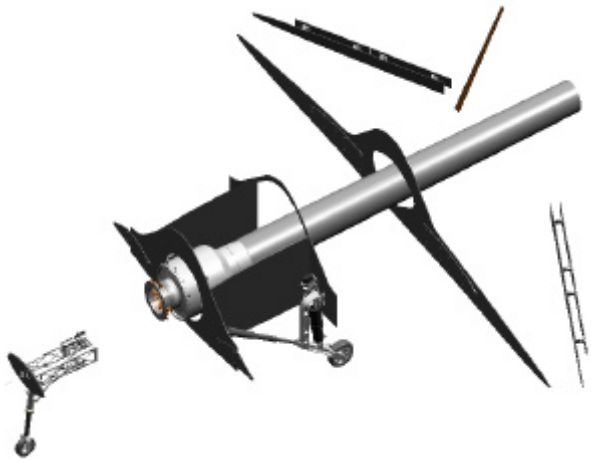


Figure 3.5: Structures for the Main Systems of the *VIPER* UAV

3.3.1. Main Center Section

Because of the limited volume available for the avionics and flight systems the structures for tying the nose landing gear into the rest of the main section were placed along the leading edge of the UAV. By examining the shape of the UAV the leading edge forms a D-shaped beam to tie the nose gear mount to the rest of the structures. The D-shaped beams allowed the nose section to have a large compartment. This compartment is located in front of the turbine, which makes it a prime location for avionics. This structural design provides ample room for moving the avionics and their sub-systems around to balance the UAV.

The rear bulkhead is used for two connection points. The rear tab of the wing bolts into this bulkhead. The tail section also connects to this bulkhead. Since there are heavy loads being transmitted to this bulkhead two semi-ribs were placed to transfer the loads forward. These ribs transfer the loads from the rear forward and from the nose gear structures rearwards towards the strong point of the internal structures of the main center section, Figure 3.6.

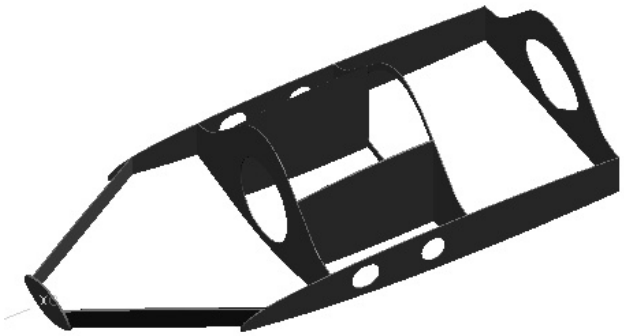


Figure 3.6: Main Center Section Structural Design

3.3.2. Rear Center Section

The rear main section of the *VIPER* UAV consists of load carrying members for the vertical and the rear section. Each section is considered separate because of the differences in loads that each has to handle. The structures in the vertical are composed of aircraft plywood.

These structures are two spars for carrying the aerodynamic loads. The rear vertical spar is also used to connect the control surface loads to the vertical structure. The rudder attaches to this spar. The rudder houses a corresponding spar for hinge mounting.

To handle the loads in the main rear section two bulkheads and two partial ribs are utilized. The front bulkhead is used to connect the rear and main center sections together. The rear bulkhead was placed in the design to help transfer the loads from the wing hinge line across the entire UAV. The partial ribs are located in front of the rear bulkhead to connect the wings to the rear section.

To connect the vertical and the rear section the loads seen by the vertical were examined. The side loading on the vertical is transmitted thru the vertical spars to the base of the vertical. The structural connection between the vertical and the rear section are overlapping structures and the combined skin. A third bulkhead was added to the internal design to connect the rear vertical rear spar into the main section. The front spar was connected to the front bulkhead. The structures for the rear center section are seen in Figure 3.7.

Figure 3.7: Structural Design for the Rear Center Section

3.3.3. Wing Panels

The design of the internal structures of the wings is to handle aerodynamic loads, control surface loads and attachment stresses. Each poses their unique problems. The main aspect of the wing's structural design is to transfer the high loads forward. The reason for this is

because more than two thirds of the surface area of the wing is located behind the center of gravity. When the loads are transferred forward the heavier structures are located near or slightly in front of the center of gravity.

From the pressure distribution of the *VIPER* UAV the main load is located near the leading edge of the wing. A spar was placed parallel to the leading edge to handle the main aerodynamic load. A series of partial ribs extend from the main spar to the leading edge to control the amount of distortion at the leading edge. The interaction point between the main spar and the center section is the location of the front attachment point.

The control surface loads are transferred in several directions. The elevon mounting structure attaches to the spar and to the joint between the wing and the rear center section. Ultimately a structure was added that spans from the split in the elevons to the tab connection location. This structure adds support to the center of the elevon mounts and reduces the twisting caused by the elevon loads.

The remaining structures of the wing are for transferring loads to the center section. The front and rear traversing structures are located where the wing connects to the center section. These two structures hold the tab connections between the wing and the center section. The middle structure is used to transmit loads to the main landing gear mounting bulkhead by surface contact. The structures for the wing panels are seen in Figure 3.8.

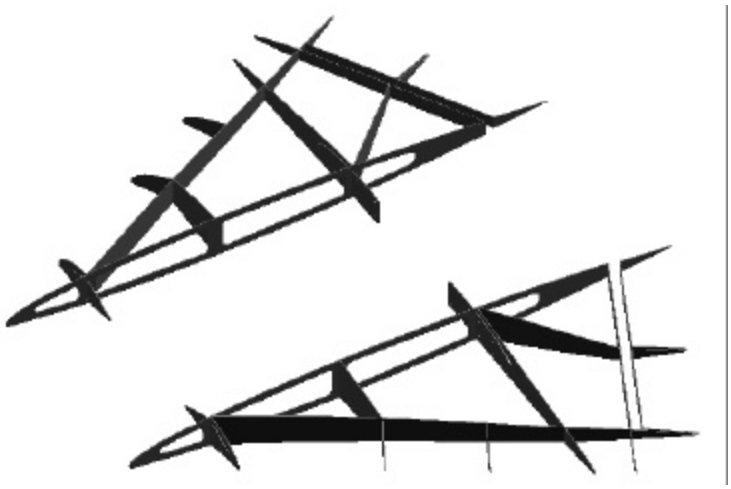


Figure 3.8: Structural Design of the Wing Panels

3.3.4. Hatches

Hatches are a critical aspect of a structural design. Hatches are designed for access to critical systems within the UAV. The hatches were placed in areas where the load paths would not be compromised by a hatch. The corners of the hatches are round to reduce the stress concentration at the corners of the skin. The hatch for the fuel tank was required to be very large and rectangular. Most of the loads that are encountered around this hatch are from the aerodynamic forces produced by the wings.

The main center section contains five hatches. There are four landing gear doors also located on the lower surface of the main center section. Two hatches are located on either side of the inlet. These hatches are utilized for access to the inlet, avionics, batteries and the front wing tabs. The round hatch on the top of the UAV is for accessing the turbojet, switches and data ports. The large rectangular hatch is for installing the fuel tank and accessing the main landing gear mounts along with the wing and tail attachment points. The final hatch in the main section is located in front of the nose gear on the lower surface. This hatch is used for accessing the pitot-static tube and the steering mechanism. This hatch was placed on the lower surface for safety reasons. The rear main section has two hatches. These hatches are

for accessing the bolts that attach the tail section to each wing. Access to the rudder actuation is accomplished when the section is removed from the rest of the UAV. Each wing houses two hatches located on the lower surface. These hatches are for accessing the control surface actuation and hinges. The *VIPER* UAV hatches layout can be seen in Figure 3.9.

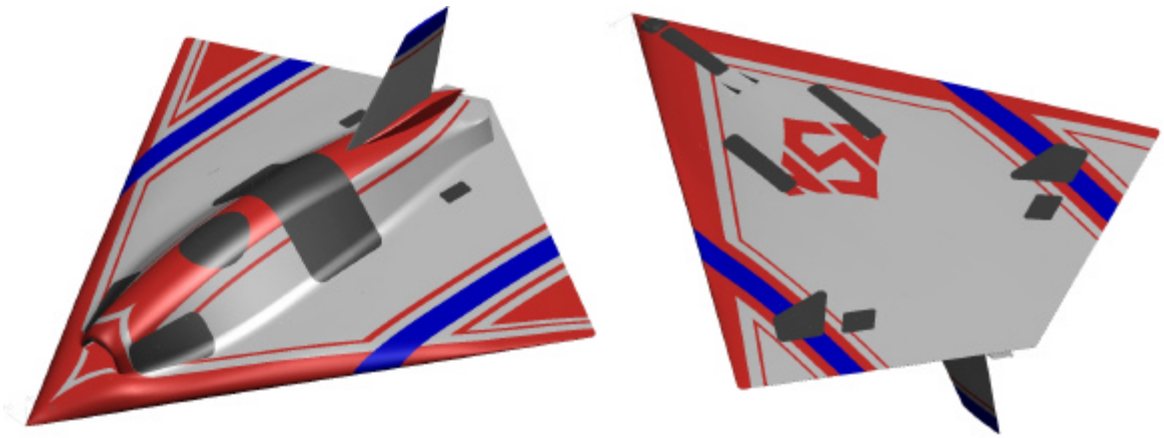


Figure 3.9: Design Layout of the Access Hatches

3.4. DESIGN CONCLUSIONS

Concurrent Engineering provided insight into the entire UAV design, analysis and manufacturing. The key areas were the landing gear system, a lightweight open structural design and the integration of the avionics and propulsion system.

The aerodynamic model provided insight into where the major loads are. This data formed the design of the internal structures. The aerodynamic model was utilized in the finite element analysis phase. Using the aerodynamic pressure loads, landing loads, propulsion system integration and the avionics requirements an initial structural design was completed. Lightening holes were not applicable to this design since all of the structures are heavily loaded except the rib between the wing panel and the center section. This rib was hollowed on the wing side for access to the internals of the wing. This allowed for verification and analysis of the design.

4. MATERIAL ANALYSIS

4.1. MATERIAL TESTING

The material testing phase was critical to the overall design of the UAV. It was necessary before any analysis could commence. Ideal numbers were obtained from using manufacturers published numbers. The drawback of utilizing these numbers was that they are for the material itself, and this UAV is composed of composite sandwiches of multiple materials. Extensive testing was conducted to obtain more accurate material properties for the structural analysis phase.

4.1.1. Calculation of Material Properties

The calculated sandwich material properties gave insight into how the sandwich will act when it is constructed. This approach is idealized because the sandwich is held together with epoxy resin which makes calculating the material properties difficult. The epoxy adds to the physical dimensions but it takes away strength of the material.

A component build-up technique[9] was used to create a baseline for the composite sandwich material properties. The material properties of each composite were obtained from each individual manufacturer. Table 4.1 lists all of the materials that were investigated.

Table 4.1: Individual composite properties

Core Material	Tensile Strength (psi)	Modulus of Elasticity (psi) $x10^3$	Shear Modulus (psi) $x10^4$	Poison's Ratio
Korex	225	7.00	1.70	1.06
Rohacell 51A	275	9.95	0.27	17.43

Face-Sheet Material	Tensile Strength (psi) $x10^5$	Modulus of Elasticity (psi) $x10^7$	Shear Modulus (psi) $x10^7$	Poison's Ratio
Carbon	5.29	3.50	1.40	0.25
Fiberglass	4.80	0.78	0.32	0.22
Kevlar 49	4.35	1.90	0.70	0.36

The equation for the composite build-up method for the modulus of elasticity, Eq. (4.1), uses volume fractions of each individual component of the sandwich. The tensile strength, shear modulus and poisson's ratio were calculated in the same manner. The results of the component build-up method are listed in Table 4.2.

$$E \; ? \; E_{core}V_{core} \; ? \; E_{Material}V_{Mateial} \tag{4.1}$$

Table 4.2: Composite sandwich build-up

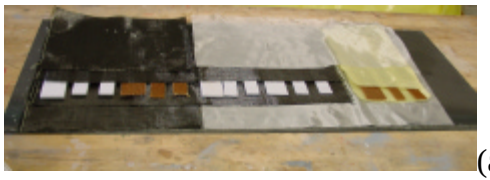
Composite Sandwich	Tensile Strength (psi) x10^4	Modulus of Elasticity (psi) x10^6	Shear Modulus (psi) x10^6	Poison's Ratio
1/8 C/C Korex	9.84	6.50	2.61	0.24
1/8 C/G Korex	7.12	3.76	1.52	0.24
1/8 K/G Korex	4.45	1.39	0.54	0.29
1/8 C/C Rohacell	9.84	6.51	2.60	0.25
1/8 C/G Rohacell	7.13	3.76	1.50	0.25
1/8 K/G Rohacell	4.45	1.40	0.53	0.33
1/4 C/C Rohacell	5.44	3.59	1.44	0.25
1/4 C/G Rohacell	3.84	2.02	0.81	0.25
1/4 K/G Rohacell	2.35	0.74	0.28	0.33

4.1.2. Composite Coupon Testing

To obtain the physical properties of the materials that were to be used in the analysis of the *VIPER* UAV, a series of tests were conducted. Coupons of various candidate sandwich configurations were tested based on previously constructed UAVs at NC State. The first step was to create the coupons for the testing. The sizes of the coupons and their testing techniques were obtained from two ASME reports[10,11]. The overall dimensions of the tested coupons were ten inches long by one inch wide. The last inch of each coupon's core was filled with an epoxy mixture to reduce the risk of failure at the mounting location.

The construction of the coupons consisted of a traditional wet lay-up method, which uses epoxy as the bonding agent between each layer. The construction of the coupons is seen in Figure 4.1. The general layout of the coupons consists of two face sheets on either side of a core material. The face sheets consist of different combinations of fiberglass, carbon fiber and Kevlar®. The core materials are Korex and two different thickness of Rohacell 51A.

Korex is a honeycomb material where the Rohacell 51A is a structural foam of the closed cell type.


(a)


(b)


(c)

Figure 4.1: Coupon Construction: (a) Layout, (b) Lay-up
(c) Coupons with Strain Gauges

The sandwich coupons were tested in an INSTRON machine at NC State. The INSTRON machine was used to conduct tensile tests. A typical setup of the INSTRON machine is seen in Figure 4.2. A data acquisition system was used to measure the loads applied to the coupons and the strains that were generated. Before the composite materials were tested a baseline calibration was performed. This calibration used a sample of known aluminum to verify the data acquired from INSTRON machine. Then the composite sandwiches were tested. These tests produced data for the entire sandwich on the face sheets.

Figure 4.2: INSTRON machine

The material properties thru the core were obtained from bending tests. These material properties were required to model the composites within the finite element solving software. The bending test setup consisted of hanging weights from the cantilevered sandwich, Figure 4.3. As the weight was applied the deflections were measured. After the bending test the material properties of the sandwich were calculate with two equations are Eq. (4.2) and Eq. (4.3).

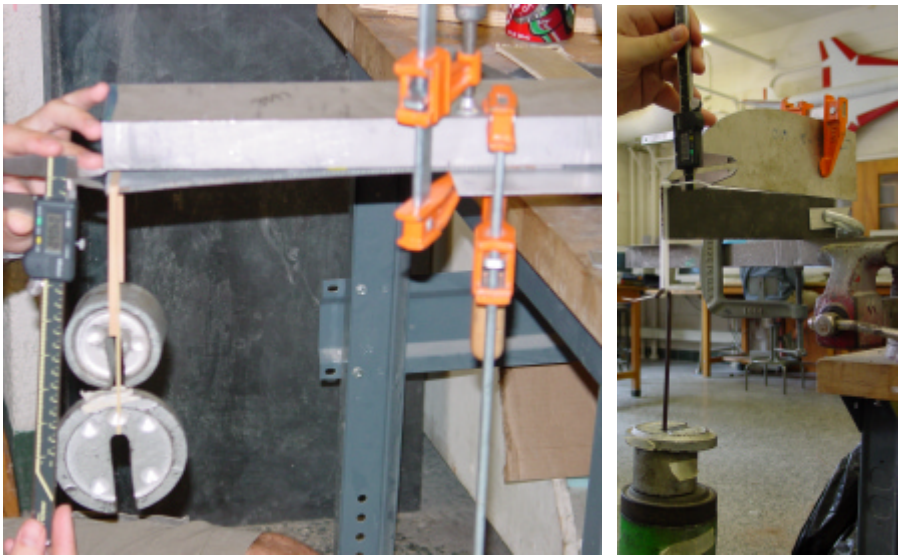


Figure 4.3: Composite Bending Test

$$? \; ? \; \frac{P_{(load)} l^3}{3EI} \tag{4.2}$$

$$I \; ? \; \frac{bh^3}{12} \tag{4.3}$$

During the testing of the sandwich coupons it was determined that the properties of carbon fiber and fiberglass were needed. These material properties improved the results of the overall sandwich materials by providing the material properties of the faces sheets. From these material properties, the core properties could be obtained by using the composite build-up equation. Kevlar® was excluded from the remainder of the testing due to its inadequate

ability to handle the necessary loads. Coupons of solid carbon fiber and fiberglass were constructed and tested in the same manner as the sandwich coupons. The results of all of the tests are tabulated in Table 4.3, and the difference between the calculated and tested values are tabulated in Table 4.4.

Table 4.3: Tabulated Data from Coupon Testing

Material	Modulus of Elasticity (psi) $x10^6$	Shear Modulus (psi) $x10^6$	Poison's Ratio
Carbon	8.68	3.88	0.12
Fiberglass	2.76	1.22	0.12
1/8 C/C Korex	1.77	0.76	0.17
1/8 C/G Korex	1.04	0.48	0.09
1/8 K/G Korex	0.33	0.14	0.15
1/8 C/C Rohacell	1.61	0.69	0.17
1/8 C/G Rohacell	0.86	0.40	0.09
1/4 C/C Rohacell	1.22	0.53	0.14
1/4 C/G Rohacell	0.56	0.25	0.12

Table 4.4: Coupon Properties Relative to Calculated Properties

Material	Modulus of Elasticity	Shear Modulus	Poison's Ratio
Carbon	75.2%	72.3%	52.0%
Fiberglass	64.6%	61.8%	45.5%
1/8 C/C Korex	72.8%	70.9%	30.5%
1/8 C/G Korex	72.3%	68.4%	62.1%
1/8 C/C Rohacell	75.3%	73.5%	32.1%
1/8 C/G Rohacell	77.1%	73.4%	63.8%
1/4 C/C Rohacell	66.0%	63.1%	44.2%
1/4 C/G Rohacell	72.3%	69.1%	51.9%

4.1.3. Analysis of Coupon Data

The material properties were verified before the structural analysis of the UAV commenced. The ANSYS was used to analyze the coupons. The error ranged from three to fifteen percent between the experimental results and ANSYS. The higher percent errors occurred on the sandwich materials with Rohacell[?] 51A as the core material. The high percent error was due to the fact that those test articles continually deformed during the testing procedures.

4.2. MATERIAL SELECTION

The core material for the entire *VIPER* UAV is Korex? . Korex? exhibited superior bonding properties over Rohacell? during the material testing. Rohacell? was also observed deforming under relatively low loads. Rohacell? was sparingly used in the corners of the hatches. Rohacell® was used for the edges of the hatches because it is easily beveled.

The majority of the skin of the UAV consists of an outer layer of fiberglass then carbon fiber before the core material. The materials on the other side of the core are carbon fiber then lightweight fiberglass. There are two reasons for using lightweight fiberglass for the outer layer of the skin. The main reason is that when the skins are joined together the resulting bodywork will not damage the structural layer of carbon fiber. The lightweight fiberglass also helps seal the surface from the coarser weave of the heavier structural materials underneath. Carbon fiber was used for the strength layer because it exhibited superior strength over fiberglass and Kevlar? .

The skin of the vertical tail is composed of different weights of fiberglass and Korex? . The outer layer is lightweight fiberglass for the fine weave then there is a heavier structural layer before the core material. The inside of this skin is composed of structural layers of fiberglass. Carbon fiber was not utilized in the vertical tail for the main reason of possible RF blanking. The radio equipment will be located in the tail and carbon might interfere with the signals.

The internal structures will consist solely of carbon fiber layers with a core of Korex? . At the mounting locations, hardpoints of carbon fiber will be used. Carbon fiber hardpoints bond well to the carbon fiber layers of the internals. A strong bond between the hardpoints and the carbon fiber layers is necessary to reduce the risk of separation between the two.

5. AIRFRAME ANALYSIS PHASE

5.1. PRE-FINITE ELEMENT ANALYSIS

The finite element analysis started with the creation of the structural mesh. Certain care was taken to make sure that the finite element geometry would produce reasonable results. The mesh shape and size were refined to produce valuable results. After the finite element mesh was created it was merged with the aerodynamic data to create a pressure file to apply to the surface of the mesh. The final step prior to finite element analysis was to add the material properties and internal loads to the structural mesh input file. After all of these steps were completed the finite element analysis could commence.

5.1.1. Finite Element Analysis Approach

The structural analysis commenced with an initial wing model that was created utilizing past experience and the results from the material testing. The finite element mesh was generated for the skin and internal. Then the material properties were applied to the finite element mesh. The initial wing finite element model was solved and then analyzed. This analysis was used to promote changes to the structural design of the entire UAV. This approach was implemented because the wing only has aerodynamic loads without any loads from other structures.

The second structural analysis commenced after the design was refined. The wing was again analyzed first. The results from the wing analysis were two fold. The analysis results verified the design of the wing and the reaction forces were obtained from the results. These forces were transferred to the main center section. The main center section was analyzed under aerodynamic and landing loads. The weight of the landing gear, turbojet and the fuel tank were applied to the model under the aerodynamic load case. These weights were scaled

according to the g loading run being analyzed. A 1g aerodynamic load was applied to the mesh to simulate the aerodynamic loads under landing conditions. Impact loads were also applied to simulate the internal components. These loads were applied to the structural mesh at their physical location. From the results of the second analysis iteration analysis modifications were made. The final modifications were to model the final weight of the UAV, the addition of the final hatch to the wing model and further refine the mesh.

5.1.2. Finite Element Selection

During the testing and analysis of the proposed materials, it was determined that the Shell 99 element should be implemented for the analysis of the *VIPER* UAV. This element is a composite layered element, which requires the material properties of each different material. The problem with this element is that it is difficult to accurately determine the direction of the weave. For this issue it was decided to orient all of the layers in the same direction. This would result in the weakest composite. Even though this approach should produce the weakest model, the results from the material testing and analysis showed that this approach correlated well between the experimental and analytical results. The Shell 99 element type offers another advantage. Since most of the preprocessing is being done outside of ANSYS, the layers can be easily modified and then re-analyzed within ANSYS.

5.1.3. Finite Element Mesh Geometry

The ANSYS analysis program utilizes a mesh or grid to analyze the structure. The mesh was created within Unigraphics™ to ease the integration with the aerodynamics. Since all of the UAV modeling was done within Unigraphics™, changes in the structural model and mesh could be made quickly. The structural environment within Unigraphics™ was set to ANSYS. This setting makes Unigraphics™ create ANSYS compliant meshes. The geometry for the Shell 99 element needs to be quadrilaterals composed of eight key points called nodes. The

nodes occur at each corner and each mid-side. The size of the mesh grid directly correlates to the accuracy of the results and was set to a maximum size of ¼ inch.

When the finite element mesh geometry was created it was critical to make sure that the mesh between adjoining panels matched. If there were gaps between the adjoining surfaces the nodes would not coincide on adjoining panels, which ultimately results in the panels separating during the analysis. ANSYS can merge nodes that are located close, but it can create problems with the imported aerodynamic data. This ability to merge nodes was carefully used between the skin and the internals of the main center section. Other key areas of the finite element mesh geometry were the leading edges and corners of hatches. The mesh in these areas needed to be uniform and not skewed.

5.1.4. Finite Element Material Properties

All of the material properties were entered into the ANSYS input file. The materials could be ordered in any configuration to simulate the structures of the UAV. The skin uses fiberglass, carbon fiber and Korex? while the internals only utilize carbon fiber and Korex?. The material properties that were entered into the ANSYS input file can be seen in Appendix 10.2.

5.1.5. Finite Element Model Loading

5.1.5.1. Boundary Conditions

The boundary conditions are used to restrain the model during analysis. The main boundary conditions for the wing were the tab locations. The nodes at the locations of the wing tabs were restrained from movement in all directions and rotation. The upper and lower skins were restrained from moving in the transverse direction. This was done to simulate the

interaction with the center section. The locked boundary conditions appear as a gold color in Figure 5.1, while the transverse boundary conditions appear as an aqua color.

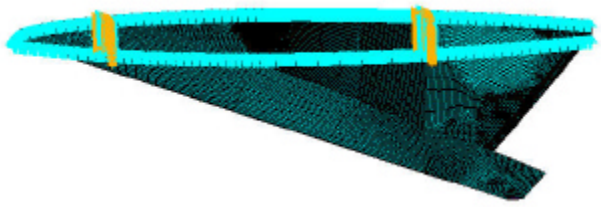


Figure 5.1: Boundary Conditions on the Wing

The boundary conditions on the center section are slightly different than the wing. The boundary conditions were utilized to simulate the opposite side of the center section. Since half of the section was modeled, the boundary conditions were applied to the centerline of the UAV. A minimum of one point had to be locked so that the model has a reference point for ANSYS. The bulkheads at the centerline of the model were restricted from movement to simulate the structures continuing to the other side. The upper and lower skins at the centerline were restrained from moving in the transverse direction, Figure 5.2.

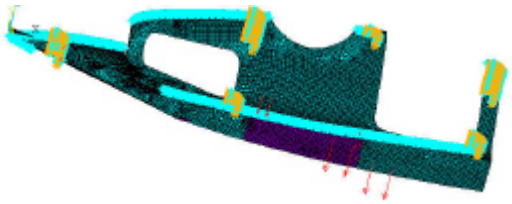


Figure 5.2: Boundary Conditions on the Center Section

5.1.5.2. Aerodynamic Loads

After the structural mesh was created the aerodynamic forces were applied to the mesh. A FORTRAN program, Appendix 10.3, was implemented to merge the aerodynamic pressure file onto the structural mesh. The program uses a gauss area interpolation to assign the coefficient of pressure from the aerodynamic nodes to the finite element nodes. Then the coefficient of pressure is converted to a pressure load. The coefficient of pressure is

converted by using physical aspects of the *VIPER* UAV; weight, surface area, lift coefficient, and the number of g's to analyze. The equation that is used to transform the coefficient of pressure to a load is Eq (5.1). A pressure plot of the merged aerodynamic data on the center section with a 9g load applied is seen in Figure 5.3.

$$L ? qSC_L$$
$$L ? mg$$
$$P ? qC_p$$
$$P ? \frac{mg}{SC_l}C_p \qquad (5.1)$$

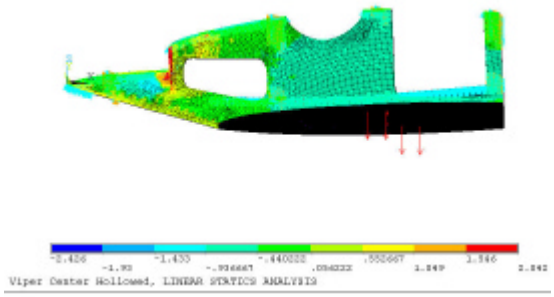


Figure 5.3: Pressure Load on Center Section at 9g's

5.1.5.3. Landing Loads

The final type of load that was simulated in the structural analysis was the landing load. The thrust loads during landing were not modeled because it is low compared to the gear impact loads. The landing loads represent the major components of the UAV. The landing loads of these components were added to the internal structures. The major components were the turbojet, fuel tank and the landing gear reaction forces. These loads correspond to the 4g impact loads imposed by each component are depicted as red arrows in Figure 5.4. Aerodynamic loads corresponding to +1g were also present on the landing load analysis.

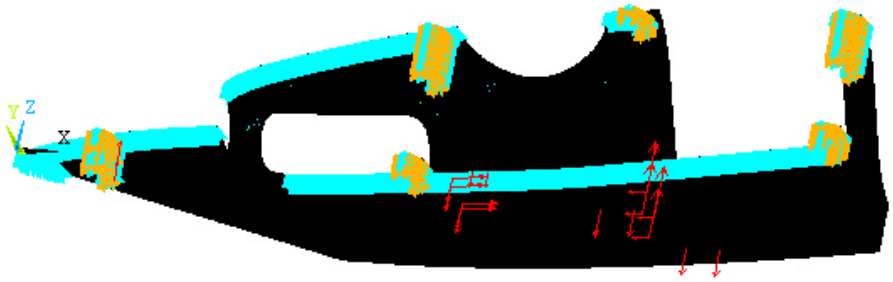


Figure 5.4: Landing Loads on Center Section

5.2. FINITE ELEMENT ANALYSIS

The finite element analysis consisted of various analyses to optimize the design. This initial run was used to determine the material makeup of the internal structures. The skins at this point were already laid up for various reasons. From the material testing it was determined that the skins would be adequately strong if they were composed of one layer of carbon fiber one either side of a core of Korex? with an outermost layer of fiberglass. This configuration is identical to a coupon that was tested during the material testing phase. The internal structures were configured with one layer of carbon fiber on either side of a core of Korex? . After the initial finite element model was analyzed, a series of modifications were implemented and re-analyzed. The final analysis was conducted after all of the final component weights were known.

The maximum strain was set to 3.5e-3 to produce a flexible airframe for future research opportunities. This value might seem low, but the material testing revealed that Korex? sandwich coupons had a maximum strain of 4.8e-3. This value was obtained from the two layers of carbon fiber Korex? coupon. The lower maximum value of strain was imposed to reduce the risk of failure due to non-ideal joint construction.

5.2.1. Initial Finite Element Analysis

The initial iteration of the structural model consisted of the wing analysis with only aerodynamic loads. These loads were applied to the surface of the wing and the results from the ANSYS analysis was analyzed. The wing's deflection, strain and strain vector plots were analyzed to promote changes of the structural design, Figure 5.5 thru Figure 5.7 respectively.

