

DESIGN OPTIMIZATION USING CAD PARAMETERIZATION THROUGH CAPRI

By

William Emerson Brock V

Approved:

Dr. Steve Karman Jr.
Professor of Computational Engineering
(Director of Thesis & Committee Member)

Dr. Abdollah Arabshahi
Research Professor of
Computational Engineering
(Committee Member)

Dr. Timothy W. Swafford
Professor and Head, Graduate School
of Computational Engineering
(Committee Member)

William Sutton
Dean of College of Engineering and Computer
Science

Jerald Ainsworth
Dean of the Graduate School

DESIGN OPTIMIZATION USING CAD PARAMETERIZATION THROUGH CAPRI

By

William Emerson Brock V

A Thesis

Submitted to the faculty of
The University of Tennessee, Chattanooga
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computational Engineering

The University of Tennessee, Chattanooga
Chattanooga, Tennessee

August 2011

Copyright © 2011
By William Emerson Brock V
All Rights Reserved

ABSTRACT

Design optimization is one of the many areas of Computational Engineering that is directly applicable to almost any engineering endeavor. The design cycle is a multi-step process, beginning with a Computer Aided Design (CAD) model and ending with an optimized version of that model relative to some objective function. This is accomplished by parameterizing the model, through one of many different algorithms, and finding the ideal value of those parameters with respect to the specified objective. The purpose of this research is to explore the use of a Computational Analysis PRogramming Interface (CAPRI [1]) for utilizing CAD parameters in design optimization along with exploring the steps taken to integrate this technology into a typical deterministic design optimization cycle. CAPRI is an interface developed by Robert Haimes of MIT which allows communication with CAD software during the various stages of computational design.

A framework was developed for implementing CAPRI within the SimCenter's current GEOMETRY libraries with the aim of improving geometry representation and accuracy. The framework also supports functionality to interface with a design optimization cycle by providing sensitivity derivatives and surface mesh coordinates throughout successive iterations. A sample wing was generated using SolidWorks®[2] as the CAD tool and the root and tip sections of a NACA 2412 airfoil. Design optimization was then performed upon this model with tip rotation and wing length as adjustable parameters.

DEDICATION

This thesis is dedicated to my brothers, Stephen and Hutch Brock.

ACKNOWLEDGEMENTS

The author expresses his sincere gratitude to the following people for pushing and pursuing him throughout his academic career. First, for Bill and Laura Brock, who provided a nourishing place of employment during his undergraduate career which later guided him towards the fields of Computer Science and Engineering. Second, for Dr. Tim Swafford and Dr. Abdollah Arabshahi and their pursuit of the author as a Candidate for Graduate School at the SimCenter, without whom this thesis would have not been possible. And Last, for Dr. Steve Karman, Dr. Chad Burdyshaw, and Robert Haines for their clear direction, approachability and assistance throughout the course of this research.

TABLE OF CONTENTS

ABSTRACT	iv
DEDICATION	v
ACKNOWLEDGEMENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	x
CHAPTER	
1 INTRODUCTION	1
Importance	1
Background	1
Statement of the Problem	1
Objectives of the Study	2
Scope and Limitations of the Study	2
Plan of Presentation	2
2 GEOMETRY PARAMETERIZATION	4
Purpose and Application	4
Discrete Parameterization	4
CAD Parameterization	5
3 COMPUTATIONAL DESIGN CYCLE	8
Parameterization	8
Optimization	9
Function Evaluation	10
Sensitivity Analysis	11
Shape and Mesh Movement	11
Final Model	11

4	CAPRI IMPLEMENTATION	13
	Client/Server Specifications	13
	Surface Mesh Synchronization	14
	Quilt Construction	17
	Snap to Surface	19
	Quilt and Closest Point Validation	20
	Generating Sensitivity Derivatives	20
	Modifying the CAD Model	23
	CAPRI Framework	30
	Geometry Software Library Additions	30
5	APPLICATION AND RESULTS	32
	Initial CAD Model	32
	Building the CAD Model	32
	Parameterization	32
	Surface Mesh Creation	35
	Design Variables	35
	Issues in Implementation	37
	Design Results	41
	Final Design	43
6	CONCLUSION	45
	Future Work	46
	REFERENCES	48
	VITA	48

LIST OF FIGURES

3.1	Flow Chart of Design Cycle Using CAPRI	12
4.1	CAD Face vs. Mesh Face	16
4.2	CAD and Surface Faces used for Quilt validation: side view	21
4.3	CAD Model and Surface Mesh used for Quilt validation: perspective view	22
4.4	CAD Parameterization: Ellipse Wing	27
4.5	Surface Mesh Modification: Ellipse Wing	28
4.6	Surface Mesh Modification: Tip of Ellipse Wing	29
4.7	Flow Chart of CAPRI Implementation	31
5.1	NACA 2412 Cross Section [3]	33
5.2	Model Used for Design	34
5.3	NACA 2412 CAD Model Parameterization: Spline Points	36
5.4	NACA 2412 CAD Model Parameterization: Tip Rotation 10°	37
5.5	Surface Mesh Resolution with Shape Deformation: Spline Point Movement	39
5.6	Surface Mesh Resolution with Shape Deformation: Tip Rotation	40
5.7	Model Movement over the Design Cycles	42
5.8	Initial and Final Iterations	42
5.9	Surface Gradients	43
5.10	CAD Representation of Wing after four design cycles	44

LIST OF TABLES

4.1	Parameter Modifications for an Elliptical Wing	25
4.2	Changes in the CAD Face (u, v) Ranges after Parameter Modification	26
4.3	Changes in the CAD Edge t Ranges after Parameter Modification	26
5.1	Design Cycles and the associated changes in the Geometry Parameters . . .	41

LIST OF SYMBOLS

t , Parametric Coordinate for CAPRI Edges

uv , Parametric Coordinate for CAPRI Faces

xyz , Cartesian Coordinate

ρ , Fluid Density

C_L , Objective

L , Integral force of components in the direction of Lift

V , FreeStream Velocity

A Surface Area

\tilde{H} , Hessian Matrix

CHAPTER 1

INTRODUCTION

Importance

The author will explain the process for applying the Computational Analysis PRogramming Interface (CAPRI[1]) for use in Design Optimization at the University of Tennessee at Chattanooga's SimCenter: National Center for Computational Engineering, hereafter referred to as SimCenter.

Background

Computational Design Optimization is a field of research that has been an area of interest at the SimCenter for a number of years. The design process involves many modular operations and is truly a multi-disciplinary approach. From the building of a model using a CAD software package to the finished result, an optimized design, there are many steps and opportunities for individual areas of research.

The CAPRI Computer Aided Engineering (CAE) Gateway is a "CAD-Vendor neutral gateway for engineering analysis and design". CAPRI was developed by Robert Haines of the Massachusetts Institute of Technology[1] and serves the purpose of providing custom communications from a computational software suite to the preferred CAD system. This allows a software developer access to the CAD system's geometry definitions and functionalities providing the ability of querying the CAD system whenever needed.

Statement of the Problem

In the current implementation of the design process, there is a large disconnect between the CAD model and the final iteration of the design cycle. In short, when a mesh is built

upon a model, that mesh is a discretization of that model. This discretization no longer carries any of the information associated with the construction of that model, other than the specified x, y, and z surface coordinates. This creates the need for advanced parameterization methods in order to extract design variables from the computational mesh. The parameters are derived out of the surface mesh from one of many different algorithms, which often times result in parameters that are difficult to control. When performing computational design on the discretized model, along with using these derived parameters, an optimized model can be achieved. However, a large drawback of this process is that obtaining a model out of this process to use for future analysis is a non-trivial task.

Objectives of the Study

The objective of this study is to explore the process of implementing the CAPRI API into a typical deterministic design cycle. CAPRI will be used to obtain the design variables (parameters) and to output a CAD model at each iteration of the design cycle, which can easily be used for future analysis. The culmination of the research presented herein is a framework, that can be used as a basis for future work. CAPRI functionality will be incorporated into the current geometry utilities that exist at the SimCenter, extending their capabilities to include CAD information and analysis in a variety of different scenarios.

Scope and Limitations of the Study

The scope of this project will be limited to the optimization of a wing based on a NACA 2412 airfoil cross section with matching root and tip segments. The variables within the CAD model which will be modified to achieve this design will be the length of the airfoil and the rotation of the airfoil tip from the quarter chord distance. The cost function used will optimize lift.

Plan of Presentation

This research will be presented as follows:

1. Discrete Parameterization will be explained in brevity along with CAD Parameterization and how CAPRI will assist in using CAD Parameters within a design cycle.
2. An explanation of the proposed design cycle and CAPRI's role within that cycle will be given.
3. How to synchronize a surface mesh with the associated CAD model via CAPRI will be explained.
4. The steps necessary to incorporating CAPRI into a computational design software suite will be discussed.
5. The results from applying this research to a 3D conceptual NACA 2412 wing will be given.

CHAPTER 2

GEOMETRY PARAMETERIZATION

Purpose and Application

Geometry parameterization serves two purposes: to reduce the number of control variables used for computational design and to create a surface with smooth deformations when these variables are modified. The issue of parameterization is crucial to computational design. How a model is parameterized has a large effect on how the design process continues and greatly influences the final outcome of the optimized model. Currently, methods of Discrete Parameterization are being used within the SimCenter's design software where applying CAPRI will allow the use of CAD parameters in computational design.

Discrete Parameterization

In its simplest form, design optimization can take the shape of a model as merely a collection of points in 3D space (without parameterization). The analysis in this simplest case involves moving each of the points around by some small amount and determining the result on the function specified for optimization. This process is both slow and very computationally expensive. Shape parameterization exists as an effort to reduce these complexities and inefficiencies.

Shape parameterization is the process of determining a set of parameters that control the size and shape of the model to be designed. By determining some driving parameters for a model, smooth derivatives can be calculated for changes to those parameters along with the added benefit of reducing the number of design variables from the number of surface points to the number of parameters.

There exist many parameterization methods in use today. They can be viewed as an attempt at reverse engineering the parameters used at building whatever model is to be optimized. Some of the more common methods involve fitting a function such as a Bezier or NURBS curve or surface to the collection of points along the geometry. While this method is often successful at parameterization, it can have difficulties reproducing the original shape.

Another method recently developed greatly reduces the need for surface fitting [4]. This method involves using an arbitrary shape as the control surface, usually a bounding box about the element. This shape includes a "net" of control points where the number and location of these points is determined according to their influence over the shape to be parameterized. In order to obtain control of the surface, each control point is assigned weights either positive or negative and then their influence is interpolated to the surface by the propagation of weights into the computational volume mesh using a Laplace equation.

There are many advantages of this method, most importantly that the surfaces do not need to be fitted with splines and parameterization for complex geometries is obtained fairly easily. While this method does a good job of reducing the number of control points and produces smooth surface derivatives, there exist many drawbacks as well. It is very difficult to control the influence of the control points upon the surface. For instance, since a linear equation is being used to drive the weights of each point in the "net", setting up the weights using non-linear functions, such as the rotation of an object, is outside of the capability of this implementation. Also, the resultant product is a collection of points, or a surface mesh and it is very hard to convert this data back into a useful form used for future analysis, such as a CAD model.

This research attempts to address these issues and provide an alternative approach where the parameterization of the model comes directly from the CAD.

CAD Parameterization

The CAD Parameterization begins when building the model within the CAD software. Different aspects of the model can be built dynamically so that when the user changes a

parameter, the entire model changes while maintaining integrity. An example of this would be the length of a wing, or a spline point used for construction of a lofted surface. This requires planning beyond the typical CAD build, and the CAD modeler to know that the model will be used dynamically.

The CAPRI CAE Gateway provides a direct interface to the CAD features and parameters. Most CAD systems represent their construction in the form of a tree, usually referred to as the Feature Tree [1]. CAPRI gives the user access to this tree, displaying the features and parameters in the form of branches. Parameters are the components within the model that contain values. CAPRI exposes all of the adjustable parameters to the user. When building a model in SolidWorks, the adjustable parameters that CAPRI is able to see most easily are Spline Points and "Smart Dimensions".

This shape parameterization must be validated during the design process. Each parameter needs to be independently modifiable, resulting in a valid geometry. Acceptable ranges for each parameter should be noted during the construction and validation of the model. This ensures that when one is modifying the parameters through the CAPRI Gateway, the changes will result in a valid geometry along with staying within the original intent of the CAD engineer. Note: when making any changes to a CAD model through CAPRI, it is always the CAD software that is operating on that model, not the CAPRI Gateway. The CAPRI server runs on a Windows machine and uses an instance of the CAD software to make changes to that model.

For the purpose of design, CAPRI offers the generation of analytic parameter sensitivities directly from CAD. Using these sensitivities results in smooth surface deformations, with the added advantage of CAD accuracy. This allows even non-linear design variables such as rotation to be used during the design process.

