Development of a Computer-Aided-Design-Based Geometry and Mesh Movement Algorithm for 3D Aerodynamic Shape Optimization

by

Anh Hoang Truong

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Copyright © 2014 by Anh Hoang Truong

Abstract

Development of a Computer-Aided-Design-Based Geometry and Mesh Movement Algorithm for 3D Aerodynamic Shape Optimization

Anh Hoang Truong
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2014

This thesis focuses on the development of a Computer-Aided-Design (CAD)-based geometry parameterization method and a corresponding surface mesh movement algorithm suitable for three-dimensional aerodynamic shape optimization. The geometry parameterization method includes a geometry control tool to aid in the construction and manipulation of a CAD geometry through a vendor-neutral application interface, CAPRI. It automates the tedious part of the construction phase involving data entry and provides intuitive and effective design variables that allow for both the flexibility and the precision required to control the movement of the geometry. The surface mesh movement algorithm, on the other hand, transforms an initial structured surface mesh to fit the new geometry using a discrete representation of the new CAD surface provided by CAPRI. Using a unique mapping procedure, the algorithm not only preserves the characteristics of the original surface mesh, but also guarantees that the new mesh points are on the CAD geometry. The new surface mesh is then smoothed in the parametric space before it is transformed back into three-dimensional space. The procedure is efficient in that all the processing is done in the parametric space, incurring minimal computational cost. The geometry parameterization and mesh movement tools are integrated into a three-dimensional shape optimization framework, with a linear-elasticity volume-grid movement algorithm, a Newton-Krylov flow solver for the Euler equations, and a gradient-based optimizer. The validity and accuracy of the CAD-based optimization algorithm are demonstrated through a number of verification and optimization cases.

Acknowledgements

I would like to thank Professor David Zingg for giving me this research opportunity and for the support and guidance he has given me through all these years. I'm especially grateful for his encouragement and understanding during the difficult times in my research. I feel extremely fortunate to have such a sympathetic, knowledgeable and inspirational supervisor.

I am also very thankful to Robert Haimes for providing me with not only a powerful tool, CAPRI, that enabled me to do my research, but also the timely support that I needed throughout my degree. Thanks to Daniel Fudge for his work with CAPRI and CATIA which set a firm foundation from which I built mine.

I am grateful to the doctoral committee members: Professors Joaquim Martins, Jorn Hansen, Clinton Groth and Craig Steeves for their advice and suggestions which improved the quality of my thesis; to Professor James Gottlieb for his encouragement and confidence in my abilities; to the system administrators in the CFD lab, at SciNet and Guillimin for their help in getting my cases running, particularly Scott Northrup for his diligence in keeping all the software and hardware up-to-date and running smoothly at all times such that I did not experience interruptions on his watch.

I wish to give grateful acknowledgement to the UTIAS community of professors, staff, and students for creating such a pleasant research environment; to all my friends and colleages in the CFD and MDO labs, especially Graeme Kennedy, Gaetan Kenway, Sandy Mader, Tim Leung, Lucian Ivan, Jason Hicken, Rhea Liem, Jenmy Zhang, Chad Oldfield, Tom Reist and Shahriar Khosravi for helping me so much technically and emotionally. Everyone have always been so kind and eager to help and share their expertise without reservations. Thank you for your wisdom and friendship.

I feel incredibly thankful and blessed to have my sons Nicholas and Benjamin along the way, and to have the love and support of my family. They have always been there for me, encouraging and helping in whatever way they can so that I can focus on my thesis. My husband, Iordan Iordanov, was instrumental in helping me to set up the computer system so that I can work remotely. I'm very grateful to have benefited from his expertise in computer science.

I am very grateful for having received financial support from the University of Toronto, the Government of Ontario, Bombardier Aerospace, MITACS, and Zonta International. Their funding of my research is very much appreciated.

Computational resources for performing all of the calculations reported herein were provided by the SciNet High Performance Computing Consortium at the University of Toronto and the Guillimin High Performance Computing Facility at McGill University, both are part of Compute Canada through funding from the Canada Foundation for Innovation (CFI) and the Province of Ontario, Canada.

Anh Truong
University of Toronto Institute for Aerospace Studies
February 17, 2014

Contents

List of Tables

List of Figures

Nomenclature

Alphanumeric

angle	Maximum angle between triangle neighbours in a tessellation, in degrees
$\mathbf{A}$	Matrix for a general linear system of equations
a	Translation of a coordinate system in the x direction
A, B, C	Coefficients in the general equation of a line
b	Translation of a coordinate system in the y direction
C_D	Drag coefficient
C_L	Lift coefficient
$\mathcal{X}$	Design variables
deviat	Maximum deviation between the parametric center of a triangle in a tessellation and the CAD surface, in model units
D	Spring constant
E	Spring energy
f	Winslow smoothing function for either x or y coordinate direction
G	Grid variables
$\mathcal{J}$	Objective function
L	Length of a segment between two nodes
$\mathcal{L}$	Lagragian
mxslen	Maximum length of a side of a triangle in a tessellation, in model units

M	Mach number
N_{surf}	Total number of surface nodes
p	Surface pressure
Q	Flow field variables
R^n	Region in space of dimension n
r	Residual
$\mathcal{C}$	Constraints
$\mathcal{R}$	Flow residual
S	Wetted planform area of the wing
s	Scale change between coordinate systems
u, v	Coordinates in parametric space
V_n	Design velocity in the normal direction to the boundary
$\mathbf{X}, \mathbf{Y}$	Vectors of unknowns and solution of a linear system
x, y, z	Coordinates in physical space
X, Y	Transformed coordinates in physical space

Abbreviations

ABF	Angle-based flattening
Adj	Adjoint
AIAA	American Institute of Aeronautics and Astronautics
API	Application Programming Interface
BREP	Boundary Representation
CAD	Computer Aided Design
CATIA	Computer Aided Three-dimensional Interactive Application
CAA	Component Application Architecture
CAPRI	Computational Analysis PRogramming Interface
CFD	Computational Fluid Dynamics
CGM	Common Geometry Module

DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V, German Aerospace Centre
EGADS	Engineering Geometry Aerospace Design System
FD	Finite-difference
FFD	Free-form Deformation
GGS	Geometry Generation Subroutine
Ganimede	Geometry ANd Inherent MEsh DEformation
IDEAS	Integrated Design and Engineering Analysis Software
IGES	Initial Graphics Exchange Specification
LE	Leading edge
NIST	National Institute for Standards and Technology
NURBS	Non-Uniform Rational B-Splines
OMG	Object Management Group
OpenCASCADE	Open-source Computer Aided Software for Computer Aided Design and Engineering
OpenCSM	Open-source Constructive Solid Modeler
MDO	Multidisciplinary Design Optimization
SDRC	Structural Dynamics Research Corporation
SNOPT	Sparse Nonlinear OPTimizer
STEP	Standard for the Exchange of Product Model Data
vte	Virtual topology editor
VDAFS	Verband des Automobilindustrie-Flächenschnittstelle, Automotive Industry Association - Surface Data Interface

Greek

α, β, γ	Baricentric coordinate, also used as coefficients
ΔA	Surface area of an element
ϵ	Finite-difference step size
ε	Nonorthorgonality angle between the axes of the two coordinate systems
ξ, η	Coordinates in computational space

Ω	Feasible region of the design space
κ	Coefficient
ψ, λ	Lagrangian multipliers for the flow and grid variables
φ	Rotation of the axes of one coordinate system with respect to the other
ω	Angle of attack
τ	Linear twist
θ	Rotation angle

Subscripts

0	Origin
i, j	Node indices
$surf$	Surface

Superscripts

T	Transpose

aut viam inveniam aut faciam.

1 Introduction

1.1 Motivation

The increased awareness of and concern for the environment have prompted many researchers to seek superior, unconventional aircraft designs to minimize environmental impact, resulting in extensive developments in the field of aerodynamic shape optimization in recent years [7, 139]. Advanced aircraft design is characterized by multiobjective, multidisciplinary requirements [122, 136]. Optimization techniques probe the aerodynamic, flight mechanical and structural design sensitivities for an optimal vehicle-system. Aerodynamic shape optimization involves four distinct components: 1) geometry parameterization and associated surface grid generation; 2) volume grid generation or deformation; 3) computational fluid dynamics (CFD); and 4) numerical optimization algorithm. The geometry parameterization technique defines the design variables that can be used by the optimizer to modify the geometry during the optimization process. When the geometry changes, the volume mesh also has to be regenerated or deformed to fit the new geometry. If the volume mesh deformation approach is used, a surface mesh that is topologically equivalent to the original surface mesh has to be generated to drive the volume mesh deformation algorithm. The process of obtaining a surface mesh is relatively straightforward if the mathematical relationship between the design variables and the surface mesh is known. For example, in the integrated geometry and mesh movement approach of Hicken and Zingg [64], the surface mesh is related to a B-spline patch through a polynomial. The design variables are the positions of the B-spline control points on the patch. When the control points are moved, the surface mesh can be calculated from the known mathematical relationship. However, this process is more complicated if

the mathematical definition of the geometry surface is unknown, such as in the case of Computer-Aided-Design (CAD) geometries. The flow solver is used to compute aerodynamic quantities such as lift and drag which can be defined as the objective or constraint of the optimization problem, (e.g. lift-constrained drag-minimization, lift-to-drag maximization). Finally, the numerical optimization algorithm controls the solution of the optimization problem. It searches the design space to find a set of design variables at which the objective is minimized and the constraint, if it is defined, is satisfied.

Although the necessary tools for three-dimensional (3D) aerodynamic shape optimization are currently available, they are quite limited in terms of geometry handling, lacking in the ability to manipulate and enforce constraints for complex three-dimensional geometries. Geometry is a common data set that must be shared and manipulated among various disciplines to maintain the accuracy and consistency of the model during the optimization process [101, 134, 137, 164]. Sharing the same geometry is important during product design and manufacturing, as it enhances communication among different engineering disciplines, improves quality by allowing the natural coupling among the disciplines, and compresses the overall design process timetable [164]. It has been demonstrated mathematically that the use of sequential discipline optimization technique may lead to suboptimal design [101, 164]. As stated in [164], in the traditional "waterfall" design approach, "as the design process goes forward, designers gain knowledge but lose freedom to act on that knowledge." Collaboration among the experts in each field can be achieved by improving accessibility of models, data and tools, which in turn will improve the reliability of the predictions of the properties of the product, which is especially critical in the early design phases [86]. Such a strategy is especially useful in the field of aircraft design where even small improvements in the lift/drag ratio can result in remarkable reductions of the direct operating costs [151]. Robust geometry modeling and grid deformation tools could significantly reduce the design cycle time and cost by allowing the optimization process to be automated [164].