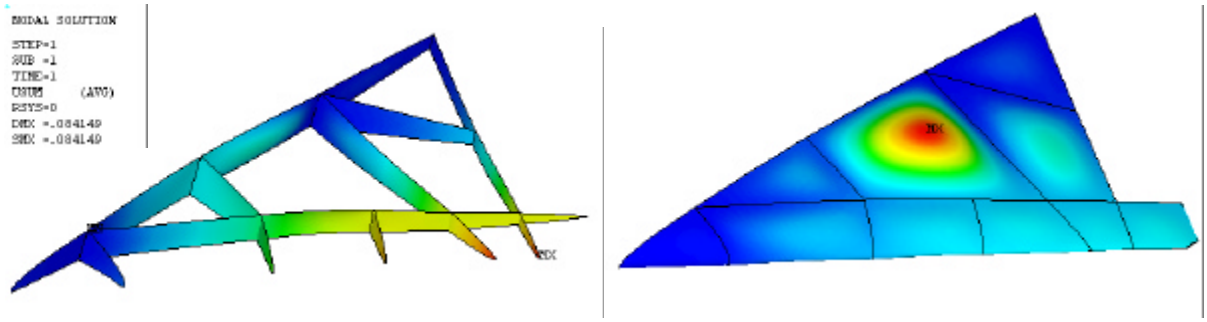


Figure 5.5: Deflections of the Initial Finite Element Wing Model

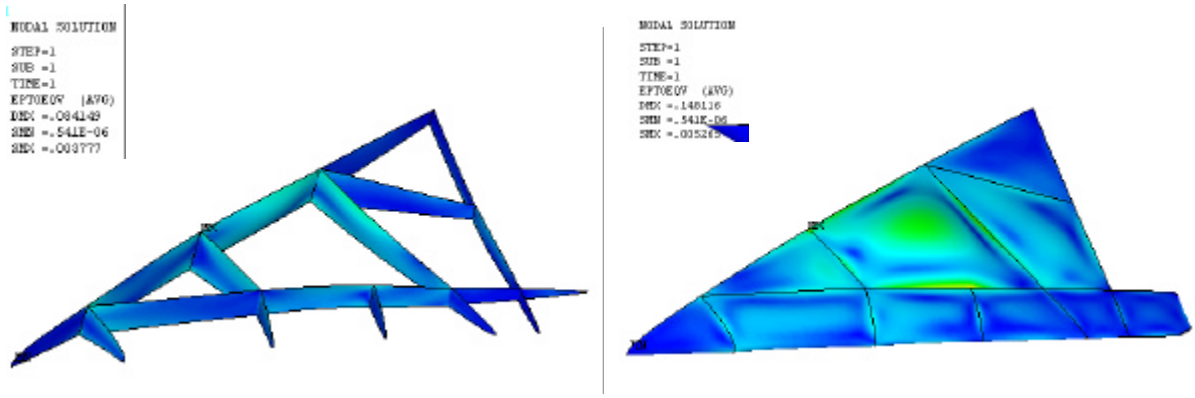


Figure 5.6: Strain Distribution of the Initial Finite Element Wing Model

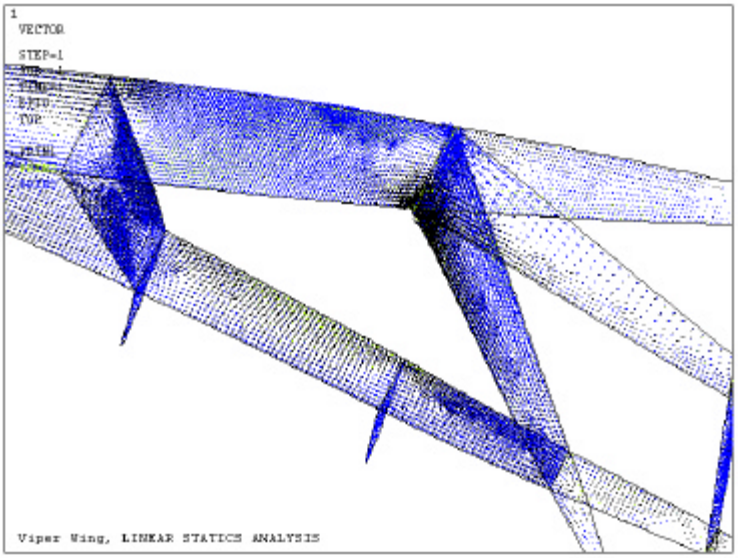


Figure 5.7: Stress Vector Plot of the Initial Finite Element Wing Model

The wing experienced excessive deflections due to the aerodynamic loading. This high deflection created high strain concentrations, Table 5.1. The strain distribution on the skin could be spread out over more of the skin by slightly changing the internal structures. Two concentrations of strain were present at the wing tab locations. The strain vector plot shows that there are strain concentrations near the tab location and that the structures are not transferring loads throughout the entire wing structure.

Table 5.1: ANSYS Analysis Results for Initial Wing

	Deflection (in)	Strain
Internals	0.084	3.777E-03
Skin	0.148	5.265E-03

5.2.2. Final Finite Element Model

There were some final modifications made to the wing's finite element model. An additional hatch was added to each wing. This hatch was added for access to the control system of the inboard elevon. The internal structures at the joint between the wing and center section, main rib, were hollowed to give access to the center of the wing panels. The internal structures

have been changed to two layers of carbon fiber on either side of a Korex[?] core. The weight of the UAV was also adjusted to the current weight. The final weight breakdown resulted in a 185lb UAV with fuel. The adjusted weight accounts for the higher amount of fuel being carried, turbojet physical weight, inlet actual weight, finalized avionics weight and the finalized airframe weight with hardpoints.

5.2.3.1. Final Wing Analysis

The final wing analysis consisted of analyzing the deflections and the strains incurred under varying load conditions, Figure 5.8 and Figure 5.9 respectively. The strain vector plots, Figure 5.10, were also analyzed to validate the final wing design. This analysis of the wing corresponds to a 9g aerodynamic loading. The boundary conditions on the wing hold the bolt locations fixed and the skin from moving inward or outward. The results of the revised wing analysis were tabulated, Table 5.2.

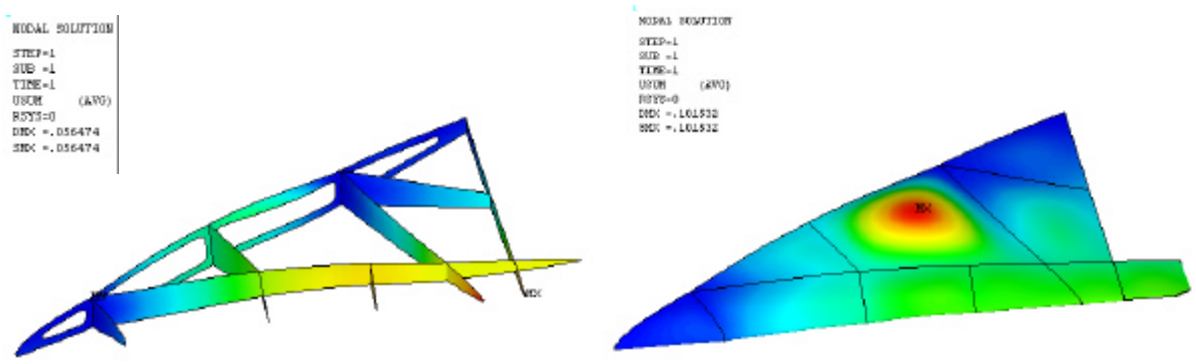


Figure 5.8: Deflections at 9 g's of the Final Wing Model

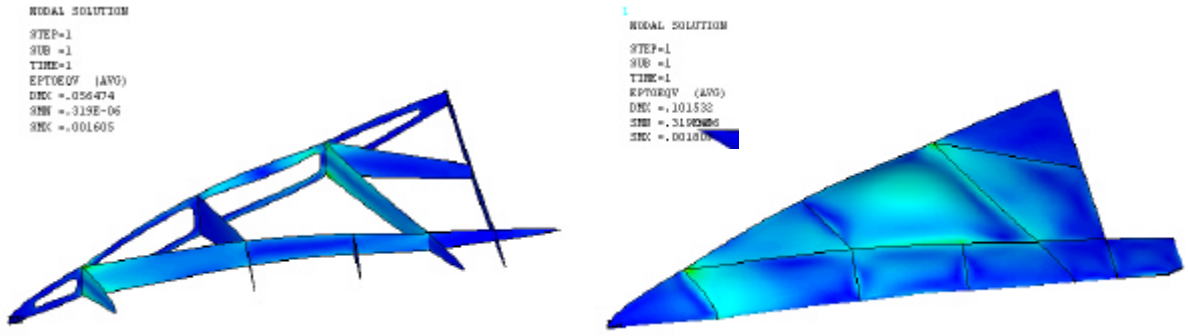


Figure 5.9:Strain Distributions at 9g's for the Final Wing

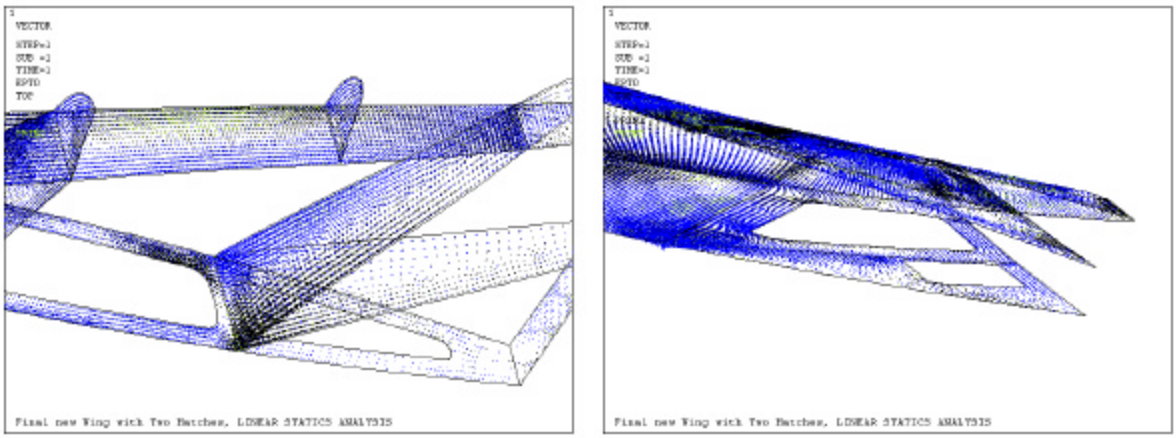


Figure 5.10: Strain Vector Distributions at 9g's for the Final Wing

Table 5.2: ANSYS Analysis Results for the Final Wing

	Deflection (in)	Strain
Internals	0.056	1.605E-03
Skin	0.102	1.803E-03

The deflections of this analysis are similar to the previous iterations. This proved that the additional hatch and hollowed main rib did not diminish the strength of the wing structure. The maximum strain also decreased for the internals and the skin in this analysis. The reduced strain values are attributed to the hollowed main rib. The hollowed main rib forces the load path to move forward along the spar. In the previous iterations the loads were transmitted to the main rib and the internals that tie into the center section. The strain vector

plots also confirms that there are no large areas of strain. This means that the internals and the skin are transferring the strain/loads more evenly throughout the wing's structure.

5.2.3.2. Final Main Center Section Aerodynamic Analysis

The final main center section's analysis was conducted in the same manner as the wings analysis. The resultant forces from the final wing analysis at the tabs were applied to the corresponding center section members. The final center section mesh was revised to remove the problems with the internals not matching the skin. The positive and negative 9g aerodynamic load cases were examined, but the results from the positive 9g case revealed the most about the design, Figure 5.11 thru Figure 5.13. The results of the final center section analysis were tabulated, Table 5.3.

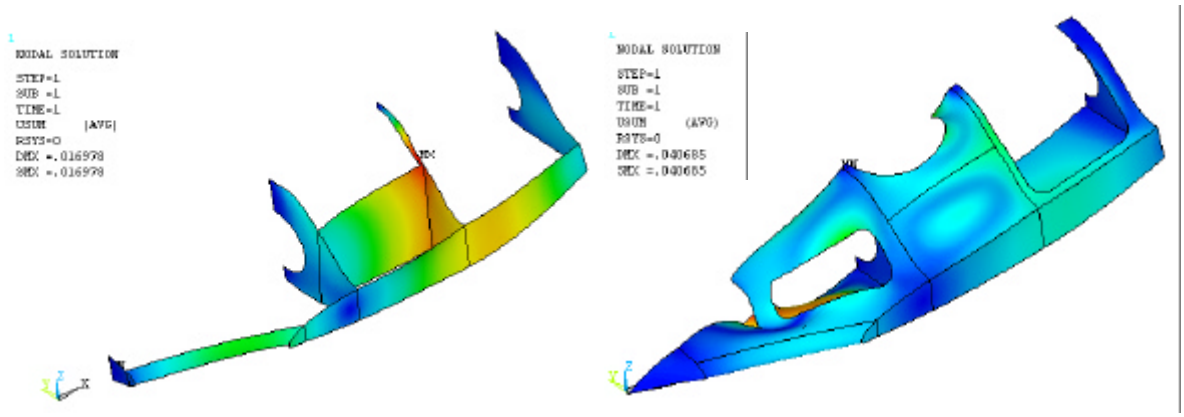


Figure 5.11: a 9g Load

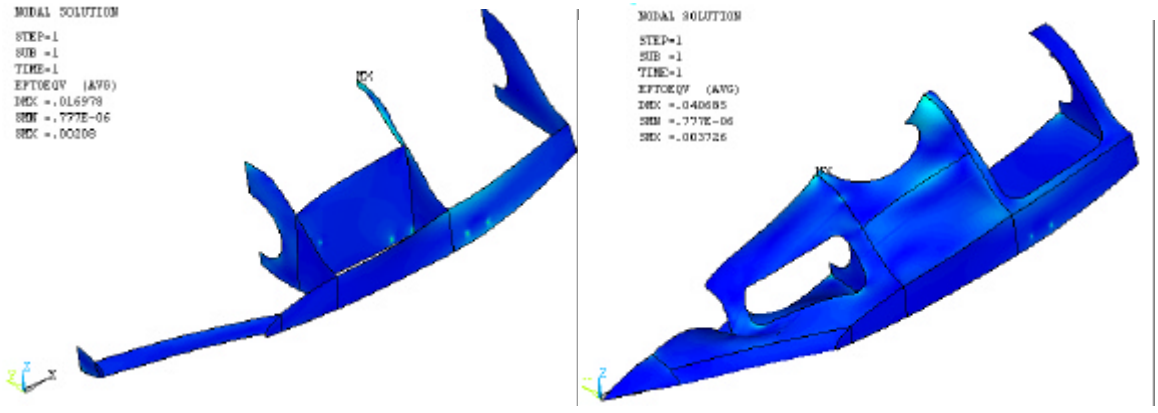


Figure 5.12: Strain of the Center Section Under a 9g Load

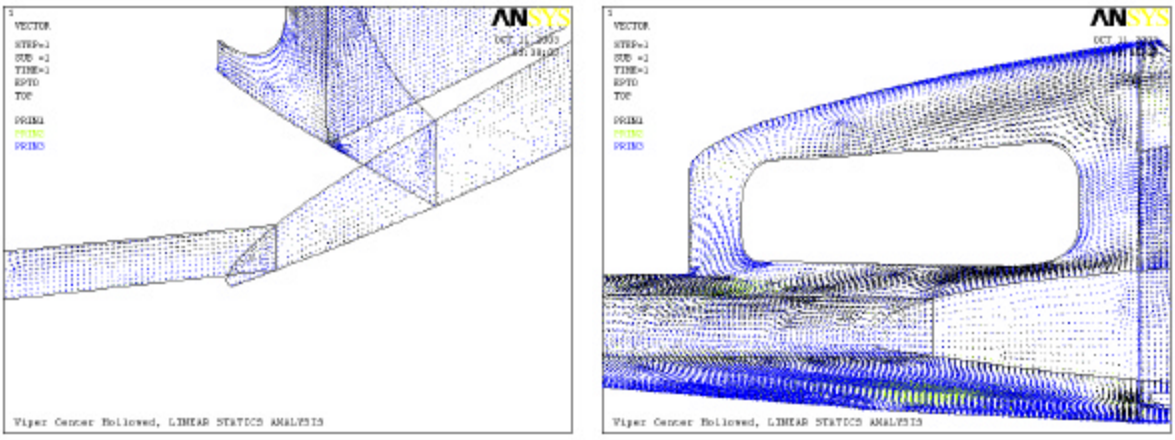


Figure 5.13: Strain Vector Distributions Under 9g Load

Table 5.3: Final ANSYS Results for the Center Section Under Aerodynamic Loads

	Deflection (in)	Strain
Internals	0.017	2.080E-03
Skin	0.041	3.726E-03

The maximum deflection of the internal structures occurs at on the bulkhead behind the turbojet, which is produced by the addition of the wing loads. The maximum deflection of the skin occurred on the lower surface around the nose landing gear doors, which is cause by the aerodynamic loads. The maximum strain of the internals under aerodynamic loads is within the limits of the materials. The maximum strain of the skin is slightly outside the imposed limits. This value was deemed acceptable since during manufacture additional material will be added to attach the skins to the internals. This additional material will strengthen the area where the maximum strain occurs. Also, by examining the strain vector plots it is seen that there are no large concentrations of strain. This means that the internals and the skin are transferring the strain/loads effectively throughout the center section's structure.

5.2.3.3. Final Main Center Section Landing Load Analysis

The final analysis of the *VIPER* UAV consisted of analyzing the landing loads. This analysis uses the same mesh that was produced for the final center section's aerodynamic analysis. A positive 1g aerodynamic load was applied to the skin. The results of the center section analysis, Figure 5.14 thru Figure 5.16, were tabulated, Table 5.4.

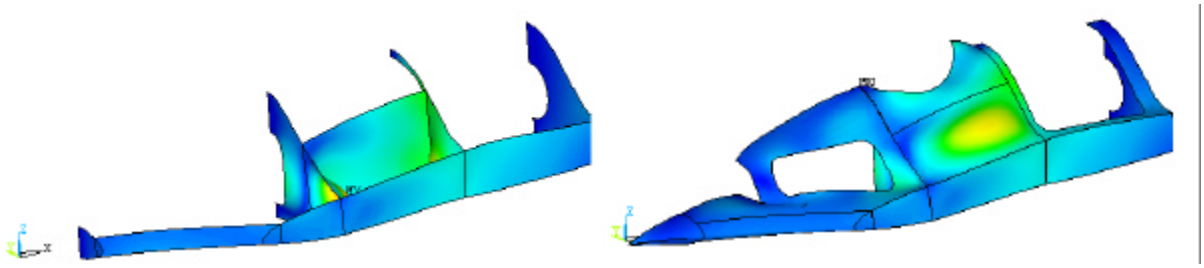


Figure 5.14: Final Center Section Deflections for Landing Loads

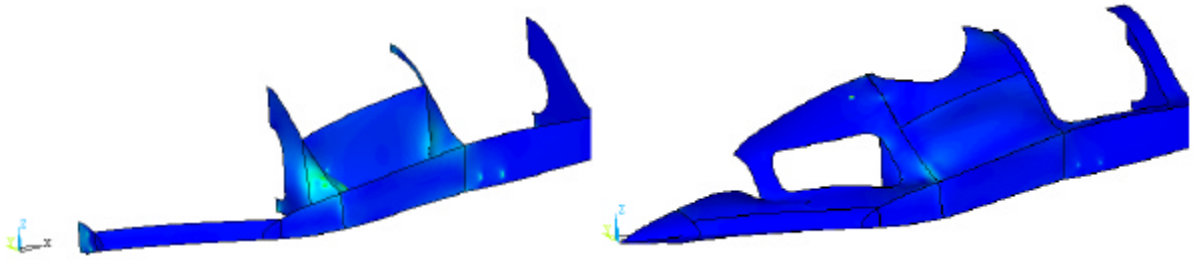


Figure 5.15: Final Center Section Strain Distributions for Landing Loads

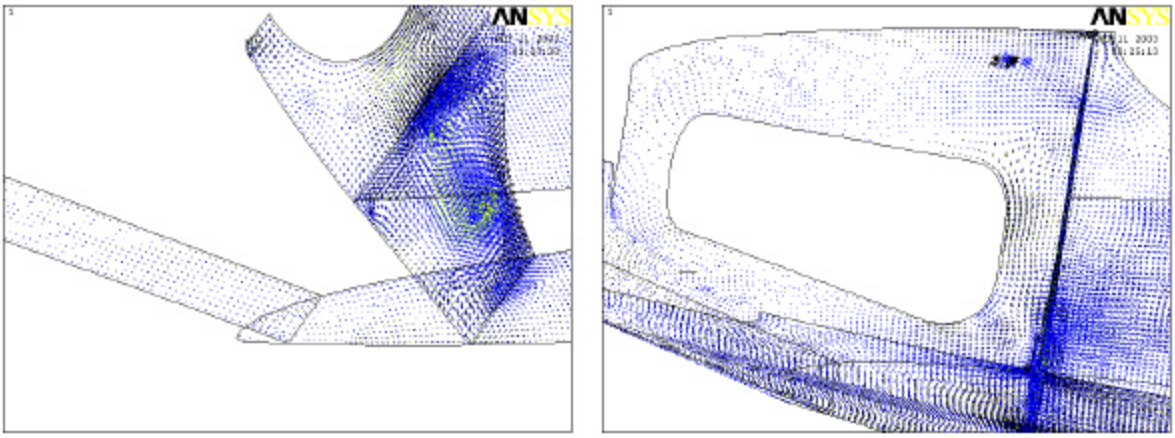


Figure 5.16: Final Center Section Strain Vector Plots for Landing Loads

Table 5.4: Final ANSYS Results for the Center Section Under Landing Loads

	Deflection (in)	Strain
Internals	0.011	1.215E-03
Skin	0.010	2.856E-03

The maximum deflection and strain of the internal structures occurs on the bulkhead that the drag link is attached to. The maximum deflection and strain of the skin occurs above the structures that house the main landing gear. The maximum strains of the final center section are within the limits of the materials. The strain vector plots shows that the strain related to the landing gear is transferred in all directions. The internals and the skin are transferring the strain/loads effectively from the mounting locations of the landing gear and the other systems located in the center section.

5.2.4. Final Finite Element Conclusions

The initial analysis of the wing produced valuable information implementing changes of the structural design. Several intermediate iterations were made before a final model was produced. The final structural model handles aerodynamic and landing loads well. The final structural design of the wing transfers the aerodynamic loads forward as it was designed to. The center section transfers the aerodynamic and landing loads throughout the entire structure.

Now that the structural analysis has concluded the final construction of the *VIPER* UAV could commence. The internals structures are made up of two layers of carbon fiber on either side of Korex®. The non-ideal interactions of the constructed UAV will be addressed by applying carbon fiber strips to the joints. The final configuration of the *VIPER* UAV is seen in Figure 5.17.

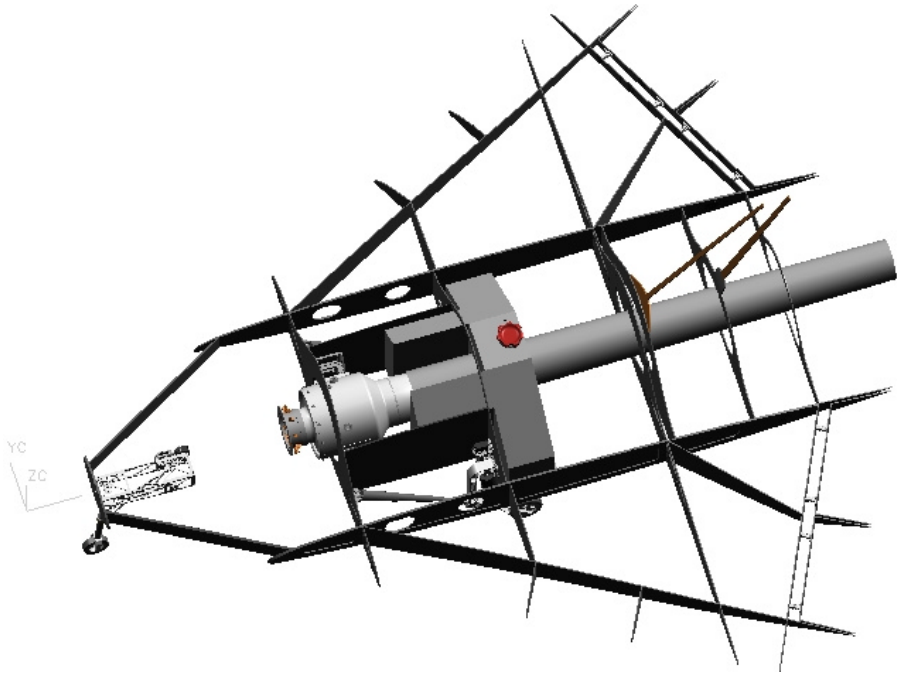


Figure 5.17: Final *VIPER* UAV Configuration

6. MANUFACTURING

6.1. VIPER *UAV AIRFRAME CONSTRUCTION*

The construction process started with hatches and then progresses to the skins. The internals were then constructed and fitted to the skins. Then the UAV was ultimately closed. The final step in the construction of the UAV was to perform final bodywork. After this was completed the structural construction of the *VIPER* UAV was finished.

6.1.1. Hatch Construction

The hatches were laid out on the surface of the molds with permanent marker. The edges of the hatches were cut out of one inch wide pieces of Rohacell®. All of the hatches are one-eighth inch thick except the fuel tank, turbojet and inlet hatches that are one-quarter inch thick. These hatches are thicker because of their unique locations and the loads that are near them. The Rohacell® was heated to form the foam to the curvature of the UAV. After the Rohacell® was completely formed around a hatch the joints were glued together. The rest of the core is Korex® that was cut to fit within the Rohacell®. The outer edges of the foam were sanded to produce a beveled edge for the hatch. The lay-up of the hatches was composed of outer layers of fiberglass and carbon fiber in a wet lay-up vacuum bag process. After the hatches were cured they were released from the mold. Then they were trimmed to the correct size. The hatches are then reattached to the mold with the use of spray contact cement in their correct location, Figure 6.1.

Figure 6.1: Hatch Construction for the *VIPER* UAV

6.1.2. Skin Construction

The start of the skin construction began with constructing formers around the molds. These formers are used to create flanges between the sections of the UAV. The forms were constructed with the same materials as the parting planes. The forms were attached to the molds using the same holes that hold the sections of the molds together. These formers are also used as a sealing surface for the vacuum bag while the skins are being constructed.

The preparation for constructing the skins started with cutting the core material. Korex® was cut to fit around the hatches and the extents of the skin. The curved sections of the UAV needed multiple pieces of Korex® to form the core material. All of the pieces of Korex® were taped together and then beveled. The bevel is used to ease the transition around hatches and the edges of the skins. Two layers of tape were applied to the molds where edges of different skins need to be joined later. The tape produces a step in the skin that permits a structural joint. The materials for the skins were cut out and then wet laid-up, Figure 6.2.

Figure 6.2: Skin Construction for the *VIPER* UAV

6.1.3. Harpoint Construction

The hardpoint material was constructed from layers of carbon fiber. It was determined that seventeen layers would produce the same thickness as the Korex from the material testing results. These layers were wet laid-up to create the hardpoint material. The designs of the hardpoints were modeled in Unigraphics™ to determine the necessary amount of material. This was done to reduce the amount of waste by either have excess or not enough material. The optimum geometric shape for the hardpoint is a circle. The circle produces the lowest and most uniform stress around at the edges.

6.1.4. Internal Structures Construction

The internals were first laid out on tables to determine the minimum amount of materials necessary. After the layout was determined the internals were taped down and the Korex was cut to size. Then the Korex was removed where the hardpoints reside, Figure 6.3. Carbon

fiber was then applied to the Korex to create the internals in a wet lay-up vacuum bag. When the lay-up cured the internal templates were taped to the carbon lay-up, and then rough cut out of the large sheets.

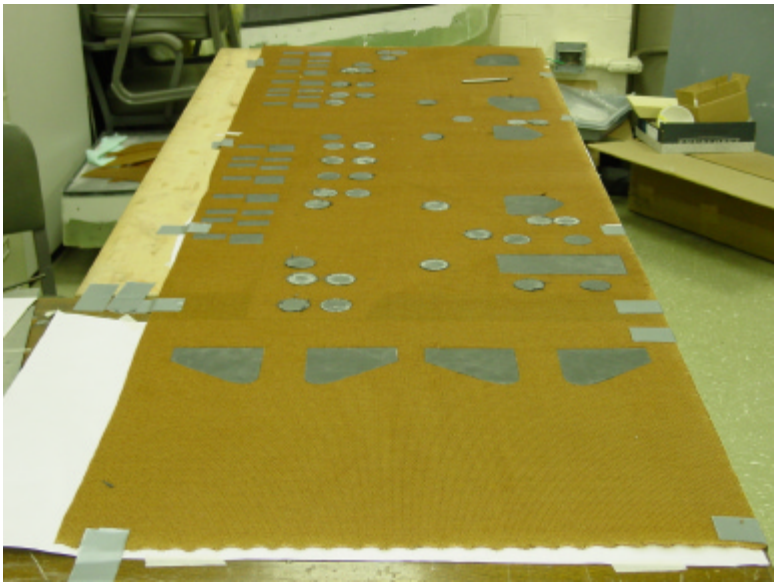


Figure 6.3: Hardpoints for the *VIPER* UAV

The internals were independently and meticulously sanded to fit precisely to the skins. After all of the internals were independently fitted to one side of the UAV, the oversized internals were trimmed to fit together in the skin. Then all of the internals were taped together on one side. Precise measurements were then taken to transfer the internals to the opposing skin. The individual internals were fit on the opposing side of the skin while keeping them attached to the first skin. This step is time consuming and critical to producing a structurally sound airframe, Figure 6.4.