The advantage of this process is having real CAD information during the design optimization process giving access to a complete CAD representation of the geometry at each stage of the design cycle. CAPRI has the functionality to implement complex changes to the geometry, such as entering new spline points to curves and element rotation. One

disadvantage of this process is the synchronization of the surface mesh with that of the CAD features. Within PointWise [5] the user is able to mark boundaries in order to match these features, but as the CAD geometry gains in complexity, this synchronization increases in difficulty. Custom methodology has been built to handle these difference. Another difficulty is creating CAD geometries that are easily manipulated in the parameterized form. While this may seem like an easy task, in practice dynamically modifying a CAD model or part can have unintended side effects to that part. Care must be taken by the designer to build the model where all changes to all parameters have the desired effect. This adds more demands upon the CAD engineer to lay a foundation that is highly flexible and to also communicate the specifications and limitations of each parameter to all others involved in the design process.

CHAPTER 3

COMPUTATIONAL DESIGN CYCLE

Computational design is an iterative process in which a model is evaluated for a desired property and then modified successively in order to improve that desired property. The design process involves bringing together many different operations. These operations can be separated into five main processes.

Parameterization

The first step of the design cycle is to parameterize the model to be optimized. This determines the variables that will be used to modify the model. The proposed parameterization method begins with the creation of the CAD model. When building the model within SolidWorks, design parameters need to be defined as "Smart Dimensions" which will enable them to be viewed and modified through the CAPRI Gateway. All parameters need to be verified so that modifications to them result in a valid geometry according to the designers intent. This verification is accomplished by modifying each of the parameters separately and checking for a valid model after each change. Sometimes these changes can have adverse effects on a model resulting in an invalid geometry, or one that is inconsistent with the CAD engineer's intent. It is important to know how each change will affect the underlying geometry. During this process valid ranges can be determined and documented for the use within design process.

CAPRI will be used in this study to expose primitives in the solid representation that can be used as design variables. CAPRI will also be used to determine the changes in any point on the surface of this model with respect to the changes in any of the design variables. These

changes in the surface mesh with respect to the design variables are called the Sensitivity Derivatives.

Optimization

The optimizer is the driver of the entire design process. It makes the calls to the function evaluation and the sensitivity analysis, and from this determines new values for each of the design variables. After these have been determined, the optimizer initiates the process which modifies the shape of the model and moves the mesh according to the displacements. This stage is where the CAPRI functionality is inserted, making calls to the CAD system to drive the modified shape of the model and to update the surface derivatives. After the mesh has been moved, and the new sensitivities obtained, the process starts over again with the calls to the function evaluation and sensitivity analysis. When an improved value tolerance has been reached, the cycle is complete.

PORT[6], KSOPT[7], OPT++[8] are some examples of optimizers used at the SimCenter. For the scope of this research, OPT++ is used implementing a Quasi Newton Method (BFGS with trust region), and side constraints on design variables[9].

A Quasi Newton's Method (BFGS) was employed in the current optimization process. This method approximates a true Newton's method which is described as follows:

$$\vec{\beta}^{n+1} = \vec{\beta}^n - \tilde{H}_{\beta^n}^{-1} \nabla_{\beta}(\vec{\beta}^n)$$

Where \tilde{H} is the Hessian Matrix defined as:

$$\tilde{H} = \frac{\partial^2 f}{\partial \vec{\beta}^2}$$

which is a second order derivative. However, in our case only first order derivative information is available. In the BFGS scheme, \tilde{H} is then approximated as the following:

$$\tilde{H}_{n+1}^{-1} \approx \tilde{B}_{n+1}^{-1} = \tilde{B}_n^{-1} - \frac{\vec{Y}_n \vec{Y}_n^T}{\vec{Y}_n^T \nabla \vec{\beta}_n} - \frac{\tilde{B}_n \nabla \vec{\beta}_n (\tilde{B}_n \nabla \vec{\beta}_n)^T}{\nabla \vec{\beta}_n^T \tilde{B}_n \nabla \vec{\beta}_n} \quad (3.1)$$

Where

$$\begin{aligned}\tilde{B}_0 &= \tilde{I} \\ \vec{Y} &= \nabla f(\vec{\beta}^{n+1}) - \nabla f(\vec{\beta}^n)\end{aligned}$$

Giving the following upon iteration:

$$\vec{\beta}^{n+1} = \vec{\beta}^n - \alpha \tilde{B}_n \nabla f(\vec{\beta}^n) \quad (3.2)$$

At $n = 0$ this is the steepest decent method. Equation 3.2 reduces the problem to a 1D minimization problem where α is the step size which is determined using either a line search or trust region method. The line search minimizes equation 3.3 for α , holding B^n constant. So the task is to minimize the following function where all but α is constant.

$$\hat{f} = f(\vec{\beta}^n - \alpha \tilde{B}_n \nabla f(\vec{\beta}^n)) \quad (3.3)$$

Function Evaluation

The evaluation of the objective function is accomplished using the Tenasi flow solver. The Tenasi flow solver is an un-structured mesh, multi-regime, parallel, finite volume flow solver. It was developed at the University of Tennessee at Chattanooga's SimCenter[10]. The objective is to minimize some function $f(\beta)$ with $\beta_{min} \leq \beta \leq \beta_{max}$.

$$f(\beta) = \sqrt{(target - obj)^2} \quad (3.4)$$

Where

$$\begin{aligned}obj &= obj(\beta) \\ f(\beta) &\geq 0\end{aligned}$$

The objective function used here is the coefficient of lift:

$$obj = C_L = \frac{L}{\frac{1}{2}\rho V^2 A} \quad (3.5)$$

Lift (L) is a numerically integrated quantity produced by Tenasi[10].

Sensitivity Analysis

Tenasi's adjoint solver capability is used to compute the sensitivity analysis of the objective function. Tenasi utilizes linear elasticity and dual adjoint calculations for field mesh sensitivities[11].

Shape and Mesh Movement

When the optimizer has determined the changes in parameters that need to be made, it uses the CAPRI API to make the required changes to the underlying CAD model. Once the parameters have been updated on the CAD model and a new model generated, the new coordinates from the surface mesh will be obtained by retrieving the new xyz coordinate for the scaled uv coordinate related to each point on the Surface Mesh. This process will be explained in greater detail in the next chapter. The xyz displacement in 3D space will be used to drive the mesh movement, which uses a Linear Elastic smoothing routine contained within Tenasi [12]. Sensitivity Derivatives are updated after each change in the underlying geometry to ensure that they are accurate to the newest version of the optimized model.

Final Model

Upon the last iteration of the design cycle, the final call to the CAD engine to adjust the parameters results in an optimized version of that model. This model can then be exported for future analysis. A flow chart explaining the design cycle can be viewed in figure 3.1.

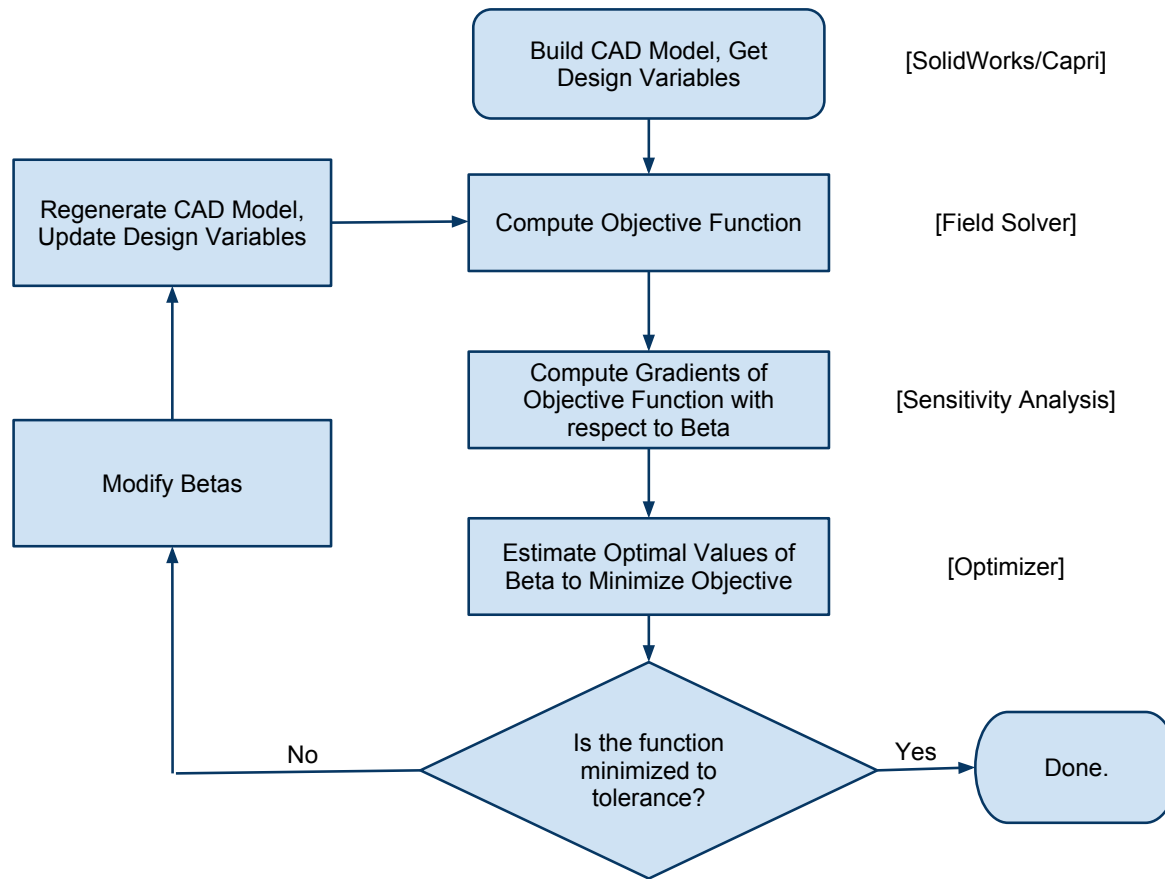


Figure 3.1 Flow Chart of Design Cycle Using CAPRI

CHAPTER 4

CAPRI IMPLEMENTATION

CAPRI is a "CAD-Vendor Neutral Gateway for Engineering Analysis and Design" [1]. What this means is that CAPRI will grant the user access to one of many supported CAD systems. The following CAD systems are supported: Pro/ENGINEER[13], CATIA[14], OpenCASCADE[15], SolidWorks[2], UGS UG NX[16], and Parasolid[17]. For the course of this research SolidWorks has been selected as the CAD system of choice.

For any of the aforementioned CAD systems the specific installation and the availability of that system needs to be considered when building a CAPRI application. In this case, SolidWorks is only available on a Windows platform and is a large program which is expensive in terms of memory use. Due to these issues, a Dell workstation running the Windows OS has been dedicated to the use of SolidWorks and the client/server version of CAPRI is used on that machine.

Client/Server Specifications

Since all development of this software has been accomplished on a MAC OS X or LINUX platform and SolidWorks only executes on the Windows OS, communications between the design and CAD software must be accomplished through the Web Services. The Web Services provided by CAPRI utilize the following web-based technologies[1]:

- SOAP: soap is a Simple Object Application Protocol. This is being used as the foundation for the CAPRI client/server.
- DIME: a Direct Internet Message Encapsulation specification which CAPRI uses to transfer the CAD/Geometry Kernel files from the client to the server.

- Message Compression: CAPRI implements message compression to reduce network traffic. All messages are decompressed when received by the client.
- Secure SSL Option: A secure SSL option exists for the CAPRI client/server interactions, however this will not be utilized for this implementation.

It should be noted that the client/server version of CAPRI will place restrictions on the uses of CAPRI within the design application. Since all data are to be communicated over the network, it is very important to limit these communications to a minimum. When hundreds or even thousands of point queries need to be executed, it is recommended to package those queries using the CAPRI functions `Begin()` and `End()` which will be explained later in detail.

Surface Mesh Synchronization

When using a CAD model as the basis for design cycles, certain steps need to be taken to ensure agreement between the computational mesh and that underlying model. Once the points on all the surfaces from the computational mesh have been matched to their corresponding CAD entities, the implementation becomes straight forward. Due to the inherent disconnect between CAD, the computational mesh, and the design software, a custom method was developed for matching these discretized surfaces to the CAD. This method needed to take into consideration the data structures used in both the GEOMETRY libraries and CAPRI. Once these relationships have been established, a robust CAPRI application can be built.