An ideal shape parametrization is one which can provide detailed definition of a surface model with only few simple parameters and at the same time the flexibility in order to achieve truly optimal designs [112]. A comprehensive survey of shape parametrization techniques can be found in [135], with more recent developments discussed in [45]. Among the techniques available, two categories of methods are suitable for use for multidisciplinary design optimization (MDO), i.e. have consistent geometry representation across different engineering disciplines: the CAD-based approaches and free-form deformation

(FFD) methods. The FFD algorithm [44, 85, 136] relates the grid point coordinates of an analysis model to a number of design variables or control lattice that is constructed around the shape. It treats the model as rubber that can be twisted, bent, tapered, compressed, or expanded, while retaining its topology. This is ideal for parameterizing airplane models that have both external skin and internal components. However, the design variables may have no physical significance for the designer, which makes it difficult to select an effective and compact set of design variables [5, 135]. Second, it may be difficult to preserve geometric features if the design variables are not properly constrained [70]. It is challenging and time-consuming to properly enforce meaningful geometric constraints such that the movement of the lattice follows the design intent [4]. Third, the geometry's movement is confined within the volume formed by the lattice. Finally, since FFD is only a geometry deformation tool, optimized geometries must be converted back to a geometry modeling tool (usually CAD) for manufacturing, a process which can be very time-consuming.

Modern CAD capabilities allow for parametrized aircraft models that can include all typical features of a design and are thus ideal for optimization [66, 136]. CAD systems provide a comprehensive set of geometric functionalities to build the model, and the parameters that are used in the creation of the model are exposed and can be modified during all stages of design [22, 78, 124]. However, one serious drawback of using CAD systems is that they do not provide analytical sensitivity derivatives which are required by a gradient based optimizer. These derivatives have to be obtained by differentiating the CAD modeler, which is not possible for commercial CAD systems since they are proprietary and therefore the source code is not available for differentiation. This is a major issue and is discussed further in Section 5.4. Nevertheless, CAD is an essential tool for modelling and an important parameterization technique for aerodynamic shape optimization. It allows for a geometry-centric optimization process whereby different engineering disciplines (aerodynamic, structures, etc.) use the same common geometry, resulting in a consistent and indeed optimal design.

1.2 CAD Parameterization

Geometry objects are usually created in a CAD environment, mainly because CAD systems have robust modeling capability and are able to capture the design intent. They allow designers to create and modify shapes effortlessly in 3D simply by altering the

dimensions of the features from which they were created. Solid modeling helps to avoid design errors. It allows the designers to better understand how the product will look and function before physical prototypes are made, and naturally became the tool to use for product design and development. The models serve as the basis for analysis, shape optimization, and ultimately manufacturing. Ideally, the CAD representation of the geometry should be used directly to create computational grids suitable for numerical simulations. With more advanced applications involving mesh adaptation, tessellations for immersive environments in virtual reality, geometry based interfaces for fluid-structure coupling in aero-elasticity, and multidisciplinary design optimization, the CAD model needs to be used throughout the process to ensure high fidelity to the true CAD surface [104]. However, CAD systems are proprietary and, traditionally, the CAD models have to be translated to some standard format in order to be used by downstream applications. The translation process removes all topological and connectivity information, and the model has to be extensively repaired. In the worst case, when the process fails, a significant amount of time is spent in looking for problems and fixing the CAD geometry. Sometimes the problems are hidden and the converted geometry is slightly altered but it is not evident. Moreover, the translated version frequently does not include important topological and construction information along with the geometry entity. These are known as interoperability issues and are estimated to cost one billion US$ each year in the US automotive sector alone, according to the US National Institute for Standards and Technology (NIST) [46].

The most critical problems in data exchange are the different internal mathematical representation schemes and internal accuracy of the geometric definitions in the modeling kernel of the various CAD systems. In particular, the problems arise from the accuracy and the convergence criteria used when performing calculations with curves and surfaces. All this can occur either within the original system or during the pre- or post-processing phases of CAD data, performed with neutral formats like IGES [81] or STEP [72, 140]. File translation, inconsistent geometry engines, and non-native point construction are all identified as sources of non-robustness [1]. The task of "geometry repair, clean-up, and preparation" on CAD models, prior to meshing, is a major bottleneck in the use of Computational Fluid Dynamic as a practical design tool. This laborious, manual-intervention process typically consumes up to 80% of the total time required for CFD analyses [32]. In 1991, the AIAA technical committee on Multidisciplinary Design Optimization (MDO) identified that one of the key tasks that need development is to establish unified numerical

modeling parmeterized in terms of the design variables – a consistent vehicle geometry to be used as a basis for all mathematical models, and changes to the geometry must be centrally coordinated [164]. To address this problem, most CAD systems began introducing toolkits such as ACIS, Parasolids, the SolidWorks API, CATIA CAA and Pro/Toolkit [9, 156] for accessing geometric data in the CAD system (see Section 2.2). This marks a major milestone as CAD entities are now accessible and modifiable and thus directly usable in shape optimization algorithms, although the applicability thus far is limited for two important reasons. First, CAD software is very complex and users have no control on the mathematical definition or the number of entities generated by a CAD program. This means that the designer cannot relate the geometry definition to the body-fitted computational mesh, particularly the surface mesh, and is limited to optimization problems in which the geometry can undergo only relatively small movements [3, 12, 166]. In these cases, the geometry definition does not change and the movement of the surface mesh points can be "tracked" by their CAD parametric coordinates [3, 12], or regenerated entirely based on the initial topology [166]. Alternatetively, Nemec *et al.* [113] used Cartesian meshes so that mesh movement or regeneration can be avoided, but they are limited to optimization problems where the flow is inviscid. Second, CAD systems do not provide derivatives of surface displacements with respect to the design variables, which are needed in the chain rule to compute the sensitivities of the objective/constraint function with respect to design parameters. Section 5.3 presents a survey of the methods for calculating the surface sensitivity derivatives. All of the methods have varying degree of robustness and/or efficiency issues. However, the most simple and general option is to apply finite differences to the CAD system.

1.3 Objective

The objective of this thesis is to develop a 3D CAD-based shape optimization tool that is unique in its ability to manipulate and control a geometry definition through CAPRI, a CAD vendor-neutral Application Programing Interface, and its ability to morph the original volume mesh to fit the new geometry as it evolves during an optimization cycle. The optimization tool consists of the following components:

1. Geometry parameterization technique - CATIA v5 is used in conjunction with CAPRI to define the design variables and to obtain updated instances of the original

geometry definition as well as surface meshes to drive the volume mesh movement algorithm.

2. Volume mesh movement algorithm - based on the linear elasticity method developed by Truong [159] is used to deform the original volume mesh to fit the modified geometry during the optimization process.

3. Flow solver - a parallel Newton-Krylov algorithm for the Euler equations, Diablo, developed by Hicken and Zingg [62], is used to compute the objectives and constraints.

4. Numerical optimization method - the Sparse Nonlinear OPTimizer (SNOPT) [48], which considers the value as well as gradient information of the objectives and constraints, is used to search the design space for the optimal solution.

The tool must have capabilities to evaluate, analyze, and deform the geometry from CATIA as well as deform the original surface mesh to fit the new geometry. It has to

1. be able to connect to the analysis and CATIA geometry module and access information in its native format to ensure fidelity in the surface mesh points and the CAD geometry;

2. provide a suitable set of functions required for the interaction with the CAD system such as projection from parametric to physical space and vice versa;

3. maintain associative information between the CAD entity and the mesh points during all stages of design;

4. use design variables provided by the optimizer to generate the modified geometry and the corresponding surface mesh. The design variables can be nodal or global. Nodal variables are control points defining a set of surface patches. Global variables translate and rotate the support plane and scale the profile of the model sketch [42]. They are used in designing the wing planform, for example, where a more intuitive engineering specification of the design such as the taper, skew, and twist of the wing, is desired.

5. provide accurate sensitivity derivatives of the surface mesh with respect to the design variables.

To achieve this goal, we introduce a clustering algorithm to control the topology of the CAD geometry, a mapping procedure to obtain a new surface mesh for the modified geometry that preserves the characteristics of the original surface mesh, and a procedure to obtain smooth and accurate surface sensitivity data for the optimizer. The strength of the combined algorithm lies in its ability to maintain, modify, and precisely control both the CAD definition and the original volume mesh, avoiding the need to regenerate a new mesh and interpolate the solution from the previous mesh, which may lead to increased overall computational time to converge the solution for each design cycle [77]. The tool will allow the aerodynamic shape optimization process to be automated, requiring no user intervention. It begins and ends with a CAD geometry representation that is guaranteed to be manufacturable (i.e. the optimal geometry satisfies all the manufacturing constraints that were included during the creation of the original CAD model).

1.4 Thesis Outline

This thesis is divided into 7 chapters and is organized as follows: Chapter 2 discusses how a CAD model is represented in CAD systems and different methods for accessing the model. It describes how to create a parametric CAD model and how to control the topology changes during optimization. Chapter 3 explains the procedure to obtain a new surface mesh for each design change by reparametrization. It first gives an overview and evaluation of the available parametrization techniques followed by implementation details. Chapter 4 defines an approach for improving the surface mesh, starting with an examination of the available smoothing techniques. Chapter 5 then discusses how to obtain surface sensitivity for gradient-based optimization. It begins with the introduction to the aerodynamic optimization process and explains why surface sensitivity derivatives are required. It provides a survey of current methods for determining the surface sensitivity derivatives and gives details of the current implementation. The above procedure for obtaining the surface mesh and gradient information is integrated with the CFD solver Diablo [64], the parallelized version of the volume grid mover of Truong [158], and the optimization software SNOPT [48]. The optimization results are presented in Chapter 6. Finally, Chapter 7 concludes with a summary of the original contributions. It gives an assessment of the developed algorithm and recommendations for future work.

2

CAD Geometry

2.1 CAD Models

Model representation has evolved in most CAD systems towards the Master-Model concept, which includes feature-based data and a boundary representation, or BREP, model. The feature-based data, also called a "Feature Tree", contains a list of sequential topological operations such as extrude, revolve, merge, subtract and intersection operations, performed to build the geometry of the solid model [58]. According to this scheme, a geometric object is composed of entities of dimension 0, 1, 2, or 3 corresponding to vertices, edges, faces, and volumes, each with its own ID string. Each entity (except a vertex) is described by its boundaries, which are lower-dimensional entities. A *mapping function* is used to embed an entity from one dimension into a higher dimension. For example, each edge of a face is accompanied by a function from an interval of R^1 to R^2 that gives the position of the edge in the parametric domain of the face. This face, with an accompanying mapping function from a region of R^2 to R^3, can then appear as the boundary of a volume. Since different faces are likely to share an edge as a boundary, it is required that both mappings are equal up to a tolerance value. Therefore, each entity also stores a tolerance to indicate how much error to expect when comparing two different embeddings [50]. Finally, the individual volumes are assembled using boolean constructs that allow the user to combine different components to create a solid model definition of the geometry [77].