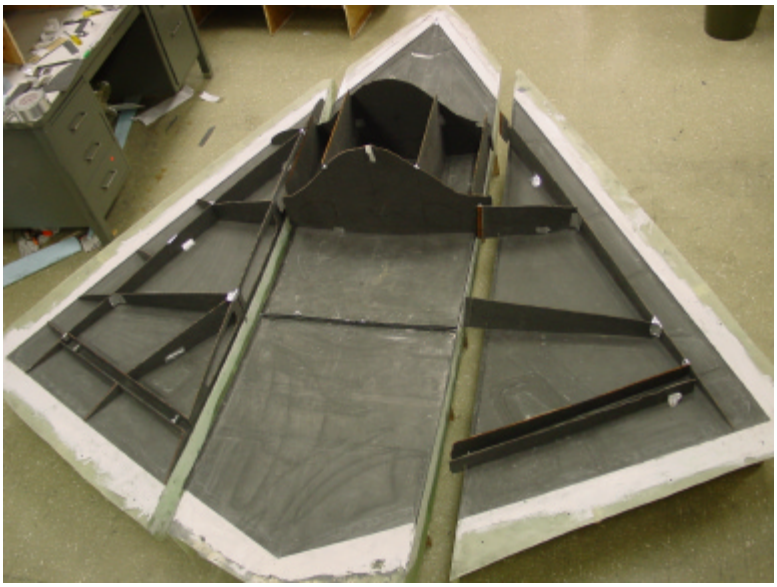


Figure 6.4: Internal Structures Construction for the *VIPER* UAV

6.1.5. Closing the Airframe

Before the airframe could be closed the internals needed to be joined and flanged. Tape and parting agent was placed on the skin of the UAV to create removable flanges. The internals were flanged from the internals down over the tape. Peel ply was added to the top of the flanges to help hold the carbon fiber down and to remove any excess epoxy. After one entire side was flanged the internals were permanently bonded to the opposing skin via small dabs of thickened epoxy. Cab-o-sil was added to the epoxy to thicken it. Cab-o-sil does not reduce the strength of the epoxy because it is not filler. After the epoxy cured, the internals and upper skin were carefully removed from the lower skin. Then the tape was removed from the lower skin. The exposed flanges were trimmed at this point. After everything was prepared for closure, the UAV was closed by applying epoxy to the exposed flanges and then closed. During the final assembly phase it was important to make sure that all of the molds were in their correct orientation to achieve the correct dihedral, Figure 6.5.

Figure 6.5: Closing the *VIPER* UAV

6.1.6. Control Surface

Before the control surfaces could be cut out of the skins the trailing edges needed to be sealed. The trailing edges of the UAV were sealed with thickened epoxy. After the trailing edges were sealed, the control surfaces could be cut out. Meticulous planning was necessary before any of the surfaces could be cut out. The edges of the control surfaces were marked on the skins. Then a straight edge was attached to the UAV to help guide the cutting. Ultimately all of the edges were cut and the control surfaces were removed from the UAV. A rounded piece of foam was applied to the leading edge of the control surface to finish them, Figure 6.6. The finished control surfaces were then reattached to the UAV.

Figure 6.6: *VIPER* UAV Elevon Control Surfaces

6.1.7. Finish Bodywork

After the UAV was closed, the seams need to be closed. Since the skin carries a good amount of the loads it was important to produce a strong joint. The adjoining surfaces were prepared before they were sealed. The edges of the UAV were sealed from the outside with strips of carbon fiber. The carbon fiber strips were held down with the use of peal ply. The peel ply was held to the UAV with tape. The joints between adjoining panels were slightly filled to make the gaps as small as possible. The slight differences in the heights between adjoining panels were also smoothed out, Figure 6.7.

Figure 6.7: Finished Construction of the *VIPER* UAV

7. STRUCTURAL TESTING

7.1. STRUCTURAL LOAD TEST PREPARATION

To verify the *VIPER* UAV structural design, the airframe was subjected to static load testing. The load testing of the UAV encompassed getting the *VIPER* UAV ready for load testing, actual load testing and the analysis of the results. Before the static load testing could commence the generation of the panels to load, the determination of the loads for each panel, creation of cradles to hold the UAV and the application of strain gauges in strategic locations needed to be performed. The load tests were then performed and data was acquired. Finally the results from the load testing and the structural analysis were analyzed.

7.1.1. Load Generation for Testing

The load generation for testing the UAV was composed of dividing the wing into panels and then determining the associated loads. The wing needed to be divided into panels for the physical load testing. The panels were used during the load test to define where the weights need to be applied. First the wing was divided into span-wise sections and then each section was sub-divided into panels. The span-wise sections were set to approximately one foot in width. This was done to produce enough area for the weights to be placed on the surface of the wing during load testing. Three sections were needed for each wing. After the sections were determined, the number of panels in each section was determined. It was important to make sure that the panels were not too small. If a panel were too small, applying the weights during load testing would be difficult. The final grid for the load testing can be seen in Figure 7.1.

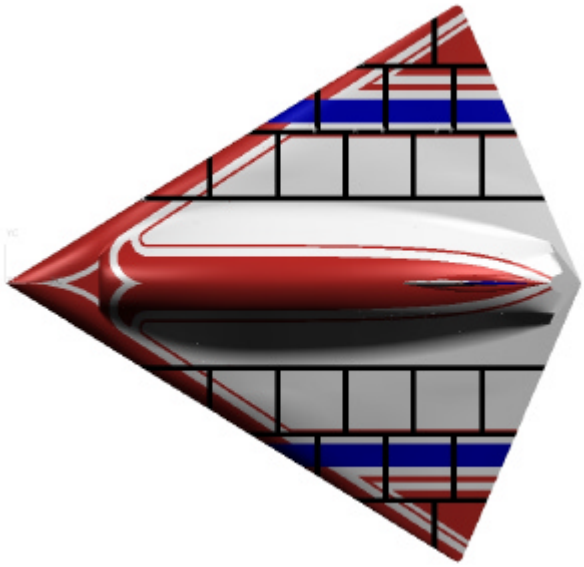


Figure 7.1: Grid for Load Testing

After the wing was divided into the load testing areas, a FORTRAN program, Appendix 10.4, was created to determine the loads for each area. A program was implemented because of the complexity of the aerodynamic model. The aerodynamic model, analyzed with the linear inviscid CMARC solver, is composed of many elements, which would have been time intensive to parse and errors could be made. The program converts the coefficient of pressure from the aerodynamic model to actual pressures corresponding to a 1g case. Then it sums up all of the pressures that reside within each area and produces an output file. The resulting loads for a 1g loading on the grid can be seen in Figure 7.2. The other g loadings are multiples of the 1g loading because the aerodynamics results are linear.

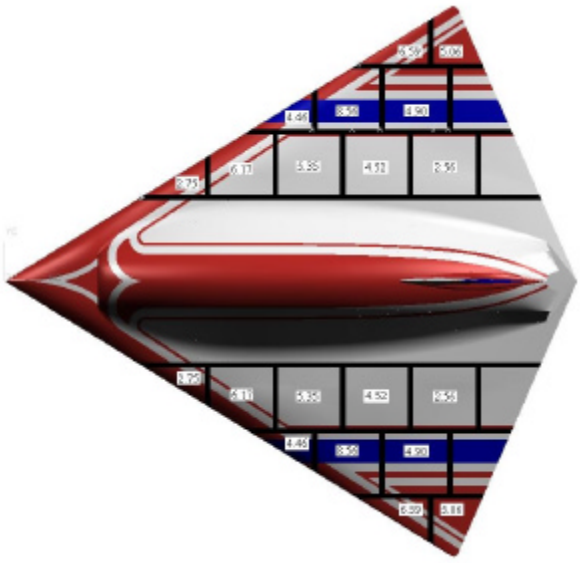


Figure 7.2: 1g Loads for Load Testing

The final aspect that needed to be implemented before the load testing could commence was the construction of support structures. These support structures were used to constrain the UAV from movement during load testing. Since the wings are the only section that loads are being applied to, two support structures were constructed to firmly hold the center section of the UAV. These support structures hold the UAV from the bottom and the top by supporting the surface above the bulkheads. For positive g loading the UAV was placed inverted and the weight was placed on the lower surface.

7.1.2. Strain Gauge Locations

Now that the *VIPER* UAV was ready for static load testing, the location of the strain gauges needed to be defined. The placement of the strain gauges was critical to obtaining useful data to relate back to the ANSYS analysis. The surface of the UAV was omitted since the weights for load testing would only be placed on one surface at a time. This does not accurately represent the aerodynamic loads seen by the UAV, but it does impose the total load for the airframe. The tabs were also omitted to reduce the risk of obtaining poor correlation to the structures model because of the boundary conditions in the analysis being different.

Ten strain gauges were located at three areas of interest on the wing. Three rosettes of strain gauges were arranged on the internals of one wing. Each rosette is composed of three strain gauges orientated 0, 45 and 90 degrees. The rosettes were placed on the internal structures of the wing. The locations correspond to a point near the front tab on the main spar, near the central point of the wing on the main spar and on the structure that the main tab mounts to. The diagonal strain gauge of front location was not used because of the number of strain gauges that could be tested. The remaining two strain gauges were placed in the other wing at the central point on the main spar and on the structure that the main tab mounts to. The locations and orientation of the strain gauges can be seen in Figure 7.3.

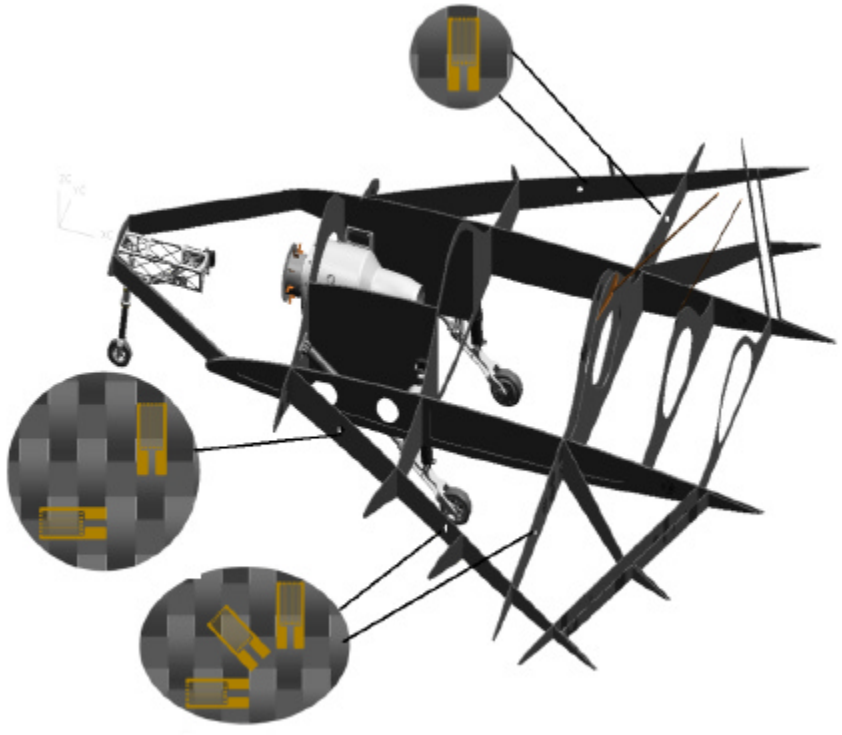


Figure 7.3: Strain Gauge Locations

7.2. LOAD TESTING

The UAV was prepared for the load tests by creating a testing plan and preparing the weights. The testing plan was necessary for making sure that the tests would not be performed without a methodology and limits. This reduced the risk of a catastrophic failure by setting guidelines to abort the test before damage would occur. The weights were prepared in 1g increments for each panel totaling 6g. The last step was to execute the load testing.

7.2.1. Testing Plan

A test plan was derived to aid the static structural load testing and pose all possible situations. The test plan is as follows:

Turn on the computer and leave the System 6000 turned off
Start the Strain Software and either open a project or create a new one

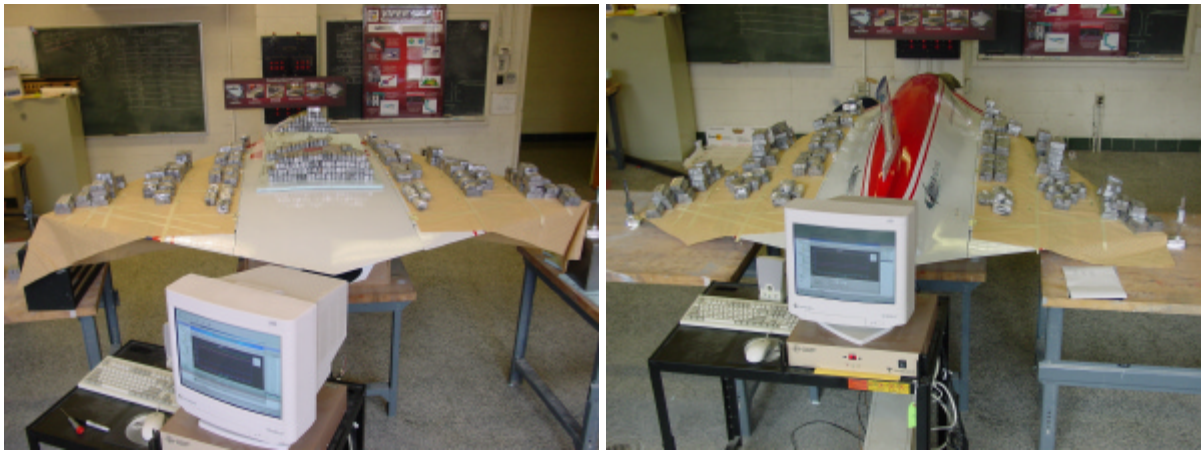
1. Connect five Strain Gauges to the corresponding channels
2. Turn on the System 6000 scanner and let it warn up for a few minutes
3. Zero and Calibrate the Strain Gauges
4. Place a set of 1g weights on the wings
5. Let the oscillations dampen out
6. Record between ten and twenty seconds of strain data
7. Record the wing tip deflections
8. Make sure that the strains are within the 3500 micro strain values
   - If the strain values are large stop and remove the loads
   - If the strain values are okay continue
9. If the maximum weight is reached proceed to step 10 otherwise repeat steps 4-8
10. Remove two sets of 1g weights from the wings
11. Let the oscillations dampen out
12. Record between ten and twenty seconds of strain data
13. Record the wing tip deflections
14. If all of the weights are off of the wings continue otherwise repeat steps 10-13
15. Turn off the System 6000 scanner
16. If there is another set of strain gauges to process proceed to step 1

Two complete iterations of static load tests were conducted for the positive and negative load cases. This was necessary because the data acquisition computer, System 6000 model 6100

scanner by VISHEY Measurements Group, that was used at NC State monitors five strain gauges at a time and ten strain gauges were applied to the UAV. The repeated tests give more accurate deflections on the second iteration because the structure had a chance to reach its equilibrium position on the first iteration. As each 1g increment was placed on the wing the deflections and strains were acquired. The strain values were constantly monitored as the weights were applied to make sure that the structures would not fail.

7.2.2. Physical Load Testing

Prior to the start of the static load testing of the airframe, approximately three hundred pounds of lead was placed in the center section around the bulkheads. This was necessary to produce a stable test platform. This reduced the risk of movement between the UAV and the supporting structures. The weight was incrementally place on the wing. If the wing was oscillating from the placement of the weights the oscillations were allowed to dampen out before data was acquired. Wing tip deflections and strain data were then acquired. Once the 4g load case was achieved, the application of weights was paused to make sure that nothing was going out of range. The load testing resumed and continued until the 6g loads were applied. The same procedure was implemented for the remaining strain gauges and the positive load cases. The static load testing can be seen in Figure 7.5.



+5g Load Case **-6g Load Case**

Figure 7.5: Static Load Testing

7.2.2. Load Testing Data

The analysis of the results started after the physical load testing was conducted. The wing tip deflections of the tests were tabulated, Table 7.1, along with the corrected deflections, Table 7.2. The wing tip deflections were corrected by removing the wing droop produced by separation from the center section. This was done so that the load test data could be compared to the ANSYS results. The separation at the center section was just over one sixteenth of an inch. The strain data was reduced to obtain the mean for each location during every load case. This data can be seen in Figure 7.6.

Table 7.1: Load Test Deflection Results

Number of g's	Deflections (inches)			
	run1		run2	
	Left	Right	Left	Right
-1	0.112	0.111	0.081	0.087
-2	0.248	0.232	0.176	0.179
-3	0.414	0.355	0.288	0.291
-4	0.596	0.486	0.397	0.394
-5	0.764	0.605	0.504	0.499
-6	0.938	0.8	0.607	0.601
1	0.092	0.08	0.061	0.06
2	0.209	0.198	0.145	0.141
3	0.346	0.346	0.245	0.239
4	0.49	0.492	0.349	0.338
5	0.646	0.67	0.451	0.44
6	0.84	0.828	0.56	0.55

Table 7.2: Corrected Load Test Deflection Results

Number of g's	Deflections (inches)			
	run1		run2	
	Left	Right	Left	Right
-1	0.041	0.040	0.010	0.016
-2	0.107	0.091	0.035	0.038
-3	0.202	0.143	0.076	0.079
-4	0.314	0.204	0.115	0.112
-5	0.411	0.252	0.151	0.146
-6	0.515	0.377	0.184	0.178
1	0.021	0.009	0.000	0.000
2	0.068	0.057	0.004	0.000
3	0.134	0.134	0.033	0.027
4	0.208	0.210	0.067	0.056
5	0.293	0.317	0.098	0.087
6	0.417	0.405	0.137	0.127

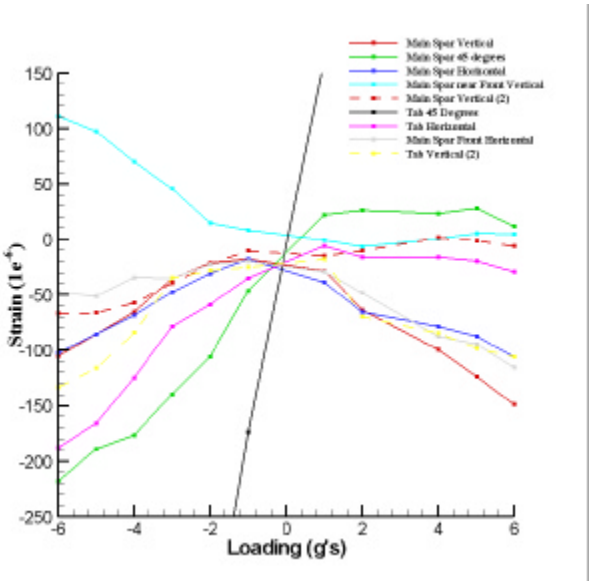


Figure 7.6: Strain Gauge Load Testing Results

7.2.3. Load Testing Comparisons

The data from the data acquisition computer was reduced via a FORTRAN program, Appendix 10.5. Once the results were reduced they were compared to the ANSYS analysis.

Another FORTRAN program, Appendix 10.6, was implemented to parse the ANSYS data file to obtain the results at the closest location to the strain gauges. The strain data was compared first and tabulated in Table 7.3 thru Table 7.5. It was observed that there was a major problem with the data. The strain values from the ANSYS analysis ranged from 15% to 9000% with an average of 313% difference from the analysis results. The wing tip deflections correlated well with the ANSYS results. ANSYS predicted a maximum deflection of 0.114 inches and the maximum load test deflection was 0.137 inches. As a result of the deflections being close and the strain results being so different more investigation needed to be conducted to define the problem.

Table 7.3: Micro Strain Gauge Testing Results

	Strain Gauge Location								
g Loading	1	2	3	4	5	7	8	9	10
-6	-105.14	-217.77	-101.82	111.45	-66.49	-1199.43	-187.47	-47.54	-132.78
-5	-85.64	-189.14	-85.54	97.41	-65.75	-1006.17	-165.88	-50.76	-115.77
-4	-64.95	-176.14	-68.00	70.27	-56.62	-764.70	-124.66	-34.07	-83.73
-3	-35.60	-139.28	-47.28	45.92	-39.03	-542.03	-78.32	-34.79	-34.56
-2	-21.53	-105.79	-31.03	15.03	-22.06	-375.46	-58.88	-22.50	-27.68
-1	-17.70	-46.40	-17.40	8.36	-10.07	-173.66	-34.65	-19.62	-24.93
1	-28.22	21.92	-38.62	-0.63	-14.43	162.69	-5.87	-29.11	-17.75
2	-63.53	26.08	-64.95	-6.65	-9.82	356.09	-16.01	-47.83	-69.32
4	-99.08	23.06	-78.02	0.68	2.13	688.18	-16.01	-87.19	-84.48
5	-123.53	27.92	-87.46	5.26	-1.10	904.77	-19.74	-94.61	-97.90
6	-148.40	12.15	-105.92	4.79	-5.50	1079.85	-29.25	-115.22	-105.89

Table 7.4: Micro Strain Data from ANSYS

	Strain Gauge Location								
g Loading	1	2	3	4	5	7	8	9	10
-6	36.045	-28.157	-28.157	-4.1764	31.855	75.951	-24.223	1.2879	-14.228
-3	18.023	-14.078	-14.078	-2.0882	15.928	37.975	-12.112	0.64394	-7.1143
3	-18.023	14.078	14.078	2.0882	-15.928	-37.975	12.112	-0.64394	7.1143
6	-36.045	28.157	28.157	4.1764	-31.855	-75.951	24.223	-1.2879	14.228

Table 7.5: Percent Difference from the ANSYS Analysis

	Strain Gauge Location								
g Loading	1	2	3	4	5	7	8	9	10
-6	-392%	673%	262%	-2768%	-309%	-1679%	674%	-3791%	833%
-3	-298%	889%	236%	-2299%	-345%	-1527%	547%	-5503%	386%
6	312%	-57%	-476%	15%	-83%	-1522%	-221%	8846%	-844%

7.3. ANALYSIS OF ANSYS VERSUS LOAD TEST DISCREPANCY

7.3.1. ANSYS Re-Analysis

The trends from the ANSYS analysis of the UAV looked correct, but the discrepancy between the analysis and the load tests continually raised questions about the structural model. Different boundary conditions were checked and the results remained consistent with the final iteration of the structures. Several new ANSYS iterations were conducted to try to determine the discrepancy between the ANSYS and load test results. The first model implemented utilized a further refined mesh. The results from this analysis were very close to the final ANSYS analysis model. A series of ANSYS warnings were discovered during this refined analysis. The warning that raised the most questions dealt with the layers of the composites. The warning stated that there is a possible inter-laminar shear stress violation and that a refinement of the mesh could eliminate this warning. From this warning a series of different shaped tests were analyzed. The results of these simple shapes revealed the identical warning. A final ANSYS analysis was conducted with the material properties of the entire laminate as one layer. The trends from this analysis do not correlate with the expected trends. It turned out that the trends from the composite layering model correlated well with the expected trends.

7.3.2. Strain Gauge Analysis

Since the ANSYS re-analysis did not give any insight into the discrepancy, a series of tests were conducted on the original material testing coupons. This time the bending tests were instrumented with strain gauges. The first test focused on checking the calibration of the strain acquisition computer. Strain gauges were placed on a known sample of aluminum and then subjected to bending loads. The results from this test revealed that the data acquisition was correct.

The load test results were re-examined and it was discovered that the strain gauge data on opposing wings were not correlating. Table 7.6 highlights the discrepancies in the load test data. This information was used in the design of another test. This test instrumented an array of different size strain gauges on a carbon fiber composite coupon. This coupon was subjected to two different bending cases. The face sheet of carbon was orientated vertically and loaded downward and the other case was orientated horizontally with the load pulling down, Figure 7.7 and 7.8 respectively.

Table 7.6: Micro Strain at the Identical Location on Opposite Wings

	Main Spar	
g Loading	Left Wing	Right Wing
-6	-105.14	-66.49
-5	-85.64	-65.75
-4	-64.95	-56.62
-3	-35.6	-39.03
-2	-21.53	-22.06
-1	-17.7	-10.07
1	-28.22	-14.43
2	-63.53	-9.82
4	-99.08	2.13
5	-123.53	-1.1
6	-148.4	-5.5

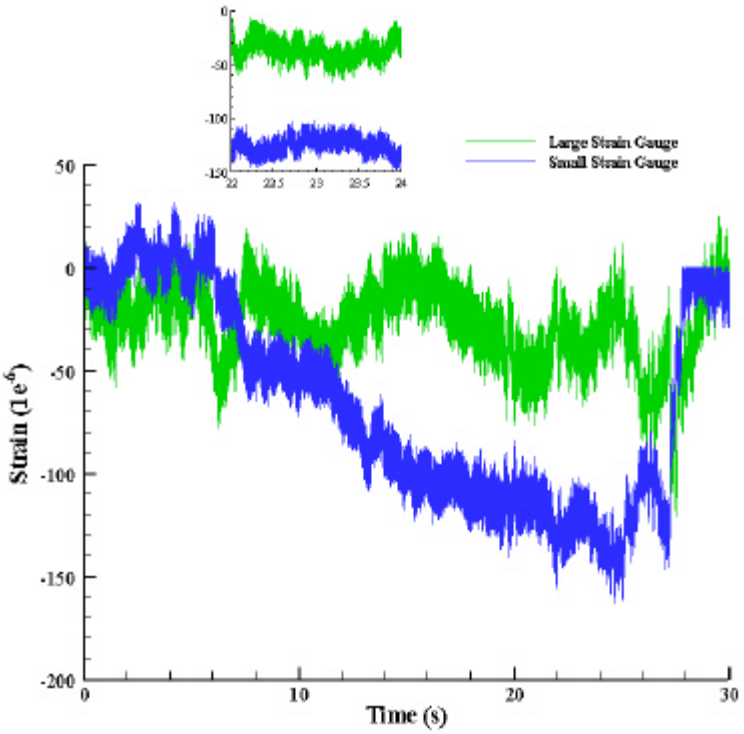


Figure 7.7: Strain Gauge Results from Vertical Bending

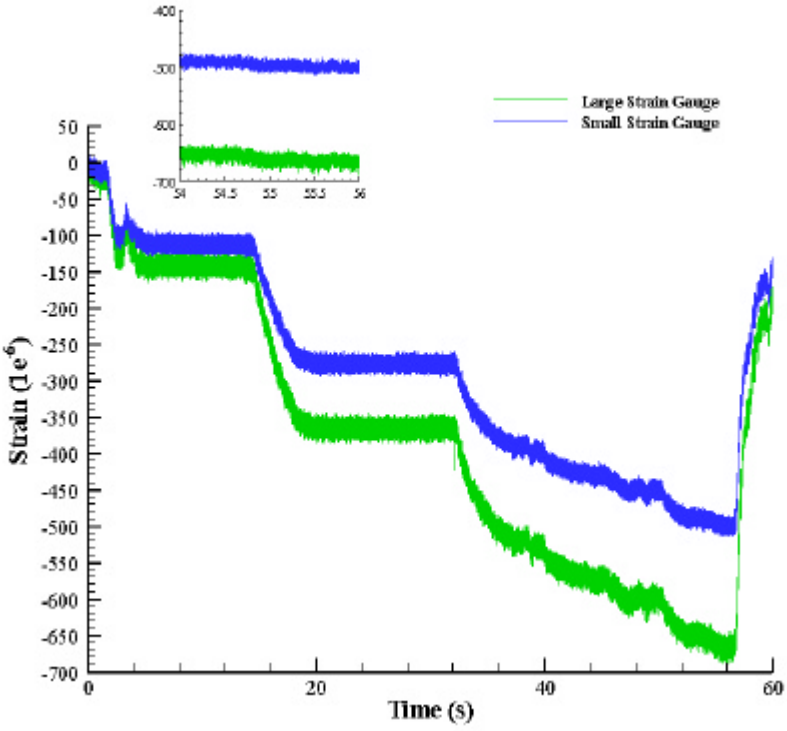


Figure 7.8: Strain Gauge Results from Horizontal Bending

The strain gauge results from these test proved beneficial. The load testing of the *VIPER* UAV utilized the small strain gauges. The plots of the strain data revealed that the strain gauge data that was acquired from the load test was inaccurate. The size of the small strain gauge only covers approximately two thirds of a complete weave of the carbon fiber. This meant that the placement of the strain gauges directly correlated to the accuracy of the resulting data. The larger strain gauge covers more than one complete weave. The larger strain gauge produces an average of the strain across multiple weaves. The ANSYS model uses the average properties across the entire composite. The material properties that were generated from the experimental tests did not use strain gauges. The larger strain gauge produced more accurate results. A depiction of the two strain gauges applied to the carbon fiber can be seen in Figure 7.9.

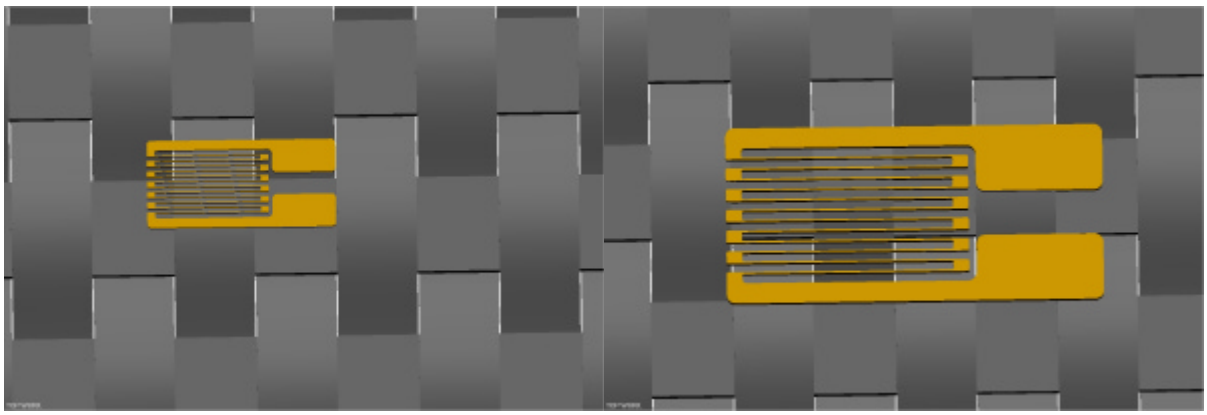


Figure 7.9: Strain Gauges Applied to Carbon Fiber

7.3.3. Sources of Error

The strain gauges were the main contributors of the load test strain errors. A variety of other aspects of the UAV also contribute to erroneous data. They range from the ANSYS model to the construction of the UAV. The joints between the skin and the internals for the structural analysis were modeled as perfect joints. The boundary conditions that were imposed on the structural models also contribute to the error. The boundary conditions that were imposed

were perfect locked nodes whereas the actual points can move slightly. The construction of the airframe is the main source of error. If the constructed airframe does not exactly match the structural model then the comparison between the two will also not match. The constructed airframe varies by the most one eighth of an inch in the placement of structures within the wing. The constructed airframe implemented a butt joint between the skin and internals along with flanges. The flanges were used to help bond the internals to the skin. The internals might not be in the exact same orientation as the model. If the constructed internals are not absolutely vertical they tend to twist under high loading.

