WingSMITH

A framework for automated synthesis and perturbation of aircraft wing structural meshes

By:
Michael Trauttmansdorff

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE
BACHELOR OF APPLIED SCIENCE

DIVISION OF ENGINEERING SCIENCE
FACULTY OF APPLIED SCIENCE AND ENGINEERING
UNIVERSITY OF TORONTO

Supervisor: J. Martins

April 2005

Abstract

This thesis describes a software tool, WingSMITH, for designing, generating, and perturbing structural meshes of aircraft wing structures. It provides useful and powerful use methods for the creation of a mesh representation of the spar/rib/stringer structural components of a wing. It also provides a powerful yet light weight structural mesh perturber which enables deformation of the mesh. WingSMITH is well suited for use in aerostructural optimization applications which perturb a structural mesh through the adjustment of design variables.

The architecture of the WingSMITH program has been designed from the ground up to be as generalized, modular, and extensible as possible. Its architecture allows programmers to quickly understand the functioning of the code, as well as to develop new modules which can be added to the tool. Open standards and programming best practices have been applied in order to ensure maintainability and long term usefulness of the program

WingSMITH combines the functions of design automation and mesh perturbation into a powerful and versatile package that will increase the productivity of engineers and researchers by allowing them to rapidly mesh, optimize, and test structural configurations that would otherwise be difficult or impractical to specify manually.

Acknowledgements

I wish to thank the following people for their support and help during the preparation of this thesis:

- Professor Joaquim Martins for his time, support, and insightful guidance, as well as for giving me the opportunity to pursue such a rewarding and interesting thesis topic while contributing to the toolset being used to expand the field of multidisciplinary optimization.

- My friends Hanh, Nick, and Anna whose support, timely suggestions, and feedback have helped me make enormous improvements in the quality of this thesis.

- My parents, Fritz and Gise Trauttmansdorff, for giving me the opportunity to pursue such an interesting field of study and for their tireless encouragement of all my endeavors.

Table of Contents

Table of Figures

Nomenclature

Term:	Definition:
chord	The straight line connecting the leading and trailing edges of a wing section.
c	Non-dimensional coordinate, defined as fraction of local chord length relative to theleading edge of a wing.
FEAP	Finite Element Analysis Program
FEM	Finite Element Model
FEA	Finite Element Analysis
GUI	Graphical User Interface
HSCT	Hypersonic Civilian Transport
MDO	Multidisciplinary Optimization
MVC	Model View Controller
Planform	The downward projected area of a wing, (ie, a 2-d approximation of the wing's characteristic shape)
TWSAP	Thin Wall Structural Analysis Program
WingSMITH	Wing Specification, Meshing, and Iteration TecHnology

1 Introduction

1.1 Background

Aircraft design cycles typically begin with the development and analysis of a conceptual model of the aircraft that provides an accurate estimate of the performance of a given design. One of the major tasks in the preliminary design stage is the synthesis of a structural model of the wing that can be used for analysis or optimization. The aim is to produce a conceptual model that is both realistic and satisfies the particular design criteria. This thesis describes a new software tool which allows for rapid production of such aircraft wing models for use in an optimization environment. The power of multidisciplinary optimization (MDO) techniques to improve the performance of a finished aircraft makes the effective application of these techniques an important task within the aerospace field.

> The greatest potential benefits can be obtained by applying formal optimization at the preliminary design stage.
>
> -Abdo et al., Bombardier Aerospace [3]

By optimizing designs early on, engineers are able to quantify the potential of a design before committing to detailed design or production. This allows for faster evaluation of design concepts, and hence more rapid development of attractive and practical viable designs.

To date, automated tools to generate a finite element model of an aircraft wing have been designed for some specific optimization methodologies, but they are either impractical for general use, or they are part of proprietary design methods. Bombardier Aerospace has developed a Thin Wall Structural Analysis Program (TWSAP) [9] as part of their MDO framework. TWSAP is used to generate a stick model representation of the wing structure for determination of the aeroelastic coefficients.

A 'finite element pre-processor' was written at the Georgia Institute of Technology to place a wing box into surface geometry of a wing, and create an input file for the ASTROS structural optimization tool [10]. However, their application focuses on system design and structural optimization, as opposed to multidisciplinary optimization.

1.2 Objectives

The aim of this thesis is the development of a tool that allows rapid generation of a realistic aircraft wing structural model. The tool should be useable both as a one-time-use utility for users to create a finite element model for pure structural optimization, and as a sub-component within an aero-structural or other generalized optimization process.

> "high-fidelity multidisciplinary design environments are still in their infancy and fast progress in research can only be achieved if a number of different ideas can be tried in quick succession"
>
> Alosno et al. [14]

In order to speed up the design cycle, the tool accepts inputs in the form of wing surface geometry and user specified parameters as to the structural layout of the wing. It outputs a finite element model containing node locations, connectivity, and design variables, to be used in the Finite Element Analysis Program (FEAP) [8].

> MDO is, emphatically, not a push-button design. Hence, the human interface is crucially important to enable engineers to control the design process and to inject their judgment and creativity into it.
>
> -Sobieszczanski-Sobieski , J. and Haftka, R. [2]

The purpose of this tool is to allow the designer the freedom to inject his/her creativity in to the design process while removing the labor intensive tasks traditionally associated with the creation of a finite element model.
WingSMITH is intended to provide the designer with the functionality of existing tools (TWSAP and Georgia Tech's 'pre-processor') for use in MDO. Furthermore, WingSMITH is capable of exceeding the capabilities of existing tools due to its extensible architecture which promotes long term code re-use, maintainability, and improvements in functionality through the creation of plug-in modules.

1.3 Thesis Outline

In chapter 2, this thesis begins with a review of modern and historical aircraft wing structural layouts, focusing on trends and patterns. Chapter 3 continues with a discussion of existing methods for mesh perturbation, and how they apply to finite element models used in MDO. In chapter 4, the WingSMITH platform is presented, and its use and merits as a structural design tool are discussed. Chapter 6 outlines how WingSMITH can be used in an optimization framework to facilitate aero-structural optimization. Chapter 7 reviews the design philosophy and architecture which shaped the WingSMITH program. Appendix A contains a user guide for setting up and running the WingSMITH platform as a stand-alone or optimization tool. Appendix B contains instructions for extending WingSMITH by adding new modules.

2 Aircraft Wing Structure

The purpose of this chapter is to provide an overview of aircraft wing structural layouts, and to illustrate the conventional abstraction of these structures into a representative finite element model. The strategy used by WingSMITH to assemble the finite element model is strongly tied to the consistent and structured nature of wing component layouts.

The first part of this chapter reviews the various structural components of wing structures, and is followed by a discussion of the Finite Element abstraction of these structural members. Next trends and patterns in Wing Structure Layouts are discussed, and the chapter ends with a set of capabilities which WingSMITH must have in order to accurately represent modern wing structural layouts.

2.1 Components

This section defines the various structural components which are frequently seen in wing designs, and surveys trends or patterns which are applicable to the WingSMITH platform.

2.1.1 Spars

Spars bear the main aerodynamic loads on a wing are form the structural interface between the fuselage and the wing. They are configured approximately normal to the flow, extending outward from the fuselage. Conventional small aircraft have 1 or 2 spars, while larger commercial aircraft typically have 2 or 3 spars [1].

Both business jets and airliners show remarkable consistency in their wing structural layouts, particularly with respect to spar location. It is clear that, for commercial transport aircraft, 2 spars spanning the length of the wing is the norm.

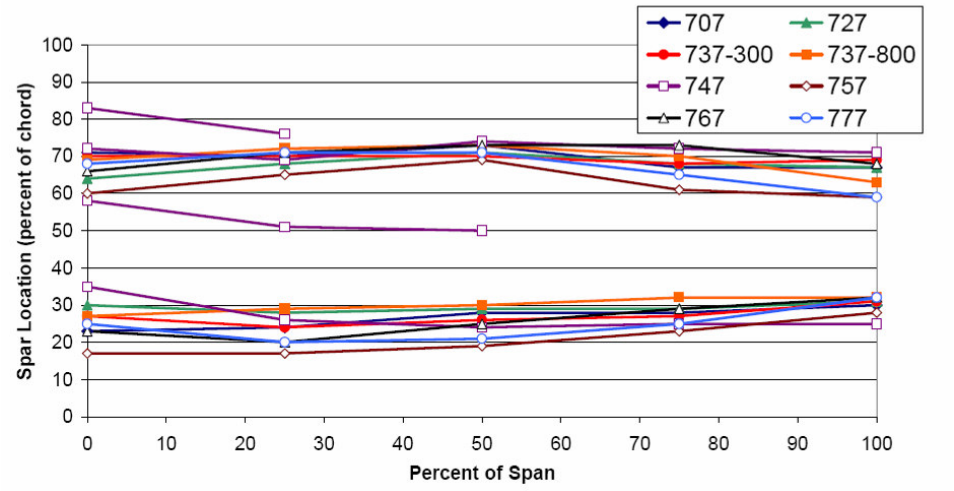


Figure 2-1: Spar location chart for Boeing 7X7 Series [1]

Sensmeier and Samareh [1] reported on patterns in wing structure layouts, and their findings were presented as graphs illustrating the chordwise distribution of spars within aircraft families (See Figure 2-1 above for an example) . Their conclusions are summarized in the table below.