A BREP model consists of geometry, topology, and tolerances, and the methods to evaluate them. The discrepancy between different CAD systems arises as a result of a lack of agreement as to what constitutes a valid BREP model [9]. CAD systems often use

relatively large variable tolerancing to provide robustness to model operations, resulting in gaps and overlaps in geometry and topology of the CAD system BREP model, which are apparent when translated to other modeling engines or mesh generation software, but are non-existent in their native environment since it has algorithms that are written specifically to deal with these tolerances [9]. Pro/ENGINEER [126], for example, uses relative tolerance by default, referring to the fact that the acceptable gap between pieces of geometry is based on the relative size of the geometry. It has a value of 0.0012. On the other hand, the default positional tolerance in CATIA (Computer Aided Three-dimensional Interactive Application) V5 [15] is a dimensional quantity given by 0.001 mm.

2.2 Accessing CAD Models

There are four possible strategies available for CAD geometry access:

1. Translation
2. Discrete Representation
3. Direct Geometry Access
4. Uniform Direct Interface

The first method has been the most commonly used. It involves the use of standard file format or direct translators. The IGES (Initial Graphics Exchange Specification) [129] format only contains geometry as a collection of disjoint and unconnected surfaces and curves, with no notion of topology [9, 58]. Because 3D meshing tools require a closed geometry, a great deal of time is expended in "healing" the model. This process usually introduces "sliver" surfaces to resolve gaps, overlaps, and tangency conditions in order to maintain the validity and integrity of the model, resulting in a large number of added faces that can cause problems to mesh generation algorithms [13]. The VDAFS (Verband des Automobilindustrie) [162] and STEP (Standard for the Exchange of Product Model Data) [72] standards do address the topology and global tolerances, but do not address issues related to characteristic features of the model and local tolerancing. Hence, their use also results in a "dirty" geometry, although typically cleaner than IGES [9].

The second method is based on the generation of a faceted model, or subdivision of surfaces, or higher order triangular patches by the CAD system and accessing the resulting discretized surfaces for mesh generation, provided that this functionality exists in the original CAD system. These facet representations are often done on a face-by-face basis and may be incompatible across face boundaries. Unigraphics [161], Parasolid [123], CATIA generate a triangulation that is guaranteed to be closed. Pro/Engineer, on the other hand, requires the extra feature called Pro/MESH to generate a closed triangulation. Finally, SDRC's Open I-DEAS API [71] does not support this feature at all [9, 23, 57, 91]. Moreover, the resulting representation is only an approximation of the true CAD model. Further processing of the tessellation may be required to refine the triangulation to a state suitable for meshing and computational analysis [58].

The third method is more robust in that accessing CAD geometry and topology is done in its native environment, through CAD system toolkits such as ACIS, Parasolids, the SolidWorks [152] API, CATIA CAA and Pro/Toolkit [9, 156]. Queries on geometric entities are made directly from the CAD geometry kernel using an Application Programming Interface (API), avoiding the need to infer topological information from imprecise data and the loss of solid geometry information in a translation [27]. Having direct access to curves and surfaces ensures that the points placed on these entities match the part being meshed. Some widely used commercial software, such as Ansys, MARC, and Patran, is designed to run from within the CAD environment. However, one is then limited to using the analysis codes native to the CAD/Analysis interfaces, and the functionalities available from within the CAD system. This also limits the ability to support evolving geometry by the CAD software available [9, 58]. A natural extension of this method is the Uniform Direct Interface approach. This approach is a modeler-independent interface to the original CAD system data. It provides a separate topology data structure that allows for multiple forms of defeaturing while retaining the original geometry and topology [9]. It is capable of accessing any CAD system in its native environment, and at the same time, maintaining the associative information between the mesh points and the geometry. OMG's CAD Services, CADScript, CGM (Common Geometry Module) and CAPRI (Computational Analysis PRogramming Interface) are examples of APIs that use this approach [58, 104, 156]. Through the vendor-neutral API, queries to the geometry can be made from within all the analysis sub-modules such as grid generators for meshing, flow solvers for grid adaptation, and post-processors for visualization. CAPRI [56, 58], one of the interfaces developed that aids in acquiring geometry data directly from CAD

files, is used in the present research. CATIA V5 is selected as the modeling engine due to its prominence in the aerospace industry.

2.3 CAPRI

2.3.1 Solid Model

CAPRI builds on the Master-Model concept with the Feature Tree presented to the programmer in the form of "branches". Each of the geometric entities in the solid model has an index to identify its relative position in the tree. A branch can be marked "suppressed" for defeaturing purposes, so that unwanted features, such as fasteners and fillets, which are too small for a fluid flow calculation, will not be expressed and can be excluded from the analysis [58]. CAPRI makes available all of the adjustable parameters so that a new member of the family can be built by following the prescription outlined in the Master-Model definition. For shape optimization, CAPRI also exposes definition parameters of curves and surfaces (such as knot points) obtained from independent sketched features in the model that are later used in solid generation as the basis for rotation, extrusion, blending and lofting. This is critical in aerodynamic shape design, for example, where the curve/surface specifications of the airfoil shape of the wing and tail component are required to be defined at a very detailed level [57, 58].

In order to manipulate the CAD model effectively, it should be constructed in such a way as to allow later suppression and modifications to the original model as the geometry evolves. For example, a simple box with filleted edge should be created by an extrusion of a rectangle, followed by another feature defining the fillets on the edges instead of the extrusion of a rectangle with filleted corners. This will allow the suppression of the fillets in the analysis, whereas the latter approach will make it impossible to later suppress the fillets [58].

CAPRI supplies the geometric information in a data hierarchy of node, edge, face, boundary, and volume, with the nodes being the lowest dimensional entity and corresponding to points in three-dimensional space (Figure 2.1). An edge is a general curve in space that begins and ends at distinct nodes, and thus a closed curve (i.e. a circle) must be formed by two or more edges. Each edge is connected to two faces. Faces are bounded by a closed set of edges organized into loops and oriented such that the face lies inside them when they are arranged in a counterclockwise direction when viewed from a point

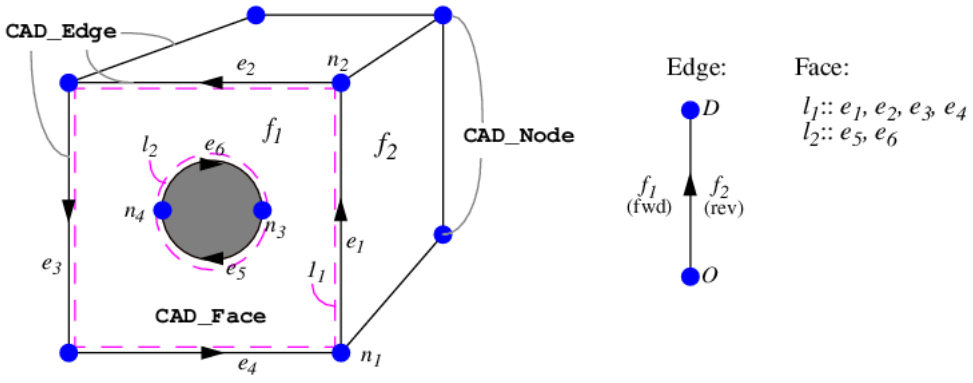


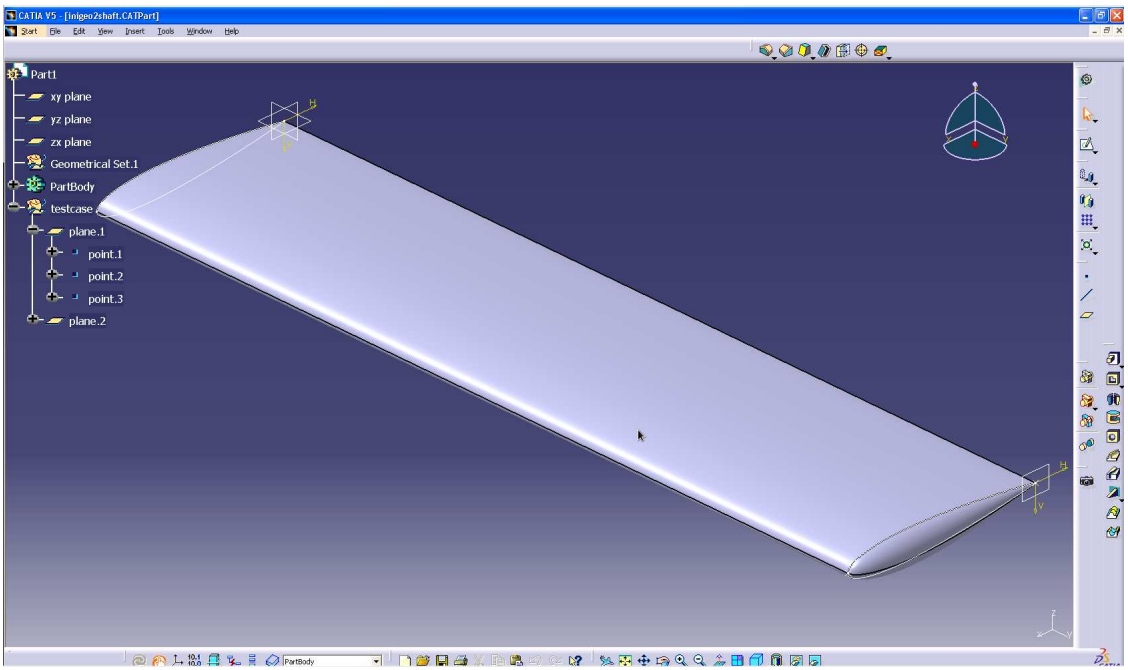
Figure 2.1: CAPRI geometry definition [56]

outside the solid volume. Following this convention, holes in the surface are described by clockwise loops. Both edges and faces are curved entities since they follow the underlying parametrization of the geometry. A collection of faces that have special significance to one or more of the systems in the solid model, such as an inflow boundary, make up a boundary entity. Finally, volumes are closed regions of three-dimensional space bounded by a set of faces [1, 10, 42].