From all of the post analysis of the ANSYS model and the constructed airframe, the source of error between the two was determined. If the airframe was constructed exactly as the ANSYS model and larger strain gauges were utilized the results should have closely matched.

8. CONCLUSIONS

The customer requirements of the *VIPER* UAV vehicle have been met during the analysis phase. The landing gear, materials and the airframe have been designed and analyzed. These have also been tested to validate the design and analysis. The first aspect of the *VIPER* UAV that was designed was the landing gear. After the design and analysis provided confidence that the manufactured landing gear would perform as designed. The gear was tested before installation into the vehicle. These tests confirmed that the landing gear performed as they were designed and analyzed.

The material properties used on this vehicle has been tested and verified to obtain all of the material properties. The verification of the material properties was accomplished. The materials were modeled by using the Shell 99 element which was used for it ability to model layered composite materials. The analysis matched well with the experimental results. The structural design of the UAV was conducted to handle the harshest flight conditions. The final airframe analysis showed that it handles the landing loads as well as the aerodynamic loads well. The internal structures along with the skin transfer the loads forward towards the center of gravity. This was important because more than two thirds of the surface area exists behind the center of gravity. During the testing of the airframe it was found that the deflections of the wings correlated well with the analysis, but the strains within the internal structures did not correlated well. It was determined that the strain gauges that were used during the load testing were too small for the materials used on the UAV.

At the time of this thesis a final Load Test verification has not been attempted. The final stages of assembly are currently underway. The final system integration testing will commence as they are installed in the *VIPER* UAV. The flights of the UAV are scheduled to commence as soon as possible.

9. REFERENCES

[1] Collie, Wallis V., N., "Design and Analysis of an Unmanned Aerial Vehicle Propulsion System with Fluidic Flow Control Inside a Highly Compact Serpentine Inlet Duct," *MS Thesis*, Department of Mechanical and Aerospace Engineering, NC State Universitey, Raleigh, NC, 2003

[2] Gridley, M. C. and Walker, S. H., "Inlet and Nozzel Technology for 21st Century Fighter Aircraft," *ASME Paper 96-GT-244*, June 1996

[3] Hamstra, J. W., Miller, D. N., Truax, P. P., Anderson, B. H. and Wendt, B. J., "Active Inlet Flow Control Technology Demonstration," *ICAS Paper 2000-6.11.2*, August 2000.

[4] Anderson, B. H. and Gibb, J., "Vortex Generator Installation Studies on Steady State and Dynamic Distortion," *Journal of Aircraft*, Vol. 35, July-August 1998, pp. 513-552.

[5] Anderson, B. H., Miller, D. N., Yagle, P. J. and Truax, P. P., "A Study on MEMS Flow Control for the Management of Engine Face Distortion in Compact Inlet Systems," ASME Paper FEDSM99-6920, July 1999.

[6] Hall, C. E., "Real-Time LINUX based System for Flight Testing of Remotely Piloted Vehicles," Proc. IEEE 19th Digital Avionics System Conference, October 2000

[7] Currey, Norman S., *Aircraft Landing Gear Design: Principles and Practices*, Published by AIAA, 1988

[8] Gibson, C. S., "Flight Testing of a 17.5% Scale F/A-18E Remotely Piloted Vehicle for Simulation Enhancement," *MS Thesis*, Department of Mechanical and Aerospace Engineering, NC State Universitey, Raleigh, NC, 1997

[9]Jones, Robert M., *Mechanics of Composite Materials*, 2nd edition, Taylor & Francis, Philadelphia, PA, 1999

[10] American Society for Testing and Materials, C 393-94, *Standard Test Methods for Flexural Properties of Sandwich Construction*, 1998.

[11] American Society for Testing and Materials, D 3039/D 3039M-95a, *Standard Test Methods for Tensile Properties of Polymer Matrix Composite Materials, 1998.*

10. APPENDIX

10.1. CONCURRENT ENGINEERING

Propulsion System
Fuel consumption
Turbine Cooling
Operating Temperature
Exhaust Duct
Double Wall
Single Wall
Insulated Combination
Engine position
Engine Controls
Fuel System
Maximum Thrust
Installation loses
Turbine Installation
S-duct Installation
Fuel Capacity
Bleed Air Requirements
Mass Flow
Inlet Design
Cooling Inlet/ducting
Cooling Flow Induction

Gear System
Location
Position and Retraction
Weight
Gear
Plane
Brakes
Drum
Locking Mechanism
Landing Loads
Shock/dampening
(oil/pneumatic)
Tire/Wheel Size
Pneumatic System
Bay Location
Drag
Materials
Aluminum
Steel
Bay Doors
Test Setup
Actuation
Drop Test
Stroke

Performance
Takeoff/Landing
Distance
Speed
G Loading
Flight Envelope
Endurance
Stall Angle/Characteristics
Avionics Weight
Airframe weight
Landing Angle
L/D
Crosswind Capability
Rate of Climb
Range

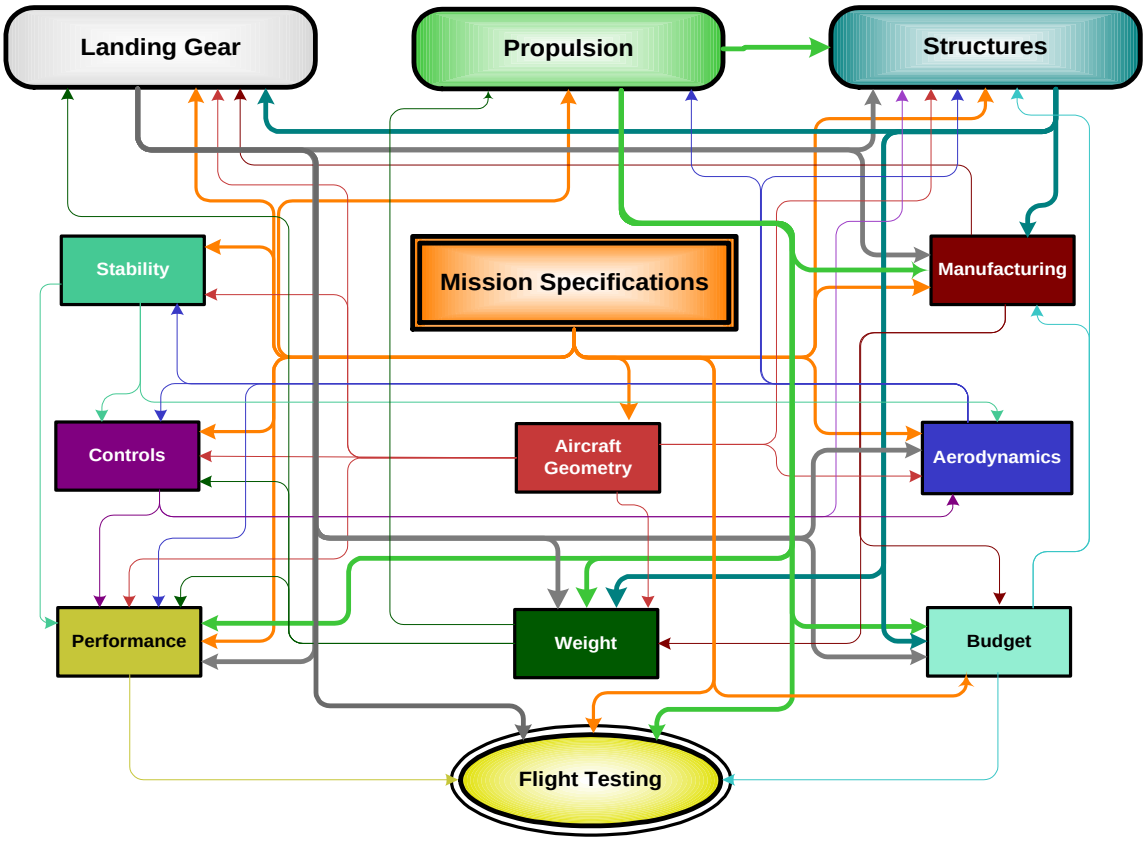
Structures
Strength vs. Weight
Wing Separation points
Structural Foam/Korex
Turbine/Gear Mounting
Configuration
Load Factors +/- 6 G's
Landing Loads 4g's
Avionics size
Data Probe mount/location
Vertical Attachment
Exhaust pipe interaction
Hardpoints/Composite Interaction

Stability and Controls
Longitudinal Control Power
Lateral Control Power
Elevator trim
Static Margin
Parallel servos
Heavy Duty servos
Fiber Optics
Control Surface sizes
Elevons
Moments of Inertia
Dynamic Stability

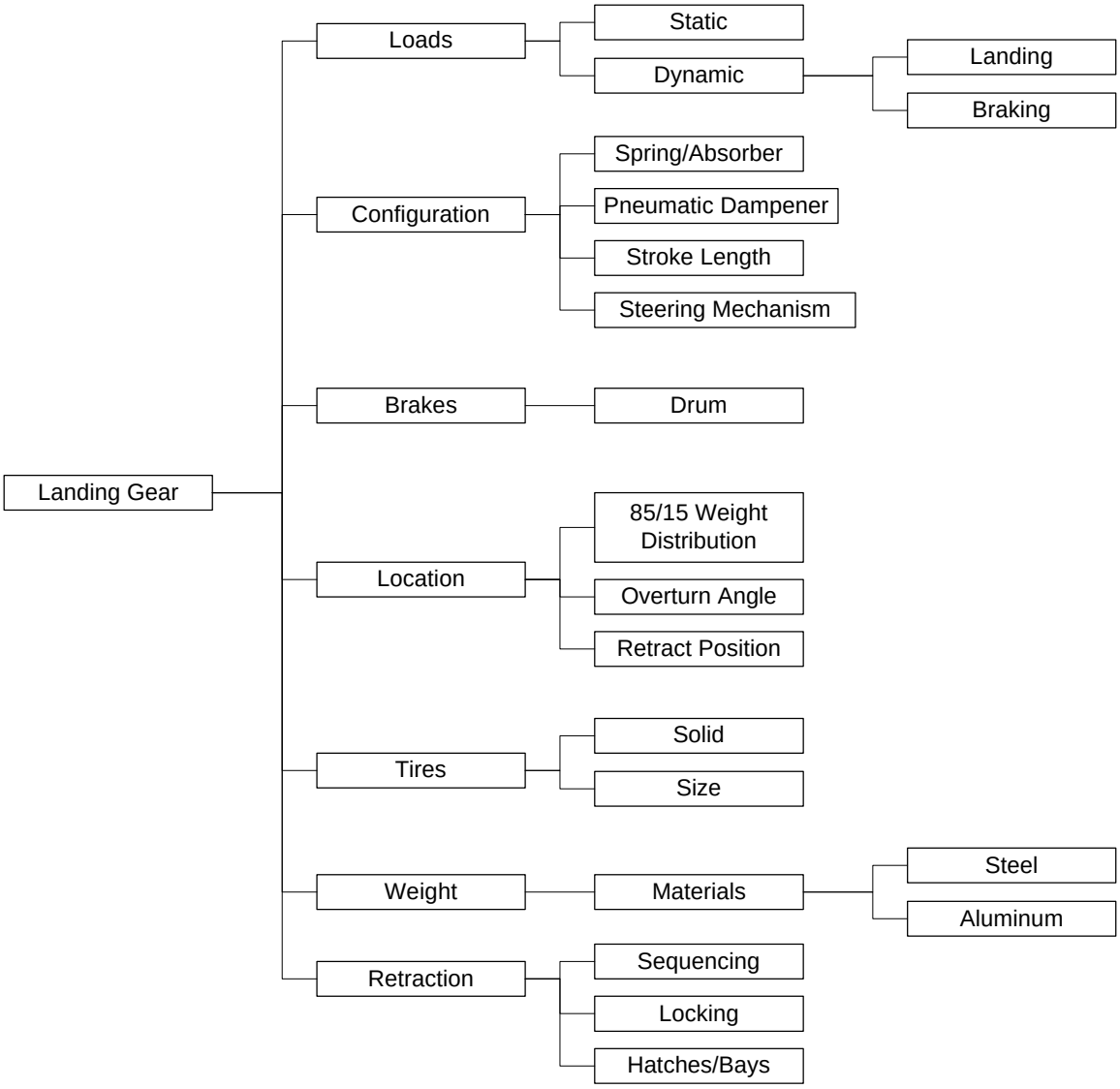
Manufacturing
Break Points/Attachments
Hatch Locations
Maintenance/Accessibility
Fuel Tanks
Composite Materials/Resins
Plug/ Molds
Landing Gear
Control Surface Installation
Strain Gauges
C.G./Moment Rigs
Internals
Wiring Harness
Transportability
Hardpoints

Flight Testing
SOP's
Pre/Post Flight Checklists
Battery Charge/Discharge
Start-Up Procedures
Emergency Procedures
Turn Around Time
Control Surface Calibration
Drop Tests
Taxi Test
Low Speed
High Speed

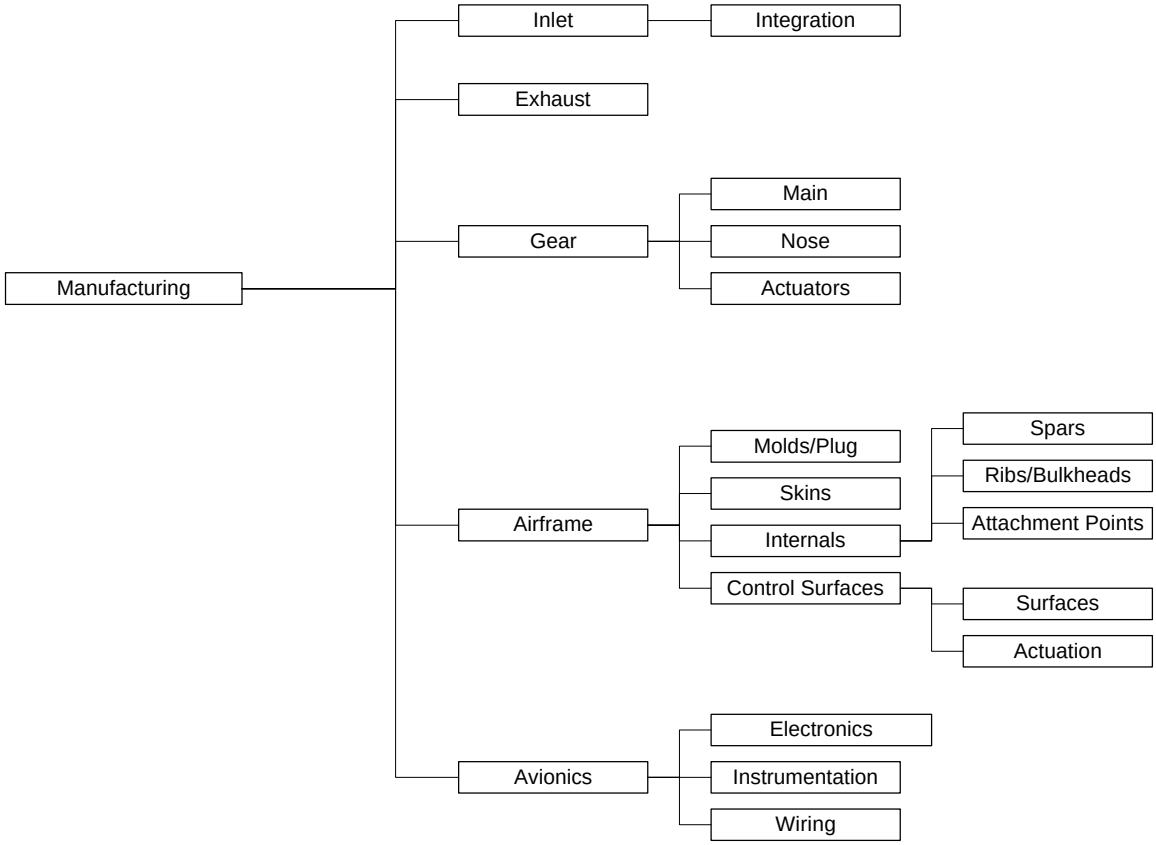
VIPER UAV Affinity Diagrams



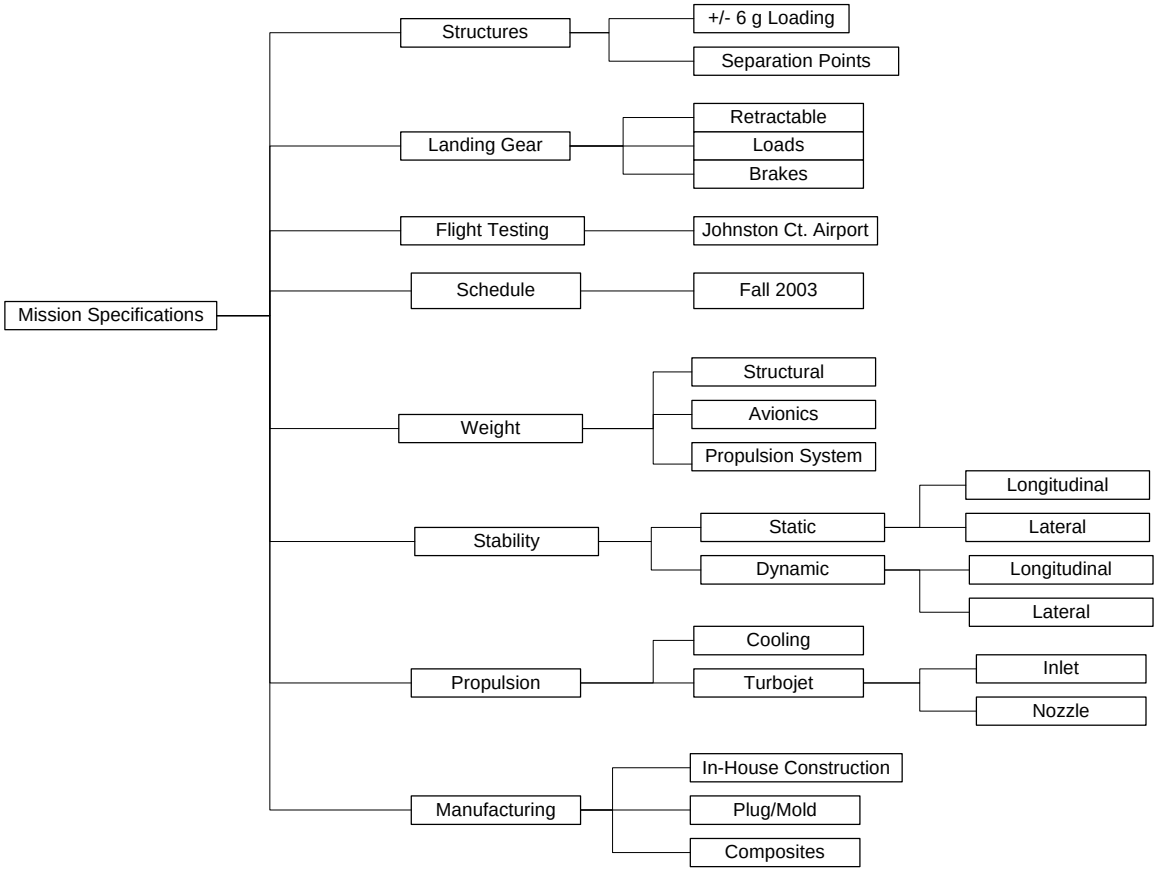
***VIPER* UAV Interrelationship Diagram**



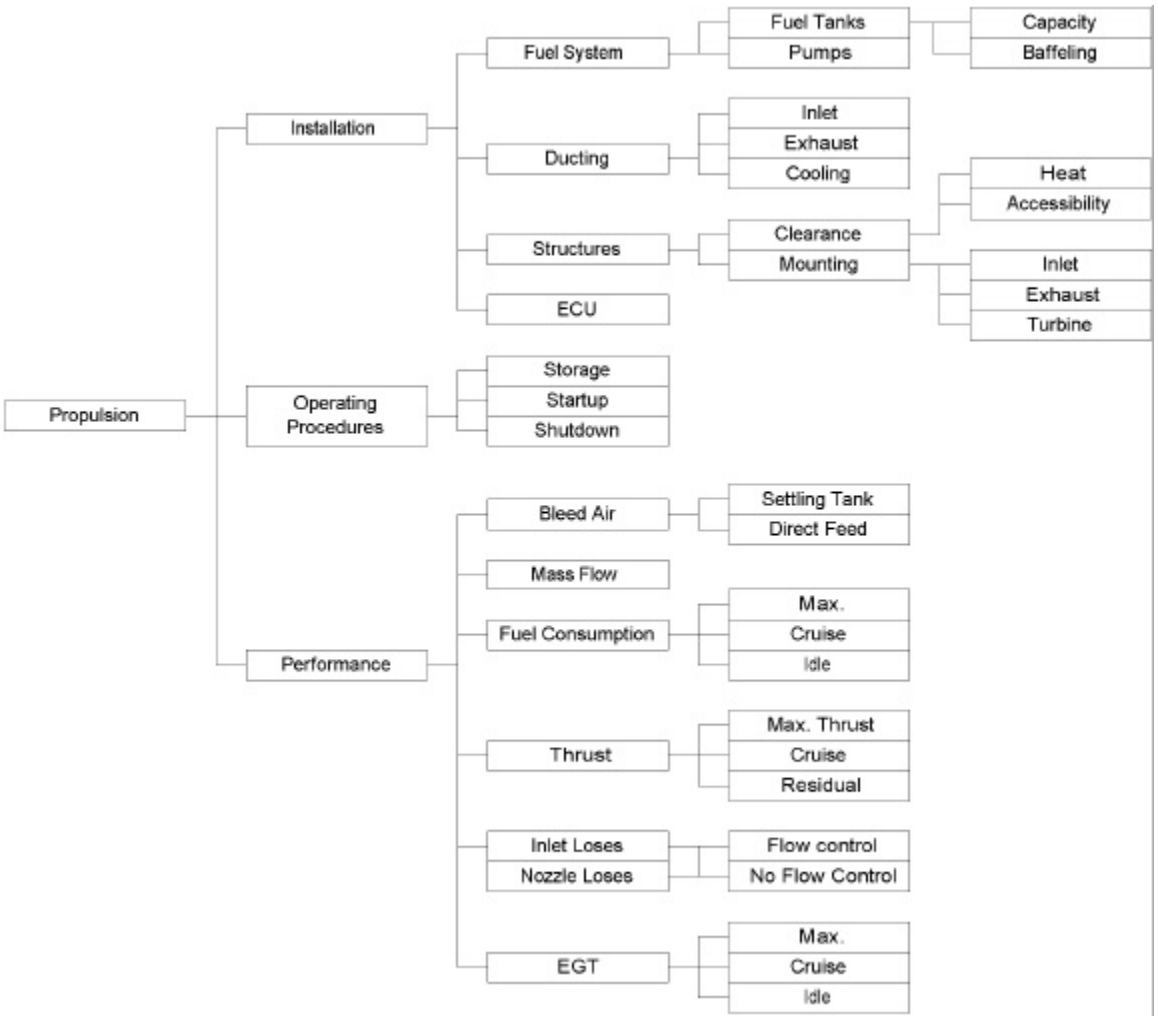
***VIPER* UAV Landing Gear Tree Diagram**



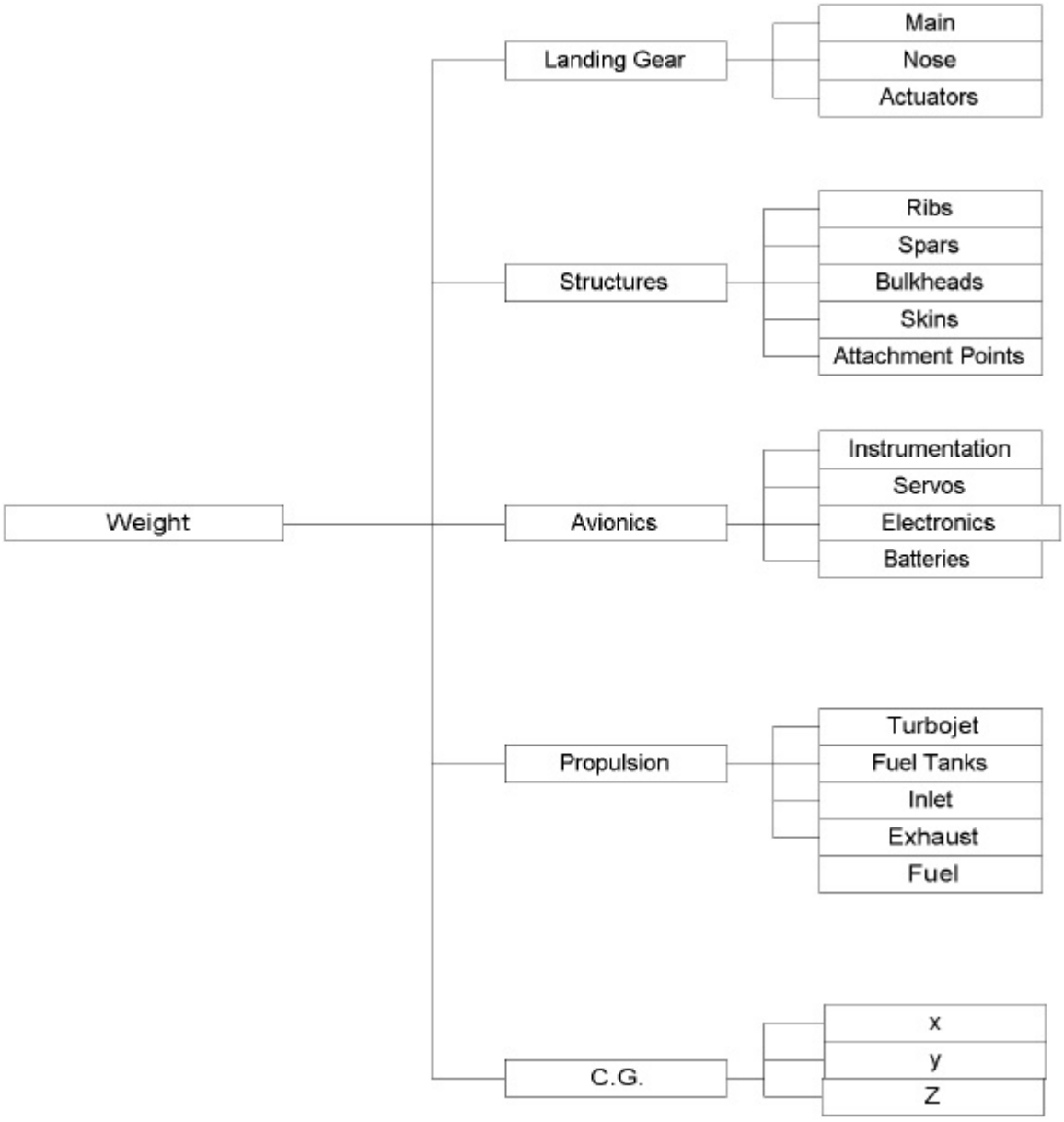
***VIPER* UAV Manufacturing Tree Diagram**



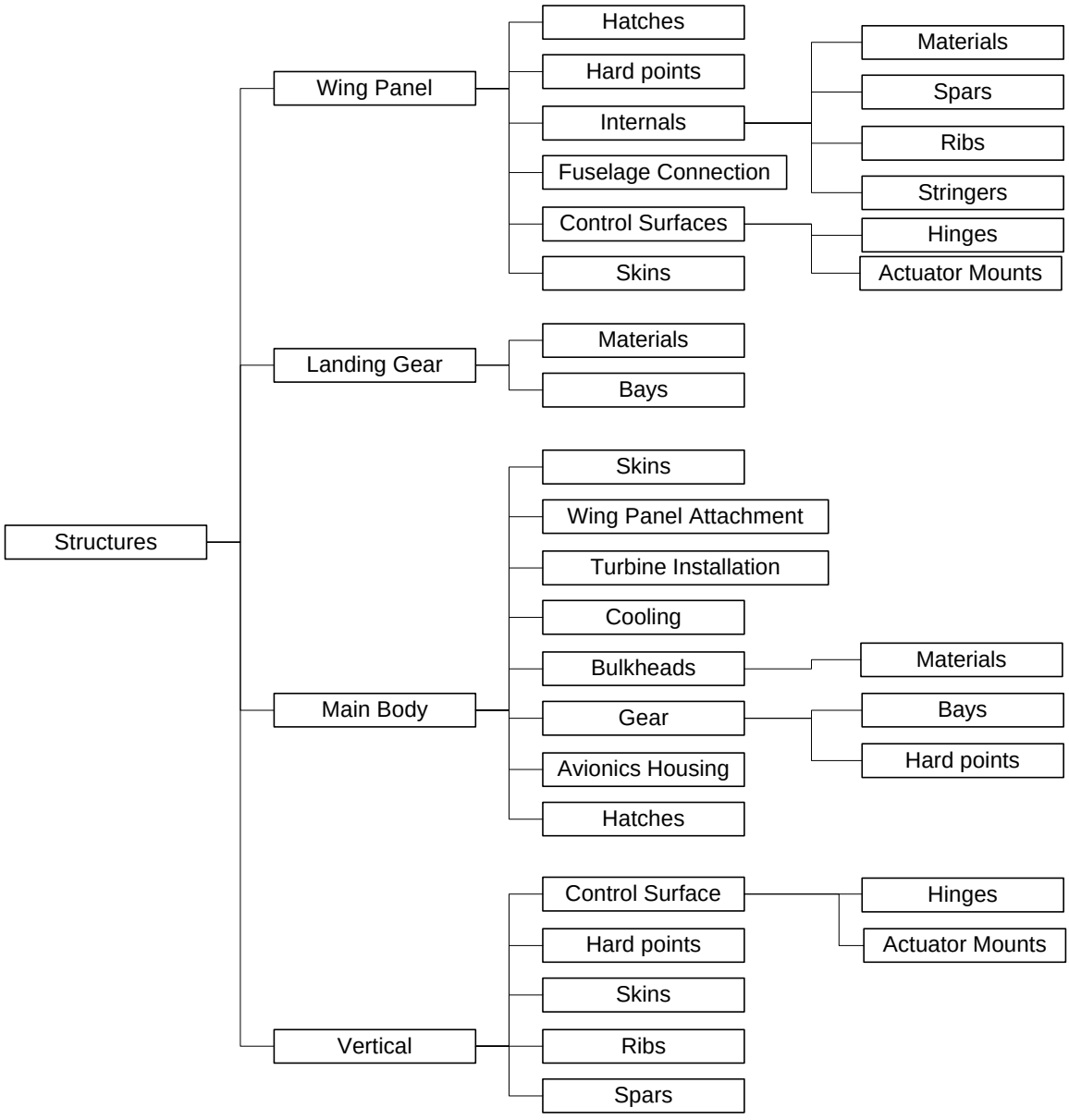
VIPER UAV Mission Specifications Tree Diagram