Within the SimCenter's current Geometry libraries exists the capability to strip the surface mesh out of a computational mesh. All the mapping work done in CAPRI uses this surface mesh. Each point on the surface mesh exists in one of three categories. Each category is determined by the number of Materials that a point touches. Materials are the boundary conditions specified per each face of the geometry. These are specified by the user when creating a computational mesh in PointWise or GridGen. For example, the materials

for an airfoil might be an upper surface, a lower surface, a tip, and a symmetry plane. A Point lies in one of the following categories:

- Critical Point: A Critical Point is associated with only one point and is the basis for all datatypes within the surface mesh. It is related to three or more materials.
- Edge: An Edge is associated with a string of points associated by the same materials. An Edge is identified if a point is associated with exactly two materials. An Edge is bounded by two Critical Points.
- Face: This is the most general of categories for a point to lie in, a point is on a Face if it is associated with only one material. A Face is bounded by Edges.
- Material: The type of boundary condition associated with a face, synonymous with Face.

Similarly, CAPRI has a hierarchy of categories which are determined by the CAD construction.

- Node: A Node is the simplest entry and is a point in 3D space.
- Edge: An Edge is a curve bounded by two unique Nodes. Edges are parameterized on a value of t .
- Face: A Face is a parameterized (u, v) surface. All faces are bounded by edges, but by no specific number.
- Boundary: A boundary is a collection of one or more faces with associated names, (i.e., turret, wing, etc)
- Volume: A volume can be either a solid or a shell/sheet body. A volume that is a body is a completely closed region in 3D space. For the course of this research, only solid bodies are used and there is the additional limitation of only one volume per design.

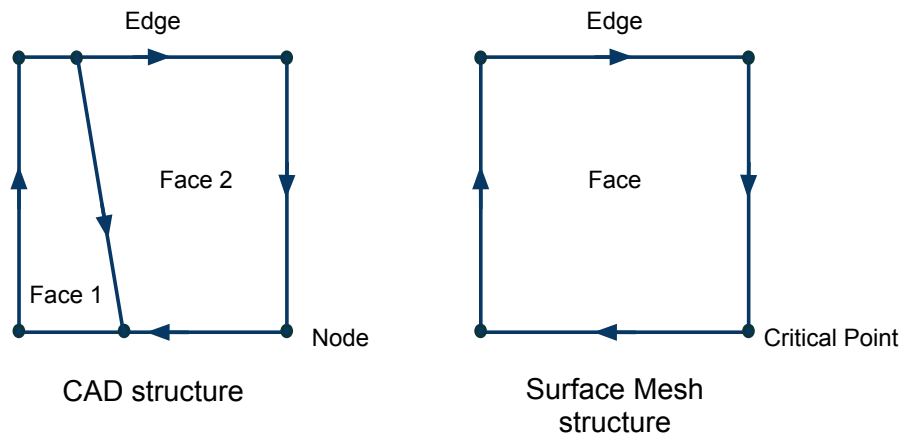


Figure 4.1 CAD Face vs. Mesh Face

An important distinction needs to be made at this point. Within the Surface Mesh data structures, each point is associated as either being a Critical Point, or existing on an Edge or a Face. However, when referencing the CAD the structures are Nodes, Edges and Faces, not points. The task here will be to determine where each point lies on the CAD model and what type of entity to which is associated with. See figure 4.1 for illustration.

Despite the apparent similarity between the topology of the surface mesh and the topology defined by CAPRI, they do not match up exactly. A few assumptions have been made for the purpose of this mapping: (1) that every Critical Point on the surface mesh maps to a Node on the CAD model, (2) that every point on an Edge in the surface mesh will map to an Edge or a Node on the CAD model, and (3) that every point that lies on a Face in the surface mesh will map to a Face, an Edge, or a Node on the CAD model.

Note that this is not true in the reverse order. For each Surface Mesh entity there is at least one, maybe multiple associated CAD entities. For each CAD entity, there may or may not be a matching Surface Mesh entity. For instance, a Node on the CAD may link two edges together on one side of a face. In the surface mesh, this may be considered one edge and the Node from the CAD might not have a Critical Point on it. So in this case, one Edge from the Surface Mesh will be related to two Edges on the CAD. This introduces the concept of a Quilt[18].

A Quilt is a mapping from one Face within the Surface Mesh to the associated CAD Face(s) of that Face. In this implementation, a Quilt related to a Face will contain unique CAD Faces, but will not be shared with other Quilts. Similarly, another mapping exists to map an Edge on the Surface Mesh to its associated Edge(s) on the CAD model. The CAD Edges contained in each Edge mapping are also unique and not shared among Edge maps. Quilt and Edge mappings will be built as the primary data structure used for reference from the surface mesh to the CAD model.

Quilt Construction

The quilt building algorithm begins by using the following CAPRI geometry based-data routines [1]:

- `GetVolume()`: returns the name of the volume and the associated counts of faces, edges, and nodes for the specified volume
- `GetFace()`: returns all associated data for the specified face: the face uv ranges, the number of edge loops, the number of edge loops, and pointers to the edge pairs that make up the loops
- `GetEdge()`: returns the edge t range values and the node endpoint indices for a specified edge
- `GetNode()`: returns the node coordinates in 3D space for a specified node
- `Box()`: returns a bounding box in 3D space for a specified volume

Once the data for a given volume have been retrieved, the first step is to match each Critical Point on the Surface Mesh to a Node on the CAD. Note that there will most likely be more nodes on the CAD than Critical Points on the Surface Mesh. This matching is accomplished by looping over all of the Critical Points within the Surface Mesh and finding the CAD Node with the minimum distance to that Critical Point. If no CAD Node is found to be within a reasonable distance from any given Critical Point, the program will exit since this violates the requirement for the Critical Points to have matching Nodes defined on the

CAD model. After all the Critical Points have been related to their corresponding CAD Nodes, the remaining CAD Nodes are looped over and associated to their closest points on the surface mesh. These points within the Surface Mesh are marked as being associated to a CAD Node along with all Nodes from the CAD structure being labeled with the entity and that entity type on the Surface Mesh that to which they are associated.

After all CAD Node associations have been made, the next step is to loop over all the faces on the CAD. For each CAD Face, the Materials associated to each CAD Node (found from each associated Critical Point) are counted. If there is a Material that is related to that CAD face more than the others, then the CAD Face is added to that Material's Quilt. If there is no Material that dominates the previous count, another method is applied to find the Material to which the CAD Face is related. The xyz coordinate of the midpoint of the CAD Face is obtained (using CAPRI function `PointOnFace()`) and compared to all points in the Surface Mesh. The node with the minimum distance contains the Face to which this CAD lies. This method is definitely more expensive than the previous, but in practice is not used very often. The CAD index is then added to that Face's quilt. Once all CAD Faces have been looped over, the quilt mapping is complete.

The process for building the Edge mappings begins by looping over all of the Edges within the CAD and checking each Edge's endpoints. If both of the CAD Edge's endpoints are related to Critical Point on the surface Mesh, then all Surface Mesh Edges are looped over and checked for matching endpoints. If a matching Edge is found then the CAD Edge is counted. If there is one CAD Edge which has a dominating count, that CAD Edge is added to that Surface Mesh Edge's Edge Map. If there isn't a dominating Edge count, then the midpoint of that CAD EDGE is taken, using CAPRI `PointOnEdge()`. That midpoint is compared to all points on all Surface Mesh Edges and the closest point determines the Edge Map to which the CAD Edge will be added. If either both or neither of the CAD Edge's Nodes are not on Critical Points then one or both of the Edge Nodes will already contain information about the Surface Mesh Edge that it lies on from the initial CAPRI Node mapping algorithm explained earlier.

Once the quilts are built for the Edges and Faces on the Surface Mesh, the associations exist to easily get all the information needed from CAPRI.

Snap to Surface

The first step to any CAPRI application is to verify that all the Surface Mesh entities have been successfully mapped to their corresponding CAD Entities. This is accomplished by projecting all points from the surface mesh onto their associated CAD entities and monitoring the maximum perturbation distance. The Snap to Surface algorithm here serves multiple purposes, first to finalize the which CAD Face or Edge each point from the surface mesh is actually associated with, and second, to get the xyz coordinate from the CAD Software, not the Surface Mesh.

When doing this projection, there needs to be consideration taken to minimize the communication to the CAD software since all communications are done over a network. The point queries provided within CAPRI can be executed in one of two modes: normal or grouped. By utilizing the functions `Begin()` and `End()`, the user can group his functions and they will not be executed until `End()` has been called. This allows the query to be packaged, taking potentially thousands communications across the network down to only one send and receive.

The projection routine takes an array of points from one Face of the Surface Mesh and that Face's associated quilt and modifies the xyz coordinates contained within those points to lie on the CAD model. It also returns the uv or t coordinates (if on a CAD Face or Edge) and the index of the CAD entity on which each point lies. The `ClosestOnCAD` function operates by projecting each point onto each CAD entity within a Quilt. The projection which has the minimum distance from the initial point is returned. The CAPRI functions used here are `ClosestOnFace` and `ClosestOnEdge`. These methods not only return the xyz coordinate that lies on the database, but the parametric uv or t coordinates as well. Once all CAD Faces and Edges have been looped over, the process has been completed.

Within this process, specific information is also stored for each point on the Surface mesh which is required by all other CAPRI functions. The most important information is what type of entity the point lies on and what the index of that entity is within the CAD system. Also, the value of t or uv for that edge or face is stored for use when generating the Sensitivity Derivatives.

Quilt and Closest Point Validation

For validation of the Quilt and Edge mappings a box was built in SolidWorks which contained features that would require Quilt and Edge maps of varying sizes. The CAD model and the associated Surface Mesh can be viewed in figures 4.2 and 4.3. The Surface Mesh is viewed as a six sided box, where the top has a ramp, however still viewed as one face or material. All the remaining five sides of the box are marked as individual faces within PointWise. This created the need for a quilt mapping on the top surface to map to three different CAD faces. Additionally, two of the Edges of the surface mesh along the top of the box need to map to three different Edges on the CAD model.

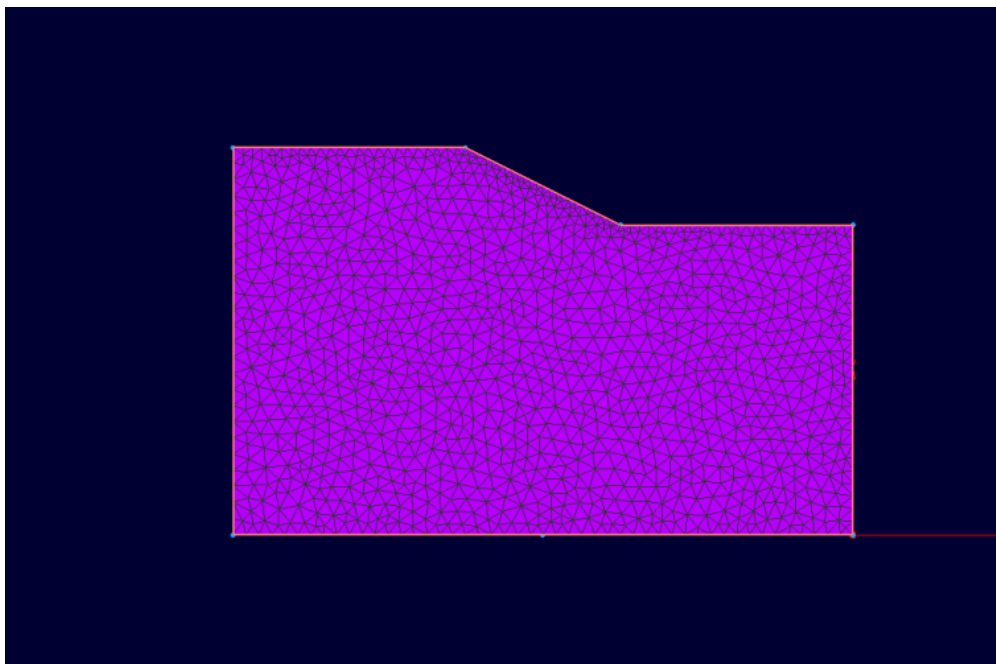
Generating Sensitivity Derivatives

The current implementation of CAPRI uses a hybrid approach of using instancing and finite differences [19] to generate sensitivities in cases where analytic methods have not yet been developed. The sensitivity derivative features of the CAPRI API are still in development and this approach has been provided so that work may be continued in this area without having all analytic definitions for each supported CAD package. For the course of the present research, the calls to retrieve the sensitivity derivatives are treated as a black box, with the only adjustable parameter being the initial finite difference step size*.