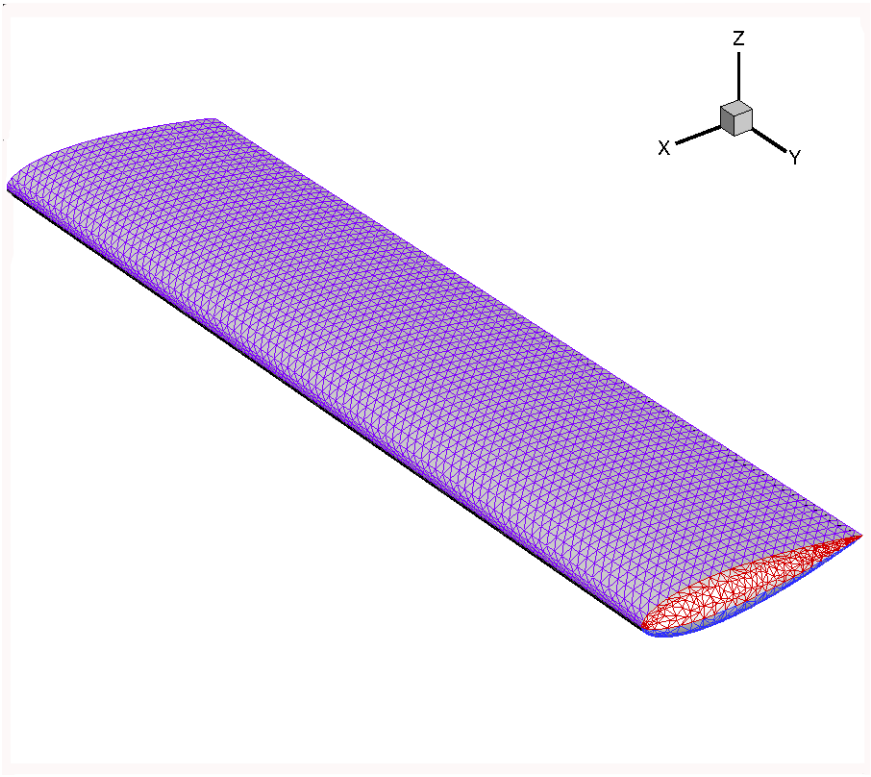
2.3.2 Tessellation

CAPRI, in particular, also provides a watertight triangular discretization of the solid geometry that can be used by the mesh generator in conjunction with the geometry and topology information to gain easier access to the CAD data [57]. In this thesis, the CAPRI tessellation is used to drive the surface mesh movement algorithm. The tessellation of an edge is an ordered stream of points while the tessellation of a face is points arranged into triangles [10] (in counter clockwise direction). An example of a triangulation is shown in Figure 2.2.

Each tessellation is created from 3 input parameters: angle, mxslen, and deviat. angle, in degrees, is the maximum angle between triangle neighbors. mxslen, in the same coordinate system as the solid model, is the maximum triangle side length. deviat, also in the same coordinate system as the solid model, is the maximum deviation between the parametric center of the triangle and the CAD surface. CAPRI will try to split up any triangles which violate these specifications. There is no guarantee that the tessellation will meet these criteria either because of the watertight requirement or because the



(a) CAD model of a wing consisting of 2 sections (flat and tip section) and 4 CAD faces (2 on top and 2 on bottom).



(b) Tessellation is done for each of the CAD faces, and the tessellations match at face boundaries.

Figure 2.2: CAD geometry and associated triangulation

boundary edges do not meet the specifications. It is easy to get "blemishes", or tightly clustered triangles, in the tessellation because the CAPRI triangulator will expose any unusual surface defect. This may disappear by increasing the angle parameter but doing so may also have other negative side-effects. There is no smoother or vertex removal phase in attempting to improve the tessellation. In fact, once a vertex is placed, it is left in the triangulation. It takes some experimentation to get an idea of what values will be suitable for a specific application. The best way to try to get uniform elements is to regenerate the tessellation with mxslen relatively tight (within the tolerance of the solid model) and with angle relatively loose (e.g. 45°). deviat usually has to be much smaller than mxslen.

It may be necessary to improve the triangulation manually. In such cases, one can use a smoother. Another idea is to try collapsing the short edge of the triangles which have high aspect ratios. This can be done by moving one vertex to another or moving them both to their average. In this case one should make sure that the resultant triangles are not inverted and to snap the result to the actual underlying CAD surface since only the vertices of the triangles are guaranteed to lie on the CAD surface.

2.4 Creating a CAD model

The first step in the optimization procedure is to define a robust CAD model that reflects the design approach and encapsulates the trends of the final design [27]. There are a number of ways to create a CAD model. The challenge is creating a design that captures features that are most likely to evolve throughout the optimization process. For example, one way to construct a wing is to define the coordinates of the airfoil section at the root. The coordinates can be read into CATIA from Excel using an Excel user-defined function. Then, two guiding curves extending from the leading and trailing edges of the airfoil are created in order to sweep the airfoil to make the wing surface. Finally, the tip of the wing is constructed by revolving a B-spline defining the airfoil section at the tip about its chord (see Figure 2.3). The intent is to use the guiding curves as design variables to control the sweep and span of the wing. However, because surfaces created this way do not have consistent orientation, and the entities do not have direct association, there could be problems later on regarding the generation of a watertight triangulation when the geometry is modified.

Another way of creating a wing is to define airfoil sections along the wing and then

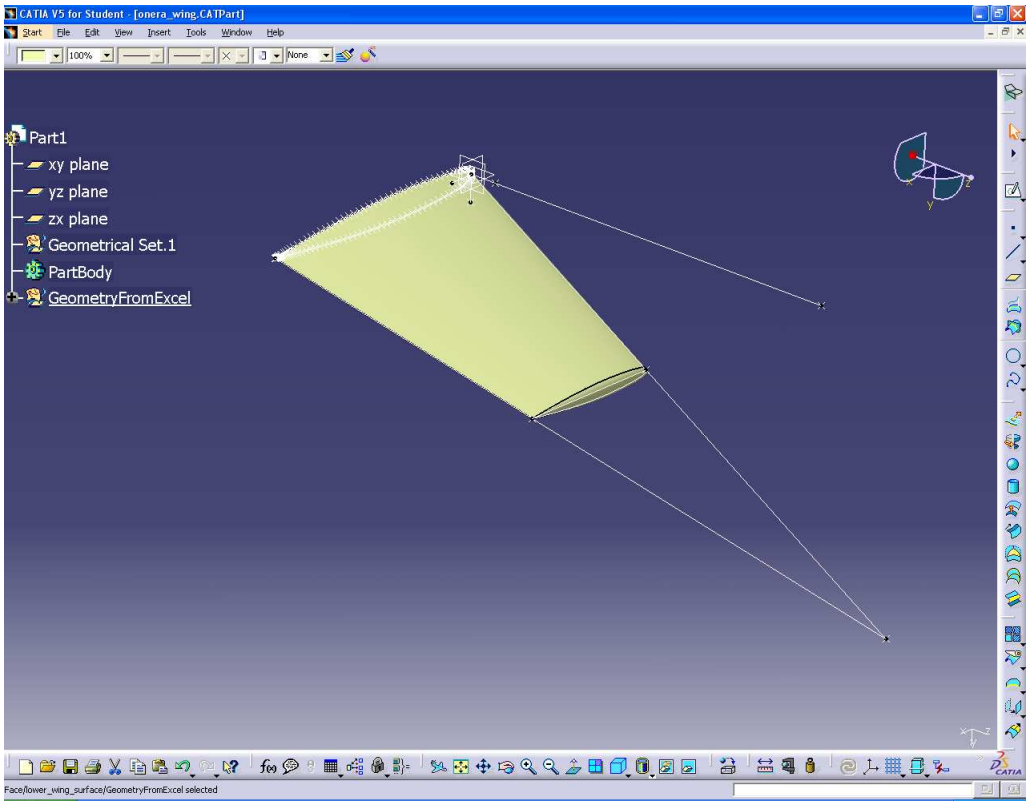


Figure 2.3: A wing created from independent surfaces (shell geometry)

create a solid (loft) using the defined sections (see Figure 2.4). A solid will be trimmed with greater precision, and the orientations will be consistent.

A geometry generation subroutine (GGS), based on the work of Fudge [43], is used to create CATIA V5 volumes from given cross sections. All of the functions in the subroutine can be manually performed in CATIA. However, this automated approach ensures that the geometry has the design variables incorporated. It also greatly reduces the construction time by eliminating the need to manually enter all of the section coordinates. The CAD volume is created using a process called lofting, over the cross sections (see Figure 2.4). This is one of the most common methods of generating CAD volumes, and is a standard procedure in the aerospace industry. For example, the DLR-F6 wing geometry used in the 2nd AIAA CFD Drag Prediction Workshop [30] was defined by four cross sections. The first was the Ha5 airfoil, and the second, third and fourth sections were the R4/4 airfoil. Each of the sections was scaled, rotated and translated through 3D space to generate the wing. A similar process is executed by the geometry generation subroutine.

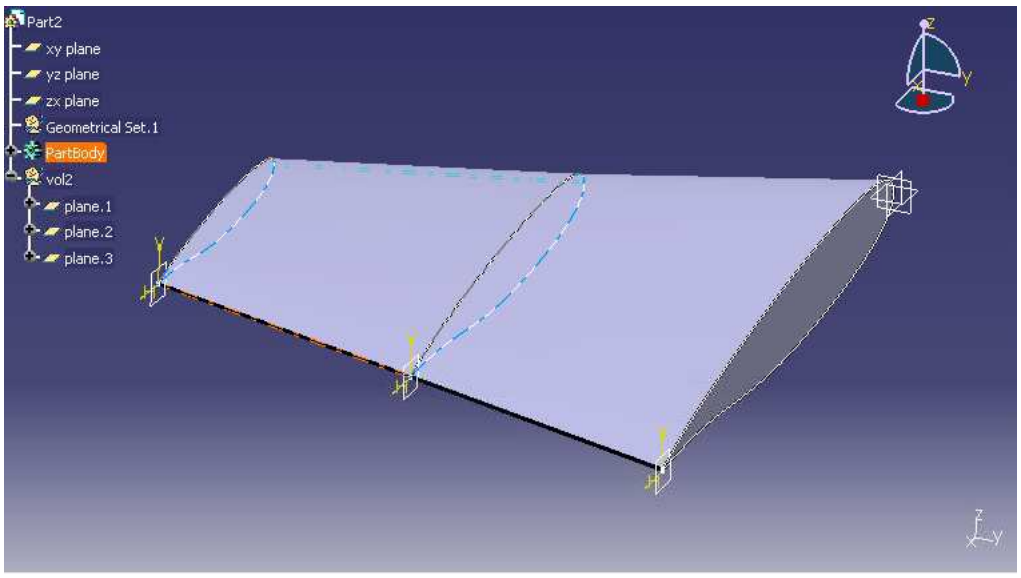


Figure 2.4: A wing created from lofting of three airfoil sections drawn on three sketches (solid geometry)

The GGS can be called several times to generate multiple volumes. These volumes can then be added to the same CATIA part to create more complex geometries, (e.g. the wing-body configuration shown in Figure 2.5). For example, the GGS can create a wing, fuselage, pylon, and nacelle, which can be joined to create a full aircraft. Internal and external details can also be added in CATIA to the base geometry, such as spars, antennae, and notation [42]. The inputs to the GGS are included in Appendix A.