***VIPER* UAV Propulsion Tree Diagram**



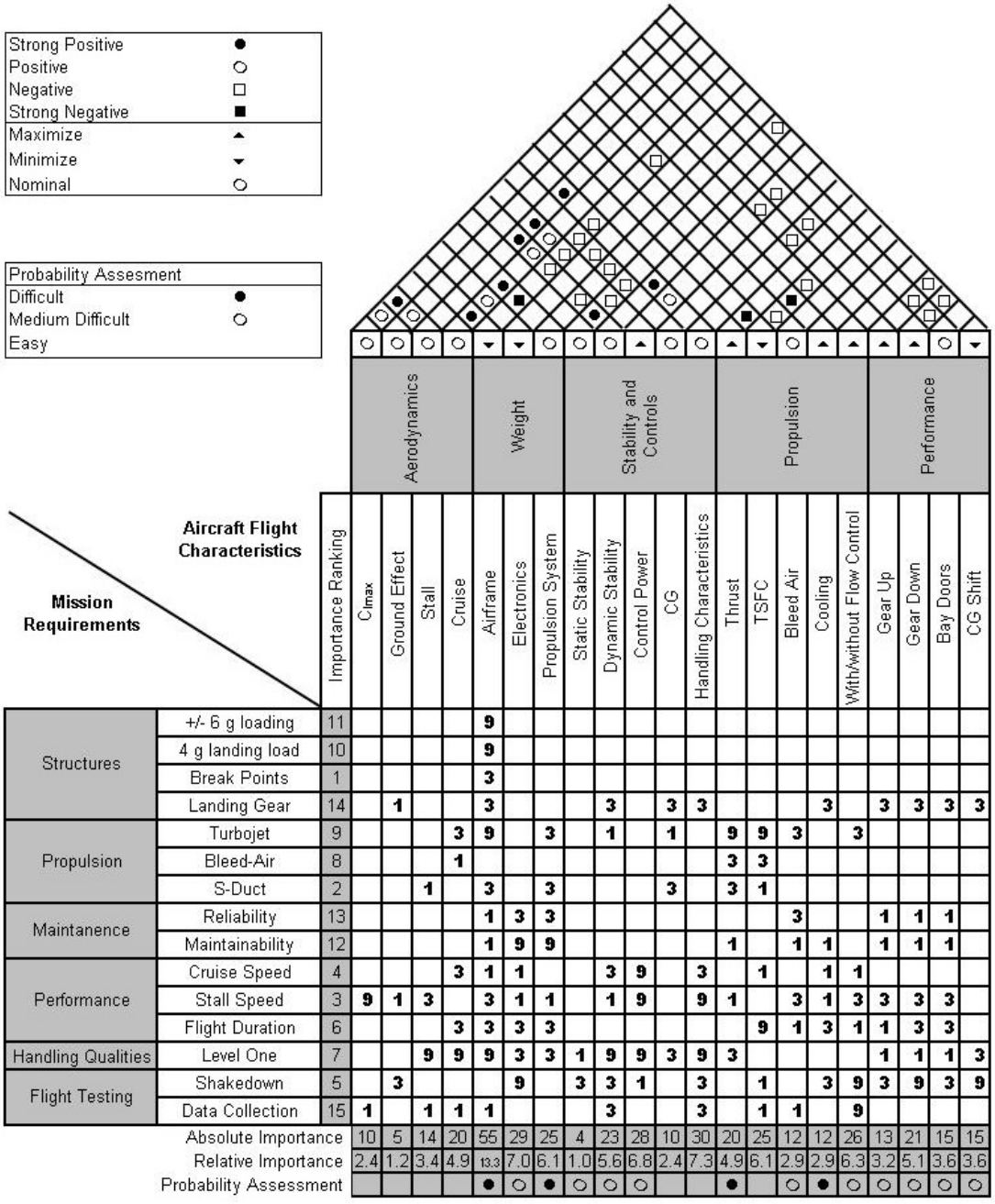
***VIPER* UAV Weight Tree Diagram**



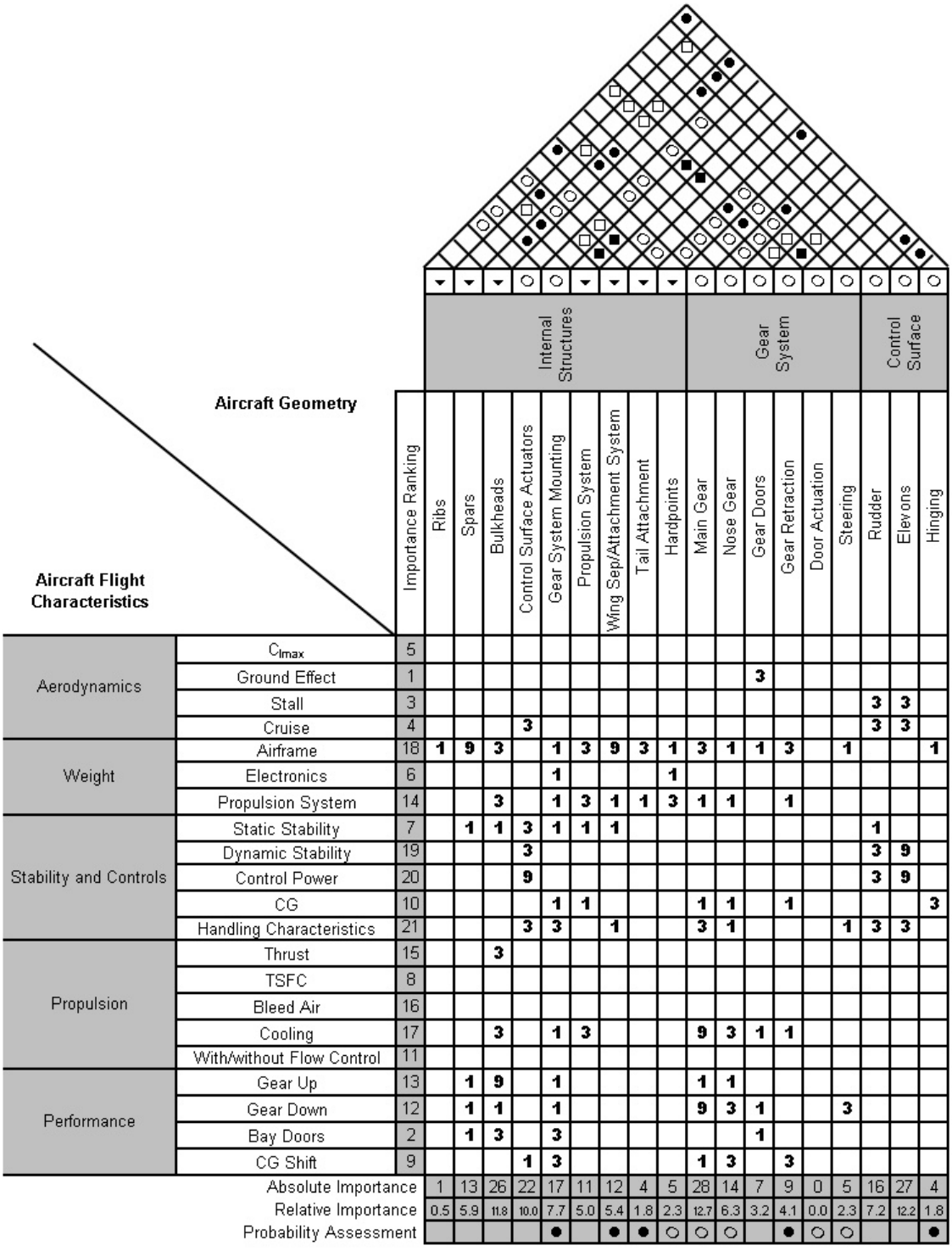
***VIPER* UAV Structures Tree Diagram**

			Stability and Controls					Landing Gear					Performance					Structures					Propulsion				
		Importance Ranking	Static Stability	Dynamic Stability	Control Surfaces	Handling	CG	Location	Loading	Materials	Retraction	Weight	Landing Distance	Takeoff Distance	Cruise Speed	Stall Speed	Endurance	Accessibility	Weight	Turbine Installation	Break Points	Landing Gear Mounting	Inlet Losses	Exhaust losses	Fuel Consumption	Turbine Cooling	Installation Weight
Stability and Controls	Static Stability	21					9																		1		3
	Dynamic Stability	13			3	9	3					3							3								3
	Control Sufaces	11		3		9	3												9								1
	Handling	6		9	9		9					3							3					1			3
	CG	2	9	3	3	9		9		1	3	9							9	1	9	3					9
Landing Gear	Location	15					9		3		3								1	1	3	1					
	Loading	10						3			1					1			9			3					9
	Materials	17					1		1		1	1	3						9			1					3
	Retraction	7					3	3	1	1		9						1		3	9	3				3	
	Weight	4		3		3	9			1	9								9	1	3	3					1
Performance	Landing Distance	14								3						9			9								3
	Takeoff Distance	12														9			9				3	3			1
	Cruise Speed	23																	3	3			3	3			1
	Stall Speed	8							1				9	9					9				3	3			1
	Endurance	16																	9				3	3	3	1	1
Structures	Accessibility	25									1								1	1	1	1					1
	Weight	1		3	9	3	9	1	9	9		9	9	9	3	9	9	1		1	3	3			1	1	3
	Turbine Installation	20					1	1			3	1			3			1	1								3
	Break Points	5					9	3			9	3						1	3			9					
	Landing Gear Mounting	9					3	1	3	1	3	3						1	3		9					1	
Propulsion	Inlet losses	19												3	3	3	3							1	3		
	Exhaust Losses	18				1								3	3	3	3						1		3		
	Fuel Consumption	22	1														3		1				3	3		1	1
	Turbine Cooling	24									3						1		1			1			1		
	Installation Weight	3	3	3	1	3	9		9	3		1	3	1	1	1	1	1	3	3					1		
	Absolute Importance		13	24	25	37	77	21	27	19	36	42	24	25	13	35	20	6	104	14	37	28	16	17	13	7	47
	Relative Importance		1.79	3.3	3.44	5.09	10.6	2.89	3.71	2.61	4.95	5.78	3.3	3.44	1.79	4.81	2.75	0.83	14.3	1.93	5.09	3.85	2.2	2.34	1.79	0.96	6.46

***VIPER* UAV Prioritization Matrix**

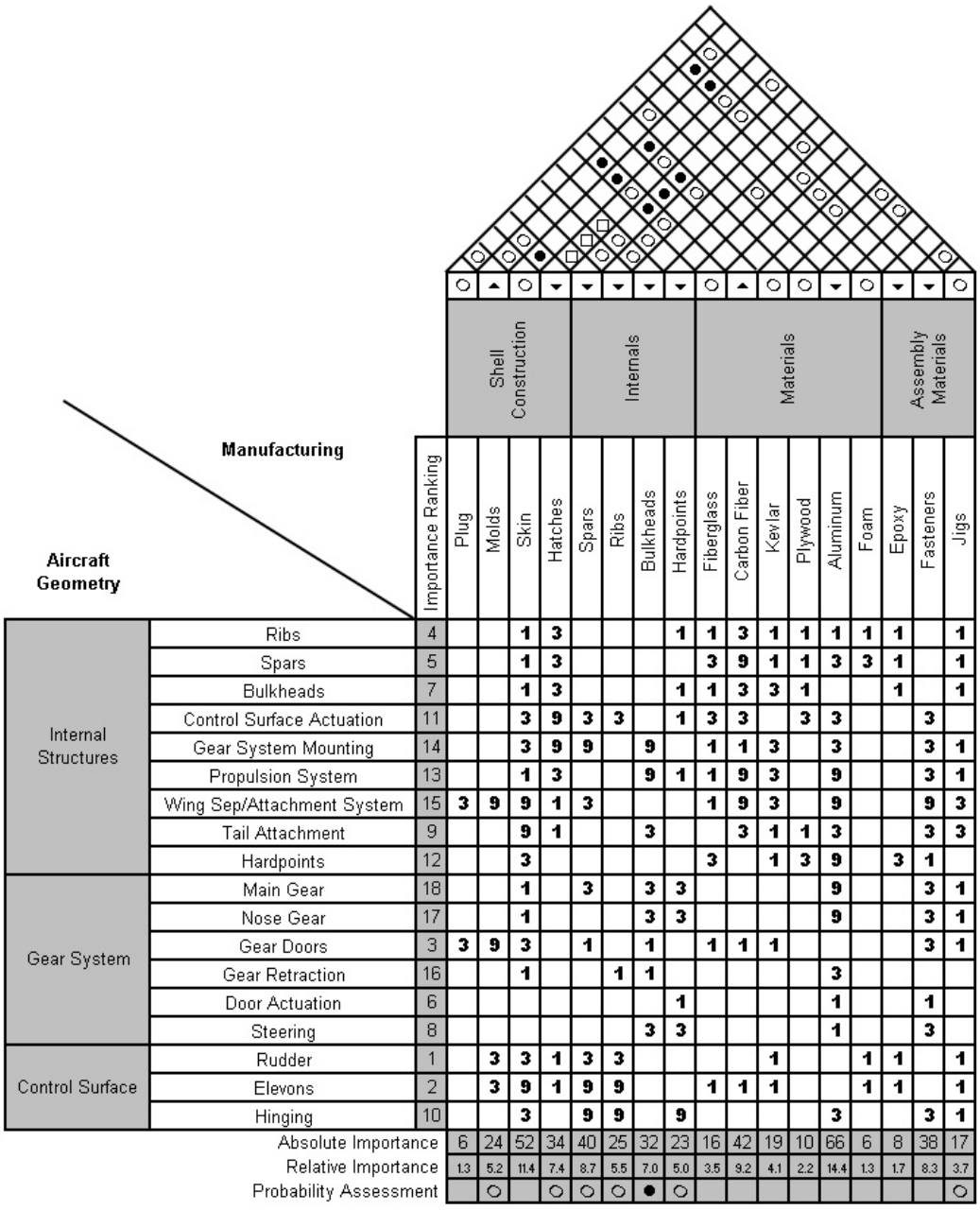


***VIPER* UAV Quality Functional Deployment Matrix #1**

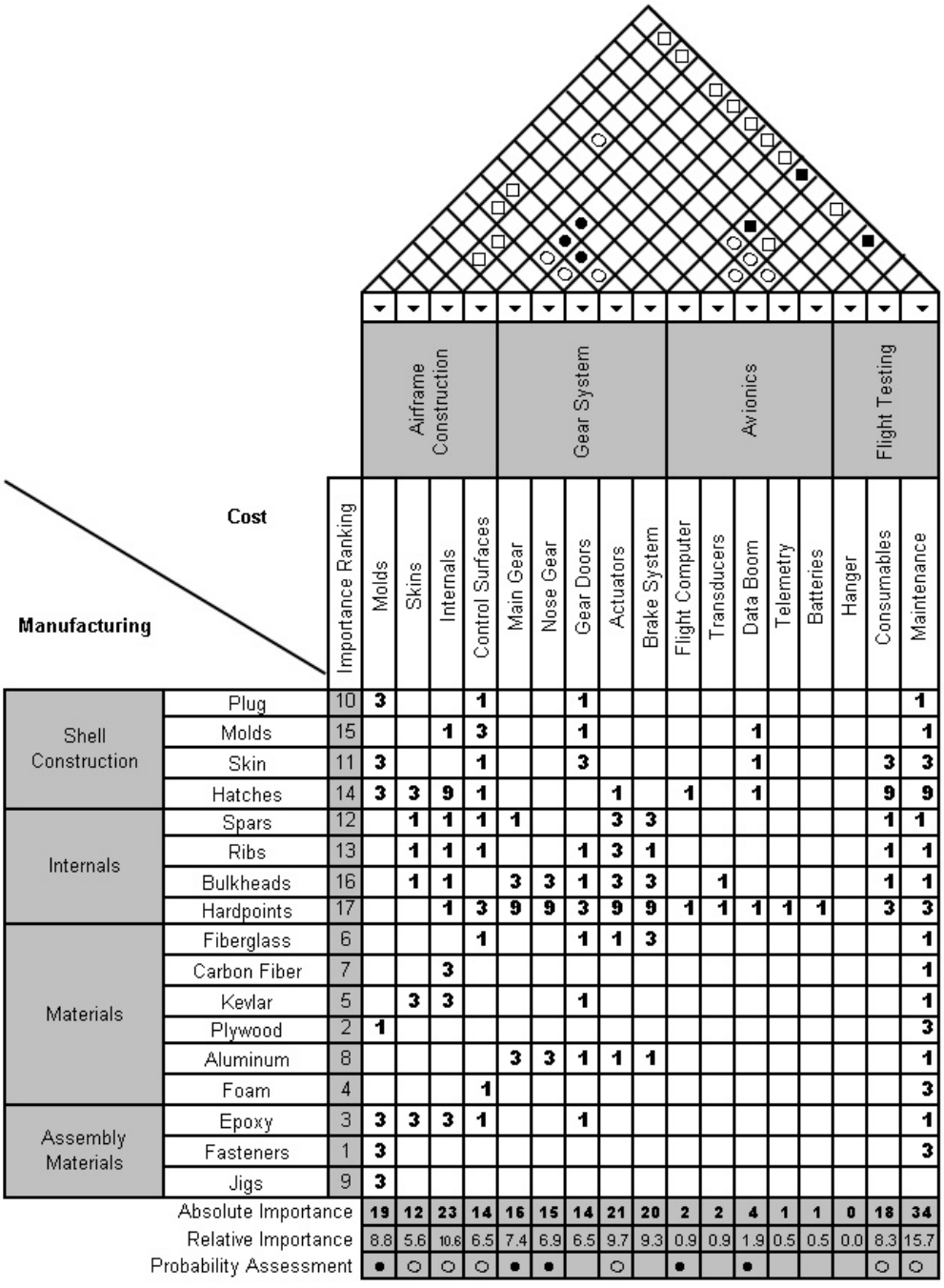


Aircraft Flight Characteristics		Importance Ranking	Internal Structures: Ribs	Spars	Bulkheads	Control Surface Actuators	Gear System Mounting	Propulsion System	Wing Sep/Attachment System	Tail Attachment	Hardpoints	Gear System: Main Gear	Nose Gear	Gear Doors	Gear Retraction	Door Actuation	Steering	Control Surface: Rudder	Elevons	Hinging
Aerodynamics	C_{lmax}	5																		
	Ground Effect	1												3						
	Stall	3																3	3	
	Cruise	4				3												3	3	
Weight	Airframe	18	1	9	3		1	3	9	3	1	3	1	1	3		1			1
	Electronics	6					1				1									
	Propulsion System	14			3		1	3	1	1	3	1	1		1					
Stability and Controls	Static Stability	7		1	1	3	1	1	1									1		
	Dynamic Stability	19				3												3	9	
	Control Power	20				9												3	9	
	CG	10					1	1				1	1		1					3
	Handling Characteristics	21				3	3		1			3	1				1	3	3	
Propulsion	Thrust	15			3															
	TSFC	8																		
	Bleed Air	16																		
	Cooling	17			3		1	3				9	3	1	1					
	With/without Flow Control	11																		
Performance	Gear Up	13		1	9		1					1	1							
	Gear Down	12		1	1		1					9	3	1			3			
	Bay Doors	2		1	3		3							1						
	CG Shift	9				1	3					1	3		3					
	Absolute Importance		1	13	26	22	17	11	12	4	5	28	14	7	9	0	5	16	27	4
	Relative Importance		0.5	5.9	11.8	10.0	7.7	5.0	5.4	1.8	2.3	12.7	6.3	3.2	4.1	0.0	2.3	7.2	12.2	1.8
	Probability Assessment						●		●	●	○	○	○		●	○	○			●

***VIPER* UAV Quality Functional Deployment Matrix #2**



***VIPER* UAV Quality Functional Deployment Matrix #3**



Manufacturing \ Cost		Importance Ranking	Molds	Skins	Internals	Control Surfaces	Main Gear	Nose Gear	Gear Doors	Actuators	Brake System	Flight Computer	Transducers	Data Boom	Telemetry	Batteries	Hanger	Consumables	Maintenance
			Airframe Construction				Gear System					Avionics					Flight Testing		
Shell Construction	Plug	10	3			1			1										1
	Molds	15			1	3			1					1					1
	Skin	11	3			1			3					1				3	3
	Hatches	14	3	3	9	1				1		1		1				9	9
Internals	Spars	12		1	1	1	1			3	3							1	1
	Ribs	13		1	1	1			1	3	1							1	1
	Bulkheads	16		1	1		3	3	1	3	3		1					1	1
	Hardpoints	17			1	3	9	9	3	9	9	1	1	1	1	1		3	3
Materials	Fiberglass	6				1			1	1	3								1
	Carbon Fiber	7			3														1
	Kevlar	5		3	3				1										1
	Plywood	2	1																3
	Aluminum	8					3	3	1	1	1								1
	Foam	4				1													3
Assembly Materials	Epoxy	3	3	3	3	1			1										1
	Fasteners	1	3																3
	Jigs	9	3																
	Absolute Importance		19	12	23	14	16	15	14	21	20	2	2	4	1	1	0	18	34
	Relative Importance		8.8	5.6	10.6	6.5	7.4	6.9	6.5	9.7	9.3	0.9	0.9	1.9	0.5	0.5	0.0	8.3	15.7
	Probability Assessment		●	○	○	○	●	●		○		●		●				○	○

***VIPER* UAV Quality Functional Deployment Matrix #4**

10.2. Material Properties for Shell 99

```
! ======================
! Material Properties
! ======================
!Material Property, Property, Material Number, Value

!Material Properties of Carbon
MAT,1                                   !Material Property Used
MP,EX,1,8.68E+06                        !Modulus of Elasticity
MP,EY,1,8.68E6                          !Modulus of Elasticity
MP,EZ,1,8.68E+06                        !Modulus of Elasticity
MP,PRXY,1,0.119                         !Poison's Ratio
MP,PRYZ,1,0.119                         !Poison's Ratio
MP,PRXZ,1,0.119                         !Poison's Ratio
MP,GXY,1,3.88E+06                       !Shear Modulus
MP,GYZ,1,3.88E+06                       !Shear Modulus
MP,GXZ,1,3.88E+06                       !Shear Modulus
MP,DENS,1,0.045                         !Density
!Thickness for single layer=.0078


!Material Properties of Fiberglass
MAT,2                                   !Material Property Used
MP,EX,2,2.76E+06                        !Modulus of Elasticity
MP,EY,2,2.76E+06                        !Modulus of Elasticity
MP,EZ,2,2.76E+06                        !Modulus of Elasticity
MP,PRXY,2,.122                          !Poison's Ratio
MP,PRYZ,2,.122                          !Poison's Ratio
MP,PRXZ,2,.122                          !Poison's Ratio
MP,GXY,2,1.22E+06                       !Shear Modulus
MP,GYZ,2,1.22E+06                       !Shear Modulus
MP,GXZ,2,1.22E+06                       !Shear Modulus
MP,DENS,2,0.057                         !Density
!Thickness for single layer=.0032


!Material Properties of Korex
MAT,3                                   !Material Property Used
MP,EX,3,7000                            !Modulus of Elasticity
MP,EY,3,7000                            !Modulus of Elasticity
MP,EZ,3,7000                            !Modulus of Elasticity
MP,PRXY,3,-0.7941                       !Poison's Ratio
```

```
MP,PRYZ,3,-0.7941                !Poison's Ratio
MP,PRXZ,3,-0.7941                !Poison's Ratio
MP,GXY,3,17000                   !Shear Modulus
MP,GYZ,3,17000                   !Shear Modulus
MP,GXZ,3,17000                   !Shear Modulus
MP,DENS,3,1.11E-02               !Density


!Material Properties of CARBON-GALSS Korex
MAT,5                            !Material Property Used
MP,EX,5,1.04E+06                 !Modulus of Elasticity
MP,EY,5,1.04E+06                 !Modulus of Elasticity
MP,EZ,5,1.04E+06                 !Modulus of Elasticity
MP,PRXZ,5,0.088                  !Poison's Ratio
MP,PRXY,5,0.088                  !Poison's Ratio
MP,PRYZ,5,0.088                  !Poison's Ratio
MP,GXY,5,4.78E+05                !Shear Modulus
MP,GXZ,5,4.78E+05                !Shear Modulus
MP,GYZ,5,4.78E+05                !Shear Modulus
MP,DENS,5,.00173                 !Density



! ===============
! Element Type
! ===============
!Element type, Element #,Element type
ET,1,SHELL99

!Key Options, Element Type, KeyOpt #, 0=Constant Thickness
KEYOPT,1,2,0

!Real Constant, Real Constant Set#, Number of Layers, Layer symmetric about ?
R,1,5,0

!Modify Real Constant, Real Constant Set#,  BLA  , Material, Theta, Thickness Layer 1,
MAterial, Theta, Thinkness Layer 2
RMODIF,1,13,        2,0,0.0032,          1,0,0.0032,
RMODIF,1,19,        3,0,.125,            1,0,0.0032,
RMODIF,1,25,        2,0,0.0032,
```