	Front Spar	Variation	Rear Spar	Variation
Boeing 7X7 series	0.25c	~ 15%	0.7c	~ 15%
Airbus Jets	0.3c	~ 20%	0.7c	~ 15%
DC-Series	0.3c	~ 25%	0.7c	~ 15%

Table 1: Approximate Spar location trends based on [1]

All of the aircraft have front spars at around 25% to 30% along the chord length, and the rear spar almost exactly at 70% [1]. While the locations vary somewhat from model to model, the ability to replicate these patterns will allow WingSMITH to accurately represent civil transport aircraft structures.

It is important to note that some of the civil transport aircraft have short spars near the root of the wing in addition to the 2 primary spars. These serve to stiffen this region where the bending moments are greatest, as well as sometimes providing hard points for landing gear or flap attachment.

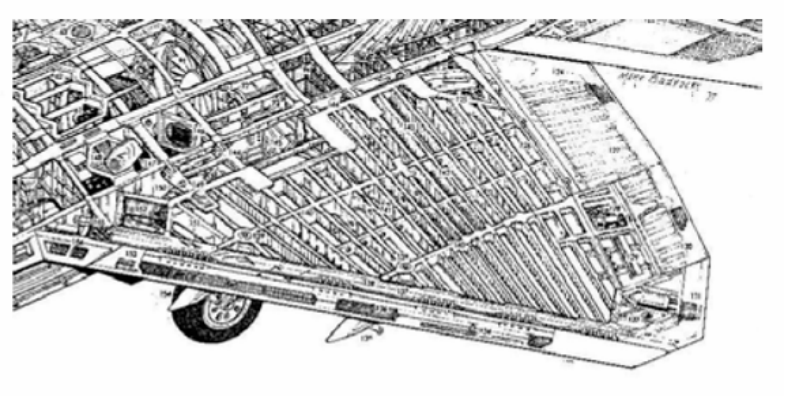


Figure 2-2: Lockheed Martin F-22A Raptor wing [1]

Military aircraft tend to have many more spars, but the exact quantity is highly variable from design to design. Sensmeier and Samareh also observed the differences in design philosophy between American aircraft, which feature many spars and few ribs, vs. Russian aircraft, which feature many ribs and fewer spars [1]. For this reason it is important that WingSMITH does not restrict the quantity of spars which the user can specify.

2.1.2 Ribs

Ribs serve to connect spars to each other and support the skin of the aircraft wing via stringers. Typically ribs are oriented in the streamwise direction, but they are sometimes also fanned in swept portions of wings (see Figure 2-5 below).

Sensmeier and Samareh [1] also reported on rib configurations, finding high regularity in spacing across most civilian aircraft. They show that ribs are typically spaced 2 feet apart (for Boeing, Airbus, and DC-series jets), and explain that this spacing is 'likely driven by maintenance access requirements'.

	Avg Spacing	Variation
Boeing 7X7 series	2 ft	0.25 ft
Airbus Jets	2.25 ft	1.5 ft
DC-Series	2 ft	1 ft

Table 2: Approximate Rib Spacing trends (perpendicular to fuselage axis) based on [1]

It is thus important that WingSMITH allows the creation of regular repeating rib patterns to accurately replicate this design trend.

As mentioned in section 2.1.1 above, the total number of ribs and spars, as well as their relative proportions, vary greatly in military aircraft. While military applications are not the intended domain of WingSMITH, it is desirable to accommodate all common structural design approaches.

2.1.3 Stringers

Stringers support skin panels and prevent thin-walled buckling. These are typically attached to or part of the skin of the aircraft wing, oriented in the same direction as the spar, and spanning between ribs. In lower cost aircraft they are installed as parts, while in higher cost (i.e. military) aircraft they are sometimes machined as an integral part of a skin panel. In order to accomplish higher fidelity modeling of a wing structure, these components are included in the WingSMITH framework.

2.2 Finite Element Abstraction of Components

This section presents the finite element analogues to the above structural members, and outlines how they are assembled to mimic the structural characteristics of a real wing.

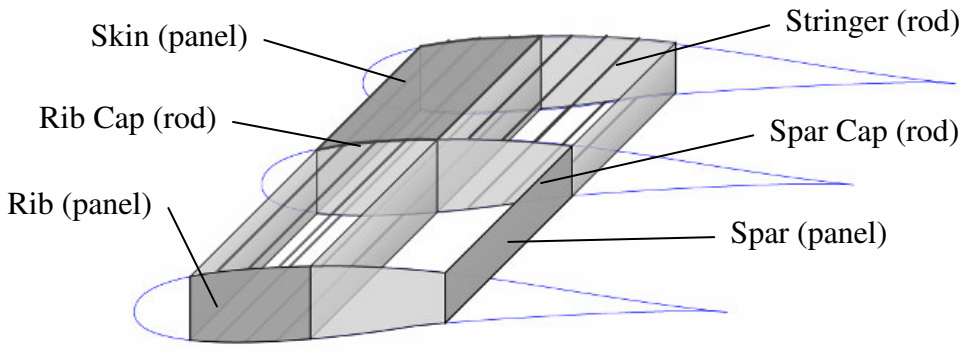


Figure 2-3: Wing Box Finite Element Model

Typically spars and ribs are modeled by a suitable quadrilateral element that can withstand shear and bending. Rib Caps and spar caps are represented by suitable rod elements such as Frames. WingSMITH adopts this abstraction by producing quadrilateral and rod type elements, independent of the particular element types used in the Finite Element Approach (FEA) solver. The specification of element properties other than node number and connectivity is specified upon transfer of the model from WingSMITH to the solver.

2.3 Wing Structure Layouts

Sensmeier and Samareh [1] reported on trends and patterns which can be found in aircraft structural layouts. Of particular interest are their observations of patterns in the layout of wing structural components within Civilian and Military aircraft.

2.3.1 Civilian Aircraft

Commercial transports and Business aircraft are designed to achieve long range flight with high fuel efficiency. They have high aspect ratio swept wings which are not subjected to the aerodynamic loads of high-G maneuvers. Wings typically have a large chord at the root, becoming more slender towards the tip. Ribs tend to be arranged in the direction of flight in the root portion of the wing. However, from the middle to the tip, ribs are frequently 'fanned' or placed in orientations that are not aligned to the direction of flight.

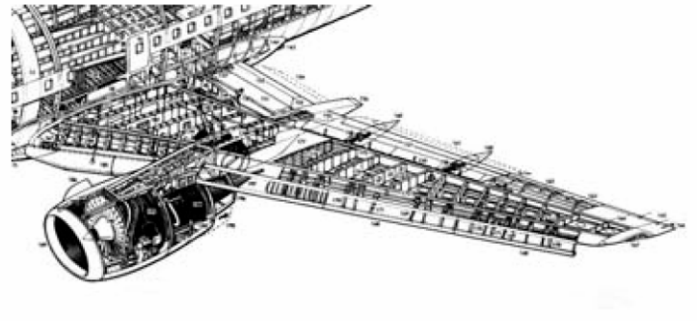


Figure 2-4: Typical transport wing structure [1]

Airliner wing configurations have historically been very stable (since the introduction of the swept wing), and innovation has come from the application of materials and the optimization of known designs.

The variations of rib alignment along the wing are shown in Figure 2-5, which illustrates a layout used by bombardier in an optimization study.

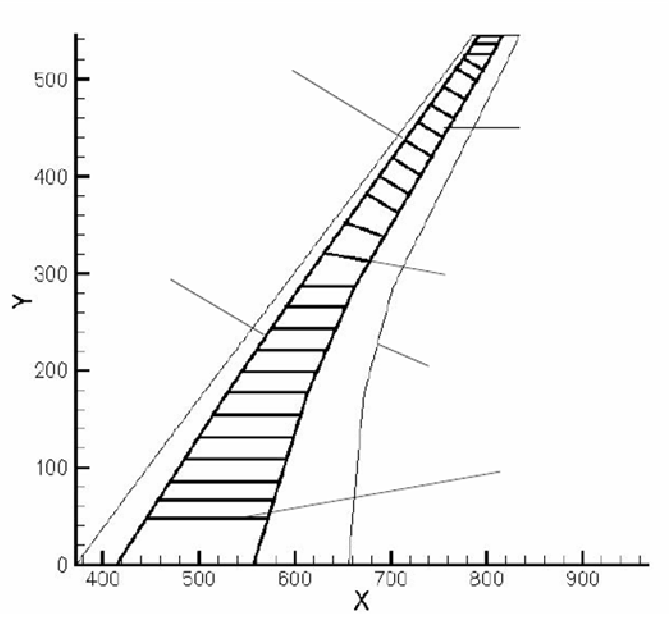


Figure 2-5: Rib and spar layout generated by Bombardier using TWSAP [3]

Lines extending from diagram should be disregarded

Ribs in the lower portion of the diagram are streamwise, while those near to the tip are 'normal-to rear'. In between there is a graduate transition where the ribs are 'fanned'. The variability in orientation of the ribs necessitates that WingSMITH allow for variation in rib orientation along the span. In particular, the definition of regions of similar orientation with a realistic interface between them is valuable for the design and testing of credible aircraft structural models.

2.3.2 Military Aircraft

High performance aircraft designed for military duty are subject to demanding wing loads [1] and tend to have slim airfoil sections for supersonic flight. This leads to structural layouts that favor a larger number of spars to provide strength across the entire chord of a wing. However, there is no discernable overall pattern which can be adopted in the WingSMITH platform.