2.5 Design Variables

The geometry generator creates a CAD loft defined by sketches. Each sketch contains a spline and support plane. To modify the geometry, each of the affected sketches are adjusted, and a new loft is created. There is no guarantee that the new loft will have the same properties (i.e. the same number of surfaces), and thus can affect subsequent sensitivity calculations. This problem can be resolved by clustering the CAD faces based on the CAD features and surface mesh topology. This process is explained in Section 2.6.1.

The geometry can be parametrized in order to reduce the number of design variables. Simple relations can be defined to link the design variables to a reduced set of CAD model parameters. The chosen design variables should provide flexibility such that all

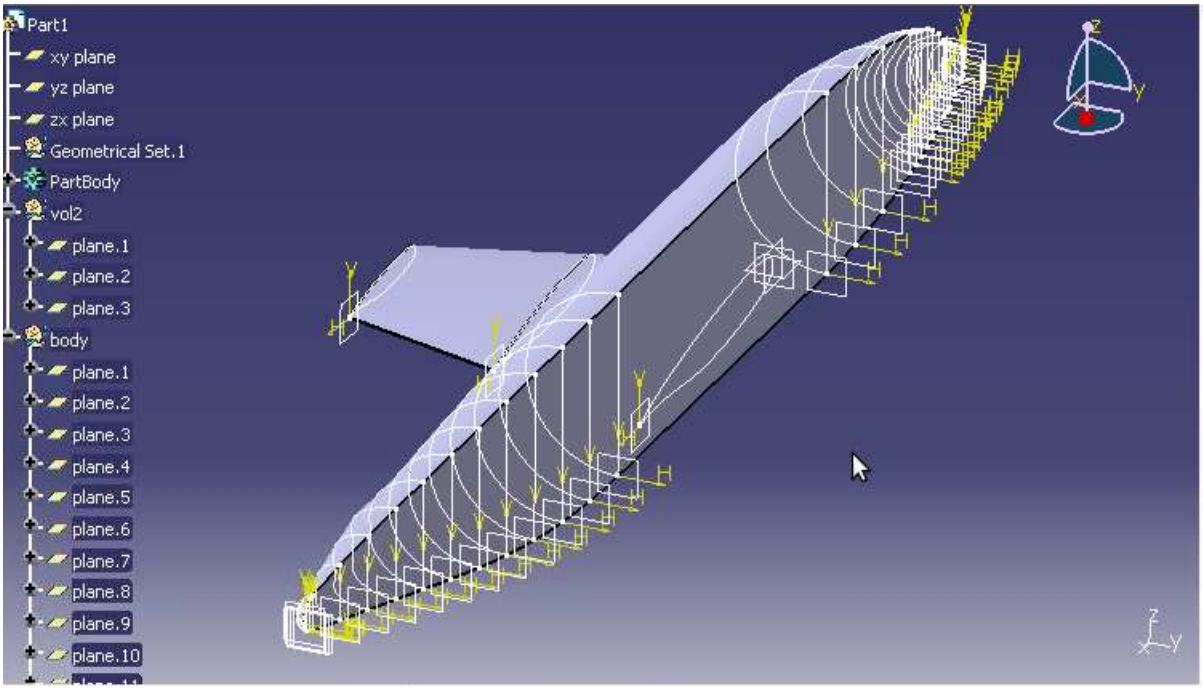


Figure 2.5: A wing-fuselage CATIA part made by adding 2 volumes

desirable shape variations can be obtained [77]. The parametric design methodology, also called variational geometry theory, was originally developed to generate variants of the design [174]. This process is realized by modifying a set of independent dimension parameters called high-level variables, which are sufficient to describe the exterior boundary of a part. However, the construction of a generalized dimensioning scheme is not simply a geometric problem. It also depends on how the structure will be fabricated, measured, toleranced and assembled [174]. Thus, it is impossible to identify automatically independent high-level design variables in general cases. Different parametrization approaches can be conceived, which lead to the same geometric representation although one may not explicitly convey the design intent. For example, a wing can be defined by the spatial location of the sections or by planform variables such as wing area, wing span, taper ratio, sweep angle, dihedral angle, thickness distribution and the x, y, z location of the wing tip at the root. In the former approach, the emphasis is on the airfoils rather than the wing itself. The latter approach, on the other hand, provides more physical meaning to the design [27].

The design variables can be global or nodal. They are:

1. Global:

- Taper, in x- and y-directions, which corresponds to leading-edge sweep and thickness distribution along the span
- Skew, in x- and y-directions, which corresponds to trailing-edge sweep and dihedral
- Angle of attack, i.e. rotation about the z-axis
- Twist
- Scale, in x-, y- and z-directions, which corresponds to chord, thickness and span
- Displacement of individual sections
- Displacement of the origin position of the part

2. Nodal:

- Number of control sections
- Position of each control section
- Number of nodes in a control section
- Position of the control nodes

Appendix A describes how the design variable can be incorporated into the geometry during construction so that they can be accessed and modify during all stages of design optimization.

2.6 Controlling Topology Changes

One of the difficulties in using CAD systems for generating parts for analysis and optimization is the lack of the ability to control the number of entities generated, or more precisely, the part's topological outcome. The topology may fundamentally change, even when two parts may be almost identical. For example, the fuselage in Figure 2.6, is a cylindrical surface that due to a specific CAD system happens to be split along an axis generally aligned with the wing juncture. Assuming that the mating position of the wing relative to the fuselage centreline is a design parameter, changing this parameter has a significant effect on the topology of the model. Maintaining topological consistency is

useful for perturbing grids in order to eliminate complete grid generation for small parameter changes, and allows the determination of geometric sensitivities when using adjoint methods [60, 137]. To address this issue, Alonso *et al.* [3] used a technique that scribes a consistent quadrilateral patch topology over the CAD geometry. The same number of points was used to discretize each side of the quadrilateral in subsequent instances. Therefore the points could be tracked by matching patch number and index in the two parametric directions.

Dannenhoffer and Haimes [25] used the CAPRI tessellation and its association to the BREP to control the topology, a process called *Quilting*. Quilting paves over the edges of the faces of the BREP that do not display sharp angles. Haimes and Dannenhoffer [60] developed a virtual topology editor (**vte**) to provide programmers with the ability to directly edit the topology of a CAD part. On **vte** initialization, a thin-skinned model consisting of Ferguson splines is created that mimics the source. The **vte** allows for topology editing (scribing new features, quilting, merging and splitting of faces) without affecting the original geometry. This process is quite complex in that it is dependent on a discretization of the model, a Ferguson spline-based solid modeling subsystem, as well as the association of the faces to the original BREP. A simpler procedure to modifying the CAD topology was proposed by Sheffer *et al.* [142]. They introduced the concept and operators of virtual topology required to edit the CAD model. The operators are applied only on the topological structure or connectivity of the model, leaving the geometric descriptions intact. It is thus relatively inexpensive, and the original model can be easily restored [143]. Originally developed to repair CAD data for automated meshing, this algorithm is quite robust and efficient. It is based solely on computations of distances and angles between planes and/or planar facets. The algorithm was modified by Inoue *et al.* [73] and further improved by Sheffer [141] by reducing the complexity cost (resulting from the use of analytical measurements of region curvature) and by addressing the issue of clustering faces into smooth but non-planar regions such as cylinders. Figure 2.7 shows how the face clustering process can simplify the CAD topology and be used to obtain consistent topology when the geometry is modified.

2.6.1 Clustering Algorithm

A clustering algorithm similar to the method of Sheffer *et al.* [74, 142, 143] is implemented here to ensure consistent topology as the geometry changes during shape optimization.

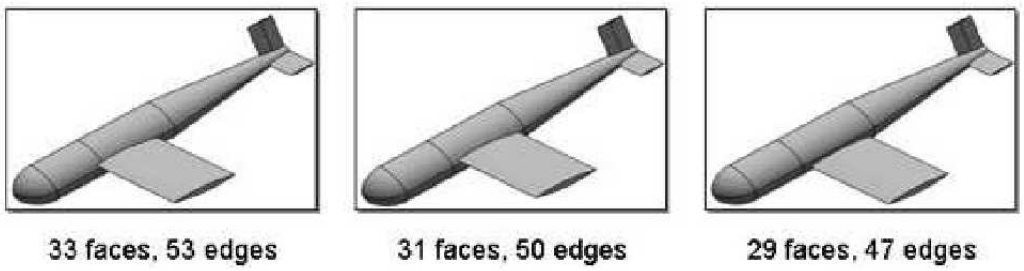


Figure 2.6: BREPs resulting from moving the wing [60]