*The sensitivity derivative functionality to CAPRI is relatively new, where all of the Sensitivity Derivative functions exist in "sensit.h" and need to be included into the source separately. Additionally, since all of these functions are new, much of the work done in this area has been accomplished with Robert Haines's direct guidance[20]

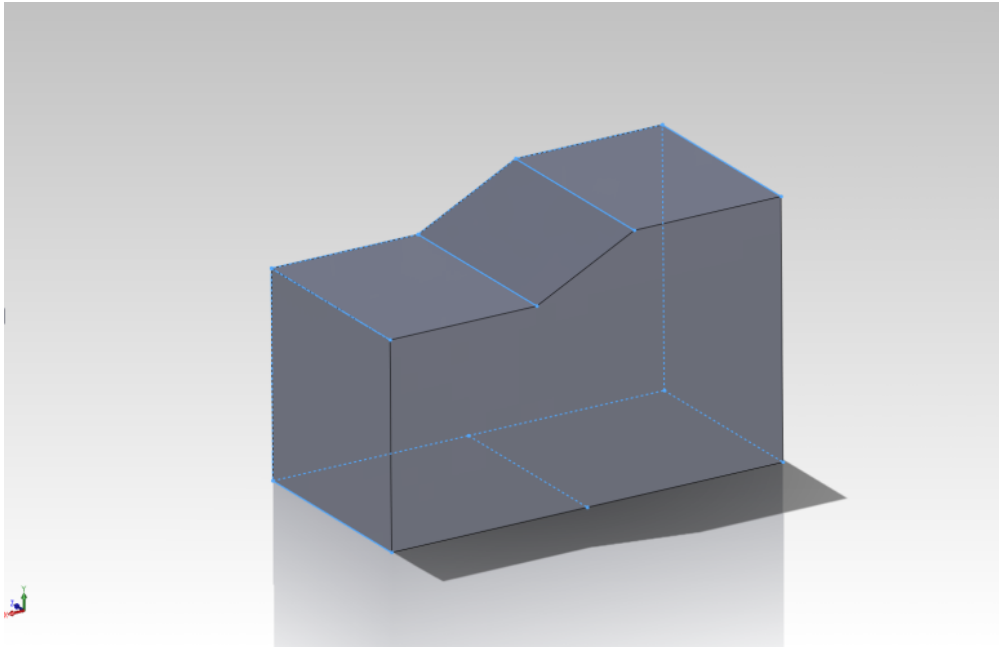


(a) CAD Face

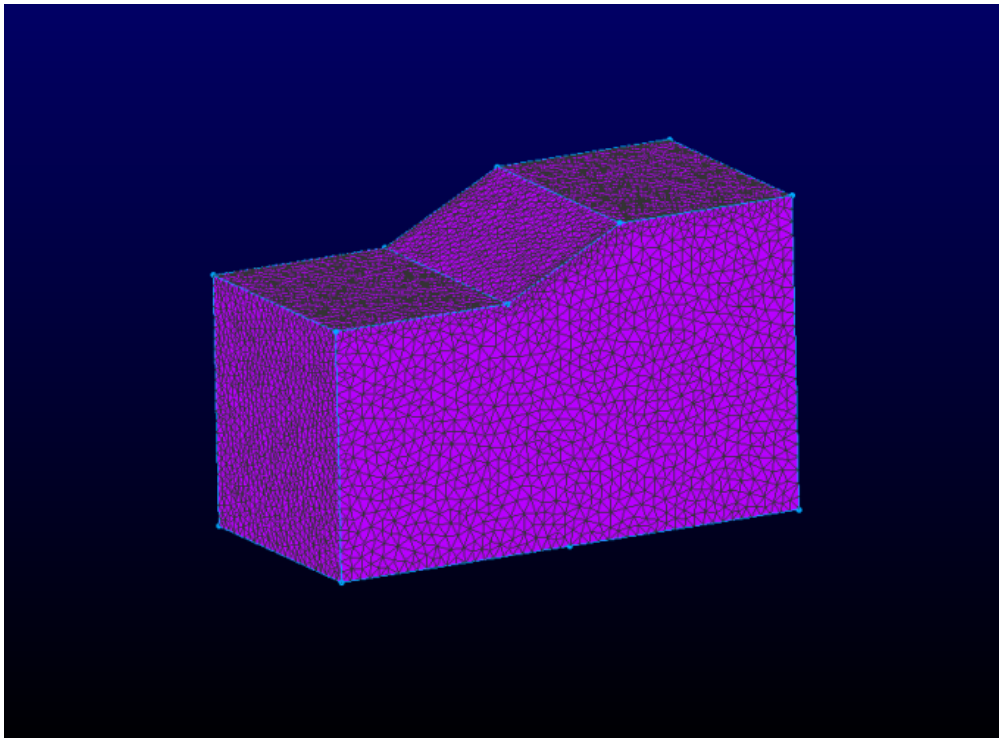


(b) Surface Mesh

Figure 4.2 CAD and Surface Faces used for Quilt validation: side view



(a) CAD Model



(b) Surface Mesh

Figure 4.3 CAD Model and Surface Mesh used for Quilt validation: perspective view

The Sensitivity Derivatives that CAPRI provides are the change of each xyz coordinate related to the change in each parameter for every point on the surface mesh. Accomplishing this, per the current implementation, is a multi-step task.

The first step is to initialize the Sensitivity Derivative Handle within CAPRI. This is accomplished by calling the `InitSDHandle` function provided, which requires the user to provide the number of parameters selected and an array containing those indexes along with the preferred finite difference step size for each parameter. This gives CAPRI a reference to the Sensitivity Derivatives that are to be used as well as the selected Parameters.

The next step is to populate an array of the CAPRI custom structure datatype `SensVerts`. Using the `InitSDVerts` function which requires all the information of the Points on the surface mesh, their CAD entity type (Edge or Face), and their respective uv or t coordinates will populate the needed arrays for the user.

Last is to call the function `EvalSD()` which returns a list of all the Sensitivity Derivatives of each parameter. Once this has been accomplished, it can easily be output to a text file for the design software. Note that it is very important to have built the Quilts and projected the Surface Mesh onto the CAD before attempting this step, as that information is needed within this process.

Modifying the CAD Model

When performing computational design, it is necessary to modify the CAD model over each iteration of the cycle. CAPRI supports the modification of parameters within a CAD model and can output a changed version of the model for each iteration. This process is fairly straightforward and can be accomplished with only a few CAPRI functions.

The first step is to retrieve the parameter to be modified; this is done using the CAPRI function, `GetParam()`. This function returns the following information about each parameter:

- Name: The name of the parameter to be modified, this is specified during the design process with the CAD software

- Type: The parameter type, either a Boolean, Integer, Real, Spline Curve Point or a String
- Bit Flag: This flag specifies if the parameter is "Read Only" and if limits for modifying the parameter exist on the model
- Length: The number of values associated with a parameter (could be the number of spline points)
- Branch: The associated branch index to the parameter for reference within the feature tree

After obtaining this information about the parameter, CAPRI offers methods to Get and Set each of the mentioned parameter types. For the course of this research, the only parameter type that will be supported is the parameter type: Real.

If valid ranges are not available to the user, it is important to manually verify allowable ranges for each parameter when building the model. If a parameter value is set outside of the allowable range and invalidates the model, CAPRI will return an error. In the current implementation, parameter ranges were not found for the parameters that were being used. For each design cycle, conservative limits were chosen to maintain model integrity throughout the entire process.

After all the necessary parameters have been modified, CAPRI needs to update the new version of the Master Model. This is accomplished using the CAPRI function Regenerate() which will return the new model index. In order to query this new model, the user needs to call GetModel() with the new model index in order to establish valid ranges for the updated model.

After the model has been modified, the next consideration is to be the Surface Mesh associated with that model. The mesh needs to move to match the changes in the underlying geometry. During the initial projection of each node onto the database, information was stored for each node; i.e, the CAD entity which with each point is associated, the CAD

entity type, and the associated uv coordinate if it lies on a CAD Face or t coordinate if a CAD Edge (there is no associated coordinate for a CAD Node). The new position of the surface mesh coordinates are found by querying the model for the xyz coordinate of each point associated with their uv or t coordinate. The functions used here are `PointOnEdge()` and `PointOnFace()`.

It is important to re-query the ranges for each CAD Edge and Face, as these might have changed while modifying the model. A linear relationship can be established from the changes in ranges from one iteration to the next. The new uv and t coordinates are calculated as follows:

$$\begin{aligned}
 t^{k+1} &= t^k * \frac{t_{max}^{k+1} - t_{min}^{k+1}}{t_{max}^k - t_{min}^k} \\
 u^{k+1} &= u^k * \frac{u_{max}^{k+1} - u_{min}^{k+1}}{u_{max}^k - u_{min}^k} \\
 v^{k+1} &= v^k * \frac{v_{max}^{k+1} - v_{min}^{k+1}}{v_{max}^k - v_{min}^k}
 \end{aligned} \tag{4.1}$$

Where $k + 1$ denotes the next iteration.

For verification, a test wing was generated with an ellipse on the root and tip of the wing see figures 4.4, 4.5, and 4.6. The adjustable parameters in this model are: tip displacement (sweep), tip height, tip width, and wing length (from root to tip). The following tables 4.1, 4.2, and 4.3 show the changes in CAD Parameters and updated uv and t limits for each of the CAD entities.

Table 4.1 Parameter Modifications for an Elliptical Wing

Parameter Name	Initial Value	Resulting Value
Wing Length	2.00	2.00
Tip Displacement (Sweep)	2.00	2.50
Tip Height	0.15	0.10
Tip Width	0.75	1.00

Table 4.2 Changes in the CAD Face (u, v) Ranges after Parameter Modification

Face Index		Initial (u, v)	Resulting (u, v)
1	min	(0.0000, 0.0000)	(0.0000, 0.0000)
	max	(0.5000, 1.0000)	(0.5000, 1.0000)
2	min	(-0.5060, -0.0759)	(-0.5060, -0.0759)
	max	(0.5060, 0.0759)	(0.5060, 0.0759)
3	min	(-0.3795, -0.0759)	(-0.5060, -0.0506)
	max	(0.3795, 0.0759)	(0.5060, 0.0506)
4	min	(0.5000, 0.0000)	(0.5000, 0.0000)
	max	(1.0000, 1.0000)	(1.0000, 1.0000)

Table 4.3 Changes in the CAD Edge t Ranges after Parameter Modification

Edge Index		Initial t	Resulting t
1	min	0.0000	0.0000
	max	1.0000	1.0000
2	min	0.0000	0.0000
	max	1.0000	1.0000
3	min	3.1415	3.1415
	max	6.2832	6.2832
4	min	-0.0000	-0.0000
	max	3.1415	3.1415
5	min	3.1415	3.1415
	max	6.2831	6.2831
6	min	-0.0000	-0.0000
	max	3.1416	3.1416