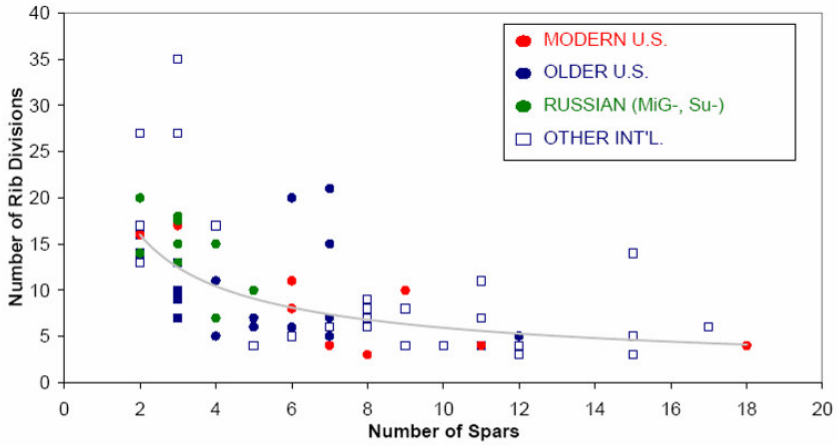


Figure 2-6: Number of ribs versus number of spars in 65 different combat jet aircraft [1]

Flexibility and creativity in the user's ability to specify a design, and generality and robustness of the builder algorithm should allow the synthesis of most spar-rib based wing models, regardless of their complexity.

2.3.3 Other features and trends

A feature seen in many aircraft is the use of wing bays to store landing gear. This necessitates the structural layout accommodating a large internal box, often interrupting the normal arrangement of spars and ribs. As seen in Figure 2-7 below the main spars may need to deviate from a straight line or constant chord fraction.

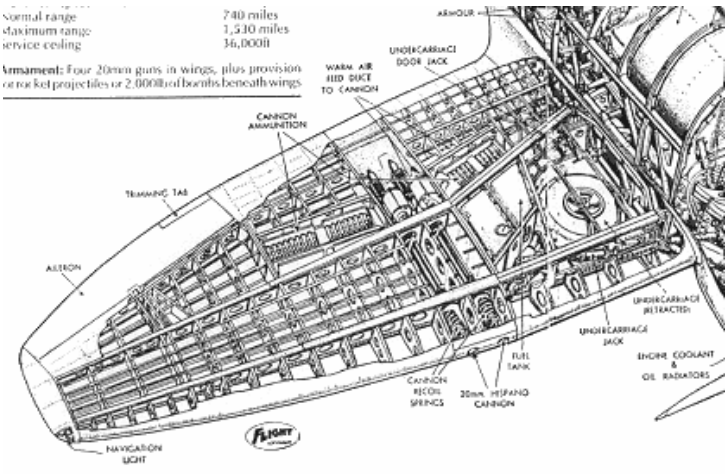


Figure 2-7: Hawker Tempest [11]

Another unconventional, though not uncommon, wing layout is the delta-wing. These typically contain a high number of spars, and may or may not use solid ribs as is visible in Figure 2-8below. Such structures are desirable to model in WingSMITH, since they are typically used in very demanding applications where optimization would be of great benefit.

Figure 2-8: Wing Structure of Space Shuttle Orbiter [13]

The wing layout produced for the Hypersonic Civilian Transport (HSCT) design at Georgia Tech provides a good baseline for the desired functionality of the WingSMITH platform, as this

structural layout includes most of the features mentioned in this chapter. With the ability to automate the modeling of such a design, WingSMITH will make a material contribution to the MDO research and design activities carried out by its users.

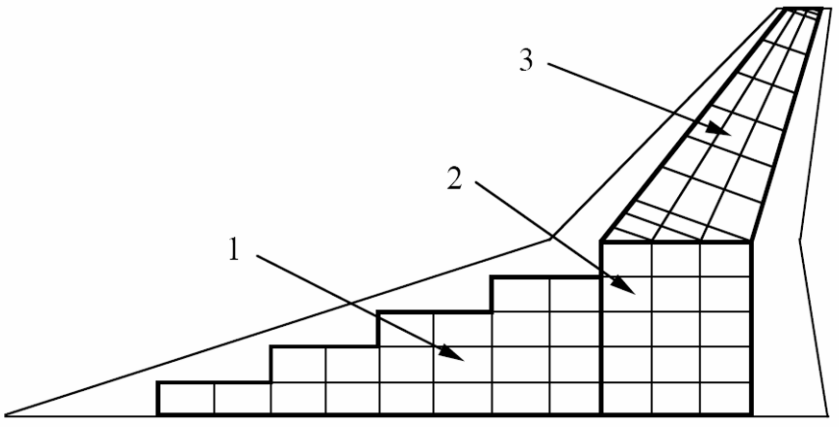


Figure 2-9: HSCT Wing Design - Structural Layout [10]

2.4 Summary

There is a wide variation of wing structural layouts between aircraft types, but within aircraft of a similar type there are often regular patterns and trends which are consistently applied to solve the same problems. The rib-spar box model discussed is a reasonable approximation of the wing structures found in aircraft ranging from the Hawker Tempest of WWII to the space shuttle orbiter and future hypersonic civilian transports.

2.4.1 WingSMITH requirements

Based on the review of known aircraft wing structures presented in this chapter, it was decided that the WingSMITH should have the following features:

- No upper limit on number of spars specified by the user
- Allow positioning of spars based on chord-fraction, with this parameter varying along the span
- Allow regular spacing of ribs based on a fixed distance offset
- Allow orientation of ribs in non-streamwise directions
- Place stringers across gaps between ribs, parallel to spars
- Produce a node-connectivity finite element model with 4 node and 2 node elements independent of the actual FEM implementation of the elements
- Maximize flexibility of user specification methods to allow unconventional designs
- Exploit regularity in wing structural layout by automatically creating patterned components without full user specification
- Incorporate default behavior which is informed by the trends observed in civil transport aircraft structures
- Allow spars to extend along arbitrary regions of the span to allow for ‘bays’ or other components embedded within a wing structure

3 Structural Perturbation

The purpose of this section is to review current methods for structural perturbation and how they apply to the problem of aero-structural optimization. This frames the way in which WingSMITH is integrated into an optimization framework.

Structural perturbation is the process of changing the node locations or otherwise allowing the configuration of a structural model to change from its initial state. During an optimization, be it structural or aero-structural, the tool must be able to vary the properties of the structure in order to explore the design space and find the optimum set of parameters and design variables driving the coupled aerodynamic and structural system [2]. For this reason, it was decided that WingSMITH needed to allow for the perturbation of its structural model to conform to changes in the wing surface geometry.

3.1 Spring analogy method

Sadri, et al. [5] discuss various mesh deformation strategies for modifying structured meshes around aircraft configurations. In particular, they identified work done by Robinson, Batina, and Yang [6] who developed a method based on a spring analogy, where "the displacement of the corner points of the blocks are determined by solving the equations of static equilibrium that result from a summation of forces". This requires a mesh moving algorithm which "continuously conforms to the instantaneous shape of the aeroelastically deforming wing" [6]

During the planning of this thesis, the application of this sort of method through use of FEAP was discussed. However, due to time constraints, the option was not fully explored or tested. It is recommended that this topic be investigated further, and that the feasibility and appropriateness of this method of structural mesh perturbation be established.

3.2 Perturbation of Design Variables

Jones et al described a simpler method of structural perturbation, based on the variation of design parameters and the re-generation of the mesh to "reflect geometric changes in the optimized system as defined by input surface(s)." [7]. They approach the problem by algebraically re-generating their meshes to account for surface changes during optimization.

In order to adopt a method whereby the re-running of WingSMITH using new surface geometry provides a reliable structural perturbation method, it is necessary for the mesh generation process to be repeatable.

Additionally, as illustrated by Reuther et al, it is important to be able to change and control other design variables which ensure the soundness of a structure's properties [15].

> Thickness constraints typical of our previous works are imposed in order to maintain the structural soundness of the final outcome of the design process. These constraints include spar depth constraints at 10% and 80% chord, a leading edge radius constraint ahead of the 2% location, a trailing edge included angle constraint behind the 95% chord location, and an additional thickness constraint to maintain maximum thickness and fuel volume at 40% chord.[15]

By perturbing design variables, and still constraining the mesh to a feasible domain, optimizers seek to find an optimal set of inputs which yields a minimum in whatever overall cost function has been set by the user. To allow this exploration of the structural design space, WingSMITH should provide access to the design variables which are used to specify and generate the structural mesh.

3.3 Summary

Based on the review of structural perturbation methods presented in this chapter, it was decided that WingSMITH needed to have the following features:

3.3.1 WingSMITH requirements

- Repeatable structure generation method to allow perturbation of mesh by changing the geometry in the wing surface input file.
- Design variables which can be modified during run time to perturb the mesh for structural optimization.

4 The WingSMITH Program

This chapter presents the WingSMITH program, the primary subject of this thesis. It is a tool with which aircraft wing structural meshes can be designed without involving the user in the technical details of finite element mesh specification. A simple and flexible specification language allows the user to specify the desired structure. WingSMITH parses this specification and a wing surface geometry file, and generates a structural mesh that fits inside the given wing envelope.

It is also possible to use WingSMITH as a structural Perturber by running it iteratively as part of an optimization code. By updating the input geometry definition, and varying design parameters, the optimization code can use the meshes generated by WingSMITH to define the finite element model used in a structural or multi disciplinary optimization problem.