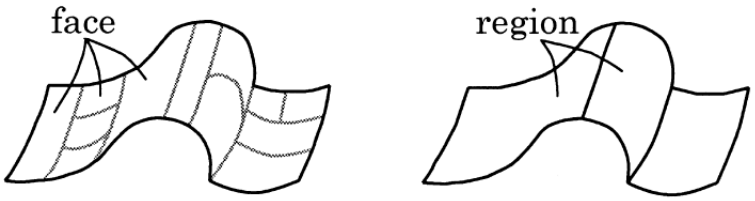
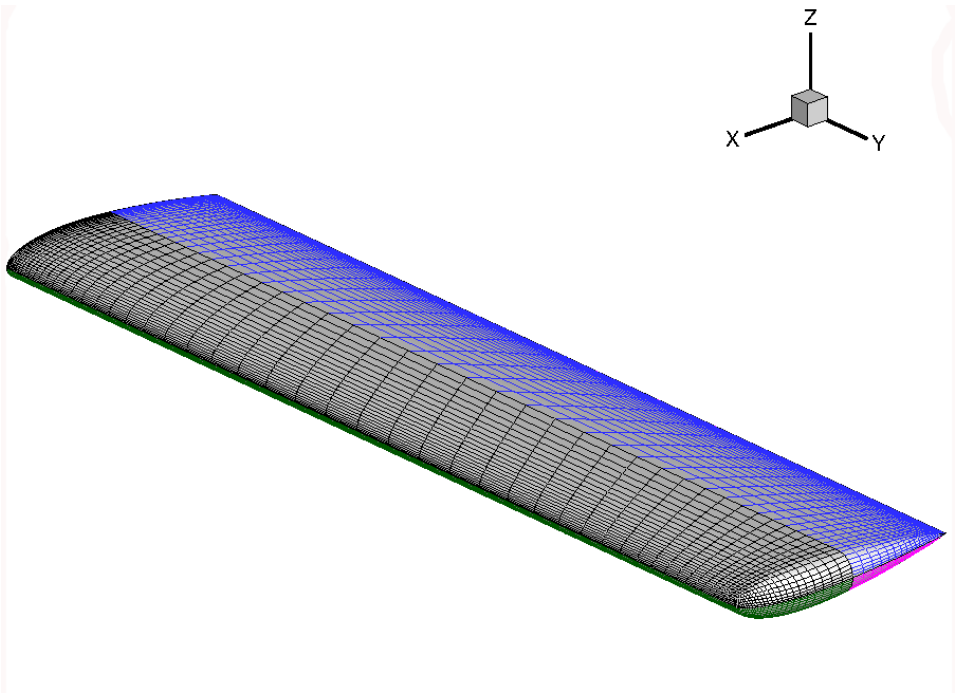
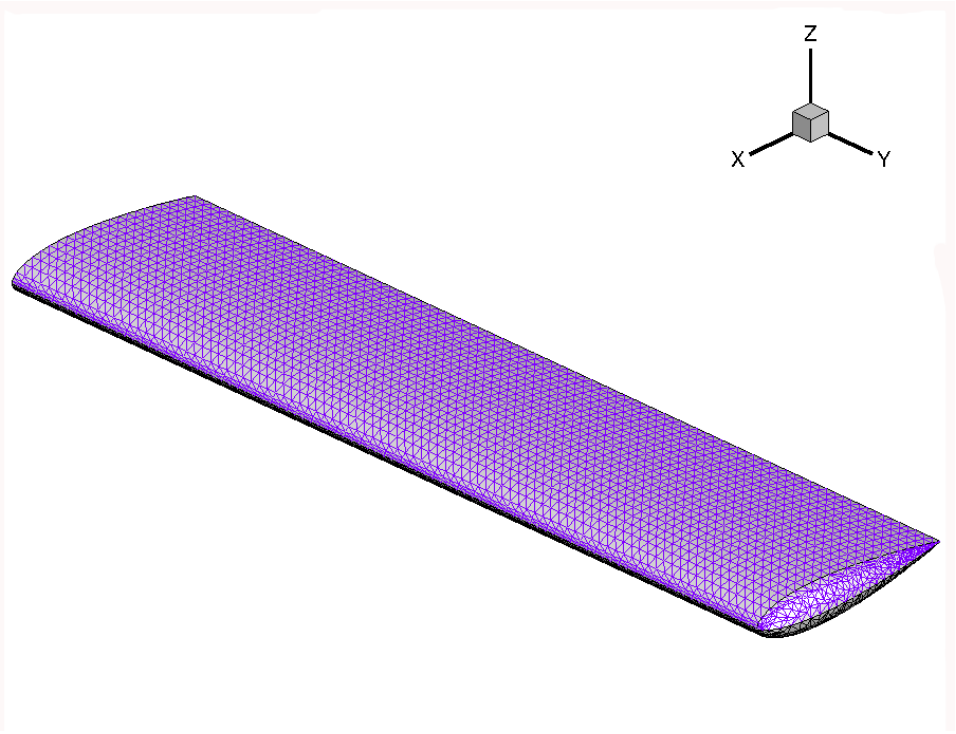


Figure 2.7: Input CAD model (left) and model after the face merging process (right) [74]

However, the criterion for the clustering of faces is based on the blocking scheme of the initial structured, multi-block volume mesh. Since the mesh will generally be deformed and reused throughout the optimization cycle, the sides of the blocks which lie on the CAD surface are used to guide the clustering algorithm such that the boundary of the block sides coincides with the boundary of the clustered CAD faces. The clustered faces are re-parametrized so that the nodes lying on the face can be tracked when the faces undergo transformation. This clustering scheme naturally satisfies the size and smoothness requirements to ensure the parametrization will be least distorted. The clustering of the wing shown in Figure 2.2(a) corresponding to a given surface mesh topology (Figure 2.8(a)) is demonstrated in 2.8(b).



(a) Surface mesh, color coded by a given blocking scheme, i.e. each colored surface patch belongs to a different block in a structured multi-block volume mesh.



(b) Clustering of CAPRI tessellation displaying one top patch and one bottom patch.

Figure 2.8: Clustering of CAPRI tessellation based on surface mesh topology.

3 Surface Mesh Deformation

During shape optimization, as the geometry changes, the surface mesh has to deform to fit the new geometry. Since there is no control over what kind or how many surfaces will be generated by the CAD program, the surface mesh movement algorithm cannot rely on the mathematical representation of the CAD surface. Instead, it uses the discrete representation, i.e. the CAPRI tessellation, whose vertices are guaranteed to be on the CAD surface, to deform the original mesh into the new shape. The deformation process involves the following steps:

1. Reparameterize the surface mesh and the CAPRI tessellation of the new shape onto the same region in the 2D plane, aligning corresponding boundaries.

2. Search for a two-dimensional (2D) triangle in the CAPRI tessellation that contains each of the surface nodes. The position of the node inside the triangle, i.e. its barycentric coordinates, is calculated.

3. Determine the CAD parametric coordinates for the surface node based on the barycentric coordinates and the CAD parametric coordinates of the vertices of the triangle which enclose it. This is an important step because it guarantees that the new mesh nodes will be on the CAD surface.

4. Convert the CAD parametric coordinates of the surface nodes into physical coordinates in 3D by interrogating the CAD program.

The background and methodology regarding surface parameterization are described in detail below.

3.1 Surface Parametrization

Parametrization of a discrete surface is a fundamental and extensively used operation in computer graphics [28, 148, 168, 173]. The initial motivation for the development of the first parameterization methods was the application to texture mapping in which properties of a 3D surface, such as colors, are sampled and stored in a texture map. The quality of the parameterization greatly affects the visual quality of polygonal models. Later, surface parameterization was used in other applications such as geometric morphing (i.e. animation), surface approximation, and remeshing due to growing demands for efficient compression methods of increasingly complex triangulated representations of 3D objects [36]. It amounts to computing a correspondence between a discrete surface patch and an isomorphic planar mesh through a piecewise linear function or mapping (see Figure 3.1). Each mesh node is assigned a pair of coordinates (u, v), referring to its position on the planar region. Such a one-to-one mapping allows one to perform any complex operation directly on the flat domain rather than on the curved surface. This facilitates most forms of mesh processing, such as morphing, surface fitting, remeshing, geometry images, and texture mapping [26, 36, 95, 127]. In texture mapping, for example, an object can be efficiently represented by a coarse geometric shape with the details corresponding to each triangle stored in a separate 2D array. Remeshing, on the other hand, is a process that corrects a given mesh geometry and connectivity (i.e. the size, vertex sampling and triangle quality), while providing decent fidelity. It is used to optimize the number of polygons to achieve acceptable visual quality and interactive rendering speeds [51].

Surface mesh deformation, also known as mesh morphing or metamorphosis, has been one of the active research themes in computer graphics in recent years [2, 82, 128]. The goal is to morph one surface G_{surf1}, called the source, to another surface G_{surf2}, called the target, which can have different topology. Usually, both the source and target surfaces are made up of triangular elements. Unlike previous work [82] in which the new object has connectivity inherited from both the source and target objects, our new surface mesh will have the same topology as the source surface mesh, since the objective is to deform the source surface mesh to conform to the new shape, represented by a triangulated mesh. It is also our goal to preserve the mesh characteristics (mesh density and distribution) of the original surface mesh.

An approach to transforming from G_{surf1} to G_{surf2} is usually divided into two steps.

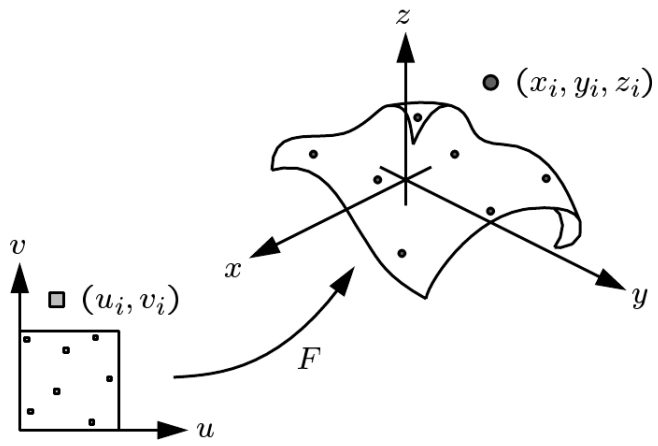


Figure 3.1: Surface parametrization [53]

The first step is to establish a correspondence between the boundaries of G_{surf1} and G_{surf2}. This is done by transforming G_{surf1} and G_{surf2} to a 2D parametric space (e.g. a unit square or any convex polygon* that is representative of G_{surf1} and G_{surf2}). The mathematical background and methods for surface mapping or parametrization are described in Section 3.1. Next, the two mappings are merged. By overlapping the mapping of the original surface patch onto the mapping of the triangulation, we can determine in which triangle each of the original surface nodes lies. Then we can determine the deformed surface by interpolating the vertices of the triangle. The inverse mapping method is described in Section 3.2.2. This approach has an additional advantage in that there is automatic matching of mesh points at patch interfaces.

3.1.1 Survey of Parametrization Techniques

The parametrization problem is inherently difficult as one is trying to flatten a surface from 3D to 2D. There is no perfect way to perform such a flattening without introducing some form of distortion [92]. Generally, parametrization techniques can be classified into two types: fixed or non-fixed boundaries in the parametric domain. Fixed boundary methods typically use very simple formulations and are very fast [148]. However, fixed boundary methods often have large distortions because the boundary shapes of the original models can be very different from those of their flattened surfaces in the 2D

*a simple polygon whose internal angles are less than or equal to 180 degrees. In other words, every line segment between two vertices remains inside or on the boundary of the polygon.

domain [92]. Free-boundary techniques, which determine the boundary as part of the solution, are often slower, but typically introduce significantly less distortion. In recent years, numerous methods for parameterizing meshes have been developed, targeting diverse parameter domain and focusing on minimizing the distortion of different intrinsic measures of the original mesh [28].