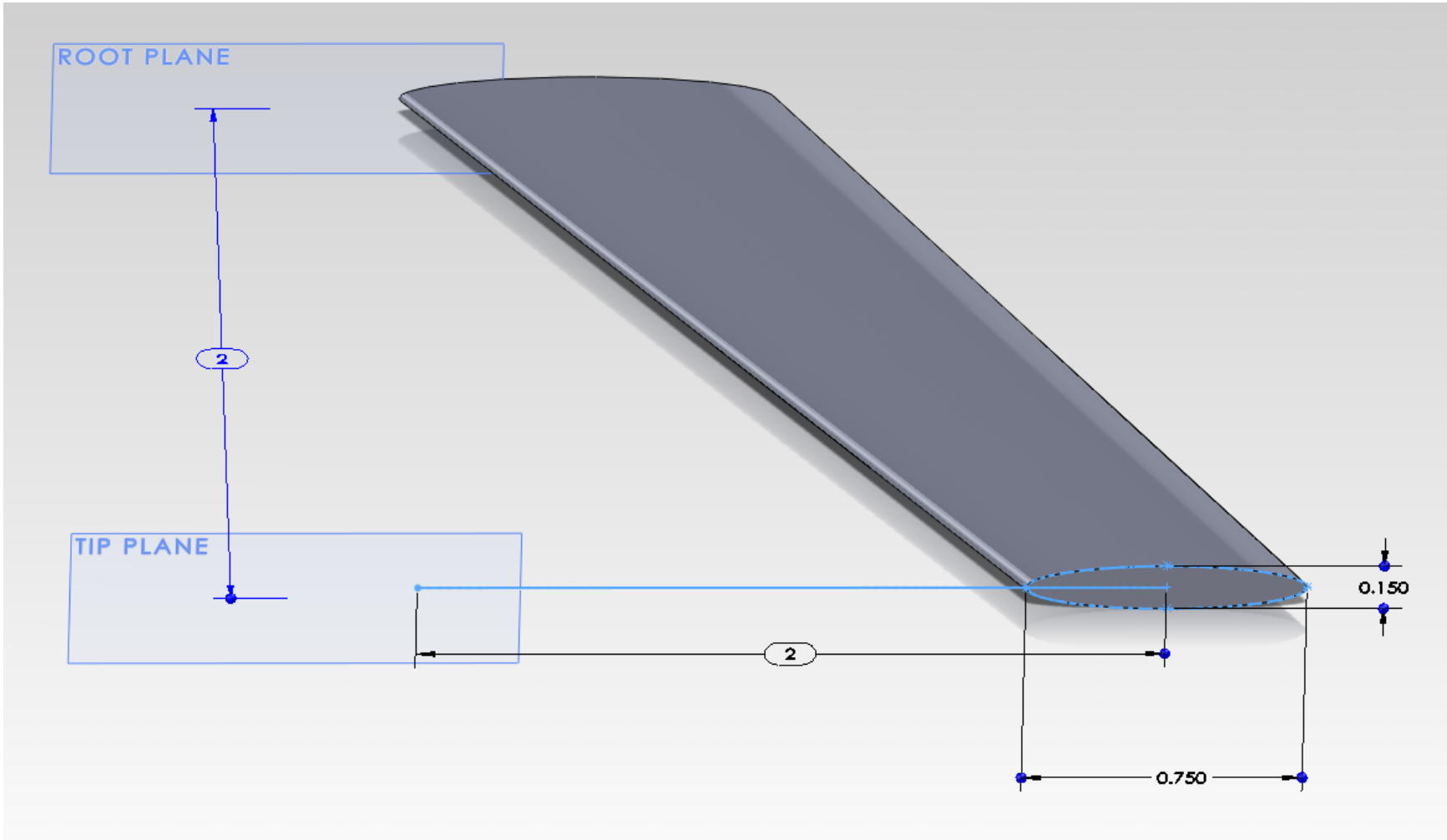
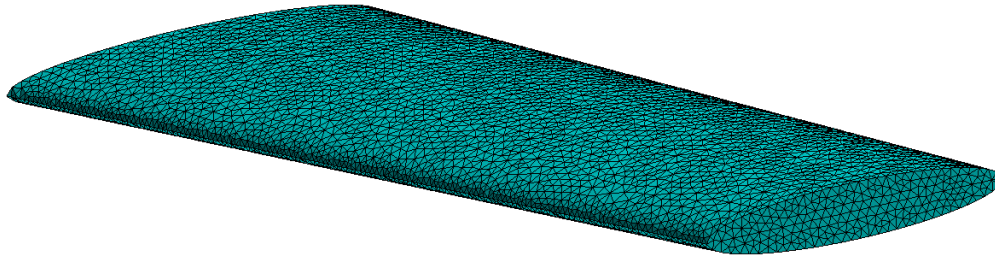
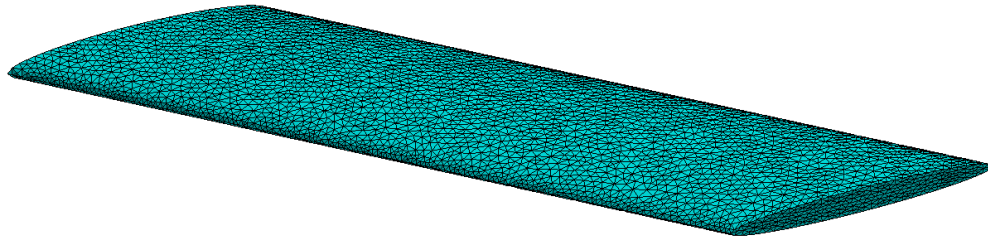


Figure 4.4 CAD Parameterization: Ellipse Wing



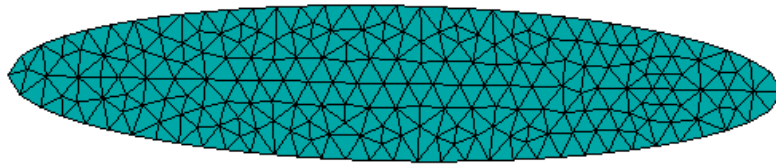
(a) Initial



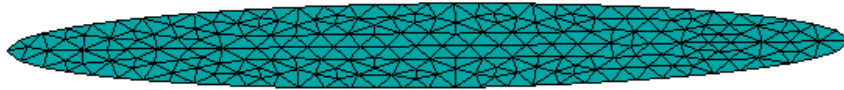
(b) Modified

Figure 4.5 Surface Mesh Modification: Ellipse Wing

The only face where the uv range changes is on the tip of the wing, Face 3. None of the Edges changed in their t ranges, however this proved effective in a later case. The formulas from equation 4.1 were applied and the mesh translation was successful as shown in figure 4.6. As the ellipse is vertically compressed, horizontally stretched and shifted in the x direction, the mesh spacing follows. This algorithm should result in similar quality meshes given reasonable geometry modifications.



(a) Initial



(b) Modified

Figure 4.6 Surface Mesh Modification: Tip of Ellipse Wing

CAPRI Framework

A framework for future work was built fully integrated into the SimCenter's current GEOMETRY libraries.

Geometry Software Library Additions

- Load CAD Model - This function starts an instance of CAPRI and loads the specified CAD model onto the server.
- Build Quilts - This function builds all of the quilts needed
- Closest On CAD - This function will return either a Point on the CAD surface or an array of Points on the CAD surface. It supports both normal and grouped modes of communication to the CAPRI Server.
- Snap Surface to CAD - This function takes the Surface Mesh and loops over all of the faces on the geometry, projecting all of the points onto the database. It also provides the functionality of determining exactly what CAD entity each point lies on and the location on that CAD entity in parametric coordinates, either t or (u, v) .
- Output CAD Sensitivity Derivatives - This function takes all the Points on the Surface Mesh and outputs the Sensitivity Derivatives for those points for each specified parameter.
- Update CAD Model - This function allows the user to modify the parameters within the current CAD model and save them to a new CAD Model along with updating the coordinates of the surface mesh to lie on the modified geometry.

A flowchart of how to obtain the Sensitivity Derivatives using this framework is shown in figure 4.7.

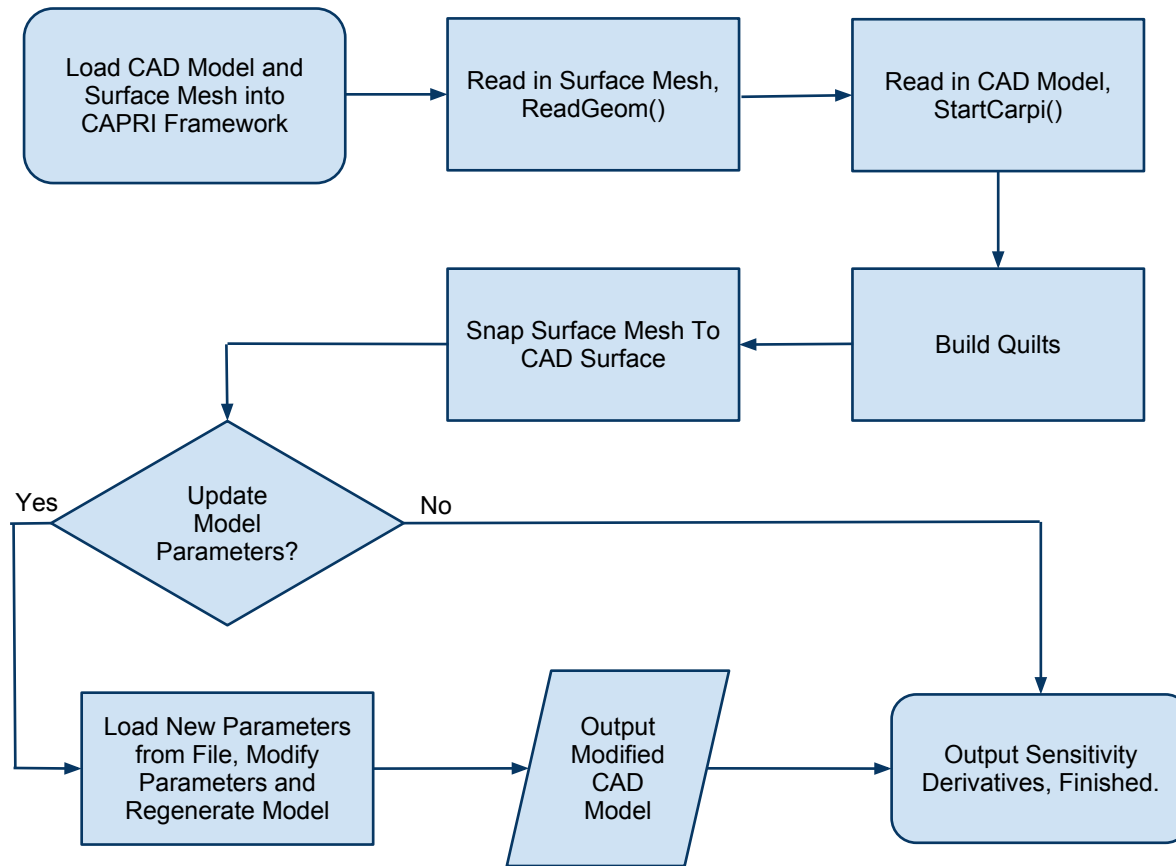


Figure 4.7 Flow Chart of CAPRI Implementation

CHAPTER 5

APPLICATION AND RESULTS

Initial CAD Model

The geometry used for computational design was built in SolidWorks[2] using the model of a NACA 2412 on both root and tip sections. The model was then exported to a format readable to Pointwise[5], a CFD meshing toolkit. From there a computational mesh was built. Figures 5.1, 5.2(a) and 5.2(b) show the 2D cross section of the wing, the initial CAD model built, and the associated surface mesh.

Building the CAD Model

The NACA 2412 Wing cross section was obtained from UIUC Airfoil Coordinates Database [3]. The data obtained for this wing were imported as spline points and entered into SolidWorks as a "Curve through XYZ points". This created a database entity on the $z = 0$ plane and will be referred to as the "root". As with all wing cross sections in the NACA family, the upper and lower curves did not connect on the trailing edge. Extra points were added to this curve to close this gap, making a sharp point on the trailing edge. This curve was then copied to another plane at $z = 1$, referred to as the "tip", and a loft created between both curves to initialize the solid model. The distance between these two planes became the first parameter.

Parameterization

The model built for this application features the following parameterized values:

1. Spline points on the upper and lower curves of the tip of the wing

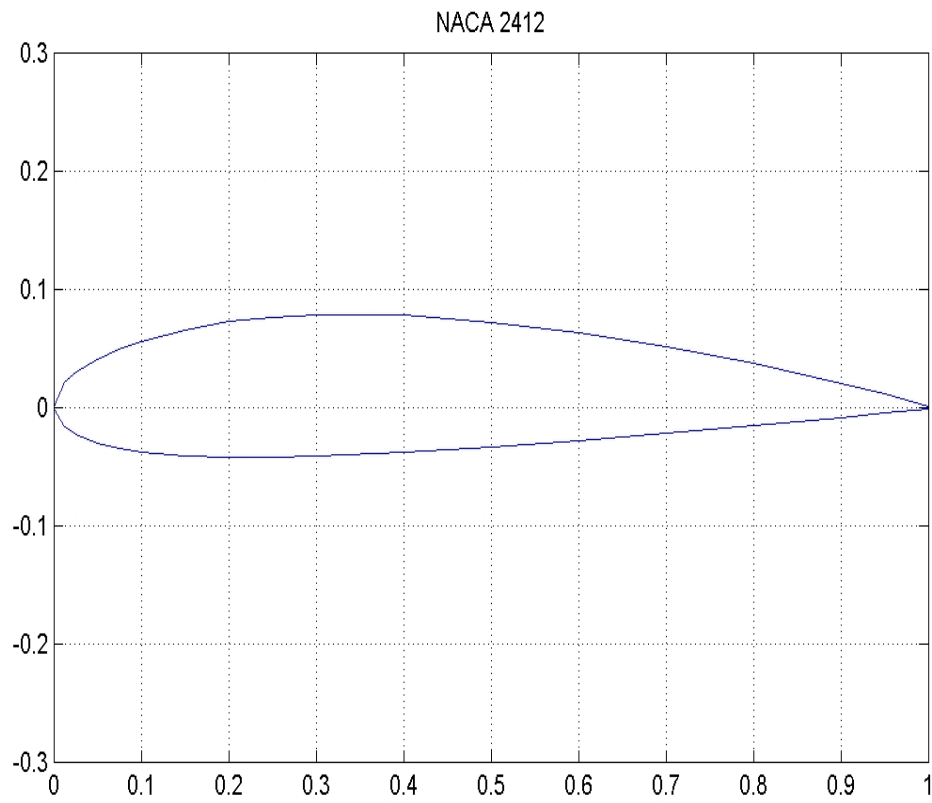
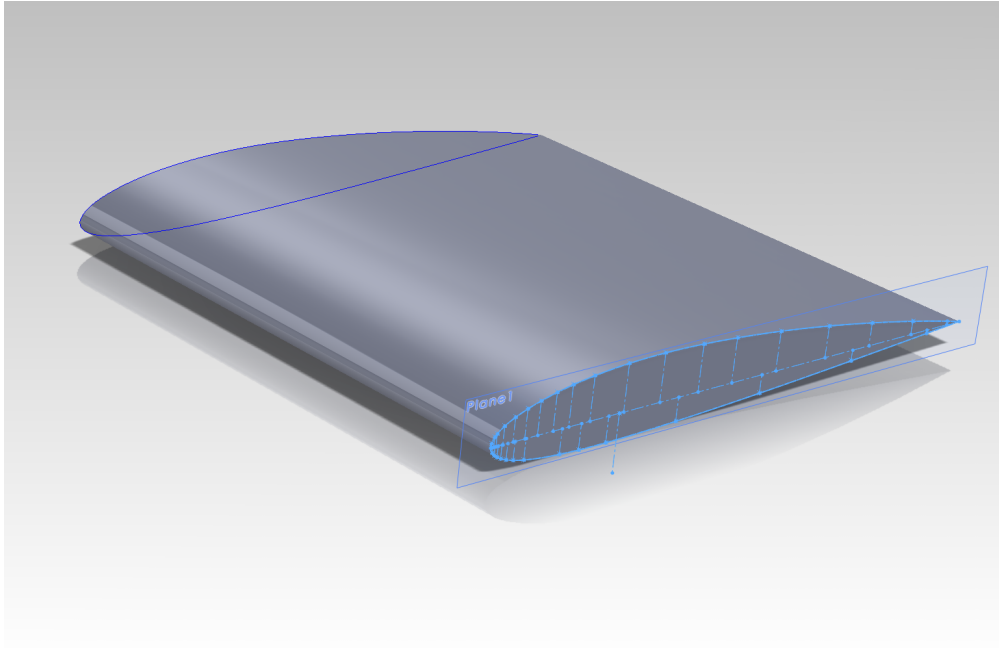
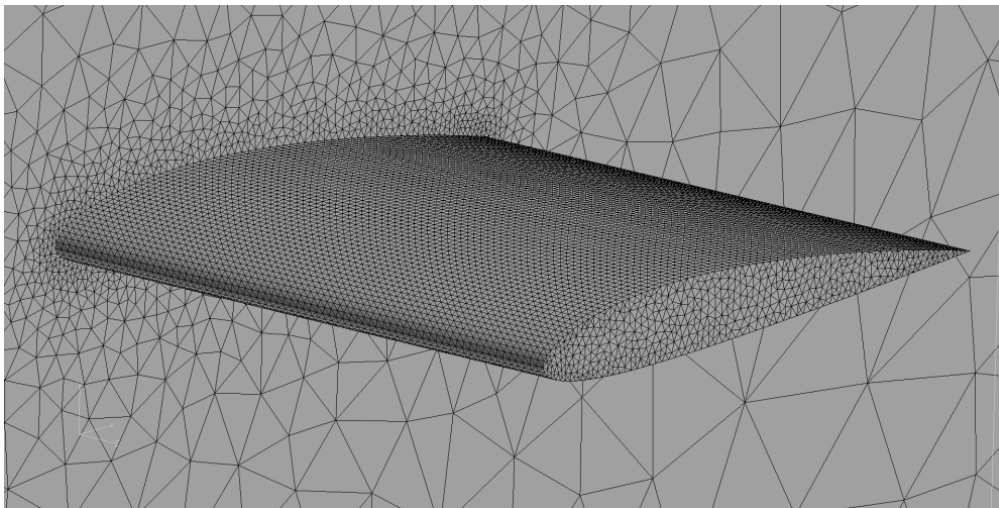


Figure 5.1 NACA 2412 Cross Section [3]



(a) CAD Model



(b) Surface Mesh

Figure 5.2 Model Used for Design

2. Wing length

3. Rotation along the wing axis at the quarter chord distance of the tip

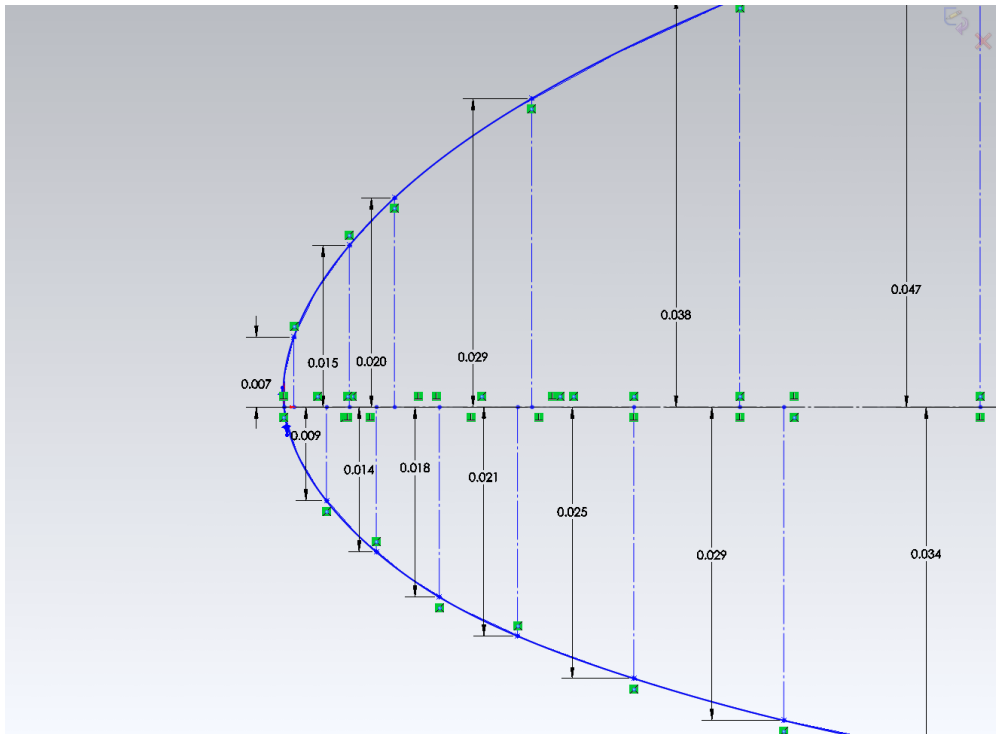
In order to maintain the integrity of the airfoil shape on the tip during rotation, special relationships needed to be set up to move the spline points in relation to the angle of rotation. Perpendicular lines were drawn from each spline point to the horizontal axis of the wing, and a relationship was set up for each of these lines to remain perpendicular to the horizontal axis. Additionally, the length of each one of these lines was established as a "Smart Dimension" so that the parameters could be modified from CAPRI. These parameters are shown in figure 5.3(a). The lengths of these spline points can be modified resulting in a valid model as in figure 5.3(b). Figure 5.4 shows an end-view of clockwise rotation of the tip of the wing. This rotation takes place at the quarter-chord distance of the wing resulting in a twist of the wing with the root section remaining stationary.

Surface Mesh Creation

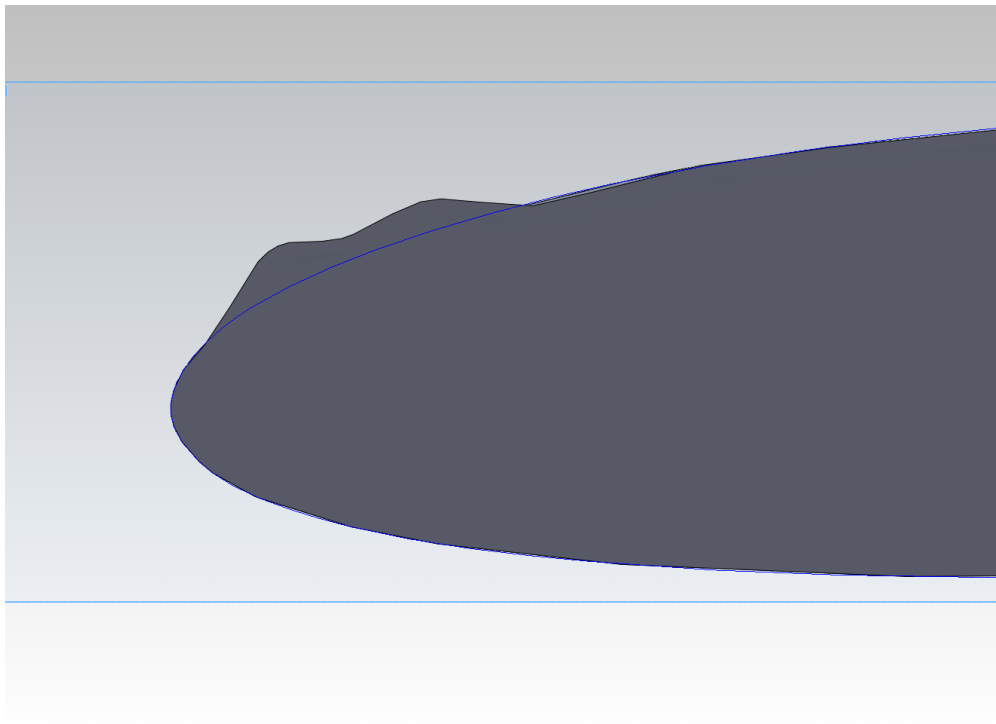
The surface mesh for the NACA 2412 wing without twist is shown in figure 5.2(b). The body of the wing had three materials defined: the upper surface of the wing, the lower surface of the wing, and the wing tip. Two more materials were specified: the symmetry plane and the far-field. Since both of these faces did not lie on the surface of the wing, their associated quilts should be empty, while the other Materials should be associated with one CAD Face each.

Design Variables

For the scope of this initial case, the parameters for tip rotation and wing length were chosen as design variables with the objective of optimizing lift. This case was selected in the attempt of using a non-linear relationship (i.e. rotation) for parameterization. Since this relationship is non-linear and the derivatives returned from CAPRI are only first order, the change in rotation was given upper and lower limits so as to only move the wing in small increments of rotation per each cycle. Each cycle was limited to a maximum rotation angle



(a) Spline Point parameterization and "Smart Dimensions"



(b) Movement of the Spline Points on the tip of the wing

Figure 5.3 NACA 2412 CAD Model Parameterization: Spline Points

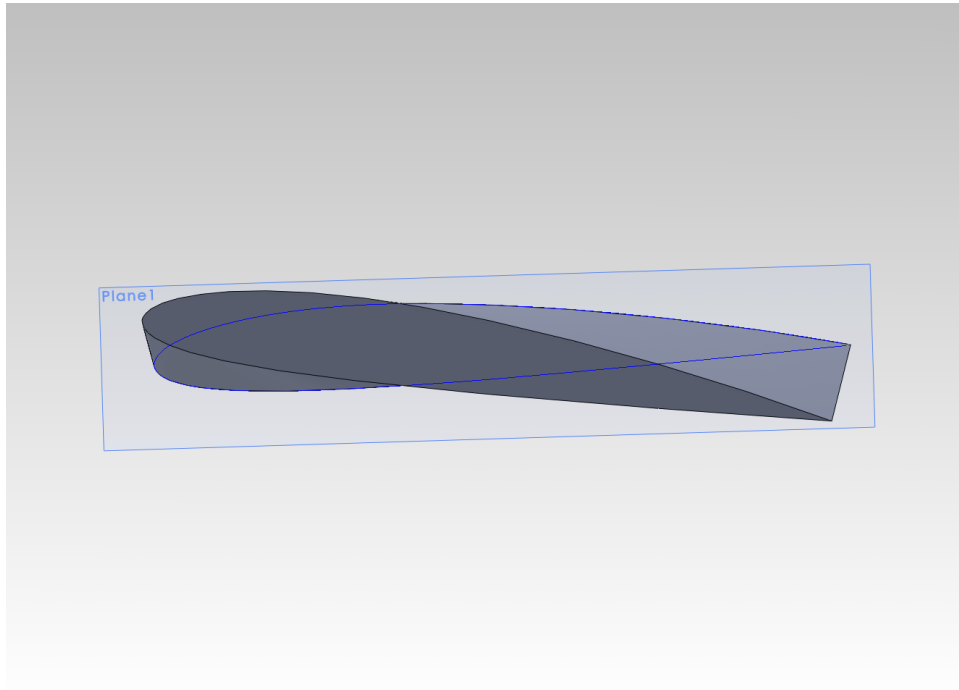


Figure 5.4 NACA 2412 CAD Model Parameterization: Tip Rotation 10°

of $\pm 5^\circ$ relative to the current value. Note that these limits were specific to each cycle and not to the entire design process. If the optimizer wished to move the wing $\pm 15^\circ$, it would take three iterations to rotate the tip that far. The Wing Length parameter was limited from 0.25 meters to 2.0 meters for the entire cycle. After each cycle the optimizer was called and new parameter values requested, because the surface derivatives needed to be updated to reflect these changes.