10.3. Merging CMARC to ANSYS FORTRAN Code

```
PROGRAM CMARCtoANSYS
!-------------------------------------------------------------!
! Programmer:     Eric N. Burcsu        06-Jul-99             !
!                 October 29, 1995                            !
! Modified by:    Michael J. Worsham    June 11, 1999         !
!                                                             !
! Converted to read CMARC aerodynamics files by:              !
!                 Demterius Siachames   November 2000         !
!                                                             !
! Converted to FORTRAN90 and modified to a new                !
!   interpolation scheme by:                                  !
!                 Rob S. Burgun           August 2002         !
!-------------------------------------------------------------!
! This program takes aerodynamic data from CMARC output and   !
! interpolates the pressures to ANSYS finite element          !
! node points.                                                !
!-------------------------------------------------------------!

IMPLICIT NONE
real,DIMENSION(5,1000000):: aero_coord !pmarc coords and pressures
real,DIMENSION(4,1000000):: node_coord !ansys node coordinates
real,DIMENSION(5,1000000):: final_data  !ansys coords and interpolated pressures
integer,DIMENSION(10,1000000):: eset !ansys element-node definition

integer,DIMENSION(1000000):: node_label !node labels
integer,DIMENSION(1000000):: elem_label !element labels

real,DIMENSION(3,3)::plane_coord
integer:: axis

integer:: node_num     !number of ansys node points
integer:: elem_num     !number of ansys elements
integer:: aero_num     !number of cmarc panels

real   :: Cl     !Lift coefficielem_numt from cmarc


CALL Get_seperation_plane(plane_coord,axis)

CALL ansread(node_coord,eset,node_label,elem_label,node_num,elem_num,plane_coord,axis)

CALL CMARC_read(aero_coord,aero_num,Cl,plane_coord,axis)
!CALL Cread(aero_coord,aero_num,Cl,plane_coord,axis)

CALL interpolate(node_coord,node_label,aero_coord,final_data,node_num,aero_num,Cl)

CALL output(final_data,eset,elem_label,elem_num)

CONTAINS

!*****************************************************************************

SUBROUTINE Cread(aero_coord,panel,CL,plane_coord,axis)
!-----------------------------------------------------------------------------
!This subroutine reads Cmarc data and puts it into an array
!The Cmarc data must be unmodified and all spacing in the file
!must be as originally created by Cmarc.
!   aero_coord = array of Cmarc pressure [x pos, y pos, z pos, Cp, above/below sep plane]
!   panel      = number of Cmarc panels
!-----------------------------------------------------------------------------
   IMPLICIT NONE
   REAL,INTENT(INOUT),DIMENSION(:,:)::aero_coord
   INTEGER,INTENT(INOUT)::panel
   REAL,INTENT(INOUT) :: Cl
   real,INTENT(IN),DIMENSION(:,:)::plane_coord
```

```
   INTEGER, INTENT(IN):: axis


   CHARACTER(7)::dummy
   CHARACTER(50)::filename
   CHARACTER(5)::hold
   CHARACTER(1)::test,limiter,spacer
   CHARACTER(7)::tempdummy
   CHARACTER(45)::longdummy
   INTEGER::something
   INTEGER::above


      !User input of data file
   WRITE(*,'(A)',ADVANCE = "NO") ' Enter filelname for Cmarc data: '
   READ(*,*) filename

   panel=0

      !OPEN Cmarc file
   OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')

      !Moves the read location to the begining of the pressure data
   DO
      READ(45,'(A7)') dummy
      IF (dummy==' AERODY') EXIT
   END DO

      !Variables set up for purposes of filtering data
   limiter='9'  !One ASCII character before "A"
   spacer='!'  !One ASCII character after "space"

      !Filters the pressure from the other text and numbers in the file
   DO
      BACKSPACE(45)
         !Reads a line from the data file. "hold" is just for place keeping
      READ(45,'(A5,A1)') hold,test

         !Checks to see if test is a character
      IF (LGE(test,limiter)) THEN
         BACKSPACE(45)

            !Puts the value of the string in to "dummy"
         READ(45,'(A7)') dummy
      END IF

         !Checks to see if "test" is a number
      IF (LGE(limiter,test).AND.LGE(test,spacer)) THEN
         BACKSPACE(45)
         panel=panel+1
            !If it is a number, set the pressure array
         READ(45,*)
something,aero_coord(1,panel),aero_coord(2,panel),aero_coord(3,panel),aero_coord(4,panel)
         CALL
seperator(plane_coord,aero_coord(1,panel),aero_coord(2,panel),aero_coord(3,panel),axis,above)
         aero_coord(5,panel) = above
      END IF

         !Filters out the region of the data file betweelem_num "SECTION PARAMETERS"
         !and "PANEL" and final_datads the END of the data.
      IF (LGE(test,limiter).AND.(dummy==' SECTIO')) THEN
         DO
            IF ((dummy==' PANEL ').OR.(dummy=='1      ')) EXIT
            read(45,'(A7)') dummy
         END DO
      END IF
      read(45,'(A45,A7)') longdummy,tempdummy
         !This denotes the END of the pressure data
      IF (tempdummy=='*******') EXIT
```

```
   END DO
   write(*,*) panel

   DO
      read(45,'(A45)') longdummy
      IF (longdummy(31:45) =='TOTAL COEFFICIE') THEN
         read(45,'(A45)') longdummy
         read(45,'(A45)') longdummy
         read(45,'(A45)') longdummy
         read(45,'(A45)') longdummy
         read(longdummy(35:40),*) Cl
         write(*,*) 'Cl (from Cmarc) = ',Cl
         EXIT
      END IF
   END DO
END SUBROUTINE Cread


SUBROUTINE CMARC_read(aero_coord,panel,CL,plane_coord,axis)
!-------------------------------------------------------------------------------
!This subroutine reads Cmarc data and puts it into an array
!The Cmarc data must be unmodified and all spacing in the file
!must be as originally created by Cmarc.
!   aero_coord = array of Cmarc pressure [x pos, y pos, z pos, Cp, above/below sep plane]
!   panel      = number of Cmarc panels
!-------------------------------------------------------------------------------

   IMPLICIT NONE
   REAL,INTENT(INOUT),DIMENSION(:,:)::aero_coord
   INTEGER,INTENT(INOUT)::panel
   REAL,INTENT(INOUT) :: Cl
   REAL,INTENT(INOUT),DIMENSION(:,:)::Plane_coord
   INTEGER, INTENT(IN):: axis

   CHARACTER(50)::filename
   CHARACTER(60)::temp
   INTEGER::above
   INTEGER:: i
   INTEGER:: counter
   INTEGER:: step


      !User input of data file
   WRITE(*,'(A)',ADVANCE = "NO") ' Enter filelname for Cmarc data: '
   READ(*,*) filename
   OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')

   counter = 0
   DO
      read(45,'(A50)') temp
      IF (temp(2:6) =='PANEL') THEN
         read(45,'(A1)') temp
         step = 1
         DO
             counter = counter + 1
            READ(45,*)
panel,aero_coord(1,panel),aero_coord(2,panel),aero_coord(3,panel),aero_coord(4,panel)
            CALL
seperator(plane_coord,aero_coord(1,panel),aero_coord(2,panel),aero_coord(3,panel),axis,above)
            aero_coord(5,panel) = above
            step = step + 1
            read(45,'(A7)') temp
            IF (temp == '') THEN
               !found the end of a sequence of panels
               counter = counter - 1
               EXIT
            END IF
            BACKSPACE(45)
         END DO
```

```
      ELSE IF (temp(31:45) =='TOTAL COEFFICIE') THEN
         read(45,'(A1)') temp
         read(45,'(A1)') temp
         read(45,'(A1)') temp
         read(45,'(A45)') temp
         read(temp(35:40),*) Cl
         write(*,*) 'Cl (from Cmarc) = ',Cl
         EXIT
      END IF
      IF (EOF(45)) THEN
          exit
      END IF
   END DO
   write(*,*) panel,counter
END SUBROUTINE CMARC_read

!*******************************************************************************

SUBROUTINE interpolate(node_coord,node_label,aero_coord,final_data,node_num,aero_num,Cl)
!-------------------------------------------------------------------------------
!This subroutine final_datads the closest Cmarc pressure point to an ANSYS point.
!There is no specified radius of convergelem_numce, so this subroutine works well
!for a variable spacing mesh in either Cmarc of ANSYS. The limitations of
!the program are inherelem_numt in its inablility to deal with total airplane
!geometries at once. The trailing (and leading) edge of an airfoil must
!be split by the y-z plane to elem_numsure that pressures are assigned to the
!proper surface (top versus bottom).
!
!   node_coord   =   array of the ANSYS nodes [node x, node y, node z]
!   node_label   =   array of the ANSYS node labels [node label]
!   elem_label   =   array of the ANSYS element labels [element label]
!   aero_coord   =   pressure array from Cmarc [x pos, y pos, z pos, Cp]
!   final_data   =   ANSYS nodes and associated Cp and pressures
!            [x pos, y pos, z pos, Cp, pressure]
!   node_num   =   number of ANSYS nodes
!   elem_num   =   number of ANSYS elements
!   aero_num   =   number of Cmarc panels
!   CL  =   lift coefficielem_numt from Cmarc
!-------------------------------------------------------------------------------

   IMPLICIT NONE
   REAL,DIMENSION(:,:),intent(INOUT)::aero_coord
   REAL,DIMENSION(:,:),intent(INOUT)::node_coord
   REAL,DIMENSION(:,:),intent(INOUT)::final_data
   INTEGER,DIMENSION(:):: node_label
   REAL::den,q                      !density, dynamic pressure,
   INTEGER::aero_num,node_num
   integer::I,J

   real,DIMENSION(4,2)::corner
   integer,DIMENSION(4)::corner_node
   real,DIMENSION(4)::radius
   real :: temp_radius
   real,DIMENSION(4,2)::mid_side
   real,DIMENSION(4)::area
   real:: Cp
   real :: temp
   integer::percentage_1
   integer::percentage_1_old
   integer::percentage_10
   integer::percentage_10_old
   integer::cube

   real :: Cl
   real :: weight
   real :: surface_area
   integer:: number_of_gs
```

```
   !User inputs to calculate dynamic pressure
write(*,'(A)',ADVANCE = "NO") ' Enter the Weight of the plane (lbs.): '
read(*,*) weight
write(*,'(A)',ADVANCE = "NO") ' Enter the Surface Area of the plane (ft^2): '
read(*,*) surface_area



   !Dynamic pressure calculation
den = 0.0023769
q=weight/(surface_area*Cl*144)

cube = 6
percentage_1_old = 0
percentage_10_old = 0
write(*,*) 'Interpolation process ...'
50 FORMAT(I5,A)

DO I=1,node_num
      !initialize radius of convergelem_numce
   corner(1,1) = -10000.0   !x- , y-
   corner(1,2) = -10000.0
   corner(2,1) = -10000.0   ! x- , y+
   corner(2,2) =  10000.0
   corner(3,1) =  10000.0   ! x+ , y-
   corner(3,2) = -10000.0
   corner(4,1) =  10000.0   ! x+ , y+
   corner(4,2) =  10000.0
   radius(:) = 10000.0

      !find closest four Cmarc points with the ANSYS node "I" in the center
   DO J=1,aero_num
         ! only process data which is on the same side of the
         ! seperation plane
      IF ((node_coord(4,node_label(i)) == aero_coord(5,j)) .AND. &
         &(aero_coord(1,j) >= (node_coord(1,node_label(i)) - (cube*1.0))) .AND. &
         &(aero_coord(1,j) <= (node_coord(1,node_label(i)) + (cube*1.0))) .AND. &
         &(aero_coord(2,j) <= (node_coord(2,node_label(i)) + (cube*1.0))) .AND. &
         &(aero_coord(2,j) >= (node_coord(2,node_label(i)) - (cube*1.0)))) THEN
         IF ((aero_coord(1,j) <= node_coord(1,node_label(i))) .AND. &
           & (aero_coord(2,j) <= node_coord(2,node_label(i))) .AND. &
           & (aero_coord(1,j) >= corner(1,1)).AND.(aero_coord(2,j) >= corner(1,2))) THEN
            IF (((aero_coord(2,j) >= 0).AND.(node_Coord(2,node_label(i)) >= 0)).OR. &
               &((aero_coord(2,j) <= 0).AND.(node_Coord(2,node_label(i)) <= 0))) THEN
               temp_radius = sqrt(((node_coord(1,node_label(i)) + aero_coord(1,j))/2)**2+&
                             & ((node_coord(2,node_label(i)) + aero_coord(2,j))/2)**2 )
               IF (temp_radius <= radius(1)) THEN
                     ! found a new node for corner 1
                  corner(1,1) = aero_coord(1,j)
                  corner(1,2) = aero_coord(2,j)
                  corner_node(1) = j
                  radius(1) = temp_radius
               END IF
            END IF
         ELSE IF ((aero_coord(1,j) <= node_coord(1,node_label(i))) .AND. &
           & (aero_coord(2,j) >= node_coord(2,node_label(i))) .AND. &
           & (aero_coord(1,j) >= corner(2,1)).AND.(aero_coord(2,j) <= corner(2,2))) THEN
            IF (((aero_coord(2,j) >= 0).AND.(node_Coord(2,node_label(i)) >= 0)).OR. &
               &((aero_coord(2,j) <= 0).AND.(node_Coord(2,node_label(i)) <= 0))) THEN
               temp_radius = sqrt(((node_coord(1,node_label(i)) + aero_coord(1,j))/2)**2+&
                             & ((node_coord(2,node_label(i)) + aero_coord(2,j))/2)**2 )
               IF (temp_radius <= radius(2)) THEN
                     ! found a new node for corner 2
                  corner(2,1) = aero_coord(1,j)
                  corner(2,2) = aero_coord(2,j)
                  corner_node(2) = j
                  radius(2) = temp_radius
               END IF
            END IF
```

```
            ELSE IF ((aero_coord(1,j) >= node_coord(1,node_label(i))) .AND. &
              & (aero_coord(2,j) <= node_coord(2,node_label(i))) .AND. &
              & (aero_coord(1,j) <= corner(3,1)).AND.(aero_coord(2,j) >= corner(3,2))) THEN
               IF (((aero_coord(2,j) >= 0).AND.(node_Coord(2,node_label(i)) >= 0)).OR. &
                  &((aero_coord(2,j) <= 0).AND.(node_Coord(2,node_label(i)) <= 0))) THEN
                  temp_radius = sqrt(((node_coord(1,node_label(i)) + aero_coord(1,j))/2)**2+&
                                & ((node_coord(2,node_label(i)) + aero_coord(2,j))/2)**2 )
                  IF (temp_radius <= radius(3)) THEN
                        ! found a new node for corner 3
                     corner(3,1) = aero_coord(1,j)
                     corner(3,2) = aero_coord(2,j)
                     corner_node(3) = j
                     radius(3) = temp_radius
                  END IF
               END IF
            ELSE IF ((aero_coord(1,j) >= node_coord(1,node_label(i))) .AND. &
              & (aero_coord(2,j) >= node_coord(2,node_label(i))) .AND. &
              & (aero_coord(1,j) <= corner(4,1)).AND.(aero_coord(2,j) <= corner(4,2))) THEN
               IF (((aero_coord(2,j) >= 0).AND.(node_Coord(2,node_label(i)) >= 0)).OR. &
                  &((aero_coord(2,j) <= 0).AND.(node_Coord(2,node_label(i)) <= 0))) THEN
                  temp_radius = sqrt(((node_coord(1,node_label(i)) + aero_coord(1,j))/2)**2+&
                                & ((node_coord(2,node_label(i)) + aero_coord(2,j))/2)**2 )
                  IF (temp_radius <= radius(4)) THEN
                        ! found a new node for corner 4
                     corner(4,1) = aero_coord(1,j)
                     corner(4,2) = aero_coord(2,j)
                     corner_node(4) = j
                     radius(4) = temp_radius
                  END IF
               END IF
            END IF
         END IF
!         END IF
      END DO

      IF ((corner(1,1) == -10000) .OR. (corner(2,1) == -10000) .OR. (corner(3,1) == 10000)
.OR. (corner(4,1) == 10000)) THEN
         IF (((corner(1,1) == -10000) .OR. (corner(4,1) == 10000)) &
               & .AND. (corner(2,1) /= -10000) .AND. (corner(3,1) /= 10000)) THEN
            !Line average between corner 2 and 3
            CALL
Gauss_linear(node_coord(1,node_label(i)),node_coord(2,node_label(i)),corner(2,1),corner(2,2),
aero_coord(4,corner_node(2)),corner(3,1),corner(3,2),aero_coord(4,corner_node(3)),cp)
         ELSE IF (((corner(2,1) == -10000) .OR. (corner(3,1) == 10000)) &
               & .AND. (corner(1,1) /= -10000) .AND. (corner(4,1) /= 10000)) THEN
            !Line average between corner 1 and 4
            CALL
Gauss_linear(node_coord(1,node_label(i)),node_coord(2,node_label(i)),corner(1,1),corner(1,2),
aero_coord(4,corner_node(1)),corner(4,1),corner(4,2),aero_coord(4,corner_node(4)),cp)
         END IF
      ELSE
            ! The following is a diagram of the points used to determine the
            ! pressure for the ansys mesh. The points marked by X are the corner
            ! points. The *'s are the mid-points
            !
            !  (2)        {2}          (4)
            !     X          *          X          ^y
            !                                      |
            !         A3        A1                |
            !                                      |
            ! {1} *         o          * {4}       |
            !                                      |          x
            !         A4        A2                .----------->
            !
            !     X          *          X
            !  (1)        {3}           (3)

         temp = (corner(1,2)-corner(2,2))
         IF (abs(temp) > 0.0 ) THEN
```

```
      mid_side(1,1) = (corner(1,1)-corner(2,1))/temp* &
                      (node_coord(2,node_label(i))-corner(2,2)) + corner(2,1)
   ELSE
      mid_side(1,1) = corner(2,1)
   END IF
   mid_side(1,2) = node_coord(2,node_label(i))

   temp = (corner(2,1)-corner(4,1))
   mid_side(2,1) = node_coord(1,node_label(i))
   IF (abs(temp) > 0.0 ) THEN
      mid_side(2,2) = (corner(2,2)-corner(4,2))/temp* &
                      (node_coord(1,node_label(i))-corner(4,1)) + corner(2,2)
   ELSE
      mid_side(2,2) = corner(2,2)
   END IF

   temp = (corner(1,1)-corner(3,1))
   mid_side(3,1) = node_coord(1,node_label(i))
   IF (abs(temp) > 0.0 ) THEN
      mid_side(3,2) = (corner(1,2)-corner(3,2))/temp* &
                      (node_coord(1,node_label(i))-corner(3,1)) + corner(1,2)
   ELSE
      mid_side(3,2) = corner(1,2)
   END IF

   temp = (corner(3,2)-corner(4,2))
   IF (abs(temp) > 0.0 ) THEN
      mid_side(4,1) = (corner(3,1)-corner(4,1))/temp* &
                      (node_coord(2,node_label(i))-corner(4,2)) + corner(4,1)
   ELSE
      mid_side(4,1) = corner(4,1)
   END IF
   mid_side(4,2) = node_coord(2,node_label(i))

   area(1) = 0.5*((mid_side(2,1)-node_coord(1,node_label(i)))* &
                  (abs(mid_side(4,2)) - abs(corner(4,2))) - &
                  (mid_side(4,1)-corner(4,1))* &
                  (abs(mid_side(2,2))-abs(node_coord(2,node_label(i)))))
   area(2) = 0.5*((node_coord(1,node_label(i)) - mid_side(3,1))* &
                   (abs(corner(3,2)) - abs(mid_side(4,2))) - &
                   (corner(3,1) - mid_side(4,1))* &
                   (abs(node_coord(2,node_label(i))) - abs(mid_side(3,2))))
   area(3) = 0.5*((corner(2,1) - mid_side(1,1))* &
                   (abs(node_coord(2,node_label(i))) - abs(mid_side(2,2))) - &
                   (node_coord(1,node_label(i)) - mid_side(2,1))* &
                   (abs(corner(2,2)) - abs(mid_side(1,2))))
   area(4) = 0.5*((mid_side(1,1) - corner(1,1))* &
                   (mid_side(3,2) - (abs(node_coord(2,node_label(i))))  ) - &
                   (mid_side(3,1) - node_coord(1,node_label(i)) )* &
                   (abs(mid_side(1,2)) - abs(corner(1,2))))

   IF ((area(1)==area(2)).AND.(area(2)==area(3)).AND.(area(3)==area(4)).AND. &
       (area(4)== 0.0 )) THEN
      Cp = 0.0
      write(*,*) 'There is a problem with the geometry and aero files.'
   ELSE
      Cp = ( aero_coord(4,corner_node(1))*abs(area(1)) + &
             aero_coord(4,corner_node(2))*abs(area(2)) + &
             aero_coord(4,corner_node(3))*abs(area(3)) + &
             aero_coord(4,corner_node(4))*abs(area(4)))/ &
           (abs(area(1))+abs(area(2))+abs(area(3))+abs(area(4)))
   END IF
END IF

   !Assign coordinates to the ANSYS node
final_data(1,node_label(I))=node_coord(1,node_label(I))
final_data(2,node_label(I))=node_coord(2,node_label(I))
final_data(3,node_label(I))=node_coord(3,node_label(I))
   !Assign Cp and pressure to the ANSYS node
```

```
      final_data(4,node_label(I))=Cp
      final_data(5,node_label(I))=Cp*q


      percentage_1 = INT(i*100.0/node_num)
      percentage_10 = INT(i*10.0/node_num)
      IF (percentage_1 > percentage_1_old) THEN
         percentage_1_old = percentage_1
         write(*,'(A)',ADVANCE = "NO") '.'
      END IF
      IF (percentage_10 > percentage_10_old) THEN
         percentage_10_old = percentage_10
         write(*,50) (percentage_10_old*10),'% complete'
      END IF
   END DO
END SUBROUTINE interpolate



!*******************************************************************************

SUBROUTINE output(final_data,eset,elem_label,elem_num)
   !-------------------------------------------------------------------------------
   !This subroutine uses the pressure data from previous subroutines on each
   !ANSYS node and creates a file of forces on each node. THEN the output is
   !writtelem_num so that ANSYS can import the file. This subroutine takes into
   !consideration all three coordinate directions and the forces involved. The
   !approximations used are as follows:
   !   The pressure on the element is the average of the pressures on the
   !      corner nodes
   !   The normal vector of the element is givelem_num by vectors eminating
   !      from opposite corners of the element towards their adjacelem_numt
   !      nodes. The normal vector is their cross product.
   !   The force on the element is givelem_num by the average pressure multiplied
   !      by the area of the element
   !   The area of the element is approximated by two triangles
   !   The force on each element corner node is equal to one fourth the
   !      force on the elem_numtire element
   !
   !The variables passed are:
   !   final_data   =   ANSYS node coordinates, Cp and pressure
   !            [x pos, y pos, z pos, Cp, pressure]
   !   eset   =   ANSYS element labels
   !            [node 1, node 2, node 3, node 4, node 4,
   !            node 5, node 6, node 7, node 8]
   !   node_label   =   ANSYS node label [node number]
   !   elem_label   =   ANSYS element label [element label]
   !   node_num   =   number of ANSYS nodes
   !   elem_num   =   number of ANSYS elements
   !
   !-------------------------------------------------------------------------------

   Implicit None

   Real,DIMENSION(:,:)::final_data
   real,DIMENSION(4,2)::pt
   real::Area
   Integer,DIMENSION(:,:)::eset
   Integer,DIMENSION(:)::elem_label
   Integer::elem_num,K
   real::g
   Integer::i
   Integer::position
   CHARACTER(20)::ansys_file
   Real::pi,pj,pk,pl,pm,pn,po,pp
   real::number_of_gs
   integer::steps
```

```
   write(*,'(A)',ADVANCE = "NO") ' Enter the number of G''s: '
   read(*,*) number_of_gs

      !elem_numtering the location of the ANSYS data
   write(*,'(A)',ADVANCE = "NO") ' Enter the name of Ansys data file:'
   Read(*,*) ansys_file

   IF (number_of_gs == 0.0) THEN
      do i=1,50
         if (ansys_file(i:i) == '.') then
            ansys_file(i:(i+6)) = '_1g.dat'
            position = i+1
            exit
         end if
      end do
      steps = 18
      g = 0.0
   else
      steps = 1
      g = number_of_gs - 1.0
   end if

      ! ASCII 49 = #1
      ! ASCII 57 = #9
   do i = 1,steps
      g = g + 1.0
      if (number_of_gs == 0.0) THEN
         if (g == 10.0) THEN
            g = -9.0
            ansys_file(position:(position+6)) = '-9g.dat'
            position = position + 1
         end if
         if (i <= 9) THEN
            !create positive g filename
            ansys_file(position:position) = 49 + INT(g) - 1
         else if (i > 9) THEN
            !create negative g filename
            ansys_file(position:position) = 57 - (INT(g) + 9)
         end if
      end if

      OPEN(82,file=ansys_file,status='unknown')

      WRITE(82,60)'C***','C*** element pressure loads','C***'
      60 FORMAT(A/A/A)

      Do k=1,elem_num
            ! determine the area enclosed by the four corners
   !      pt(1,1) = final_data(1,eset(1,elem_label(k)))
   !      pt(1,2) = final_data(2,eset(1,elem_label(k)))
   !      pt(2,1) = final_data(1,eset(2,elem_label(k)))
   !      pt(2,2) = final_data(2,eset(2,elem_label(k)))
   !      pt(3,1) = final_data(1,eset(3,elem_label(k)))
   !      pt(3,2) = final_data(2,eset(3,elem_label(k)))
   !      pt(4,1) = final_data(1,eset(4,elem_label(k)))
   !      pt(4,2) = final_data(2,eset(4,elem_label(k)))
   !      CALL Quadralateral_Area(pt, area)
area = 1.0
            !Setting the pressures at the nodes for element "k"
         pi=area*g*final_data(5,eset(1,elem_label(k)))
         pj=area*g*final_data(5,eset(2,elem_label(k)))
            pk=area*g*final_data(5,eset(3,elem_label(k)))
            pl=area*g*final_data(5,eset(4,elem_label(k)))


         IF (eset(1,elem_label(k)) == 0) pi = 0
         IF (eset(2,elem_label(k)) == 0) pj = 0
         IF (eset(3,elem_label(k)) == 0) pk = 0
```

```
         IF (eset(4,elem_label(k)) == 0) pl = 0
         IF (eset(5,elem_label(k)).ne.0) THEN
            !write(*,*) '                  **********',final_data(5,eset(5,elem_label(K)))
            pm=area*g*final_data(5,eset(5,elem_label(k)))
            pn=area*g*final_data(5,eset(6,elem_label(k)))
            po=area*g*final_data(5,eset(7,elem_label(k)))
               pp=area*g*final_data(5,eset(8,elem_label(k)))
         END IF
            !Outputs to ANSYS
         IF (eset(5,elem_label(k))==0) THEN
            IF (pl == 0) THEN
               WRITE(82,72)'SFE',elem_label(K),'1,PRES,',pi,pj,pk,pl
            ELSE IF (pk == 0) THEN
               WRITE(82,72)'SFE',elem_label(K),'2,PRES,',pi,pj,pk,pl
            ELSE IF (pi == 0) THEN
               WRITE(82,72)'SFE',elem_label(K),'3,PRES,',pi,pj,pk,pl
            ELSE IF (pj == 0) THEN
               WRITE(82,72)'SFE',elem_label(K),'4,PRES,',pi,pj,pk,pl
            ELSE
               WRITE(82,72)'SFE',elem_label(K),'2,PRES,',pi,pj,pk,pl
            END IF
            72 FORMAT(A,',',I5,',',A,4(',',E10.4))
         ELSE
            WRITE(82,74)'SFE',elem_label(K),'2,PRES,',pi,pj,pk,pl,pm,pn,po,pp
            74 FORMAT(A,',',I5,',',A,8(',',E10.4))
         END IF
      END Do
      write(82,*) '/NERR,,200000'

      Close(82)
   end do

END SUBROUTINE output

!*****************************************************************************

SUBROUTINE ansread(node_coord,eset,node_label,elem_label,node_num,elem_num,plane_coord,axis)
   !-----------------------------------------------------------------------------
   !This subroutine reads the input from a UG mesh file (*.inp) as outputted
   !from the structures scelem_numario. The variables used are:
   !
   !   eset   =   ANSYS element labels
   !            [element number, node 1, node 2, node 3, node 4]
   !   node_label   =   ANSYS node label [node number]
   !   elem_label   =   ANSYS element label [element label]
   !   node_num   =   number of ANSYS nodes
   !   elem_num   =   number of ANSYS elements
   !
   !-----------------------------------------------------------------------------

   IMPLICIT NONE

   character(50) filename
   character(50) filename2

   real,DIMENSION(:,:),INTENT(INOUT):: node_coord
   integer,DIMENSION(:,:),INTENT(INOUT):: eset
   integer,DIMENSION(:),INTENT(INOUT):: node_label
   integer,DIMENSION(:),INTENT(INOUT):: elem_label
   real,INTENT(IN),DIMENSION(:,:)::plane_coord
   INTEGER, INTENT(IN):: axis

   integer,DIMENSION(4):: ansys_coord_sys
   integer,INTENT(INOUT):: node_num,elem_num
   integer:: a,b,c,d,e,f,g,h,j,exiter
   integer:: i
   integer:: blocknum
               ! these are used for the solid routines
   integer::start_skin
```

```
   integer::start_of_pannel   !ponits to the first element of the panode_numel
   integer::end_of_pannel      !points to the last element of the panode_numel
   integer::solid_found
   integer::num_match
   integer::counter
   integer::above

   CHARACTER(1)::limiter
   CHARACTER(2)::dummy
   CHARACTER(1)::test
   CHARACTER(7)::elem_type
   CHARACTER(20)::longdummy
   CHARACTER(120),DIMENSION(20000)::bigass
   CHARACTER(2),DIMENSION(20000)::dummy1


   write(*,'(A)',ADVANCE = "NO")' Enter filename from UG geometry... '
   read(*,*)filename

   OPEN(99,file=filename,status='old',position='rewind')

      !Initialize
   I=0
   limiter='!'
   exiter=1
   node_num=0
   node_label=0
   solid_found = 0
   ansys_coord_sys = 0

20 FORMAT(3(BZ,I8),F16.0,F16.0,F16.0)

   DO
         !Reads a line from the data file.
      READ(99,'(A1)') test

         !Checks to see if "test" is a space
      IF (test==' ') THEN
         BACKSPACE(99)
         I=I+1
         READ(99,20) node_label(I),a,b,node_coord(1,node_label(I))&
            &,node_coord(2,node_label(I)),node_coord(3,node_label(I))
         CALL
seperator(plane_coord,node_coord(1,node_label(I)),node_coord(2,node_label(I)),node_coord(3,no
de_label(I)),axis,above)
         node_coord(4,node_label(I)) = above
         READ(99,'(BZ,A2)') dummy
         IF ((dummy=='0').OR.(dummy=='-1')) EXIT
         BACKSPACE(99)
         node_num=node_num+1
      END IF
   END DO

   write(*,*) 'nodenums',node_num
30 FORMAT(18(BZ,I7))
40 FORMAT(14(BZ,I7))

   I=0
   elem_num=0
   eset=0
   start_of_pannel = 0
   start_skin = 1
   DO
         !Reads a line from the data file.
      READ(99,'(BZ,A2)') dummy

      IF (dummy=='ET') THEN
         BACKSPACE(99)
         READ(99,*) longdummy,blocknum,elem_type
```

```
      END IF

         !Checks to see if "test" is an element reference
      IF (dummy=='  ') THEN
         BACKSPACE(99)
         IF (elem_type=='SHELL93') THEN      ! process the 8 node shell elements
            DO
               I=I+1
               READ(99,30)a,b,c,elem_label(I),d,e,f,g,h,j,eset(1,elem_label(I)),&
                  &eset(2,elem_label(I)),eset(3,elem_label(I)),&
                  &eset(4,elem_label(I)),eset(5,elem_label(I)),&
                  &eset(6,elem_label(I)),eset(7,elem_label(I)),&
                  &eset(8,elem_label(I))
               READ(99,'(BZ,A2)') dummy
               IF ((dummy=='0').OR.(dummy=='-1')) EXIT
               backspace(99)
            END DO
         ELSE IF (elem_type=='SHELL63') THEN      ! process the 4 node shell elements
            DO
               I=I+1
               READ(99,40)a,b,c,elem_label(I),d,e,f,g,h,j,eset(1,elem_label(I)),&
                  &eset(2,elem_label(I)),eset(3,elem_label(I)),&
                  &eset(4,elem_label(I))
               READ(99,'(BZ,A2)') dummy
               IF ((dummy=='0').OR.(dummy=='-1')) EXIT
               backspace(99)
            END DO
         ELSE IF (elem_type=='SOLID92') THEN   ! process the 10 node solid element
            solid_found = 1
            DO
               I=I+1
               READ(99,30)a,b,c,elem_label(I),d,e,f,g,h,j,eset(1,elem_label(I)),&
                  &eset(2,elem_label(I)),eset(3,elem_label(I)),&
                  &eset(4,elem_label(I)),eset(5,elem_label(I)),&
                  &eset(6,elem_label(I)),eset(7,elem_label(I)),&
                  &eset(8,elem_label(I)),eset(9,elem_label(I)),&
                  &eset(10,elem_label(I))
               READ(99,'(BZ,A2)') dummy
               IF ((dummy=='0').OR.(dummy=='-1')) EXIT
               backspace(99)
            END DO
            end_of_pannel = i
            IF (ansys_coord_sys(1) == 0) THEN
               filename2 = ''
               counter = 1
               DO   WHILE (filename(counter:counter) /= '.')
                  filename2(counter:counter) = filename(counter:counter)
                  counter = counter + 1
               END DO
               filename2(counter:counter+9) = '_coord.dat'
               OPEN(84,file=filename2,status='unknown')
               write(84,60)'C***','C*** coordinate system definitions','C***'
               60 FORMAT(A/A/A)
            END IF
            CALL
Determine_outter_sheet_of_solid(start_of_pannel,end_of_pannel,eset,node_coord,start_skin,elem
_label,plane_coord,num_match,ansys_coord_sys)

write(84,88)'CS',(a+10),'CART',ansys_coord_sys(1),ansys_coord_sys(2),ansys_coord_sys(3)
            88 FORMAT(A,',',I4,',',A,3(',',I7),', ,')
         END IF
      ELSE
         READ(99,'(BZ,A2)') dummy
         IF (dummy=='FI') EXIT
         BACKSPACE(99)
      END IF
      IF (EOF(99)) THEN
         exit
      END IF
```

```
   END DO

   elem_num=i
   If (solid_found == 1) THEN
      elem_num = num_match
      elem_num=start_skin-1
      CALL convert_surface_to_sheet(node_num,node_coord,node_label,elem_num,eset)
   END IF

   write(*,*) 'elemnums',elem_num

   CLOSE(84)
   CLOSE(99)

END SUBROUTINE ansread

!*****************************************************************************


SUBROUTINE seperator(plan_coord,x0,y0,z0,axis,above)

   IMPLICIT NONE
   REAL,DIMENSION(3,3)::plan_coord
   REAL::x0,y0,z0
   INTEGER::axis,above
   REAL::a,b,c,y,x,z
   real::d


   above=-1
   a=(plan_coord(2,2)-plan_coord(2,1))*(plan_coord(3,3)-plan_coord(3,1))-(plan_coord(2,3)-
plan_coord(1,3))*(plan_coord(3,2)-plan_coord(3,1))

   b=(plan_coord(1,3)-plan_coord(1,1))*(plan_coord(3,2)-plan_coord(3,1))-(plan_coord(1,2)-
plan_coord(1,1))*(plan_coord(3,3)-plan_coord(3,1))

   c=(plan_coord(1,2)-plan_coord(1,1))*(plan_coord(2,3)-plan_coord(2,1))-(plan_coord(1,3)-
plan_coord(1,1))*(plan_coord(2,2)-plan_coord(2,1))

   d = a*plan_coord(1,1) + b*plan_coord(2,1) + c*plan_coord(3,1)

   x = 0.0
   y = 0.0
   z = 0.0
   IF (axis == 1) THEN
      IF (abs(a) >= 1e-9) THEN
         x = (d - b*y0 - c*z0)/a
      END IF
      IF (x0 >= x) THEN
         above=1
      END IF
   ELSE IF (axis == 2) THEN
      IF (abs(b) >= 1e-9) THEN
         y = (d - a*x0 - c*z0)/b
      END IF
      IF (y0 >= y) THEN
         above=1
      END IF
   ELSE IF (axis == 3) THEN
      IF (abs(c) >= 1e-9) THEN
         z = (d - a*x0 - b*y0)/c
      END IF
      IF (z0 >= z) THEN
         above=1
      END IF
   END IF

   IF ((axis.ne.1).and.(axis.ne.2).and.(axis.ne.3)) THEN
      write(*,*) 'axis =',axis
```

```
      write(*,*) '*************      ERROR IN SEPERATION     ****************'
   END IF

END SUBROUTINE seperator



!*********************************************************************
SUBROUTINE Get_seperation_plane(plane_coord,axis)

   IMPLICIT NONE
   real,INTENT(INOUT),DIMENSION(:,:)::plane_coord
   INTEGER, INTENT(INOUT):: axis

   INTEGER :: i

   CHARACTER(50)::filename


      !User input of data file
   write(*,'(A)',ADVANCE = "NO") ' Enter filename for the seperation plane or type "KEY IN" :
'
   read(*,*) filename

   IF ((filename(1:1) == 'K') .OR. (filename(1:1) == 'k')) THEN
      write(*,*) 'enter the three seperation points. eg.  x1,y1,z1'
      DO I=1,3
         READ(*,*) plane_coord(1,I),plane_coord(2,I),plane_coord(3,I)
      END DO
   ELSE
         !OPEN Cmarc file
      OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')
      DO I=1,3
         READ(45,*) plane_coord(1,I)
         read(45,*) plane_coord(2,I)
         read(45,*) plane_coord(3,I)
      END DO
      CLOSE(45)
   END IF

   write(*,'(A)',ADVANCE = "NO") ' Enter the principle normal axis  (x=1,y=2,z=3)'
   read(*,*) axis

END SUBROUTINE Get_seperation_plane


!-------------------------------------------------------------
SUBROUTINE
Determine_outter_sheet_of_solid(start_of_pannel,end_of_pannel,eset,node_coord,start_skin,elem
_label,plane_coord,num_match,ansys_coord_sys)

   IMPLICIT NONE

   real,DIMENSION(:,:),INTENT(IN):: node_coord
   integer,DIMENSION(:,:),INTENT(INOUT):: eset

            ! these are used for the solid routines
   integer,DIMENSION(6,10000):: surface     !array for the outter surface of the solid
   integer,DIMENSION(10000)::surface_label
   integer,INTENT(INOUT)::start_skin        ! points to the start of the new surface
panode_numel
   integer,INTENT(INOUT)::start_of_pannel
   integer,INTENT(INOUT)::end_of_pannel     !points to the last element of the panode_numel
   real,INTENT(IN),DIMENSION(:,:)::plane_coord

   integer,DIMENSION(1000000,4,3)::elem     !store the 3-D element into 2-D parts
   integer,DIMENSION(:),INTENT(INOUT):: elem_label
   integer,DIMENSION(:),INTENT(INOUT):: ansys_coord_sys
   integer::i,j,k,l,m
```

```
      real :: x1,x2,x3,y1,y2,y3,z1,z2,z3
      real :: n,o,p,d
      real :: n1,o1,p1,d1
      real :: magnitude
      real :: average_z_coord
      real :: average_z_plane
      integer:: above
      real :: max_x,max_y,max_z
      real :: min_x,min_y,min_z
      integer::max_i,max_j
      integer::min_i,min_j
      integer::erase
      integer::num_elements
      integer::num_surface

      integer::origin
      real :: origin_x,origin_z
      integer::perpendicular
      real::perpendicular_x,perpendicular_z
      integer::in_plane
      real::in_plane_x,in_plane_y
      real::temp_x,temp_y,temp_z
      real::ave_x,ave_y,ave_z

      integer,DIMENSION(1000000,3)::found      !stores the determined outter surface data
      integer,INTENT(INOUT):: num_match
      integer::index,complete,exists


      num_elements = (end_of_pannel - start_of_pannel)

         ! Copy the 3-D elements into 2-D elements
         !---------------------------------------------------------------------
         !   original: element#, 10 nodes
         !        new: element#, side#, 3 nodes
         !
         !       ex: 1|1|1,2,3
         !           1|2|1,2,4   this pattern was set from shell92 ordering
         !          1|3|2,3,4
         !          1|4|3,1,4
         !---------------------------------------------------------------------
      elem(:,:,:) = 0
      DO I=1,num_elements
         elem(I,1,1) = eset(1,(start_of_pannel+i))
         elem(I,1,2) = eset(2,(start_of_pannel+i))
         elem(I,1,3) = eset(3,(start_of_pannel+i))
         elem(I,2,1) = eset(1,(start_of_pannel+i))
         elem(I,2,2) = eset(2,(start_of_pannel+i))
         elem(I,2,3) = eset(4,(start_of_pannel+i))
         elem(I,3,1) = eset(2,(start_of_pannel+i))
         elem(I,3,2) = eset(3,(start_of_pannel+i))
         elem(I,3,3) = eset(4,(start_of_pannel+i))
         elem(I,4,1) = eset(3,(start_of_pannel+i))
         elem(I,4,2) = eset(1,(start_of_pannel+i))
         elem(I,4,3) = eset(4,(start_of_pannel+i))
      END DO
   92 FORMAT(6(I5))
         ! Algorythm to get the shell from the solid element
      erase = 0
      num_surface = 0
      DO I = 1,num_elements
         DO J = 1,4
            DO k = (I+1), num_elements
               DO l = 1,4
                     ! check elem(I,J,nodes) to elem(K,L,nodes)
                     !   If the same set both to zero because the face
                     !     does not share any other common face.
                  IF (elem(I,J,1)==elem(K,L,1)) THEN
                     IF ((elem(I,J,2)==elem(K,L,2)).AND.(elem(I,J,3)==elem(K,L,3))) THEN
```

```
                  !set an erase flag
                  erase = 1
               ELSE IF ((elem(I,J,2)==elem(K,L,3)).AND.(elem(I,J,3)==elem(K,L,2))) THEN
                  !set an erase flag
                  erase = 1
               END IF
            END IF
            IF (elem(I,J,1)==elem(K,L,2)) THEN
               IF ((elem(I,J,2)==elem(K,L,1)).AND.(elem(I,J,3)==elem(K,L,3))) THEN
                  !set an erase flag
                  erase = 1
               ELSE IF ((elem(I,J,2)==elem(K,L,3)).AND.(elem(I,J,3)==elem(K,L,1))) THEN
                  !set an erase flag
                  erase = 1
               END IF
            END IF
            IF (elem(I,J,1)==elem(K,L,3)) THEN
               IF ((elem(I,J,2)==elem(K,L,2)).AND.(elem(I,J,3)==elem(K,L,1))) THEN
                  !set an erase flag
                  erase = 1
               ELSE IF ((elem(I,J,2)==elem(K,L,1)).AND.(elem(I,J,3)==elem(K,L,2))) THEN
                  !set an erase flag
                  erase = 1
               END IF
            END IF
            IF (erase == 1) THEN
               !clear the two adjoining faces and continue with a new J search
               DO M=1,3
                  elem(I,J,M)=0
                  elem(K,L,M)=0
               END DO
               EXIT
            ELSE
            END IF
         END DO   ! L loop
         IF (erase == 1) THEN
            erase = 0
            EXIT
         END IF
      END DO ! K loop
   END DO ! J loop
END DO ! I loop


   ! Get rid of the faces that are vertical.  The reason why
   ! the vertical faces can be eliminated is because the solids
   ! should have been cut with either vertical or separation planes.
   !
   ! To get rid of the vertical elements the surfaces is checked to
   ! see if it is perpendicular to the x-y plane with the following:
   !   points (x1,y1,z1), (x2,y2,z2), (x3,y3,z3)
   !     (X2 - X1)(Y3 - Y1) - (X3 - X1)(Y2 - Y1) :
   !
   !     When the result of the above equation is zero then the
   !     element is perpendicular to the x-y plane.
DO I = 1,num_elements
   DO J = 1,4
      IF (elem(I,J,1) > 0 ) THEN
            ! node_coord(coord,node)
         x1 = node_coord(1,elem(I,J,1))
         x2 = node_coord(1,elem(I,J,2))
         x3 = node_coord(1,elem(I,J,3))
         y1 = node_coord(2,elem(I,J,1))
         y2 = node_coord(2,elem(I,J,2))
         y3 = node_coord(2,elem(I,J,3))
         z1 = node_coord(3,elem(I,J,1))
         z2 = node_coord(3,elem(I,J,2))
         z3 = node_coord(3,elem(I,J,3))
```

```
            IF (abs(x1) <= 1e-6) x1=0
            IF (abs(x2) <= 1e-6) x2=0
            IF (abs(x3) <= 1e-6) x3=0
            IF (abs(y1) <= 1e-6) y1=0
            IF (abs(y2) <= 1e-6) y2=0
            IF (abs(y3) <= 1e-6) y3=0
            IF (abs(z1) <= 1e-6) z1=0
            IF (abs(z2) <= 1e-6) z2=0
            IF (abs(z3) <= 1e-6) z3=0
            IF (abs(((x2-x1)*(y3-y1))-((x3-x1)*(y2-y1))) <= 1e-3) THEN
               DO M=1,3
                  elem(I,J,M)=0
               END DO
            END IF
         END IF
      END DO
   END DO
   num_surface = 0

      ! This part compares the separation plane to the plane definal_dataed by the
      ! corner points of the element to see if it is located very close to
      ! the seperation plane.
      !   variables:
      !      n - the coefficielem_numt for the x-axis
      !      o - the coefficielem_numt for the y-axis
      !      p - the coefficielem_numt for the z-axis
   n = (plane_coord(2,2)-plane_coord(2,1))*(plane_coord(3,3)-plane_coord(3,1)) - &
      & (plane_coord(2,3)-plane_coord(2,1))*(plane_coord(3,2)-plane_coord(3,1))

   o = (plane_coord(3,2)-plane_coord(3,1))*(plane_coord(1,3)-plane_coord(1,1)) - &
      & (plane_coord(3,3)-plane_coord(3,1))*(plane_coord(1,2)-plane_coord(1,1))

   p = (plane_coord(1,2)-plane_coord(1,1))*(plane_coord(2,3)-plane_coord(2,1)) - &
      & (plane_coord(1,3)-plane_coord(1,1))*(plane_coord(2,2)-plane_coord(2,1))

   d = n*plane_coord(1,1) + o*plane_coord(2,1) + p*plane_coord(3,1)
   magnitude = sqrt(n**2 + o**2 + p**2)
   n = n/magnitude
   o = o/magnitude
   p = p/magnitude
   d = d/magnitude
   DO I = 1,num_elements
      DO J = 1,4
         IF (elem(I,J,1) > 0 ) THEN
            x1 = node_coord(1,elem(I,J,1))
            x2 = node_coord(1,elem(I,J,2))
            x3 = node_coord(1,elem(I,J,3))
            y1 = node_coord(2,elem(I,J,1))
            y2 = node_coord(2,elem(I,J,2))
            y3 = node_coord(2,elem(I,J,3))
            z1 = node_coord(3,elem(I,J,1))
            z2 = node_coord(3,elem(I,J,2))
            z3 = node_coord(3,elem(I,J,3))

            n1 = (y2-y1)*(z3-z1) - (y3-y1)*(z2-z1)
            o1 = (z2-z1)*(x3-x1) - (z3-z1)*(x2-x1)
            p1 = (x2-x1)*(y3-y1) - (x3-x1)*(y2-y1)
            d1 = n1*x1 + o1*y1 + p1*z1
            magnitude = sqrt(n1**2 + o1**2 + p1**2)
            n1 = n1/magnitude
            o1 = o1/magnitude
            p1 = p1/magnitude
            d1 = d1/magnitude
            IF ((abs(abs(n) - abs(n1)) <= 1e-6).AND.(abs(abs(o) - abs(o1)) <= 1e-
6).AND.(abs(abs(p) - abs(p1)) <= 1e-6).AND.(abs(abs(d) - abs(d1)) <= 1e-6)) THEN
               DO M=1,3
                  elem(I,J,M)=0
               END DO
            ELSE
```

```
               num_surface = num_surface + 1
            END IF
         END IF
      END DO
   END DO

         ! This part determines if the pannel is above or below the seperation
         ! plane and then it finds the highest or lowest point respectively
   above = 0
   DO I = 1,num_elements
      DO J = 1,4
         IF (elem(I,J,1) /= 0 ) THEN
            IF (above == 0) THEN
               average_z_coord =
(node_coord(3,elem(i,j,1))+node_coord(3,elem(i,j,2))+node_coord(3,elem(i,j,3)))/3
               average_z_plane = (((d-n*node_coord(1,elem(I,J,1))-
o*node_coord(2,elem(I,J,1)))/p)+((d-n*node_coord(1,elem(I,J,2))-
o*node_coord(2,elem(I,J,2)))/p)+((d-n*node_coord(1,elem(I,J,3))-
o*node_coord(2,elem(I,J,3)))/p))/3
               IF (average_z_coord > average_z_plane) THEN
                  above = 1      ! above the seperation plane
               ELSE
                  above = -1      ! below the seperation plane
               END IF
               max_z = node_coord(3,elem(I,J,1))
               max_i = i
               max_j = j
            ELSE
               IF (((above == 1) .AND.(node_coord(3,elem(I,J,1)) > max_z)) .OR.&
                  &((above == -1).AND.(node_coord(3,elem(I,J,1)) < max_z))) THEN
                  max_z = node_coord(3,elem(I,J,1))
                  max_i = i
                  max_j = j
               END IF
            END IF
         END IF
      END DO
   END DO

      ! This part oders the remaining elements to the outter shell so that
      ! the pressures can be placed onto the outter surface.
   num_match = num_match + 1
   index = 1
   exists = 0
   complete = 0
   found(:,:) = 0
   found(1,1) = max_i   ! the element number
   found(1,2) = max_j   ! the element side
   found(max_i,3) = 1
   DO WHILE (complete /= 1)
      IF (found(index,1) /= 0) THEN
         DO i= 1,3
            DO j = 1,num_elements
               DO k = 1,4
                  IF (elem(j,k,1) > 0) THEN
                     IF ((elem(found(index,1),found(index,2),i) == elem(j,k,1)).OR.&
                        &(elem(found(index,1),found(index,2),i) == elem(j,k,2)).OR.&
                        &(elem(found(index,1),found(index,2),i) == elem(j,k,3))) THEN
                        DO l = 1,num_match   ! check to see if found is already in list
                           IF ((found(l,1) == j).AND.(found(l,2) == k)) THEN
                              exists = 1
                           END IF
                        END DO
                        IF (exists /= 1) THEN
                           num_match = num_match + 1
                           found(num_match,1) = j
                           found(num_match,2) = k
                        ELSE
                           exists = 0
```

```
                  END IF
               END IF
            END IF
         END DO
      END DO
   END DO
   ELSE
      complete = 1
   END IF
   index = index + 1
END DO

   ! This part rearanges the elemnts back to the original pattern
surface(:,:) = 0
surface_label(:) = 0
DO i = 1,num_match
   surface_label(i) = elem_label(start_of_pannel+found(i,1))
   IF (found(i,2) == 1) THEN   !Side one => add the mid nodes
      surface(i,1) = elem(found(i,1),found(i,2),1)
      surface(i,2) = elem(found(i,1),found(i,2),2)
      surface(i,3) = elem(found(i,1),found(i,2),3)
      surface(i,4) = 0
      surface(i,5) = 0
      surface(i,6) = 0
   ELSE IF (found(i,2) == 2) THEN   !Side two => add the mid nodes
      surface(i,1) = elem(found(i,1),found(i,2),1)
      surface(i,2) = elem(found(i,1),found(i,2),2)
      surface(i,4) = elem(found(i,1),found(i,2),3)
      surface(i,3) = 0
      surface(i,5) = 0
      surface(i,6) = 0
   ELSE IF (found(i,2) == 3) THEN   !Side three => add the mid nodes
      surface(i,2) = elem(found(i,1),found(i,2),1)
      surface(i,3) = elem(found(i,1),found(i,2),2)
      surface(i,4) = elem(found(i,1),found(i,2),3)
      surface(i,1) = 0
      surface(i,5) = 0
      surface(i,6) = 0
   ELSE IF (found(i,2) == 4) THEN   !Side four => add the mid nodes
      surface(i,3) = elem(found(i,1),found(i,2),1)
      surface(i,1) = elem(found(i,1),found(i,2),2)
      surface(i,4) = elem(found(i,1),found(i,2),3)
      surface(i,2) = 0
      surface(i,5) = 0
      surface(i,6) = 0
   END IF
END DO
DO i = 1,num_elements
   DO j = 1,10
      eset(j,i) = 0
   END DO
END DO
DO i = 1,num_match
   elem_label(start_skin+(i-1)) = surface_label(i)
   eset(1,surface_label(i)) = surface(i,1)
   eset(2,surface_label(i)) = surface(i,2)
   eset(3,surface_label(i)) = surface(i,3)
   eset(4,surface_label(i)) = surface(i,4)
END DO
start_skin = start_skin + num_elements



   ! This part determines the minimum vertical face to start defining
   ! the coordinate system on
min_i = 0
DO I = 1,num_elements
   DO j = 1,4
      IF (elem(i,j,1) /= 0 ) THEN
```

```
            DO k=1,3
               IF (min_i == 0) THEN
                  min_i = i
                  max_i = i
                  min_j = j
                  max_j = j
                  min_x = node_coord(1,elem(i,j,1))
                  min_y = node_coord(2,elem(i,j,1))
                  min_z = node_coord(3,elem(i,j,1))
                  max_x = min_x
                  max_y = min_y
                  max_z = min_z
               ELSE IF ((node_coord(1,elem(i,j,k)) <=
min_x).AND.(abs(node_coord(2,elem(i,j,k))) <= abs(min_y))) THEN
                  min_i = i
                  min_j = j
                  min_x = node_coord(1,elem(i,j,k))
                  min_y = node_coord(2,elem(i,j,k))
                  min_z = node_coord(3,elem(i,j,k))
               ELSE IF ((node_coord(1,elem(i,j,k)) >=
max_x).AND.(abs(node_coord(2,elem(i,j,k))) >= abs(max_y))) THEN
                  max_i = i
                  max_j = j
                  max_x = node_coord(1,elem(i,j,k))
                  max_y = node_coord(2,elem(i,j,k))
                  max_z = node_coord(3,elem(i,j,k))
               END IF
            END DO
         END IF
      END DO
   END DO

!write(*,*) 'min',min_x,min_y,min_z
!write(*,*) 'max',max_x,max_y,max_z

      !Find the origin node which is between the two extremes on the face
   ave_x = (min_x + max_x)/2
   ave_y = (min_y + max_y)/2
   ave_z = (min_z + max_z)/2
   DO I = 1,node_num
      IF ((ave_x /= 0.0).AND.(ave_y /= 0.0).AND.(ave_z /= 0.0)) THEN
         IF (node_coord(1,i) /= 0.0) THEN
            IF ((abs(ave_x/node_coord(1,i)) >= 0.995).AND.(abs(ave_x/node_coord(1,i)) <=
1.005)) THEN
               IF (node_coord(2,i) /= 0.0) THEN
                  IF ((abs(ave_y/node_coord(2,i)) >= 0.995).AND.(abs(ave_y/node_coord(2,i))
<= 1.005)) THEN
                     IF (node_coord(3,i) /= 0.0) THEN
                        IF ((abs(ave_z/node_coord(3,i)) >=
.995).AND.(abs(ave_z/node_coord(3,i)) <= 1.005)) THEN
                           origin = i
                        END IF
                     END IF
                  END IF
               END IF
            END IF
         END IF
      ELSE IF ((ave_x == 0.0).AND.(ave_y /= 0.0).AND.(ave_z /= 0.0)) THEN
         IF (abs(node_coord(1,i)) <= 1e-2) THEN
            IF (node_coord(2,i) /= 0.0) THEN
               IF ((abs(ave_y/node_coord(2,i)) >= 0.995).AND.(abs(ave_y/node_coord(2,i)) <=
1.005)) THEN
                  IF (node_coord(3,elem(i,j,k)) /= 0.0) THEN
                     IF ((abs(ave_z/node_coord(3,i)) >=
0.995).AND.(abs(ave_z/node_coord(3,i)) <= 1.005)) THEN
                        origin = i
                     END IF
                  END IF
               END IF
```

```
            END IF
         END IF
      ELSE IF ((ave_x /= 0.0).AND.(ave_y == 0.0).AND.(ave_z /= 0.0)) THEN
         IF (node_coord(1,i) /= 0.0) THEN
            IF ((abs(ave_x/node_coord(1,i)) >= 0.995).AND.(abs(ave_x/node_coord(1,i)) <=
1.005)) THEN
               IF (abs(node_coord(2,elem(i,j,k))) <= 1e-2) THEN
                  IF (node_coord(3,elem(i,j,k)) /= 0.0) THEN
                     IF ((abs(ave_z/node_coord(3,i)) >=
0.995).AND.(abs(ave_z/node_coord(3,i)) <= 1.005)) THEN
                        origin = i
                     END IF
                  END IF
               END IF
            END IF
         END IF
      ELSE IF ((ave_x /= 0.0).AND.(ave_y /= 0.0).AND.(ave_z == 0.0)) THEN
         IF (node_coord(1,i) /= 0.0) THEN
            IF ((abs(ave_x/node_coord(1,i)) >= 0.995).AND.(abs(ave_x/node_coord(1,i)) <=
1.005)) THEN
               IF (node_coord(2,i) /= 0.0) THEN
                  IF ((abs(ave_y/node_coord(2,i)) >= 0.995).AND.(abs(ave_y/node_coord(2,i))
<= 1.005)) THEN
                     IF (abs(node_coord(3,i)) <= 1e-2) THEN
                        origin = i
                     END IF
                  END IF
               END IF
            END IF
         END IF
      END IF
   END DO

!write(*,*) 'origin =',origin,node_coord(1,origin),node_coord(2,origin),node_coord(3,origin)

   origin_x = max_x - min_x
   origin_z = max_z - min_z
   magnitude = sqrt(origin_x**2 + origin_z**2)
   origin_x = origin_x/magnitude
   origin_z = origin_z/magnitude
   DO I = 1,node_num
      IF ((abs(node_coord(2,i)-node_coord(2,origin)) <= 1e-1).AND.(i /= origin)) THEN
                  !see if the vector from the origin is perpendicular to the
                  !vector that was used to find the origin.
         temp_x = node_coord(1,i) - node_coord(1,origin)
         temp_y = node_coord(2,i) - node_coord(2,origin)
         temp_z = node_coord(3,i) - node_coord(3,origin)
         magnitude = sqrt(temp_x**2 + temp_z**2)
         temp_x = temp_x/magnitude
         temp_z = temp_z/magnitude
         IF (((temp_x*origin_x + temp_z*origin_z) <= 1e-6).AND.((node_coord(3,i) +
node_coord(3,origin))/2 >= 5e-2  )) THEN
               !found a perpendicular vector node
            perpendicular = i
            magnitude = sqrt(temp_x**2 + temp_z**2)
            perpendicular_x = temp_x/magnitude
            perpendicular_z = temp_z/magnitude
         END IF
      END IF
   END DO

!write(*,*)'perpendicular
=',perpendicular,node_coord(1,perpendicular),node_coord(2,perpendicular),node_coord(3,perpend
icular)

   DO I = 1,node_num
      IF ((abs(node_coord(3,i)-node_coord(3,origin)) <= 1e-1).AND.(i /= origin)) THEN
               !see if the vector from the origin is perpendicular to the
               !perpendicular vector.
```

```
         temp_x = node_coord(1,i) - node_coord(1,origin)
         temp_y = node_coord(2,i) - node_coord(2,origin)
         magnitude = sqrt(temp_x**2 + temp_y**2)
         temp_x = temp_x/magnitude
         temp_y = temp_y/magnitude
         IF (((temp_x*perpendicular_x + temp_z*perpendicular_z) <= 1e-
6).AND.(abs((node_coord(2,i) + node_coord(2,origin))/2) >= 1e-1  )) THEN
               !found a perpendicular vector node
            in_plane = i
            magnitude = sqrt(temp_x**2 + temp_y**2)
            in_plane_x = temp_x/magnitude
            in_plane_y = temp_y/magnitude
         END IF
      END IF
   END DO

!write(*,*)'in_plane
=',in_plane,node_coord(1,in_plane),node_coord(2,in_plane),node_coord(3,in_plane)

   ansys_coord_sys(1) = origin
   ansys_coord_sys(2) = perpendicular
   ansys_coord_sys(3) = in_plane


END SUBROUTINE Determine_outter_sheet_of_solid



!-------------------------------------------------------------------------------------------
SUBROUTINE convert_surface_to_sheet(node_num,node_coord,node_label,elem_num,eset)
   implicit NONE

   real,DIMENSION(:,:),INTENT(INOUT):: node_coord
   integer,DIMENSION(:,:),INTENT(INOUT):: eset
   integer,DIMENSION(:),INTENT(INOUT):: node_label
   integer,INTENT(INOUT):: node_num,elem_num

   real,DIMENSION(3,1000000):: temp_node_coord

   integer :: i,j,k
   integer :: node_exists
   integer :: counter


   counter = 0
   node_exists = 0

   temp_node_coord = node_coord
   node_coord = 0.0
   node_label = 0

   DO i = 1,elem_num
      DO j = 1,6
         IF (eset(j,i) /= 0) THEN
            DO k = 1,counter
               IF (eset(j,i) == node_label(k)) THEN
                  node_exists = 1
               END IF
            END DO
            IF (node_exists == 0) THEN
               IF (eset(j,i) /= 0) THEN
                  counter = counter + 1
                  node_label(counter) = eset(j,i)
               END IF
            ELSE
               node_exists = 0
            END IF
         END IF
      END DO
```

```
   END DO
   DO j = 1,counter
      DO i = 1,3
         node_coord(i,j) = temp_node_coord(i,node_label(j))
      END DO
   END DO
   node_num = counter
   node_label(counter+1) = 0
   node_coord(1,(counter+1)) = 0
   node_coord(2,(counter+1)) = 0
   node_coord(3,(counter+1)) = 0

END SUBROUTINE convert_surface_to_sheet

SUBROUTINE Gauss_linear(x,y,x1,y1,cp1,x2,y2,cp2,cp)
   IMPLICIT NONE
   real,intent(IN)  :: x
   real,intent(IN)  :: y
   real,intent(IN)  :: x1
   real,intent(IN)  :: y1
   real,intent(IN)  :: x2
   real,intent(IN)  :: y2
   real,intent(INOUT)::cp
   real,intent(IN)  :: cp1
   real,intent(IN)  :: cp2
   real::d1, d2


   d1 = sqrt((x-x1)**2 + (y-y1)**2)
   d2 = sqrt((x-x2)**2 + (y-y2)**2)
   cp = (d1*cp2 + d2*cp1)/(d1 + d2)

END SUBROUTINE Gauss_linear


SUBROUTINE Quadralateral_Area(pt, area)
  IMPLICIT NONE
  real,INTENT(IN),DIMENSION(:,:)::pt
  real,INTENT(INOUT)::Area


  Area = 0.5*(pt(2,1)*pt(1,2) - pt(1,1)*pt(2,2) + pt(3,1)*pt(2,2) - pt(2,1)*pt(3,2) +
pt(4,1)*pt(3,2) - pt(3,1)*pt(4,2) + pt(1,1)*pt(4,2) - pt(4,1)*pt(1,2))
  Area = abs(Area)

END SUBROUTINE Quadralateral_Area


END PROGRAM CMARCtoANSYS
```