In general, we want to preserve as much of the intrinsic qualities of a surface as we can during its parametrization. This implies that we need to first define what these intrinsic qualities are for a discrete surface: length, angle and area. A length-preserving or isometric mapping is ideal in that it preserves not only length, but also angle and area. However, it is well known that isometric mappings only exist in very special cases, where the surface is developable.[†] Therefore, many approaches to surface parametrization attempt to find a mapping which minimizes distortion of either angle or area, or some combination of these [36].

Maps that minimize the angular distortion are called *conformal*, and maps that minimize area distortion are called *authalic*. In theory, angle distortion can be eliminated completely by conformal mapping, but it is impossible for conformal mappings to further eliminate area distortion completely, except for developable surfaces. As pointed out by Floater and Hormann [36], though authalic parameterizations are achievable, they are not very useful by themselves, as they allow extreme angular and linear distortion. Thus, area preservation [26, 28] methods are typically combined with angle preservation [148] to minimize the distortion.

A common approach is to minimize a certain energy to control the distortion. A simple, yet effective fixed-boundary parametrization method that uses linear spring energy with uniform spring constant was introduced by Tutte [160] in 1963. Tutte's embedding method is established in the following steps: First, n vertices, which make up a boundary segment of G_{surf}, are positioned on some convex polygon in R^2. The positions of the interior vertices are calculated so that the total energy is minimized. This energy can be represented as a sum of the energy of a configuration of springs with one spring placed along each edge of G_{surf}. Tutte's method results in a sparse, diagonally dominant linear system that can be solved easily with any iterative method [106]. Floater [34] proposes a different set of weights for the edge spring model. He generalized Tutte's procedure to generate all possible valid embeddings of the 3D graph in the plane, given the (convex)

[†]surfaces with zero Gaussian curvature, which means that it is a "surface" that can be flattened onto a plane without distortion (i.e. "stretching" or "compressing"). Examples are cylinders and cones.

positions of the boundary. Floater's shape preserving parametrization method has the advantage that the weights are always positive. This guarantees that the linear system derived can be solved robustly [106]. Others [35, 94, 125] have developed variations of Tutte's method, aiming for some effect related to reflecting the geometry of the mesh in the parametrization, i.e. minimizing its metric distortion. However, some of the methods [94, 125], cannot guarantee a valid map. Variations of harmonic energies were also optimized using discrete Laplace-Beltrami operators in [31, 34, 35, 54, 125, 146]. However, harmonic maps may contain face flips which violate the bijectivity [‡] of a parametrization [26].

Some methods decompose complex objects into a set of simpler shapes which can be parametrized without incurring too much stretch [153, 173]. In some cases, the object is mapped onto a sphere [51], which has equivalent topology to it. This can significantly reduce the distortion introduced by the parametrization without resorting to methods which introduce other artifacts, such as cutting seams. Virtual boundaries were applied in [93, 173] to absorb distortions introduced by the convex boundary conditions in cases where the boundary of the surface does not resemble a convex shape. Alternatively, [28, 94] provided parameterizations which require fixing only a few vertices in the parametric domain. Karni *et al.* [84] discussed the design of geometrically complex boundary conditions with constraints. Zayer *et al.* [172] applied discrete tensorial quasi-harmonic maps to improve the boundary and reduce the distortion.

Lévy *et al.* [94] define an energy to approximate the Cauchy-Riemann equation. They formulate the discrete conformality problem as an unconstrained quadratic minimization problem and prove the uniqueness and existence of its solution. Using a standard numerical conjugate gradient solver, they are able to compute least squares approximations to continuous conformal maps very efficiently without requiring fixed boundary texture coordinates. However, in rare cases triangle flips may occur. This mapping approach does not need to fix the boundary, and the computation can be performed efficiently. However, this approach may generate fold-over maps after parametrization. Desbrun *et al.* [28] define a space of measures spanned by a discrete version of the Dirichlet energy and a discrete authalic (area-preserving) energy. While the authalic energy remedies local area deformations, it requires fixed boundaries and results cannot achieve the quality of methods targeted at global length preservation such as Sander *et al.* [138]. They define a

[‡]there is a one-to-one correspondence between the 3D surface and its parametrization

non-linear texture stretch metric for surface parametrization. Sander *et al.* [138] proposed stretch metrics that are now commonly used in the graphics community as standard measures of distance preservation. He observed that a linear map over each triangle can be decomposed into a translation, rotation and non-uniform scale along two orthogonal axes. The two scale factors $0 \leq \gamma \leq \Gamma$ are the singular values of the transformation matrix, or the square roots of the eigenvalues of the integrated metric tensor (the transpose of the matrix times the matrix itself). Intuitively, the linear transformation map will stretch a unit circle to an ellipse with axes γ and Γ. The L_{inf} stretch for a triangle is defined by Sander *et al.* as $\max(\gamma, \Gamma) = \Gamma$ while the L_2 stretch is defined as $\sqrt{\frac{\gamma^2+\Gamma^2}{2}}$. This technique uses a relaxation approach to iteratively flatten the 3D surfaces. Their texture stretch metric produces better results than those of several previous methods ([31] and [69]), but requires many iterations for computing the optimal mapping.

More general energy forms can be found in [26, 69, 138, 153, 169, 173]. In Hormann and Greiner [69], mostly isometric parameterizations are introduced that minimize a nonlinear energy. Mostly isometric parameterizations do not require boundary texture coordinates to be fixed and avoid face flips. Additionally they approximate mathematically well studied continuous conformal maps (i.e. maps that perfectly preserve angles). Since the method's goal is to minimize an L_2 norm, the method sometimes compensates for shrinkage in one direction by stretching in another, introducing shearing [148].

Another characteristic of conformal mapping is that it can map infinitesimal circles to infinitesimal circles and preserve their intersection angles. This inspired the circle packing method in [155, 168], which uses a certain configuration of circles with a specified pattern of tangencies known to provide a way to approximate a conformal mapping. Circle packing and circle patterns replace infinitesimal circles with finite circles. In the limit of refinement, the continuous conformal maps are recovered. However, building circle packings is quite expensive [94].

Another approach to minimize angular distortion is proposed by Sheffer and de Sturler [144, 148]. They define a non-linear energy in terms of the corner angles of the mesh in texture space. Angle-based flattening (ABF) is a robust parametrization method that finds a quasi-conformal mapping by solving a constrained nonlinear optimization problem in terms of angles [145, 172]. The discrete conformal mapping is derived by minimizing the ABF energy, which is defined as differences between the corner angles of faces on the original mesh and their images in the parameter space. During the process, the boundary evolves freely to further reduce the distortion. Recently, the

method has been improved by several derivative works [147, 171] in terms of speed and robustness.

Besides angle preserving methods, only a few approaches explicitly optimize global area or global length distortion: Maillot *et al.* [99] minimize an edge length distortion, but cannot guarantee the absence of face flips. The authors also propose an area preserving energy and combine both energies in a convex combination. Iterative smoothing of an overlay grid is proposed by Sheffer and de Sturler [146] as a post-processing step for angle preserving parametrization algorithms. However, it is not clear what impact the post-processing has on the angle preservation.

Degener [26] propose a metric energy that simultaneously measures angular and global area deformations imposed by a parametrization. On surfaces with nonzero Gaussian curvature, the unavoidable deformation of angles and areas is traded off by the energy in a user-controlled way. This functional can be used to optimize parameterizations for a uniform surface sampling. It is designed to prevent face flips during optimization and does not require a fixed boundary. It is invariant under rotation and translation of the domain. Although the derived energy is nonlinear, it is differentiable and well suited for a hierarchical minimization as proposed by Hormann *et al.* [69]. Angle and global area optimized parameterizations can be computed efficiently with guaranteed convergence using nonlinear conjugate gradient methods. This method does not prevent self intersections which can occur in rare cases and can be handled in a post-processing step.

Clarenz *et al.* [24] present a new parametrization method that are described in terms of minimizers of suitable energy functionals. It provides built-in parameters for the control of angle, area and length distortions. Their approach is closely related to the theory of nonlinear elasticity where a finite element discretization is introduced and a constrained Newton method is used to minimize a corresponding discrete energy.

3.2 Implementation

In choosing an appropriate method, the following factors are taken into consideration:

- Free vs fixed boundary: Since we need to find a correspondence between the source and target surface meshes, the boundaries of the surfaces in parametric space must be the same. This limits our choice of methods to those with fixed boundaries. For

simplicity, a unit square is used as the boundary for the mapped surfaces. For our application, it is sufficient (or even desirable) to take a square as the parametric domain, with the advantage that such a convex shape guarantees the bijectivity of the parametrization if positive barycentric coordinates like the mean value coordinates are used to compute the parameter points for the interior vertices. The four edges on the surface patch can be related to the four sides of the square. For simplicity, the parameter points are uniformly spaced on the rectangular domain. Although some heuristic procedures for placing the boundary points have been proposed ([68]) and tried, having to fix the boundary vertices is already a limitation.

- Robustness: Most applications of a parametrization require it to be bijective (invertible). The bijectivity can be local, which means that there are no triangle flips, or global, which means that the boundary does not self-intersect. Since we use the map as a means to locate the position of any point on the 3D surface, any technique which provides a one-to-one mapping is sufficient.

- Numerical Complexity: Parametrization methods which require a linear solution method (as opposed to non-linear method) to solve are typically significantly faster and simpler to implement, at a cost of increased distortion. Since our problems contain hundreds of design variables and we need to find the sensitivity of the surface with respect to each of the design variables, a method that can provide a fast solution is required.

Tutte's method satisfies these criteria. However, since Tutte's method does not preserve any intrinsic property of the mesh, we need a triangulation that has elements of uniform length. Any non-uniformity or irregularity in the mesh will introduce distortion in the mapping, and subsequently, the final mesh. The quality of the parametrization is directly affected by the regularity of the triangulation, both topologically and geometrically. Topological regularity refers to meshes where the vertices have the same degree (connectivity), and geometric regularity implies that the triangles are similar to each other in terms of shape and size, and have vertices close to the centroid of their neighbors [148]. For example, the regularity of the triangulations in Figures 3.3(b) and 3.4(b) result in smooth surface meshes shown in Figures 3.3(f) and 3.4(d) upon inverse map. This regularity cannot be obtained very easily using CAPRI, only in cases where the

boundary of a CAD face is similar to that of a square, such as the straight portion of the wing shown in Figure 4.3. This regularity is lost when the wing is given some angular deformation, such as a sweep of the leading edge, resulting in distortion of the surface mesh in the region where the irregularities occurred. This is evident in Figures 3.6 and 3.7. Section 3.2.1 describes how the CAPRI triangulation is parametrized on a unit square using Tutte's method.