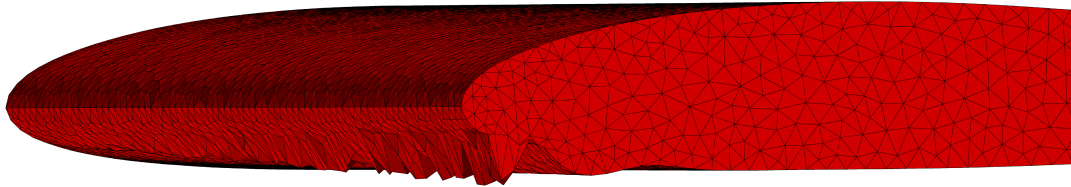
Issues in Implementation

Two issues were found when watching the surface mesh change according to modifications in the design variables.

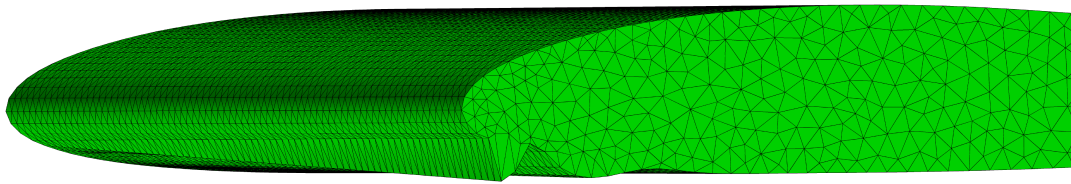
The first issue, as seen in figure 5.5 was a result of checking for smooth surface deformations while moving the spline points on the bottom of the wing near the tip. Although the changes were consistent among the points, the unstructured surface mesh still appeared jagged and inconsistent in the span-wise direction of the modified spline point as shown in figure 5.5(a). This did not reflect an error in the methodology so much as indicating the type

of mesh that is needed when making changes to a spline point on an edge. This issue was resolved by creating a structured surface mesh, and diagonalizing that mesh for the support of tetrahedra. When moving the spline points over this new mesh, there were still sharp changes in the mesh, however they were consistent as seen in figure 5.5(b). This second method was then applied to the surface mesh used for the design cases. This illustrates how the method of discretization of the surface can create discrepancies when dealing with mesh movement.

The second issue found came during the mesh movement operation during the actual design cycle. When moving the points to the new xyz coordinates according to their original uv locations, a failure in this algorithm became apparent. These changes in the surface mesh can be viewed in figure 5.6(a) and 5.6(b). For the first design iteration, the tip of the wing was moved 5.0° and the wing extended $1.17m$. The surface points on the tip were successfully translated on each of the edges and the upper and lower surface. However, the rotation of the points on the tip did not seem to operate as expected. As a quick workaround, the points on the tip of the wing were smoothed using a Linear Elastic smoothing routine and confined to the area between the upper and lower curves on the tip of the wing. This however, is not a permanent solution. The issue was discussed with Robert Haines [20] and the problem is believed to lie in the definition of the CAD model. A more robust solution to this needs to be investigated as a future research topic.

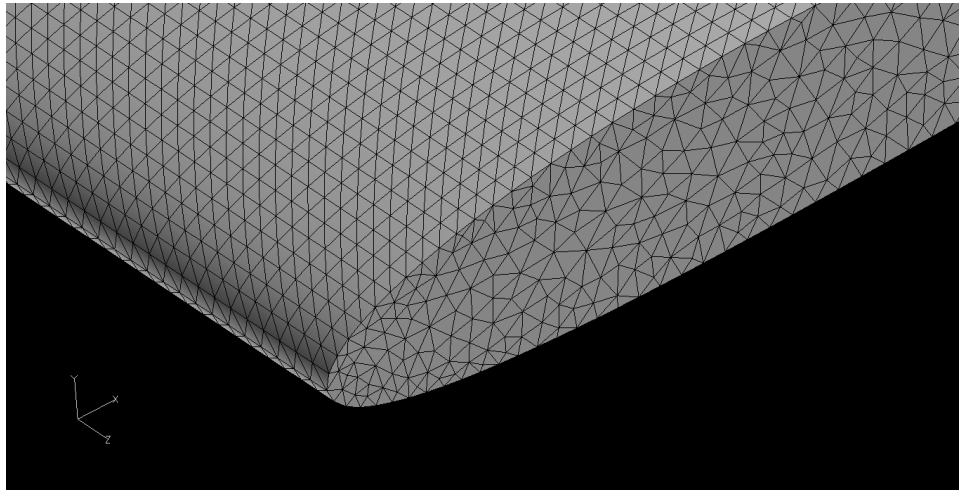


(a) Unstructured Surface Mesh

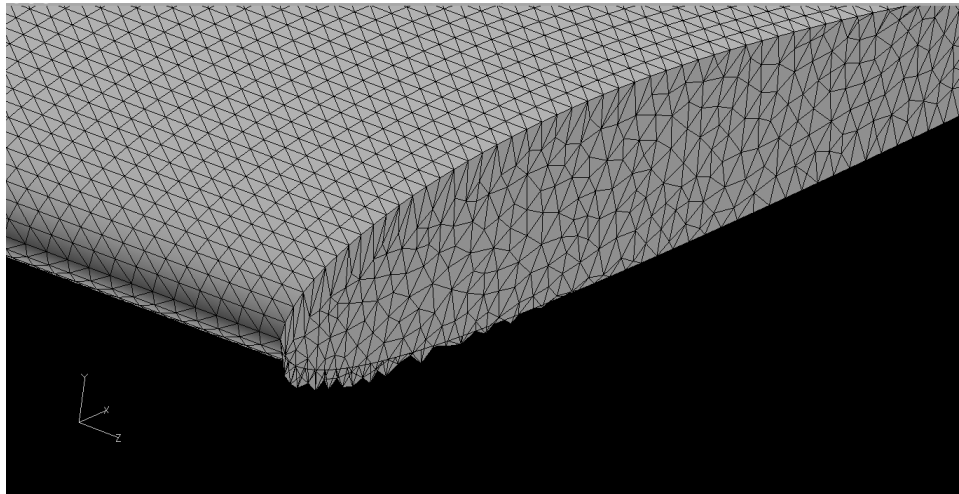


(b) Structured Surface Mesh: Diagonalized

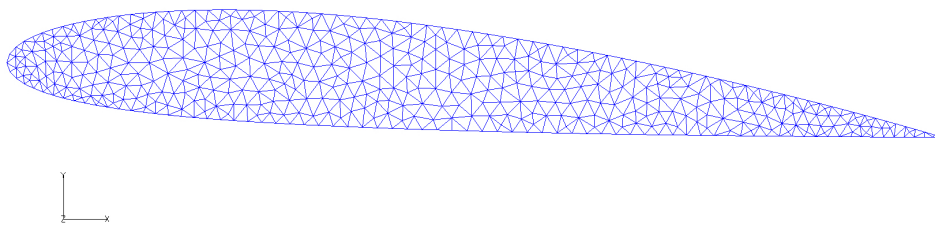
Figure 5.5 Surface Mesh Resolution with Shape Deformation: Spline Point Movement



(a) Surface Mesh Before Rotation



(b) Surface Mesh After Rotation



(c) Fixing Surface Mesh on Tip via Linear Elastic Smoothing

Figure 5.6 Surface Mesh Resolution with Shape Deformation: Tip Rotation

The initial goal of this research was to implement all of the parameters available through CAPRI while performing design optimization. However, when running the optimizer, spline points seemed to be the variables that the optimizer was modifying, not the rotation. So in order to demonstrate the effects of using non-linear relationships for design variables, the case was modified to use only wing length and tip rotation as parameters for this experiment. Further work with spline points will be also be included in future work.

Design Results

The NACA 2412 case was run a through a total of four design cycles with the goal of finding the parameters which would result in $C_L = 1.1$. The parameters used in this case were tip rotation and wing length.

Table 5.1 Design Cycles and the associated changes in the Geometry Parameters

Iteration	Wing Length	Tip Rotation	Objective	Lift Coefficient
0	1.000	0.0°	1.884e-01	0.912
1	1.174	5.0°	3.210e-02	1.132
2	1.083	2.37°	8.689e-02	1.013
3	1.160	4.60°	1.372e-02	1.114
4	1.158	4.54°	1.103e-02	1.111

As shown in table 5.1 the objective function was reduced on nearly every iteration as the Lift Coefficient became closer to the desired value. This suggests that the derivatives obtained from CAPRI were successfully incorporated into a typical design cycle. The following are some images of the changes in the model over the various design cycles. Figure 5.7 shows the change in the model for each cycle: red - initial model, turquoise - cycle 1, green - cycle 2, magenta - cycle 3, and dark blue - cycle 4. Figure 5.8 shows the difference between the initial (red) and final (dark blue) models.

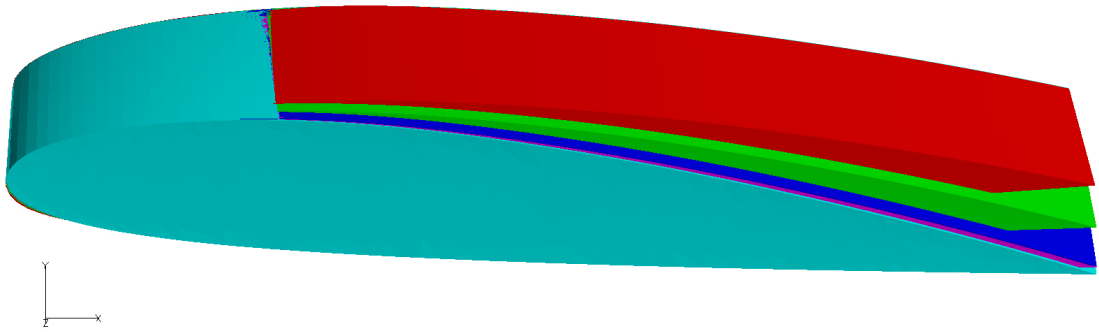


Figure 5.7 Model Movement over the Design Cycles

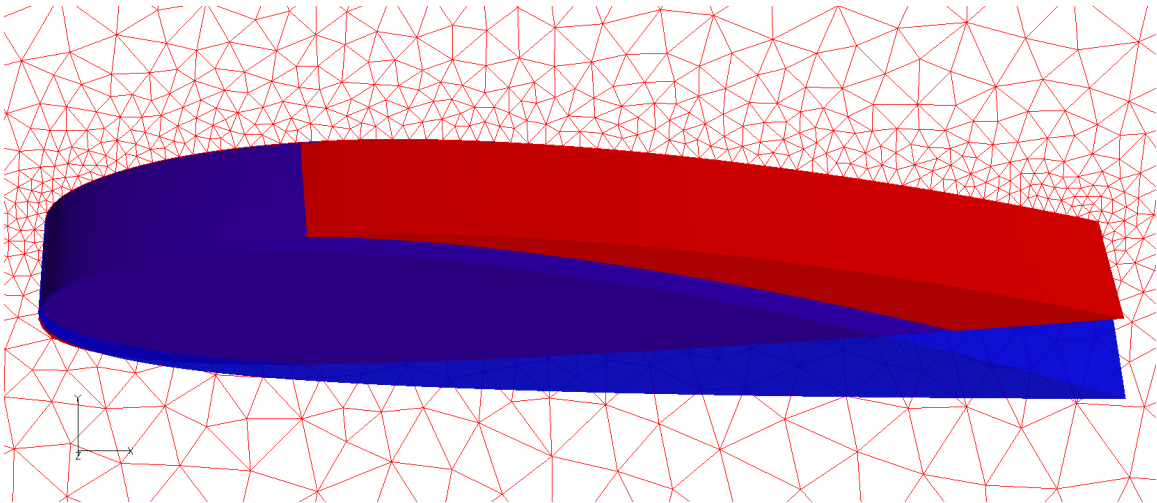
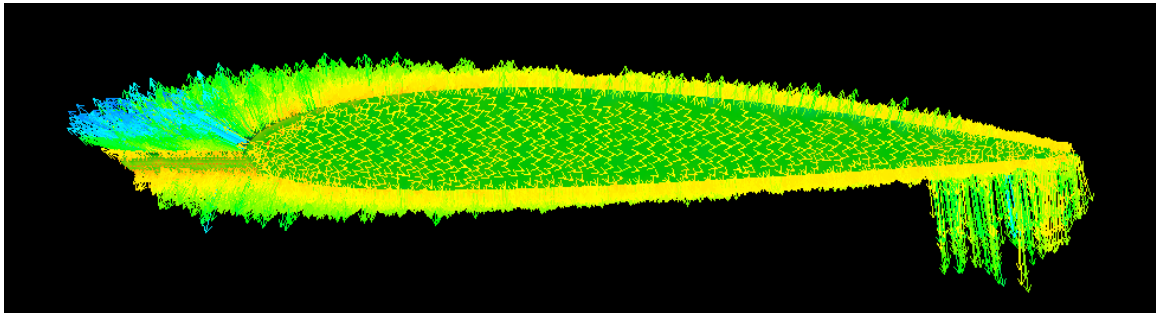
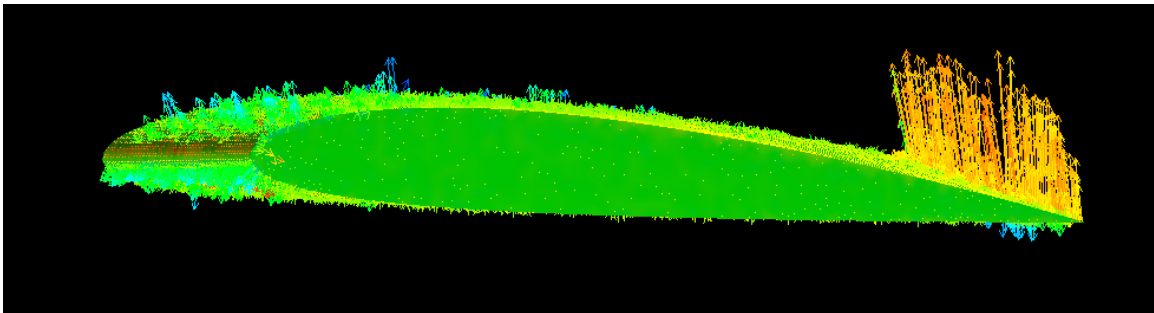