10.4. FORTRAN Program for Generation of Loads

```
PROGRAM CMARC_to_Load_test
!-------------------------------------------------------------!
! Programmer:     Rob S. Burgun        September 2003         !
!-------------------------------------------------------------!
! This program takes aerodynamic data from CMARC output and   !
! creates the loads for the load testing.                     !
!-------------------------------------------------------------!

IMPLICIT NONE
real,DIMENSION(6,100000):: aero_coord !pmarc coords and pressures
real,DIMENSION(200,200,7):: Panel_data

real,DIMENSION(500,500)::Grid_data

integer:: aero_num     !number of cmarc panels

real   :: Cl     !Lift coefficielem_numt from cmarc


aero_coord = 0.0
Grid_data = 0.0
Panel_data = -1000.0

CALL CMARC_read(aero_coord,aero_num,Panel_data,Cl)

CALL Get_Grid_Data(Grid_data,Panel_data,aero_coord,aero_num)

CALL Generate_the_loads(Grid_data,Panel_data,Cl)


CONTAINS


SUBROUTINE CMARC_read(aero_coord,panel,Panel_data,CL)

!-------------------------------------------------------------------------------------
!This subroutine reads Cmarc data and puts it into an array
!The Cmarc data must be unmodified and all spacing in the file
!must be as originally created by Cmarc.
!   aero_coord = array of Cmarc pressure [x pos, y pos, z pos, Cp, above/below sep plane]
!   panel      = number of Cmarc panels
!-------------------------------------------------------------------------------------

   IMPLICIT NONE
   REAL,INTENT(INOUT),DIMENSION(:,:)::aero_coord
   REAL,INTENT(INOUT),DIMENSION(:,:,:)::Panel_data
   INTEGER,INTENT(INOUT)::panel
   REAL,INTENT(INOUT) :: Cl

   CHARACTER(50)::filename
   CHARACTER(60)::temp
   INTEGER:: i
   INTEGER:: counter
   INTEGER:: step


      !User input of data file
   WRITE(*,'(A)',ADVANCE = "NO") ' Enter filelname for Cmarc data: '
   READ(*,*) filename
   OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')

   counter = 0
   DO
```

```
      read(45,'(A50)') temp
      IF (temp(2:6) =='PANEL') THEN
         read(45,'(A1)') temp
         counter = counter + 1
         step = 1
         DO
            READ(45,*)
panel,aero_coord(1,panel),aero_coord(2,panel),aero_coord(3,panel),aero_coord(4,panel)
            do i=1,4
               Panel_data(counter,step,i) = aero_coord(i,panel)
            end do
            step = step + 1
            read(45,'(A7)') temp
            IF (temp == '') THEN
               !found the end of a sequence of panels
               EXIT
            END IF
            BACKSPACE(45)
         END DO
      ELSE IF (temp(31:45) =='TOTAL COEFFICIE') THEN
         read(45,'(A1)') temp
         read(45,'(A1)') temp
         read(45,'(A1)') temp
         read(45,'(A45)') temp
         read(temp(35:40),*) Cl
         write(*,*) 'Cl (from Cmarc) = ',Cl
         EXIT
      END IF
   END DO

END SUBROUTINE CMARC_read

!*******************************************************************************************
!*************************************************************************

SUBROUTINE Get_Grid_Data(Grid_data,Panel_data,aero_coord,aero_number)
   IMPLICIT NONE
   real,INTENT(INOUT),DIMENSION(:,:)::Grid_data
   real,INTENT(INOUT),DIMENSION(:,:,:)::Panel_data
   real,INTENT(INOUT),DIMENSION(:,:)::aero_coord
   integer,INTENT(IN)::aero_number

   CHARACTER(50)::filename
   integer::i,j
   real::Number_of_Pannels
   real::temp_data
   real:: min
   real:: max
   real:: steps


      !User input of data file
   write(*,'(A)',ADVANCE = "NO") ' Enter the filename for the grid data or type "KEY IN" : '
   read(*,*) filename

   IF ((filename(1:1) == 'K') .OR. (filename(1:1) == 'k')) THEN
   ELSE
         !OPEN Loadt Testing Grid Input file
      OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')
         !Parse the spanwise panels
      read(45,*) Number_of_Pannels
      Grid_Data(0,1) = Number_of_Pannels
      DO i=1,int(Number_of_Pannels)
         read(45,*) Grid_data(i,1)
         if( Grid_data(i,1) == -1.0) THEN
            min = aero_coord(2,1)
            max = aero_coord(2,1)
            do j=2,aero_number
               if(aero_coord(2,j) < min) then
```

```
                  min = aero_coord(2,j)
               else if (aero_coord(2,j) > max) then
                  max = aero_coord(2,j)
               end if
            end do
            steps = (max - min)/int(Number_of_Pannels)
            do j=1,int(Number_of_Pannels)
               Grid_data(j,1) = max - steps*j
            end do
            Grid_data(Grid_data(0,1)+1,1) = max
            EXIT
         end if
      END DO
      CALL Sort_CMARC_to_Grid_spanwise(Grid_data,Panel_data,aero_coord,aero_number)
         !Parse each spanwise panel into sub-panels
      DO i=1,int(Number_of_Pannels)
         read(45,*) Grid_data(i,5)
         read(45,*) temp_data
         if (temp_data == -1.0) then
            min =  1000.0
            max = -1000.0
            do j=1,aero_number
               if(aero_coord(5,j) == i) THEN
                  if(aero_coord(1,j) < min) then
                     min = aero_coord(1,j)
                  else if (aero_coord(1,j) > max) then
                     max = aero_coord(1,j)
                  end if
               end if
            end do
            steps = (max - min)/int(Grid_data(i,5))
            do j=1,int(Grid_data(i,5))
               Grid_data(i,10+j) = min + steps*j
            end do
         else
            Grid_data(i,11) = temp_data
            DO j=2,int(Grid_data(i,5))
               read(45,*) Grid_data(i,10+j)
            end do
         end if
      end do
      CALL Sort_CMARC_to_Grid_panels(Grid_data,Panel_data,aero_coord,aero_number)
   END IF

END SUBROUTINE Get_Grid_Data



SUBROUTINE Sort_CMARC_to_Grid_spanwise(Grid_data,Panel_data,aero_coord,aero_number)
   Implicit None
   real,INTENT(INOUT),DIMENSION(:,:)::Grid_data
   real,INTENT(INOUT),DIMENSION(:,:,:)::Panel_data
   real,INTENT(INOUT),DIMENSION(:,:)::aero_coord
   integer,INTENT(IN)::aero_number

   integer::i,j,k


   do i=1,Grid_data(0,1)
      do j=1,aero_number
         if ((i == 1) .AND. (aero_coord(2,j) >= Grid_data(i,1))) then
            aero_coord(5,j) = real(i)
            Grid_data(i,2) = Grid_data(i,2) + 1.0
         else if ((aero_coord(2,j) >= Grid_data(i,1)) .AND. (aero_coord(2,j) <= Grid_data(i-
1,1))) then
            aero_coord(5,j) = real(i)
            Grid_data(i,2) = Grid_data(i,2) + 1.0
         end if
      end do
```

```
      do j=1,200
         do k=1,200
            if ((Panel_data(j,k,1) /= -1000.0).AND.(Panel_data(j,k,5) == -
1000.0).AND.(Panel_data(j,k,2) >= Grid_data(i,1))) THEN
               Panel_data(j,k,5) = real(i)
            end if
            if (Panel_data(j,k,1) == -1000.0) THEN
               exit
            end if
         end do
      end do
   end do
END SUBROUTINE Sort_CMARC_to_Grid_spanwise


SUBROUTINE Sort_CMARC_to_Grid_panels(Grid_data,Panel_data,aero_coord,aero_number)
   Implicit None
   real,INTENT(IN),DIMENSION(:,:)::Grid_data
   real,INTENT(INOUT),DIMENSION(:,:,:)::Panel_data
   real,INTENT(INOUT),DIMENSION(:,:)::aero_coord
   integer,INTENT(IN)::aero_number

   integer::i,j,k,l
   integer::total


   total = 0
   do i=1,Grid_data(0,1)
     do j=1,Grid_data(i,5)
       do k=1,aero_number
         if (aero_coord(5,k) == i) then
           if ((aero_coord(1,k) <= Grid_data(i,10+j)) .AND. (aero_coord(6,k) == 0.0)) then
            aero_coord(6,k) = j
          end if
         end if
       end do

       do k=1,200
         do l=1,200
           if (Panel_data(k,l,5) == i) then
             if ((Panel_data(k,l,1) <= Grid_data(i,10+j)) .AND. (Panel_data(k,l,6) == -
1000.0) .AND. (Panel_data(k,l,1) /= -1000.0)) then
               Panel_data(k,l,6) = real(j)
               total = total + 1
             end if
           end if
           if (Panel_data(j,k,1) == -1000.0) THEN
             exit
           end if
         end do
       end do
     end do
   end do
   write(*,*) 'total =',total

END SUBROUTINE Sort_CMARC_to_Grid_panels



SUBROUTINE Generate_the_loads(Grid_data,Panel_data,Cl)
   IMPLICIT NONE
   real,INTENT(INOUT),DIMENSION(:,:)::Grid_data
   real,INTENT(INOUT),DIMENSION(:,:,:)::Panel_data
   real,INTENT(IN) :: Cl

   real,DIMENSION(4,3)::point
   integer::i,j,k,l
   real:: temp
   real:: area
```

```
real:: Cp
real:: Pressure
real:: number
real:: total
real:: den
real:: q

real :: weight
real :: surface_area
integer:: number_of_gs


   !User inputs to calculate dynamic pressure
write(*,'(A)',ADVANCE = "NO") ' Enter the Weight of the plane (lbs.): '
read(*,*) weight
write(*,'(A)',ADVANCE = "NO") ' Enter the number of G''s: '
read(*,*) number_of_gs
write(*,'(A)',ADVANCE = "NO") ' Enter the Surface Area of the plane (ft^2): '
read(*,*) surface_area

   !Dynamic pressure calculation
den = 0.0023769

surface_area = 2*weight/Cl/den/12**3

q=(weight*number_of_gs)/(surface_area*Cl)

total = 0.0
number = 0.0
do i=1,int(grid_data(0,1))
  do j=1,int(Grid_data(i,5))
    temp = 0.0
    do k=2,200
      do l=2,200
        if (Panel_data(k,l,5) == -1000.0) then
          exit
        end if
        if ((Panel_data(k,l,5) == i).AND.(Panel_data(k,l,6) == j)) then
             !calculate the Cp at the mid-point of the area
             !calculate the area with corners at k,(k-1),l,(l-1)
             !load equals the Cp*area*q

          if (Panel_data(k,l,2) >= 0.0) THEN
                !
                !   (k-1),(l)    (k-1),(l-1)
                !         4            1
                !         3            2
                !     (k),(l)       (k),(l-1)
            point(1,1) = Panel_data(k-1,l-1,1)
            point(1,2) = Panel_data(k-1,l-1,2)
            point(1,3) = Panel_data(k-1,l-1,4)

            point(2,1) = Panel_data(k,l-1,1)
            point(2,2) = Panel_data(k,l-1,2)
            point(2,3) = Panel_data(k,l-1,4)

            point(3,1) = Panel_data(k,l,1)
            point(3,2) = Panel_data(k,l,2)
            point(3,3) = Panel_data(k,l,4)

            point(4,1) = Panel_data(k-1,l,1)
            point(4,2) = Panel_data(k-1,l,2)
            point(4,3) = Panel_data(k-1,l,4)
          else
                !
                !   (k-1),(l)    (k-1),(l-1)
                !         3            2
                !         4            1
                !     (k),(l)       (k),(l-1)
```

```
              point(1,1) = Panel_data(k,l-1,1)
              point(1,2) = abs(Panel_data(k,l-1,2))
              point(1,3) = Panel_data(k,l-1,4)

              point(2,1) = Panel_data(k-1,l-1,1)
              point(2,2) = abs(Panel_data(k-1,l-1,2))
              point(2,3) = Panel_data(k-1,l-1,4)

              point(3,1) = Panel_data(k-1,l,1)
              point(3,2) = abs(Panel_data(k-1,l,2))
              point(3,3) = Panel_data(k-1,l,4)

              point(4,1) = Panel_data(k,l,1)
              point(4,2) = abs(Panel_data(k,l,2))
              point(4,3) = Panel_data(k,l,4)
            end if

            CALL Calculate_CP_at_center(point, Cp)
            CALL Quadralateral_Area(point, area)

            pressure = Cp*Area*q
            temp = temp + pressure
            number = number + 1.0
          end if
        end do
      end do

      write(*,*) Grid_data(i,1),Grid_data(i,10+j),temp
      total = total + temp
    end do
  end do
  write(*,*) 'total = ',total,total*2
  write(*,*) 'panels =',number

END SUBROUTINE Generate_the_loads

SUBROUTINE Calculate_CP_at_center(pt, Cp)
   IMPLICIT NONE
   real,INTENT(IN),DIMENSION(:,:)::pt
   real,INTENT(INOUT)::Cp

   real::area1
   real::area2
   real::area3
   real::area4
   real,DIMENSION(4,2)::mid_pts
   real,DIMENSION(4,2)::corners
   real,DIMENSION(2)::center

      !            d               y
      !    4                1
      !    c       X        a      ^
      !    3                2      |
      !            b               +----> x
      !

   center(2) = (pt(1,2) + pt(2,2))/2
   mid_pts(1,1) = (pt(1,1) + pt(2,1))/2
   mid_pts(1,2) = center(2)
   mid_pts(2,1) = (pt(2,1) + pt(3,1))/2
   mid_pts(2,2) = pt(2,2)
   mid_pts(3,1) = (pt(3,1) + pt(4,1))/2
   mid_pts(3,2) = center(2)
   mid_pts(4,1) = (pt(4,1) + pt(1,1))/2
   mid_pts(4,2) = pt(1,2)
   center(1) = (mid_pts(2,1) + mid_pts(4,1))/2

   corners(1,1) = pt(1,1)
   corners(1,2) = pt(1,2)
```

```
   corners(2,1) = mid_pts(1,1)
   corners(2,2) = mid_pts(1,2)
   corners(3,1) = center(1)
   corners(3,2) = center(2)
   corners(4,1) = mid_pts(4,1)
   corners(4,2) = mid_pts(4,2)
   CALL Quadralateral_Area(corners, area1)

   corners(1,1) = pt(2,1)
   corners(1,2) = pt(2,2)
   corners(2,1) = mid_pts(2,1)
   corners(2,2) = mid_pts(2,2)
   corners(3,1) = center(1)
   corners(3,2) = center(2)
   corners(4,1) = mid_pts(1,1)
   corners(4,2) = mid_pts(1,2)
   CALL Quadralateral_Area(corners, area2)

   corners(1,1) = pt(3,1)
   corners(1,2) = pt(3,2)
   corners(2,1) = mid_pts(3,1)
   corners(2,2) = mid_pts(3,2)
   corners(3,1) = center(1)
   corners(3,2) = center(2)
   corners(4,1) = mid_pts(2,1)
   corners(4,2) = mid_pts(2,2)
   CALL Quadralateral_Area(corners, area3)

   corners(1,1) = pt(4,1)
   corners(1,2) = pt(4,2)
   corners(2,1) = mid_pts(4,1)
   corners(2,2) = mid_pts(4,2)
   corners(3,1) = center(1)
   corners(3,2) = center(2)
   corners(4,1) = mid_pts(3,1)
   corners(4,2) = mid_pts(3,2)
   CALL Quadralateral_Area(corners, area4)

   Cp = (area1*pt(3,3) + area2*pt(4,3) + area3*pt(1,3) +
area4*pt(2,3))/(area1+area2+area3+area4)

END SUBROUTINE Calculate_CP_at_center

SUBROUTINE Quadralateral_Area(pt, area)
   IMPLICIT NONE
   real,INTENT(IN),DIMENSION(:,:)::pt
   real,INTENT(INOUT)::Area


   Area = 0.5*(pt(2,1)*pt(1,2) - pt(1,1)*pt(2,2) + pt(3,1)*pt(2,2) - pt(2,1)*pt(3,2) +
pt(4,1)*pt(3,2) - pt(3,1)*pt(4,2) + pt(1,1)*pt(4,2) - pt(4,1)*pt(1,2))
   Area = abs(Area)

END SUBROUTINE Quadralateral_Area


END PROGRAM CMARC_to_Load_test
```