4.1 Background

WingSMITH was written to support the development of the pyMDO framework being developed at the University of Toronto Institute for Aerospace Studies (UTIAS) by Martins et al [14]. A need for a tool to perform wing structural mesh design and perturbation was identified, and became the topic of this thesis. WingSMITH's intended application is to interface with the existing python-based optimization framework to enable perturbation of realistic structural meshes during aero-elastic optimization.

In the development of WingSMITH, an effort has been made to follow good software practices throughout the program. The tool was designed with the developer in mind, with an extendible structure that allows for the rapid addition of new or customized functionality. The design philosophy and its impact on WingSMITH are discussed later, in chapter 6.

4.2 WingSMITH Description

This section describes the WingSMITH program from the perspective of a user, emphasizing inputs and outputs of the tool without elaborating on the internal model or implementation. The reader interested in programming WingSMITH extensions, or gaining a deeper understanding of WingSMITH is referred to Appendix B where the architecture is discussed at length.

4.2.1 Components

An interactive user interface that walks the user through the steps of creating a wing structural mesh is included with WingSMITH. The user interface provides access to the Specifier tools which allow the user to set up customized wingspec.xml specifications. It then controls the various components involved in interpreting the user's specification, and outputs the structural mesh in the format requested by the user. The components of the WingSMITH program are tabulated below:

Inputs:	Wing Surface Geometry	.geo file containing wing sections that define the aerodynamic surface
	Structural Specification	.wingspec.xml file that defines the quantities, arrangements, and connections of structural parts
Input Tools:	Specifiers	Interactive tools for generating Structural Specifications
Controllers:	User Menu	Interactive console for human control of the WingSMITH program
	Perturber	Enables programmatic control of WingSMITH, for iteration and variation of design variables.
Outputs:	Structural Mesh	Data structure containing nodes and elements of mesh
	Visualization of Mesh	.dx files for rendering in OpenDX [17], a scientific data visualization tool.
	Finite Element Model	.inp files to transfer mesh into FEAP solver used in the pyMDO framework [14]

Table 3: Components of the WingSMITH program

4.2.2 Generalized Process

Figure 4-1 below illustrates the generalized process of using the WingSMITH tool.

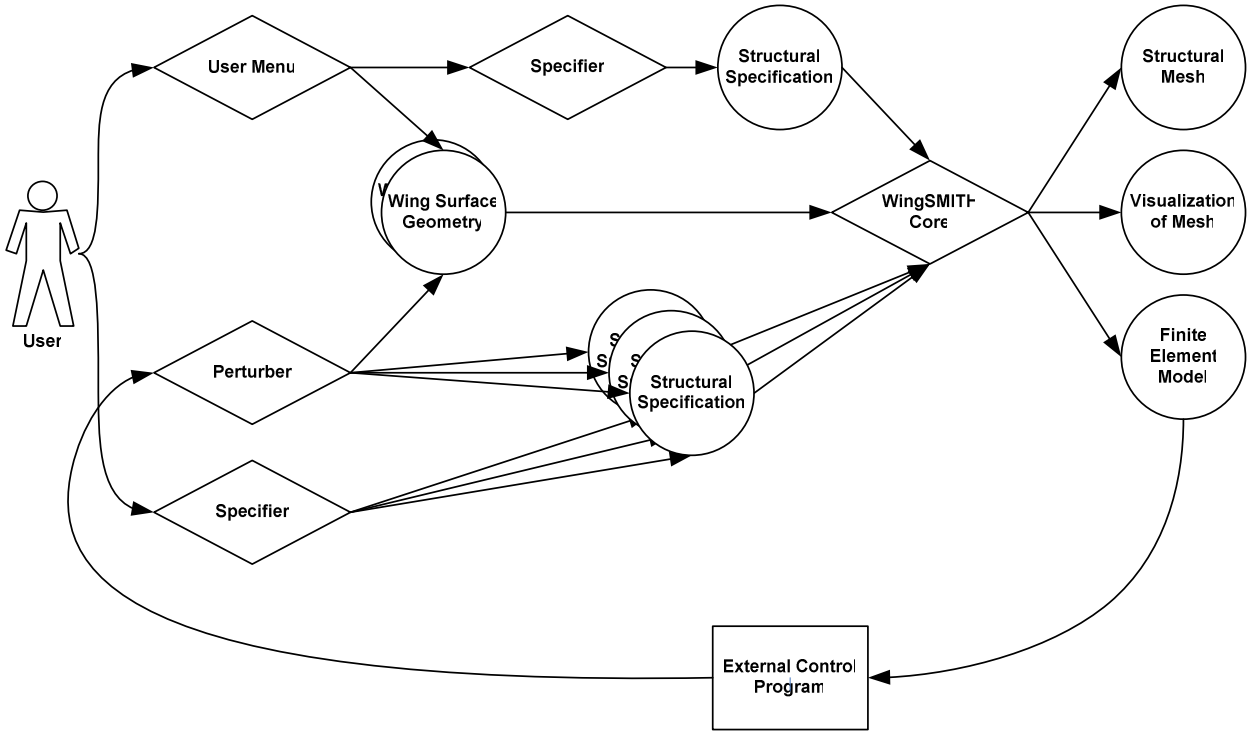


Figure 4-1: Generalized process diagram

The top branch of the process diagram illustrates how a user would control WingSMITH via the user menu. A wing surface geometry is selected, and a structural specification is prepared. These are sent to the WingSMITH core which creates one or more output files containing the generated mesh.

The bottom branch illustrates how the user would control WingSMITH via an external control program. First the user would need to create Structural Specifications using the Specifier tool. The Perturber would then be used to pass a particular wing geometry and structural specification to the core program. The outputs of the core program would be used by the external control program to re-initialize the surface geometry and other design variables which the Perturber can access. The Perturber then re-executes the WingSMITH core, and produces an updated set of outputs.

4.2.3 Program Architecture Outline

The relationships between the components of the WingSMITH program are best illustrated in graphical form, and are summarized in Figure 4-2 below. Blue rectangles represent software modules, and green rectangles represent files which contain input, specification, and output data. The modules are run, and pass information, from left to right as indicated by the green arrows. The process can be controlled via the UserMenu, which lends itself to the creation of single meshes tailored to a specific design (blue arrows). It can also be controlled via the Perturber module which is entirely code based. This module allows iteration of the run, which allows deformation of the mesh by altering the design variables between execution runs (red arrows and boxes).

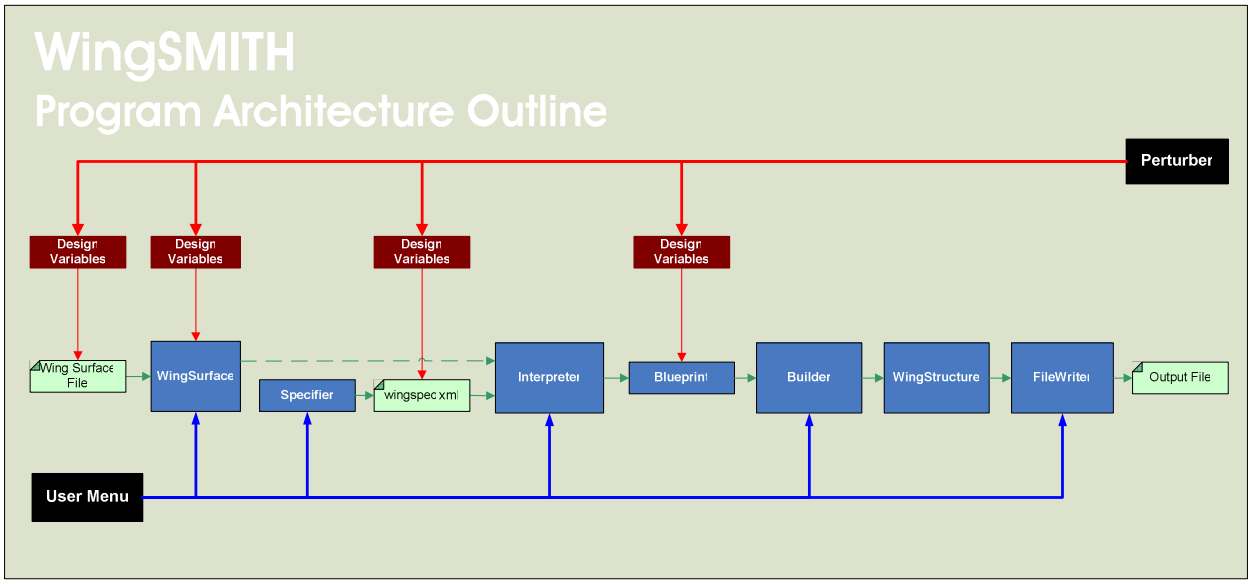


Figure 4-2: Program Architecture Outline

A more detailed version of this diagram, illustrating subclasses, data structures, and extendibility of the framework can be found in Appendix B 1.

4.3 Features and Capabilities

In chapters 2 and 3 above the features required for WingSMITH to be an effective tool for structural layout and perturbation were identified. This section revisits these requirements, explaining whether and to what degree they are satisfied.

4.3.1.1 Structural Layout

There is no upper limit on number of spars or ribs specified by the user. With an appropriate template, the user could even specify a wing with no spars at all. However, the current WingSMITH core requires that specifications always define at least two ribs, one of which can be suppressed. By default, spars are positioned based on chord-fraction, and with the more advanced Specification template the user can set the chordwise position of a spar at points along the span. Ribs are automatically spaced at regular intervals, but the current implementation does not yet support offset spacing or fanned alignment of ribs. However, these requirements are feasible within the WingSMITH architecture, and will be implemented at a future date.