Figure 5.8 Initial and Final Iterations

Figures 5.9(a) and 5.9(b) show the changes in the Surface Gradients from the first to the last iterations. In the first design cycle, the optimizer indicated that points on the top of the wing near the leading edge needed to be shifted vertically to increase the lift, along with a shift in the opposite direction for the points on the lower surface near the trailing edge. The optimizer also indicated that the wing needed to increase in length as denoted by the arrows normal to the tip of the wing. By the final iteration, most of the sensitivity had diminished on the top of the wing where the remaining sensitivities suggested moving points on the top of the tail vertically. It is evident from figure 5.9(b) that the optimizer was satisfied with the location of the majority of the surface points by the fourth design cycle.



(a) Initial Surface Gradients

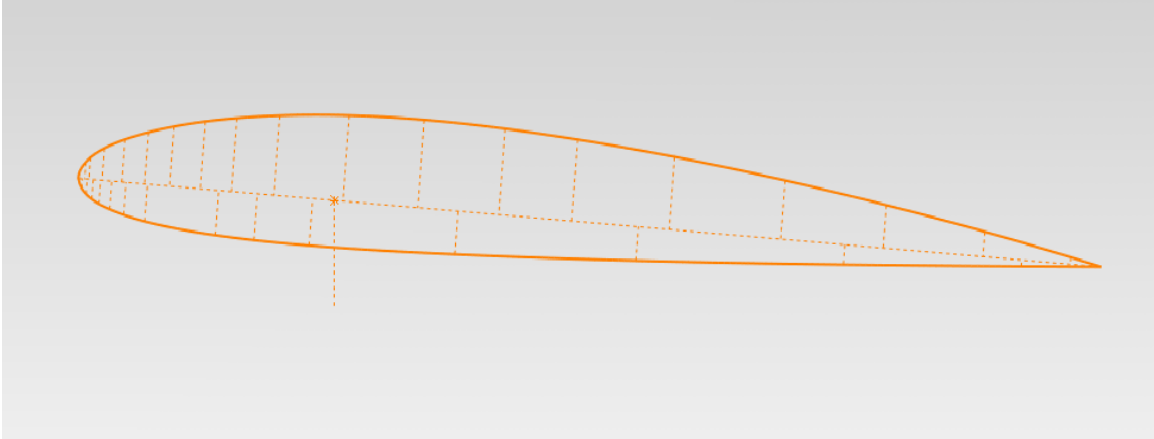


(b) Surface Gradients after four design cycles

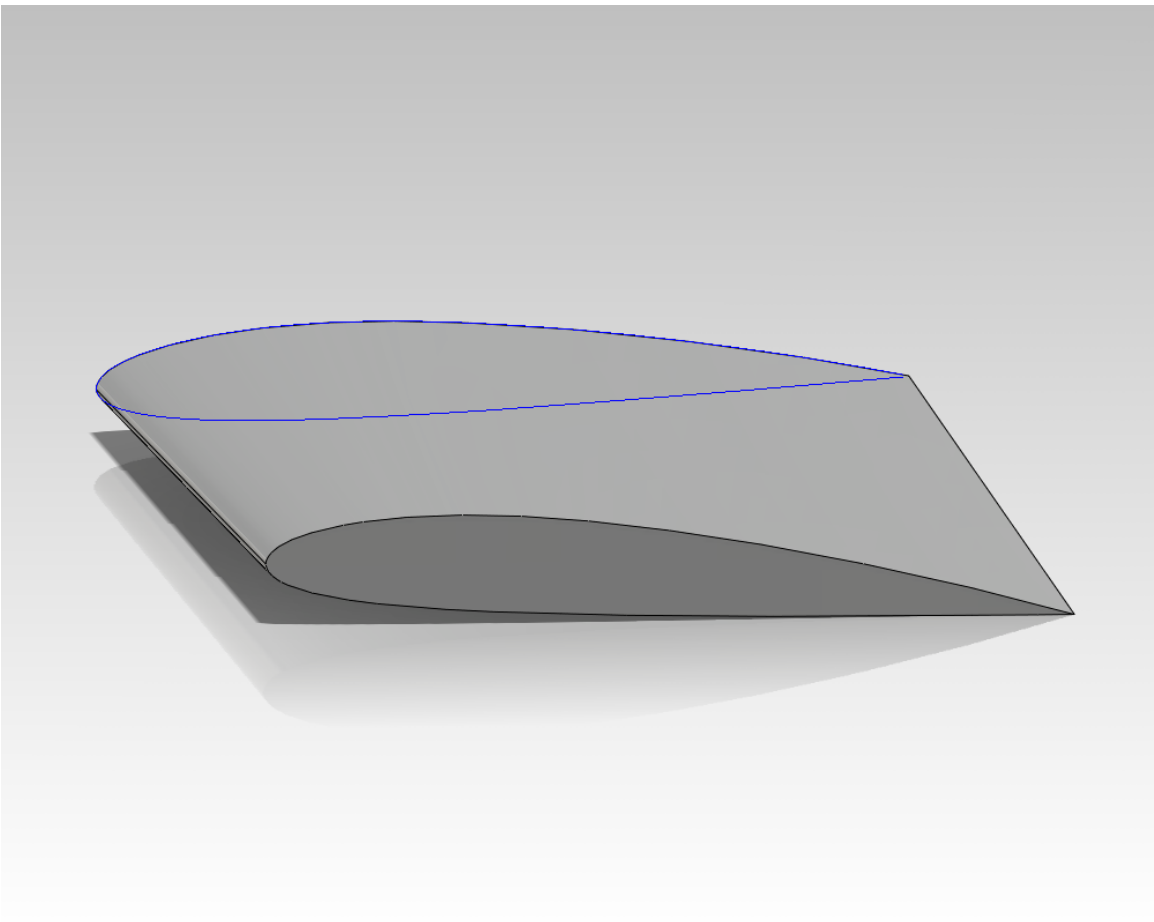
Figure 5.9 Surface Gradients

Final Design

The following images were produced within SolidWorks showing that the final result was indeed a CAD model expressed in a format which is easily used for future analysis. This model was a result of the last iteration of the design cycle.



(a) Final Tip Cross Section



(b) Final Modified NACA 2412 wing

Figure 5.10 CAD Representation of Wing after four design cycles

CHAPTER 6

CONCLUSION

It has been shown that existing functionality within CAPRI has the potential for implementing changes to the underlying CAD geometry during a deterministic design cycle.

This process involves a new method of parameterization for computational design which places new requirements on the CAD engineer. Communication between the design team and the CAD engineers is of the utmost importance, as the CAD engineers lay the framework for what parameters can be adjusted during the design cycle and how they are able to change.

Custom methodologies have been introduced which take the CAD model and a related surface mesh and map those two entities together. These mappings allow for easy use of CAPRI's geometry querying libraries and tools. Additional functionality is supported to obtain design variables from the CAD and relate the changes in each of those variables to every point on the Surface Mesh. Also supported is the updating of a CAD model and the transformation of the associated Surface Mesh as that model changes.

The pairing of existing design software available to the SimCenter and these new technologies involving CAPRI resulted in a successful implementation of a deterministic design cycle. The cycle was iterated over four times and the resulting geometric parameters, the changes in derivatives, and the final CAD model were presented.

Using CAD parameters for design optimization is still in its early stages of development. It is not a substitution for current methods, nor is it superior. However, CAD parameterization and integrating CAD geometry information into the design cycle does have great potential. It offers the user control over the design variables along with a result that is ready

for continued analysis and manufacturing. With continued development this methodology has significant potential to be a valuable resource for computational design.

Future Work

It is suggested that future work include the following:

- Further optimization of the Quilt building and Edge mapping routines.
- Additional testing of this software's capability to handle more complex geometries.
- More generalized surface mesh mapping routines (less assumptions).
- Using the CAPRI methodology to support the use of higher order elements
- Determine the inconsistencies between the surface mesh movement near the tip of the 2412 case.

REFERENCES

- [1] Haimes, R., “CAPRI CAE Gateway,” April 2007. iv, 1, 6, 13, 17
- [2] SoidWorks, “3D CAD Design Software,” <http://www.solidworks.com/>. iv, 13, 32
- [3] “UIUC Airfoil Coordinates Database,” 2011, http://www.ae.illinois.edu/m-selig/ads/coord_database.html. ix, 32, 33
- [4] Anderson, W. K., Karman, S. L., and Burdyshaw, C., “Geometry Parameterization Method for Multidisciplinary Applications,” *AIAA*, Vol. 47, No. 6, June 2009. 5
- [5] PointWise, “Mesh and Grid Generation Software,” <http://www.pointwise.com/>. 7, 32
- [6] Bell Labs, “PORT Mathematical Subroutine Library,” <http://www.bell-labs.com/project/PORT/>. 9
- [7] Wrenn, G., “An indirect method for numerical optimization using the Kreisselmeier–Steinhauser function,” Tech. rep., Contractor Report NASA CR-4220, NASA Langley Research Center, Hampton, VA., 1989. 9
- [8] Meza, J. C., Oliva, R. A., Hough, P. D., and Williams, P. J., “OPT++: An object-oriented toolkit for nonlinear optimization,” June 2007, <https://software.sandia.gov/opt++/>. 9
- [9] Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design*, VRnD, Colorado Springs, CO, 3rd ed., 1999. 9
- [10] Sreenivas, K., Hyams, D. G., Nichols, D. S., Mitchell, B., Taylor, L. K., Briley, W. R., and Whitfield, D., “Development of an Unstructured Parallel Flow Solver for Arbitrary Mach Numbers,” *AIAA 2005-0325*, Vol. 43, January 2005. 10, 11
- [11] Burdyshaw, C. E., *Achieving Automatic Concurrency Between Computational Field Solvers and Adjoint Sensitivity Codes*, Ph.D. thesis, University of Tennessee at Chattanooga, May 2006. 11
- [12] Karman, S., “Unstructured Viscous Layer Insertion Using Linear-Elastic Smoothing,” *AIAA*, Vol. 45, No. 6, June 2006. 11
- [13] “PTC - Pro/Engineer,” 2011, <http://www.ptc.com/products/creo/parametric>. 13
- [14] “Catia - Virtual Product,” <http://www.3ds.com/products/catia>. 13
- [15] “Open CASCADE Technology,” <http://www.opencascade.org/>. 13

- [16] “NX: Siemens PLM Software,” 2011, http://www.plm.automation.siemens.com/en_us/products/nx/index.shtml. 13
- [17] “Parasolid: 3D Geometric Modeling Engine,” 2011, http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml. 13
- [18] Dannenhoffer, J. F. and Haines, R., “Quilts: A Technique for Improving Boundary Representations for CFD,” *AIAA*, Vol. 4131, June 2003, pp. 1–9. 16
- [19] Haines, R., Jones, W. T., and Lazarra, D., “Evolution of Geometric Sensitivity Derivatives from Computer Aided Design Models,” *AIAA*, Vol. 2010-816099, September 2010. 20
- [20] Haines, R., May 2011, Robert Haines was in direct correspondence throughout the course of this research from May through August 2011. 20, 38

VITA

William Emeson Brock V was born in Chattanooga, TN on May 16, 1983. He is the son of Bill and Maureen Brock, and the elder brother to Stephen and Michael Brock. He graduated from Red Bank High School, class of 2001. William earned a Bachelor's of Science in Applied Mathematics and a Minor in Computer Science in December of 2008 from the University of Tennessee at Chattanooga.