10.5. Strain Gauge Data Reduction FORTRAN Program

```
Program Load_Test_Data_Reduction
!-------------------------------------------------------------!
! Programmer:     Rob S. Burgun        September 2003         !
!-------------------------------------------------------------!
! This program reduces the strain gauge data that was         !
! acquired from the System 6000 machine.                      !
!-------------------------------------------------------------!
   IMPLICIT NONE
   real,DIMENSION(5,1000000):: Strain_Gauge_data
   real*8,DIMENSION(5):: Strain_Min
   real*8,DIMENSION(5):: Strain_Mean
   real*8,DIMENSION(5):: Strain_Max
   real*8,DIMENSION(5):: Strain_RMS
   real*8,DIMENSION(5):: Strain_Std_Deviation

   INTEGER,DIMENSION(7):: Tabs

   CHARACTER(50)::filename
   CHARACTER(50)::filename2
   CHARACTER(50)::temp_data
   INTEGER:: i,j
   INTEGER:: counter
   INTEGER:: counter2
   INTEGER:: step
   INTEGER:: raw_data
   REAL::temp


      !User input of data file
   WRITE(*,'(A)',ADVANCE = "NO") ' Enter the Strain Gauge Data Filename : '
   READ(*,*) filename
   OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')

   Strain_Min = 0.0
   Strain_Mean = 0.0
   Strain_Max = 0.0
   Strain_RMS = 0.0
   Strain_Std_Deviation = 0.0
   counter = 1
   do
      read(45,'(A50)') temp_data
      IF (temp_data(1:2) =='ID') THEN
         exit
      end if
   end do

   read(45,'(A50)') temp_data
   counter2 = 1
   do i=1,50
      if (temp_data(i:i) == '   ') THEN
         Tabs(counter2) = i
         counter2 = counter2 + 1
      end if
   end do
   raw_data = 5
   IF (tabs(2)+1 == tabs(3)) raw_data = raw_data - 1
   IF (tabs(3)+1 == tabs(4)) raw_data = raw_data - 1
   IF (tabs(4)+1 == tabs(5)) raw_data = raw_data - 1
   IF (tabs(52)+1 == tabs(6)) raw_data = raw_data - 1
   IF (tabs(6)+1 == LEN_TRIM(temp_data)) raw_data = raw_data - 1

   BACKSPACE(45)
   do
      IF (raw_data == 1) THEN
```

```
         read(45,*) step,temp,Strain_Gauge_data(1,counter)
      ELSE IF (raw_data == 2) THEN
         read(45,*) step,temp,Strain_Gauge_data(1,counter),Strain_Gauge_data(2,counter)
      ELSE IF (raw_data == 3) THEN
         read(45,*)
step,temp,Strain_Gauge_data(1,counter),Strain_Gauge_data(2,counter),Strain_Gauge_data(3,count
er)
      ELSE IF (raw_data == 4) THEN
         read(45,*)
step,temp,Strain_Gauge_data(1,counter),Strain_Gauge_data(2,counter),Strain_Gauge_data(3,count
er),Strain_Gauge_data(4,counter)
      ELSE
         read(45,*)
step,temp,Strain_Gauge_data(1,counter),Strain_Gauge_data(2,counter),Strain_Gauge_data(3,count
er),Strain_Gauge_data(4,counter),Strain_Gauge_data(5,counter)
      END IF
      IF (EOF(45)) THEN
         exit
      END IF
      counter = counter + 1
   end do
   close(45)

   do i=1,step
      do j=1,raw_data
         Strain_Mean(j) = Strain_Mean(j) + Strain_Gauge_data(j,i)
      end do
   end do
   do j=1,raw_data
      Strain_Mean(j) = Strain_Mean(j)/step
   end do

      !calculate the standard deviation
      !            _____________
      !           /         2    '
      ! sigma =  / (x - mean)
      !        \/
   do i=1,step
      do j=1,raw_data
         Strain_Std_Deviation(j) = Strain_Std_Deviation(j) + (Strain_Gauge_data(j,i) -
Strain_Mean(j))**2
      end do
   end do
   do j=1,raw_data
      Strain_Std_Deviation(j) = sqrt(Strain_Std_Deviation(j)/step)
   end do

      !find the minumum and maximum of the strain gauge data
   do i=1,step
      if (i==1) THEN
         do j=1,raw_data
            Strain_Min(j) = Strain_Gauge_data(j,i)
            Strain_Max(j) = Strain_Gauge_data(j,i)
         end do
      ELSE
         do j=1,raw_data
            IF (Strain_Gauge_data(j,i) < Strain_Min(j)) THEN
               Strain_Min(j) = Strain_Gauge_data(j,i)
            END IF
            IF (Strain_Gauge_data(j,i) > Strain_Max(j)) THEN
               Strain_Max(j) = Strain_Gauge_data(j,i)
            END IF
         end do
      END IF
   end do

   filename2 = filename
   do i=1,50
      if (filename(i:(i+4)) == '.txt') then
```

```
            filename2(i:(i+11)) = '_reduced.txt'
            exit
         end if
      end do

      OPEN(UNIT=55,file=filename2,STATUS='NEW',POSITION='REWIND')
      do j=1,raw_data
         write(55,*) 'Strain Gauge',j
         write(55,*) 'Minimum Strain =',Strain_Min(j)
         write(55,*) 'Maximum Strain =',Strain_Max(j)
         write(55,*) '   Mean Strain =',Strain_Mean(j)
         write(55,*) ' Std Deviation =',Strain_Std_Deviation(j)
         write(55,*)
      end do
      close(55)

End Program Load_Test_Data_Reduction
```

10.6. FORTRAN Code for Parsing ANSYS Results at Strain Gauge Locations

```
PROGRAM ANSYS_Strain_finder
!-------------------------------------------------------------!
! Programmer:     Rob S. Burgun        September 2003         !
!-------------------------------------------------------------!
! This program takes the strain information from ANSYS and    !
! strain gauge locations and finds the strain values.         !
!-------------------------------------------------------------!

IMPLICIT NONE
real,DIMENSION(6,1000000):: node_results
real,DIMENSION(4,1000000):: node_coord !ansys node coordinates
real,DIMENSION(4,100)::     strain_coord !strain coordinates

integer,DIMENSION(1000000):: node_label

integer:: strain_number

integer:: node_num     !number of ansys node points
integer:: strain_num     !number of cmarc panels
integer::i

node_results = 0.0

CALL Get_node_data(strain_coord,strain_number)

CALL Ansys_nodes_read(node_coord,node_label,node_num)

CALL Strain_read(node_results,strain_num)

CALL Omit_unused_nodes(node_coord,node_num,node_results)

CALL Find_Closest_node(strain_coord,strain_number,node_coord,node_num)

CALL output(strain_coord,strain_number,node_results)

CONTAINS

!*****************************************************************************

SUBROUTINE Get_node_data(strain_coord,strain_number)

   IMPLICIT NONE
   real,INTENT(INOUT),DIMENSION(:,:)::strain_coord
   INTEGER, INTENT(INOUT):: strain_number

   INTEGER :: i

   CHARACTER(50)::filename


      !User input of data file
   write(*,'(A)',ADVANCE = "NO") ' Filename for the strain gauge locations or type "KEY IN" :
'
   read(*,*) filename

   IF ((filename(1:1) == 'K') .OR. (filename(1:1) == 'k')) THEN
      write(*,'(A)',ADVANCE = "NO") ' Enter the number of strain locations to search for :'
      read(*,*) strain_number
      DO I=1,strain_number
         write(*,*) 'Enter the three strain points for case',i
         READ(*,*) strain_coord(1,I),strain_coord(2,I),strain_coord(3,I)
      END DO
   ELSE
         !OPEN Cmarc file
```

```
      OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')
      read(45,*) strain_number
      DO I=1,strain_number
         READ(45,*) strain_coord(1,I)
         read(45,*) strain_coord(2,I)
         read(45,*) strain_coord(3,I)
      END DO
      CLOSE(45)
   END IF

END SUBROUTINE Get_node_data


!*******************************************************************************

SUBROUTINE Ansys_nodes_read(node_coord,node_label,node_num)
   IMPLICIT NONE

   real,DIMENSION(:,:),INTENT(INOUT):: node_coord
   integer,DIMENSION(:),INTENT(INOUT):: node_label

   integer,INTENT(INOUT):: node_num
   integer,DIMENSION(4):: ansys_coord_sys
   integer:: a,b
   integer:: i

   character(50) filename
   CHARACTER(2)::dummy
   CHARACTER(1)::test


   write(*,'(A)',ADVANCE = "NO")' Enter filename that contains the mesh nodes : '
   read(*,*)filename

   OPEN(99,file=filename,status='old',position='rewind')

      !Initialize
   I=0
   node_num=0
   node_label=0
   ansys_coord_sys = 0

20 FORMAT(3(BZ,I8),F16.0,F16.0,F16.0)

   DO
         !Reads a line from the data file.
      READ(99,'(A1)') test

         !Checks to see if "test" is a space
      IF (test==' ') THEN
         BACKSPACE(99)
         I=I+1
         READ(99,20) node_label(I),a,b,node_coord(1,node_label(I))&
            &,node_coord(2,node_label(I)),node_coord(3,node_label(I))
         READ(99,'(BZ,A2)') dummy
         IF ((dummy=='0').OR.(dummy=='-1')) EXIT
         BACKSPACE(99)
         node_num=node_num+1
      END IF
   END DO

END SUBROUTINE Ansys_nodes_read

!*******************************************************************************

SUBROUTINE Strain_read(node_results,strain_num)
   IMPLICIT NONE
   REAL,INTENT(INOUT),DIMENSION(:,:)::node_results
   INTEGER,INTENT(INOUT)::strain_num
```

```
   CHARACTER(50)::filename
   CHARACTER(60)::temp
   CHARACTER(60)::temp2
   INTEGER:: counter
   INTEGER:: step


      !User input of data file
   WRITE(*,'(A)',ADVANCE = "NO") ' Enter filelname of the ANSYS Strain data: '
   READ(*,*) filename
   OPEN(UNIT=45,file=filename,STATUS='OLD',POSITION='REWIND')

!   69136       0       06.074949292E+001-3.46589128E+001-1.39849868E+000
!  149323  0.72548E-07-0.99330E-05 0.93959E-05 0.12833E-15 0.14191E-04 0.19049E-15
30 FORMAT(I9,6(F12.0))

   counter = 0
   DO
      read(45,'(A50)') temp
      read(45,'(A50)') temp2
      BACKSPACE(45)

      IF ((temp(1:8) =='    NODE').AND.(temp2 /= '')) THEN
         step = 1
         DO
             counter = counter + 1
            READ(45,30)
strain_num,node_results(1,strain_num),node_results(2,strain_num),node_results(3,strain_num),n
ode_results(4,strain_num),node_results(5,strain_num),node_results(6,strain_num)
            step = step + 1
            read(45,'(A17)') temp
            IF (temp == '') THEN
               !found the end of a sequence of panels
               counter = counter - 1
               EXIT
            END IF
            BACKSPACE(45)
         END DO
      ELSE IF (temp(1:15) ==' MINIMUM VALUES') THEN
         EXIT
      END IF
   END DO

END SUBROUTINE Strain_read

!*************************************************************************************

SUBROUTINE Omit_unused_nodes(node_coord,node_num,node_results)
   IMPLICIT NONE

   real,DIMENSION(:,:),INTENT(INOUT):: node_coord
   integer,INTENT(IN):: node_num
   real,DIMENSION(:,:),INTENT(IN):: node_results
   integer:: i


   do i=1,node_num
      if ((node_results(1,i) == 0.0) .AND. (node_results(1,i) == 0.0)) THEN
         node_coord(1,i) = 0.0
         node_coord(2,i) = 0.0
         node_coord(3,i) = 0.0
      end if
   end do

END SUBROUTINE Omit_unused_nodes

!*************************************************************************************
```

```
SUBROUTINE Find_Closest_node(strain_coord,strain_number,node_coord,node_num)
   IMPLICIT NONE

   real,DIMENSION(:,:),INTENT(INOUT):: strain_coord
   real,DIMENSION(:,:),INTENT(INOUT):: node_coord
   INTEGER, INTENT(INOUT):: strain_number
   integer,INTENT(IN):: node_num

   real :: radius
   real :: radius_new
   real :: dx
   real :: dy
   real :: dz
   integer:: i,j


   do i=1,strain_number
      radius = 1000.0
      do j=1,node_num
         if (node_coord(1,j) /= 0.0) THEN
            dx = strain_coord(1,i) - node_coord(1,j)
            dy = strain_coord(2,i) - node_coord(2,j)
            dz = strain_coord(3,i) - node_coord(3,j)
            radius_new = sqrt(dx**2 + dy**2 + dz**2)
            IF (radius_new < radius) THEN
               radius = radius_new
               strain_coord(4,i) = j
            END IF
         END IF
      end do
   end do

END SUBROUTINE Find_Closest_node

!*******************************************************************************

SUBROUTINE output(strain_coord,strain_number,node_results)
   IMPLICIT NONE

   real,DIMENSION(:,:),INTENT(IN):: strain_coord
   real,DIMENSION(:,:),INTENT(IN):: node_results
   INTEGER, INTENT(IN):: strain_number

   integer:: i
   CHARACTER(50)::filename


      !User input of data file
   WRITE(*,'(A)',ADVANCE = "NO") ' Results filelname for the ANSYS Strain data: '
   READ(*,*) filename
   OPEN(UNIT=45,file=filename,STATUS='UNKNOWN',POSITION='REWIND')

   write(45,*) '   gauge#        EPTOX           EPTOY            EPTOZ            EPTOXY
EPTOYZ      EPTOXZ  '

   do i=1,strain_number

write(45,*)i,node_results(1,strain_coord(4,i)),node_results(3,strain_coord(4,i)),node_results
(3,strain_coord(4,i)),node_results(4,strain_coord(4,i)),node_results(5,strain_coord(4,i)),nod
e_results(6,strain_coord(4,i))
   end do
   close(45)

END SUBROUTINE output

END PROGRAM ANSYS_Strain_finder
```