Stringers are automatically placed, evenly spaced, in the gaps between spars. The WingSMITH core currently produces a node-connectivity finite element model with 4 node and 2 node elements which are used to create a specific finite element model for FEAP. Users are able to specify wing layouts using a variety of templates corresponding to different complexity levels and structural types. Additionally, WingSMITH can be extended by the user to define and interpret almost any structural configuration or pattern. The included Specification templates and internal routines are set with defaults based on the trends observed in civil transport aircraft structures. Spars can be defined along incomplete portions of the span to allow for 'bays' or compartments inside a wing structure.

4.3.1.2 Structural Perturbation

The WingSMITH program's core uses a repeatable mesh generation method which always produces the same mesh given the same inputs. This allows perturbation of a structural mesh to follow geometry deformation by supplying WingSMITH with the perturbed geometry in the wing surface input. Additionally, a large number of design variables, at several scales, can be modified via the Perturber controller. This makes WingSMITH suitable for coupled aero-structural optimization.

4.4 Comparison to existing tools

This section discusses how WingSMITH's features and capabilities compare to other known structure mesh generating tools.

Reuther et al. [15] developed a geometry generation system called AeroSurf, which was created for "the analysis and design of aircraft configurations including fuselage, wings, pylons, nacelles, and empennage". Their tool works by intersecting 'geometry components' with eachother and with the surface geometry. Perturbation to accommodate aerodynamic shape changes are performed by re-intersecting the original geometries with a new surface, a strategy analogous to

the one implemented in WingSMITH. While the AeroSurf tool can be used for non-wing structures and offers a greater depth of features than WingSMITH is currently capable of, AeroSurf does not offer the same ease of use. WingSMITH interprets simple user commands and templates in creating its structural mesh.

Bombardier's TWSAP program "is capable of generating streamwise, fanned and normal-to-rear spar ribs" [3]. TWSAP is a much larger tool than WingSMITH, and it is used to design the entire wing box, including the sizing of components and mass estimation. TWSAP's wing box layout capabilities exceed those of the current version of WingSMITH. As noted in section 4.3 above, fanned ribs are not yet supported. However, continuation of the development of WingSMITH will remedy this, and give it comparable wing box layout functionality.

4.5 Summary

The WingSMITH program is capable of representing all the common features found in aircraft wing structural layouts. Users can easily and quickly define layouts using the specification scheme, and WingSMITH functionality can be extended as needed by the extension of its core classes. WingSMITH also provides structural perturbation methods which make it applicable to a wide range of optimization problems.

WingSMITH is an appropriate tool for the design and optimization of aircraft structures, and provides the ability to quickly and easily design and test a concept. It is suitable for use in both academic and engineering settings to explore the merits and characteristics of a large array of wing structure configurations.

5 Applications of the WingSMITH program

The purpose of this chapter is to demonstrate to the reader how WingSMITH can be used both as a standalone mesh generator and as a Perturber as in an MDO problem. The two use cases presented below encapsulate the basic requirements which shaped the WingSMITH program, and serve to effectively demonstrate its capabilities.

5.1 Structural Design

This use case illustrates how WingSMITH can be used to produce wing box structures that correspond to known aircraft configurations. These structural meshes could be used by a tool that estimates wing structural mass, or they could be passed to a finite element analysis suite which would apply load cases and determine the response of the wing.

5.1.1 Synthesis of a B737 Wing Box

The wing box was generated using the Basic Specifier tool, and the following parameters were entered via the User Menu:

Rib quantity	50
Suppress Root Rib	Yes
Suppress Rib Caps	No
Front Spar	30% chord
Rear Spar	70% chord,
Suppress Spar Caps	No
Stringer Quantity	10

The three suppression properties allow the user to decide whether the corresponding finite elements are included when meshing takes place. The root rib is suppressed to more accurately represent the wing-fuselage connection. Similarly, ribcaps and sparcaps can be disabled. The front and rear spars are located according to the graph in Figure 2-1. The rib quantity is set to produce about 2' spacing assuming that the native units of the geometry file are 10 cm increments (i.e. 100 units → 10m).

The resultant wing structure is shown in Figure 5-1 below.

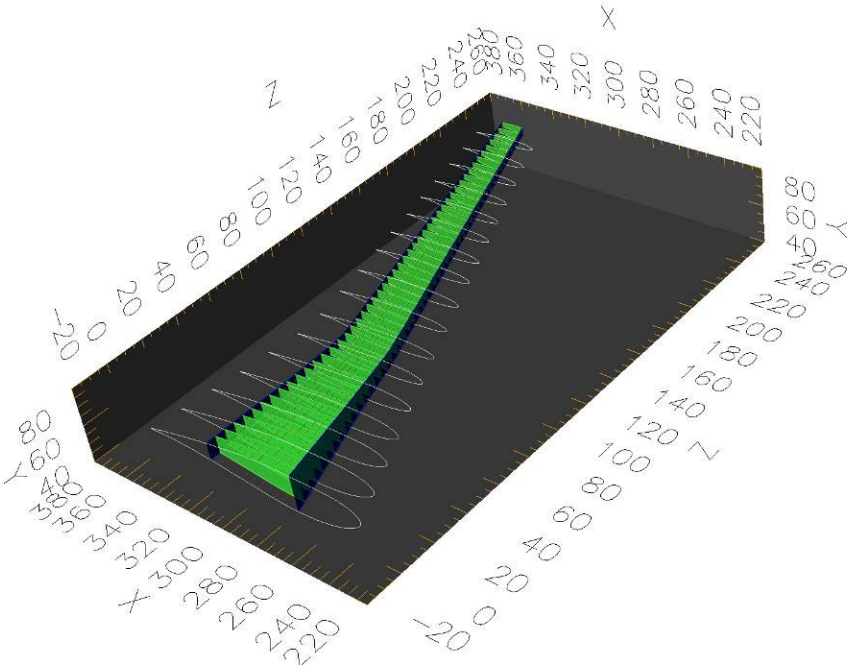


Figure 5-1: Wing box for B737 airliner. Generated using WingSMITH

A slightly different specification, featuring 3 spars, fewer ribs, and a tailing edge spar can look like the wing in Figure 5-2 below. Though it is an impractical structural configuration due to the small number of ribs, it shows how a user can easily produce a great variety of layouts for a given surface. This is useful in the analysis of preliminary designs, where it is important to evaluate as many ideas as possible.

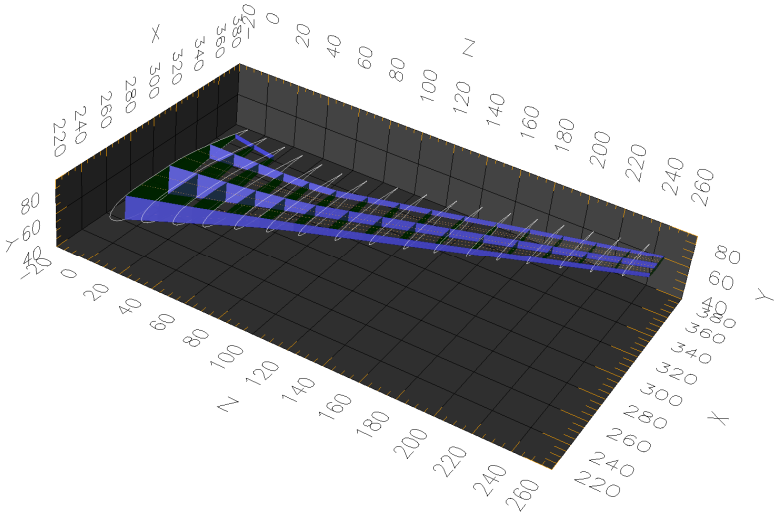


Figure 5-2: Alternative layout of wing box.

The ease with which a user can create detailed wing boxes demonstrates the power of the layout Specification approach. The WingSMITH program clearly fulfils the "rapid, 'hands-off' grid generation needs of [the] MDO cycle." [7].

5.2 Aero-Structural Optimization

This use case illustrates the application of WingSMITH to an aerostructural optimization problem, where the structural mesh must deform to match the changes in the aerodynamic surface. Sobieski et al describe the nature of the optimization challenge as follows:

> "One of the most common applications of multidisciplinary optimization techniques is in the field of simultaneous aerodynamic and structural optimization, in particular for the design of aircraft wings or complete aircraft configurations. The reason for this prominence is that the tradeoff between aerodynamic and structural efficiency has always been the major consideration in aircraft design: slender shapes have lower drag but are heavier than the stubby, more draggy shapes." [2]

5.2.1 Perturbation of an SSBJ wing structure

WingSMITH's Perturber module can be used to perturb design variables which correspond to transformations of the wing's aerodynamic surface. In particular, the sweep angle of a wing is an important factor in the drag of a wing at high speeds. Perturbation of the sweep angle is thus expected during aeroelastic optimization.

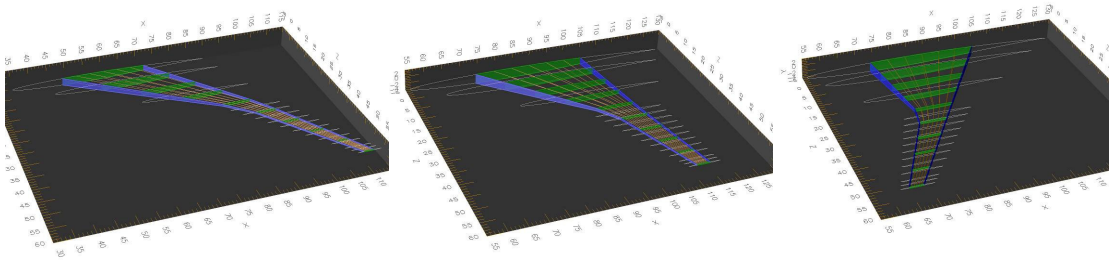


Figure 5-3: Structural mesh perturbation using the wing sweep design variable.

This series of structural configurations was obtained by iterating the Perturber cycle for various sweep angles. The wing box conforms to the geometry through a drastic change of shape, showing the robustness of the design variable perturbation approach.

5.3 Integration with pyMDO

Within the pyMDO framework, the mesh generated and perturbed by WingSMITH will be used as an input to the 'Structure' class which is already integrated into the framework. The mesh, combined with the Perturber design variables, will provide several of the necessary inputs to UTIAS's MDO platform (see Figure 5-4 below).

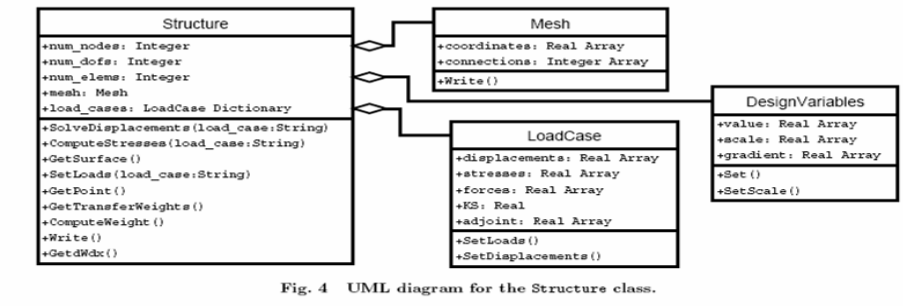


Fig. 4 UML diagram for the Structure class.

Figure 5-4: UML diagram for the pyMDO Structure class [14]

Once integration with the pyMDO framework is complete, WingSMITH will provide flexible and reliable structural design and aerostructural perturbation functionality.

6 Design Philosophy

The development of WingSMITH placed particular emphasis on the application of good software development practices. The primary goal was the development of an architecture which is both easy to use and easy to extend. The secondary goal was the development of feature complete extensions was the secondary goal. The importance of good software design and documentation has been articulated well by Abdo et al, who stated that "Although these are primarily computer science and information transfer and storage issues, they are equally important to the successful implementation of an MDO environment." [14].

6.1.1 Extensible framework

It is important to note that application of good software practice is not only a matter of information transfer and storage. Simpson et al. report on strategies used for design, and they advocate the design of 'open engineering systems' [4]:

> "Open engineering systems are systems of industrial products, services, and/or processes that are readily adaptable to changes in their environment…
>
> … Potential benefits of designing open engineering systems include increased quality, decreased time-to-market, improved customization and increased return on investment which are enhanced through the system's flexibility, i.e., its capability to be adapted to change. Generally speaking, the more a company can quickly adapt its products to changing demands and requirements, the more it can saturate its market and the more benefits it will enjoy. [4]

WingSMITH uses the open standards and formats where possible. In particular, the XML standard is used for the Specification files, as this standard is open and modular, ensuring that format can be adapted as needed.
While the WingSMITH program is intended for use in a research setting, researchers can derive similar benefits from having a tool that can be quickly adapted to new problems. For this reason it has been developed in such a way as to maximize long term extensibility through the addition of user written modules that extend the capabilities of several classes within the WingSMITH program. The modularity of the program architecture allows the adoption of open standards for other file formats without great difficulty.

6.1.2 Model-View Separation

In order to achieve the above stated goal, namely to develop an extendible and adaptable tool, WingSMITH implements the Model View Controller (MVC) pattern.

> "The application of Model-View-Controller has several benefits: Multiple views of the same model. MVC strictly separates the model from the end user-interface components. Multiple views can therefore be implemented and used with a single model… The conceptual separation of MVC allows you to exchange the view and controller objects of a model."
>
> Buschmann et al. [16]

In the case of WingSMITH, the 'view' is implemented in the abstraction of a wing structure as a set of specifications, and the output of a structural mesh either for viewing in an application such as FEAP [8] or OpenDX [17]. The model which is encapsulated within WingSMITH is the particular set of Interpreter, Blueprint, and Builder (described below) which take the user's specifications as inputs, and turn them into a finite element mesh. The model could be exchanged for a completely different one, which would read the same specifications, and produce the same mesh output, but might use a completely different algorithm in the creation of the mesh. Similarly, as will be discussed in chapter 5, the MVC interface allows the same model to be controlled either by a human user or by an optimizer. The MVC pattern would allow for the creation of a Graphical User Interface (GUI) based interface for WingSMITH which would be impossible if the model and view were closely coupled.

6.1.3 Programming Language

WingSMITH is written in python, which is the language of choice within the MDO research group at UTIAS. Python is used as the main programming language due to its concise syntax, object oriented nature, and portability [14], and these characteristics have aided the rapid development of the WingSMITH platform.

7 Conclusions

The goal at the outset of this thesis was to develop a tool which enables a designer to rapidly generate a realistic aircraft wing structural model as well as incorporate that model in an aerostructural optimization process. The achievement of this goal required a tool that could accurately represent the large variety of wing structure layouts which are in use, as well as allow perturbation of the structural mesh in order to follow perturbations of the aerodynamic surface of the wing.

WingSMITH successfully realizes this goal, and is a powerful, flexible, and most importantly, extensible and maintainable tool. It allows a developer to rapidly specify a wing structure without subjecting him or her to the details of the meshing operation. It is able to represent a wide range of structural configurations, which will grow wider as further specification templates and meshing tools are defined. WingSMITH also provides the ability to perturb design variables on scales ranging from the macroscopic sweep and taper, down to the coordinates of individual nodes in the mesh.

In the near future, the WingSMITH program will be integrated into the pyMDO framework, and will provide an important tool within the multidisciplinary optimization framework being developed by the MDO group at UTIAS.

In the longer term, as users and developers extend the capabilities of the WingSMITH program, it will become a formidable tool for the preliminary design and analysis of aircraft wing structures. Its ease of use and flexibility will enhance the productivity of researchers and wing designers, and hopefully lead to the investigation of innovative designs that would otherwise have been impractical to explore.

8 References

[1] Sensmeier, M., Samareh, J. A Study of Vehicle Structural Layouts in Post-WWII Aircraft 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference 19 - 22 April 2004, Palm Springs, California. AIAA 2004-1624

[2] Sobieszczanski-Sobieski , J., Haftka, R. Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments NASA-AIAA-96-0711

[3] Abdo, M., Piperni, P., Kafyeke, F. Optimization of a Business Jet Canadian Aeronautics and Space Institute 52 Annual General Meeting and Conference. Toronto, April 26 and 27, 2005.

[4] Simpson, T., Rosen, D., Allen, J. Mistree, Farrokh Metrics for assessing design freedom and information certainty in the early stages of design The 1996 ASME Design Engineering Technical Conferences. 96-DETC/DTM-1521

[5] Sadri, R., Leblond, D., Piperni, P. Coupling of a Wing Inverse Design Code to an Euler Navier-Stokes Flow Solver using Mesh Movement Capabilities Proceedings of the 50th AGM & Conference, CASI, 2003

[6] Robinson, B., Batina, J., Yang, H. Aeroelastic analysis of wings using the Euler equations with a deforming mesh AIAA PAPER 90-1032

[7] William T. Jones; Jamshid Samareh-Abolhassani A Grid Generation System for Multi-disciplinary Design Optimization NASA CP-3291; Proceedings of the Workshop on Surface Modeling, Grid Generation, and Related Issues in CFD Solutions , May 9-11, 1995

[8] Robert L. Taylor FEAP - - A Finite Element Analysis Program Version 7.5 User Manual University of California at Berkeley, February 2005

[9] P. Piperni, M. Abdo, F. Kafyeke The Building Blocks of A Multi-Disciplinary Wing Design Method Canadian Aeronautics and Space Institute 52 Annual General Meeting and Conference. Toronto, April 26 and 27, 2005.

[10] Rohl, P., Mavris, D., Schrage, D. HSCT Wing Design Trough Multilevel Decomposition School of Aerospace Engineering, Georgia Institute of Technology. Atlanta, GA 30332-0150

[11] Christer Landberg The Hawker Tempest Page URL: http://www.hawkertempest.se/cutaway.gif Last updated: March 22 2005.

[12] Hubert Peitzmeier 916-Starfighter.de URL: http://www.916-starfighter.de/Gallery/gallery_sch03.htm

[13] NASA Cutaway of left wing of Space Shuttle Orbiter URL: http://www.nasa.gov/images/content/2663main_COL_orbiter_wing_lo1.jpg

[14] Alonso, Juan J., LeGresley, Patrick., van der Weide, Edwin., Martins, Joaquim R. R. A., Reuther, James J. pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimization AIAA 2004–4480

[15] Reuther, James J., Alonso, Juan J., Martins, Joaquim R. R. A. A Coupled Aero-Structural Optimization Method For Complete Aircraft Configurations MCAT Institute 1999

[16] Buschmann, Frank et al. Pattern-Oriented Software Architecture, Volume 1: A System of Patterns John Wiley & Sons; 1 edition (August 8, 1996) pp.141

[17] OpenDX, the Open Source Software Project Based on IBM's Visualization Data Explorer URL: http://www.opendx.org/index2.php

[18] Kruchten, Philippe. "The 4+1 View Model of Architecture." IEEE Software 12.6 (1995) 45-50.

[19] Anderson, John D. Fundamentals of Aerodynamics Third Edition McGraw-Hill, ©2001

Appendix A WingSMITH User Manual

Appendix A 1 Installing WingSMITH

Appendix A 1.1 Required Files

WingSMITH was written and tested using python 2.3, and should be run on this version or higher in order to guarantee successful execution. The program consists of a number of files which must be located in the WingSMITH base folder for it to function:

```
BlueprintModule.py
BuilderModule.py
FileReaderModule.py
FileWriterModule.py
FiniteElementModule.py
HelperModule.py
InterpreterModule.py
LinkedListModule.py
NodeModule.py
PartModule.py
PerturberModule.py
RibDescriptionModule.py
SpecifierModule.py
StructureModule.py
UserInterfaceModule.py
WingSMITH.py
WingSurfaceComponentModule.py
WingSurfaceModule.py
```

Appendix A 1.2 Required Libraries

The following site packages are used by WingSMITH, and must be installed on the user's system. Version numbers used during development are noted, and users are advised to use site packages with equal or higher version numbers.

Scientific Python version 2.4.6 - can be found at http://www.scipy.org/
Numerical Python version 23.8 - can be found at http://numeric.scipy.org/

Additionally, WingSMITH uses the following built in libraries.

math
xml
sys
string
time

Appendix A 2 Running WingSMITH

To become familiar with the program, the user is advised to begin by directly executing WingSMITH.py. This calls up a console based interface that walks the user through the steps of creating a wing structural mesh. The UserMenu provides access to the Specifier tools which allow the user to set up customized wingspec.xml specifications. Completed meshes can be

output to a .dx file or to an .inp file. The user should initially use the .dx format, and then use OpenDX to visualize the results of his/her wing surface and specification choices. Once the user is happy with the mesh, it can be written to a .inp file for use with FEAP.

Appendix A 2.1 Required Inputs

WingSMITH requires 2 inputs; a definition of the wing's surface geometry, and a set of specifications which describe the desired structural layout.

Appendix A 2.1.1 Wing Surface

The wing surface must be given in standard '.geo' format, which specifies a set of wing sections located in a 3-D Cartesian space. This surface is parsed and then used as a helper for the meshing operation. Other formats must be added by extending the WingSurface class.

Appendix A 2.1.2 Wing Specifications

The user can create WingSpecification files by executing SpecificationModule.py
Additionally, Specifications can be created, or loaded from file, while running WingSMITH via the UserMenu. When running WingSMITH via the perturber, the filename must be specified.
The specification templates correspond to different configurations which can be interpreted by the WingSMITH program. The user is encouraged to review the default specifications in the **/defaults/** directory of WingSMITH to gain an understanding of their composition.
Component locations can be specified in terms of Fractional or actual coordinates. The user is also able to specify the quantities, and connections of parts (within the restrictions of a particular layout Specification class). The user is able to suppress features such as the root rib of a wing, which are required by the Builder algorithm but are not necessarily features of the desired wing structure.

Appendix A 3 Design variables

The following classes of design variables are exposed by the Perturber class found in PerturberModule.py. During an optimization process, these may be adjusted and the structural mesh will then be re-generated for the next solution step.

Overall shape: Sweep, Taper, Camber
Parts and Features: All parameters available in the Specification class which defines the mesh.
Node Coordinates: All coordinates used in WingSMITH's internal representation of the mesh

Appendix A 4 Connecting to FEAP

In order to facilitate FEAP perturbation of material properties, the INP files that are created by WingSMITH contain named design variables material properties which correspond to structural families of parts.

Spars, ribs, stringers, spar caps, and recaps all have independent and parameterized elastic properties.

Appendix B WingSMITH Developer Guide

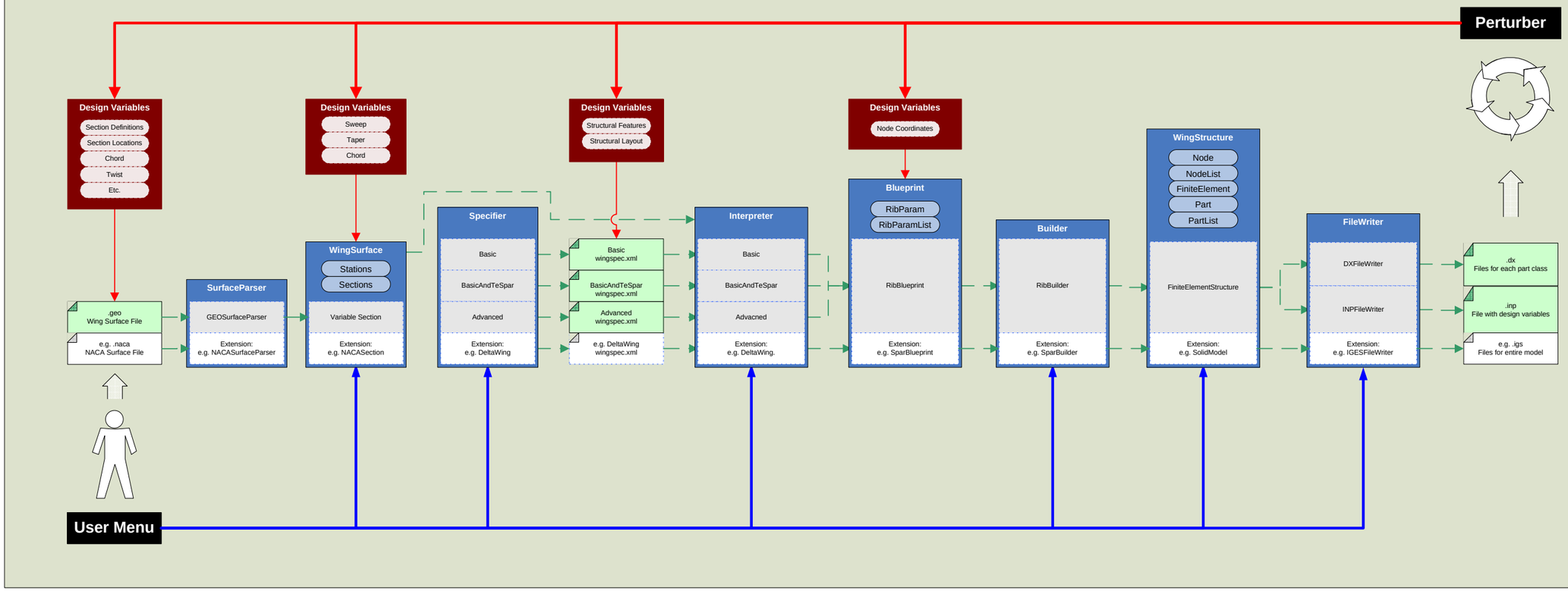
This Appendix presents the architecture, interfaces, and uses of the WingSMITH program, and illustrates how the requirements identified in the previous two chapters have been met. The programming and design philosophy are discussed, as are the architecture and algorithms which form the underlying components of WingSMITH. This chapter concludes with a summary of the capabilities and limitations which are inherent to the WingSMITH architecture and implementation.

Appendix B 1 Detailed Program Architecture

Figure B-1 on the following page illustrates the class hierarchy and program control of the WingSMITH tool.

WingSMITH

Detailed Program Architecture



Appendix B 2 Program Components

WingSMITH is made up of a number of components which comprise the various parts of the mesh building procedure. This section describes the components and how they relate to each other. Table 4 below contains a list of the main components of WingSMITH, and describes their functional role in the process of generating a structural mesh.

Component	**Description**
WingSurface	This class reads an input file such as .geo and constructs a representation of the wing surface. It allows another module to determine the coordinates of the top and bottom surfaces of the wing skin at any point within the planform.
Specifier	This class prompts a user for the values and options required to complete a wingspec.xml file, and then saves the user's choices to file. It is the primary means by which a user defines the wing structural layout.
*.wingspec.xml	These template files store the values and specifications which the user has entered for a particular class of wing. They are read by the Interpreter class, and depending on the particular template, can describe a very simple or very complex layout.
Interpreter	This class reads a wingspec.xml file, and determines where on the wing planform the user wants components placed. It then uses the WingSurface component to determine the coordinates of the top and bottom surfaces of the wing skin at each point of the planform. Lastly, it assembles the part layout and coordinates data into a Blueprint component.
Blueprint	This class contains the layout information and coordinates over which the builder needs to iterate over in order to construct the coordinate mesh. It is exposed as a distinct component to allow perturbation of mesh coordinates prior to processing by the Builder.
Builder	This class takes the information contained in the blueprints and creates a finite element mesh of the specified layout. A WingStructure class (below) is populated with finite elements and nodes organized to represent the wing's part structure.
WingStructure	This class contains the finite elements and nodes which define the mesh, which is the main output of WingSMITH. This mesh can be written to file as required.
UserMenu	This class allows a user to control the other components and to create a structural mesh by selecting the surface file, and creating or loading the wingspec.xml file. This class then outputs the mesh for visualization in DX or for inputting into FEAP.
Perturber	This class controls allows another program to control the components of WingSMITH. Allows repeated generation of a mesh and setting of design variables. Values in a set of accessible design variables.

Table 4 - WingSMITH components and functional descriptions.

Detailed documentation of all classes can be found in the **/docs/** folder of the WingSMITH program. Additionally, the source code is documented extensively, and can be made to run in a verbose mode to display the state of key internal variables during runtime.

Appendix B 3 Structural Layout Meshing Algorithm

At this time, WingSMITH contains only one Blueprint and Builder set, called RibBlueprint and RibBuilder. This section describes the way in which these work to construct the model, and how their algorithm is general enough to allow for considerable extension of the Specifier and Interpreter classes before needing an upgrade.

Appendix B 3.1 Rib based meshing

The data structure contained within the Blueprint class describes the coordinates of the rib's nodes, and also identifies the parts which connect to either side of the rib by type and index (e.g. spar,0). The RibBuilder takes this data and, one rib at a time, ties together all the parts that traverse the span of the wing. Below is pseudo-code which describes the algorithm as implemented in the RibBuilder module:

```
read rib definitions in blueprint
iterate over coordinates of rib,
    create nodes
    create elements
    place elements in rib part containers
iterate over rib part containers
    read connection definitions in blueprint
    iterate over elements that extend towards the wingtip
          Create elements
          Connect one end of elements nodes
          Place elements on unfinished stack
    iterate over elements that approach from the root
          Get elements from unfinished stack
          Connect free end of elements to nodes
          Put elements into part containers
store completed part containers in structure object.
```

During the development of WingSMITH, after developing this algorithm, it was learned that a rib-centric approach was taken by Abdo et al [3]:

> "The design process begins by distributing the ribs within the wing box and designing the sections at the ribs' locations one after the other along the wing span."[3]

The strength of the rib-centered approach as implemented in WingSMITH is that the ribs, spars, and stringers can be placed arbitrarily by defining coordinates appropriately during the interpretation phase. Spanwise components like spars and stringers do not need to follow straight lines along the entire span, and they can snake from coordinate to coordinate as they traverse various ribs.

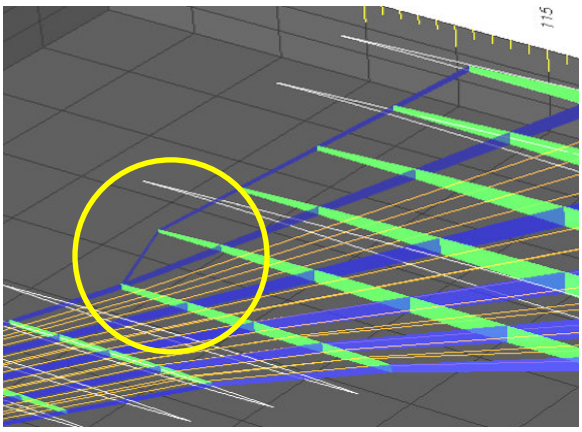


Figure B-2: Spar with an arbitrary path, generated in WingSMITH

Figure B-2 shows the capability to have spanwise components which intersect other components. While the geometry shown above is not realistic, it illustrates the flexibility of the RibBulider in meshing unstructured shapes.

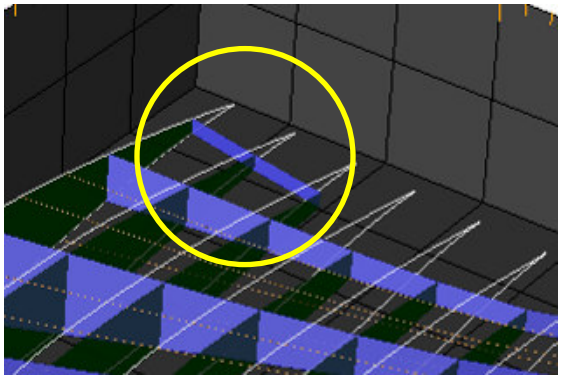


Figure B-3: Spar bridging arbitrary set of ribs, generated in WingSMITH

The ability to create truncated spars is important in replicating features such as landing gear bays, where a spar may not reach all the way to the root of the wing. The RibBlueprint and RibBuilder classes are able to mesh such a configuration. While the current Interpreter feature set is limited, the Builder will be able to mesh more finely detailed layouts when given the appropriate Blueprint.

Appendix B 4 Structure Specification Files

Appendix B 4.1 Basic Specification

This specification creates a simple wing box, inserted into the surface geometry.
Important features to note are the type variable in the <wing> element. This identifies the wignspec.xml file as being of 'basic' type, and is used to match wingspec.xml files to the correct Interpreter class.

```
<?xml version="1.0" ?>
<wing type="basic">
    <ribs rootrib="true" ribcaps="true">
          <rib quantity="10"/>
    </ribs>
    <spars front="0.25" rear="0.675" unit="fract" sparcaps="true">
          <spar quantity ="2" />
    </spars>
    <stringers>
          <stringer quantity="5"/>
    </stringers>
</wing>
```

Appendix B 4.2 Basic and Trailing Edge Spar Specification

This specification creates a simple wing box, inserted into the surface geometry, and adds a trailing edge spar as is sometimes found on airliner wings. Note that the <tespar> element within the <spars> element is the only difference between this Specification and the 'basic' one. This demonstrates the extensibility of the Specification format.

```
<?xml version="1.0" ?>
<wing type="basicandtespar">
    <ribs rootrib="true" ribcaps="true">
          <rib quantity="10"/>
    </ribs>
    <spars front="0.25" rear="0.675" unit="fract" sparcaps="true">
          <spar quantity ="2" />
          <tespar>
                <span rootrib="0" tiprib="4" rootchord="0.9"
                      tipchord="0.8"  unit="fract" />
          </tespar>
    </spars>
    <stringers>
          <stringer quantity="5"/>
    </stringers>
</wing>
```

Appendix B 4.3 Advanced Specification

This section describes the Advanced Specification format. This format is intended to give the user full control over the structural layout, allowing the creation of complex geometries.

The reader should review the sample Specification below, as it contains comments explaining the structure and syntax of the file.

As of the submission date of this thesis, this specification was not fully implemented in WingSMITH.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<wing>
    <ribs>
          <!-- Rib ID#s are assigned in order of specification (root to \
               tip)-->

          <!--First rib must be at span fraction 0
                Last rib must be at span fraction 1 -->
```

```
      <!-- Can set quantity just by repeating empty rib elements.. \
            default parameters will be applied -->
      <rib />
      <rib />
      <rib />
      <rib />
      <rib />

      <!-- Can set quantity by explicitly setting value.
            Will be interpreted as the appropriate number
            of ribs with other parameters as indicated
            quantity attribute excludes span and unit
            attributes -->
      <rib quantity="10" />

      <!-- Can set front and rear edge chord fractions for \
            a rib or set of ribs.\
            Only supported unit is fract.. will eventually \
            allow absolute distance.\
            Unspecified units not allowed\
            Unspecified ribs should be interpolated between \
            These -->
      <rib quantity="1" front="0.1" rear="0.5" unit="fract" />
      <rib quantity="3" front="0.2" rear="0.8" unit="fract" />-->

      <!-- Can set span fractions for individual ribs. \
            Only supported unit is fract.. will eventually allow \
            absolute distance.\
            Unspecified ribs should be interpolated between these.-->
      <rib span="0" unit="fract" />
      <rib quantity="1" span="0.5" unit="fract" />

      <!-- THIS WOULD BE INVALID:
      <rib quantity="5" span="1" unit="fract" />
       -->

      <!-- Can set the fan angle by explicitly setting value (in
            degrees). Fan angle should be left-handed rotation in
            plane of wing coordinate system.
            Unspecified ribs should not be interpolated between
            these.-->
      <rib fan="20"  />

      <!-- Can combine various parameters as needed to
            set up a wing with regions of differing ribs-->
      <rib quantity="5" />
      <rib span="0.3" unit="fract" />
      <rib quantity="10"  fan="15" />


</ribs>
<spars>
      <!-- can just specify quantity-->
      <spar quantity="5" />

      <!-- can specify quantity and define the span over which the \
            spars exist. default rootrib is "0"-->
      <spar quantity="3">
```

```
            <span tiprib="4" />
        </spar>


        <!-- can change the chordwise location of hte spar at \
             various points along the span of the wing by \
             defining a series of 'spans'-->
        <spar>
             <span tiprib="5" rootchord="0.8" unit="fract"/>
             <span rootchord="0.6" unit="fract"/> <!-- extend to tip--
             >
        </spar>

        <!-- can define a spar segment that is located out on the wing
             but does not reach the tip or the rooot-->
        <spar>
             <span rootrib="2" tiprib="4" rootchord="0.1"
                  tipchord="0.2" unit="fract" />
        </spar>

        <!-- fully specified defaults (tiprib=end)\
             multiple spars will be spaced out at appropriate chords\
             -->
        <spar>
             <span rootrib="0" tiprib="end" rootchord="0.3"
                  tipchord="0.3" unit="fract" />
        </spar>

        <lespar>
             <span rootchord="0.2" tipchord="0.3" unit="fract" />
        </lespar>

        <!-- fully specified defaults (tipchord=joined)\
             at tip rib, chord is equal to that of following spar-->
        <lespar>
             <span rootrib="0" tiprib="4" rootchord="0.1"
                  tipchord="joined"  unit="fract" />
        </lespar>


        <tespar>
             <span rootrib="0" tiprib="4" rootchord="0.9"
                  tipchord="joined"  unit="fract" />
        </tespar>
        <!-- fully specified defaults (tipchord=joined)\
             at tip rib, chord is equal to that of previous spar-->
        <tespar>
             <span rootrib="0" tiprib="4" rootchord="0.9"
                  tipchord="joined"  unit="fract" />
        </tespar>

    </spars>
    <stringers>
         <stringer quantity="10" />
    </stringers>

</wing>
```