The source surface is parametrized based on the idea of scaling, to maintain the ratio of the mesh cell with respect to the given patch. To preserve the mesh spacing of the source surface, G_{surf1} is parametrized onto a unit square using the arclength method, where u_i is the normalized arc length given by:

$$u_1 = 0 \tag{3.1}$$

$$u_j = \frac{1}{L_g} \sum_{i=2}^{j} L_i \qquad j = 2, \ldots, j_{max-1} \tag{3.2}$$

where L_i is the length of a segment between nodes j and $j-1$. L_g is the grid line length from one side of the patch to the other:

$$L_g = \sum_{i=2}^{j_{max}} L_i \tag{3.3}$$

A similar calculation is performed for v_i.

3.2.1 Barycentric Mapping

A simple idea for constructing a parametrization of a triangular mesh is based on the spring model, where the edges of the triangular mesh are springs that are connected at the vertices. If the boundary of this spring network is fixed somewhere in the plane, then the interior of this network will relax in the energetically most efficient configuration, and we can simply assign the positions of the vertices as parameter points [68].

Following the derivation given by Hormann [68], each spring is assumed to be ideal in the sense that the rest length is zero and the potential energy is $\frac{1}{2}Ds^2$, where D is the spring constant and s the length of the spring, we can formalize this approach as follows. First, the parameter points $\mathbf{u}_i = (u_i, v_i)$, $i = n+1, \ldots, n+b$ for the boundary vertices $\mathbf{p}_i \in \mathcal{V}_\mathcal{B}$ of the mesh are projected onto a planar convex polygon. Then minimize the overall spring energy:

$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N_i} \frac{1}{2} D_{ij} \|\mathbf{u}_i - \mathbf{u}_j\|^2 \tag{3.4}$$

where $D_{ij} = D_{ji}$ is the spring constant of the spring between $\mathbf{p}_i$ and $\mathbf{p}_j$, with respect to the unknown parameter positions $\mathbf{u}_i = (u_i, v_i)$ for the interior points. As the partial derivative of E with respect to $\mathbf{u}_i$ is

$$\frac{\partial E}{\partial \mathbf{u}_i} = \sum_{j \in N_i} D_{ij}(\mathbf{u}_i - \mathbf{u}_j) \tag{3.5}$$

the minimum of E is obtained if

$$\sum_{j \in N_i} D_{ij}\mathbf{u}_i = \sum_{j \in N_i} D_{ij}\mathbf{u}_j \tag{3.6}$$

for all $i = 1, \ldots, n$, where n is the number of interior nodes and N_i is the number of vertex p_i's neighbors. In other words, each interior parameter point $\mathbf{u}_i$ is an *affine combination* of its neighbors,

$$\mathbf{u}_i = \sum_{j \in N_i} \kappa_{ij}\mathbf{u}_j \tag{3.7}$$

with normalized coefficients

$$\kappa_{ij} = \frac{D_{ij}}{\sum_{k \in N_i} D_{ik}} \tag{3.8}$$

that sum to 1.

By separating the parameter points for the interior and the boundary vertices in the sum on the right hand side of Eq. (3.7) we get

$$\mathbf{u}_i - \sum_{j \in N_i, j \leq n} \kappa_{ij}\mathbf{u}_j = \sum_{j \in N_i, j > n} \kappa_{ij}\mathbf{u}_j \tag{3.9}$$

and see that computing the coordinates u_i and v_i of the interior parameter points $\mathbf{u}_i$ requires the solution of the linear systems

$$AU = \bar{U} \qquad \text{and} \qquad AV = \bar{V} \tag{3.10}$$

where $U = (u_1, \ldots, u_n)$ and $V = (v_1, \ldots, v_n)$ are the column vectors of unknown coordinates, $\bar{U} = (\bar{u}_1, \ldots, \bar{u}_n)$ and $\bar{V} = (\bar{v}_1, \ldots, \bar{v}_n)$ are the column vectors with coefficients

$$\bar{u}_i = \sum_{j \in N_i, j > n} \kappa_{ij} u_j \qquad \text{and} \qquad \bar{v}_i = \sum_{j \in N_i, j > n} \kappa_{ij} v_j \tag{3.11}$$

and $A = (a_{ij})_{i,j=1,\ldots,n}$ is the $n \times n$ matrix with elements

$$a_{ij} = \begin{cases} 1 & \text{if} \quad i = j, \\ -\kappa_{ij} & \text{if} \quad j \in N_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

The question remains how to choose the spring constant D_{ij} in the spring model. The simplest choice of $D_{ij} = 1$ goes back to the work of Tutte [160], who used it in a more abstract graph-theoretic setting to compute straight line embeddings of planar graphs. This approach lacks the linear reproduction characteristic, which is a production of an isometric (and thus optimal) mapping in the case where the surface is contained in a plane so that its vertices have coordinates $p_i = (x_i, y_i, 0)$ with respect to some appropriately chosen orthonormal coordinate system. In this case, the parameter points themselves are defined using the local coordinates, i.e. $\mathbf{u}_i = \mathbf{x}_i$, for $i = 1, \ldots, n+b$. Other researchers [31, 35, 37, 38, 125] have developed various values for D_{ij} to achieve the linear reproduction property, which is useful for computer graphics, geometric modeling and other applications [68], but can affect the solvability of the linear systems defined in (3.14). Since the linear reproduction characteristic is inconsequential to our application, the method of Tutte is chosen for the reparameterization of the target surface (i.e. the CAPRI triangulation).

3.2.2 Inverse Mapping

When a vertex p is included in a face of an input mesh $f = (p_i, p_j, p_k)$ in the parametric domain, the 3D position p is calculated by a barycentric coordinate (α, β, γ):

$$p = \alpha p_i + \beta p_j + \gamma p_k \tag{3.13}$$

Before the calculation of p, we need to find a face f in which p is included. In general, this problem can be regarded as a point location problem in the parametric domain, and can be processed in $\mathcal{O}(\log n)$-time for each vertex. Then, the calculation for the uniform subdivision fitting at each level is processed in $\mathcal{O}(m \log n)$-time, where m is the number of vertices in the 3D surface patch and n is the number of faces in the CAPRI triangulation.

In order to ensure that the surface node is on the CAD surface, we obtain the coordinates of the node in 2D using the CAD parametric coordinates of the triangle vertices, then a CAPRI function gi_qNormalToFace [56] is called to obtain the corresponding physical coordinates in 3D space. For nodes lying on boundary edges, their parametric positions are obtained by interpolating the edge vertices of the triangles which bound the nodes. This is possible because the vertices in the CAPRI tessellation are guaranteed to be on the CAD surface (for interior nodes) and on the CAD edge (for boundary nodes).

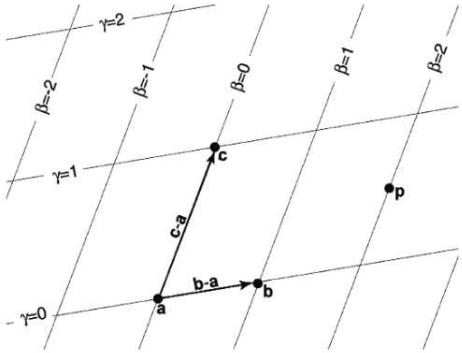


Figure 3.2: Barycentric coordinates [90]

Since the gi_qNormalToFace query is needed for all surface nodes, it can be executed in a "grouped" mode where all calls are collected and executed in a single command to minimize communication time.

3.2.2.1 Barycentric coordinates in 2D

Consider a 2D triangle whose vertices are $a = (x_a, y_a)$, $b = (x_b, y_b)$, $c = (x_c, y_c)$ and a point $p = (x, y)$. Barycentric coordinates allow us to express the coordinates of p in terms of a, b, c. More specifically, the barycentric coordinates of p are the numbers α, β and γ such that

$$p = \alpha + \beta(b - a) + \gamma(c - a) \tag{3.14}$$

If we regroup a, b and c, we obtain

$$\begin{aligned} p &= \alpha + \beta b - \beta a + \gamma c - \gamma a \\ p &= (1 - \beta - \gamma)a + \beta b + \gamma c \end{aligned} \tag{3.15}$$

If we define α as

$$\alpha = 1 - \beta - \gamma \tag{3.16}$$

we then have

$$p = \alpha a + \beta b + \gamma c \tag{3.17}$$

Figure 3.2 shows the meaning of the numbers β and γ.

Barycentric coordinates are defined for all points in the plane. They have the following properties:

1. A point p is inside the triangle defined by a, b, c if and only if

$$\begin{aligned} 0 < \alpha < 1 \\ 0 < \beta < 1 \\ 0 < \gamma < 1 \end{aligned} \tag{3.18}$$

   This is very important as it can be used to test if a point is inside a triangle.

2. If one of the barycentric coordinates is 0 and the other two are between 0 and 1, the corresponding point p is on one of the edges of the triangle.

3. If two of the barycentric coordinates are 0 and the third is 1, point p is at one of the vertices of the triangle

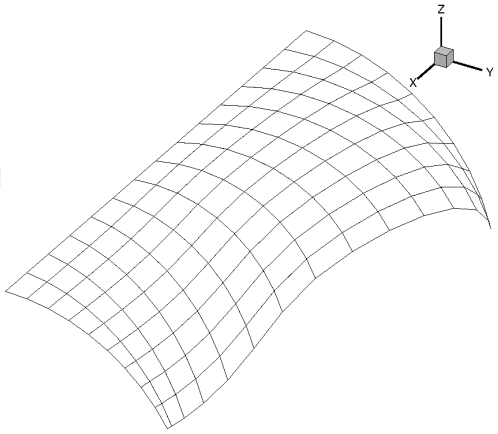
The values of α, β, γ can be determined by writing equation (3.14) in terms of the coordinates of the various points involved to yield the following system of equations:

$$\begin{cases} x &= x_a + \beta(x_b - x_a) + \gamma(x_c - x_a) \\ y &= y_a + \beta(y_b - y_a) + \gamma(y_c - y_a) \end{cases} \tag{3.19}$$

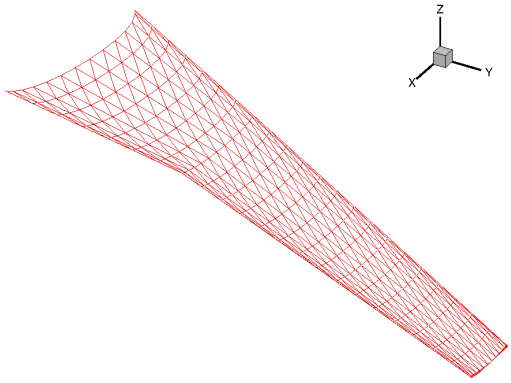
Once the values for α, β and γ are obtained, they are substituted in Equation (3.17) to determine the position of point p in the 3D space.

3.3 Surface Mesh Deformation Examples

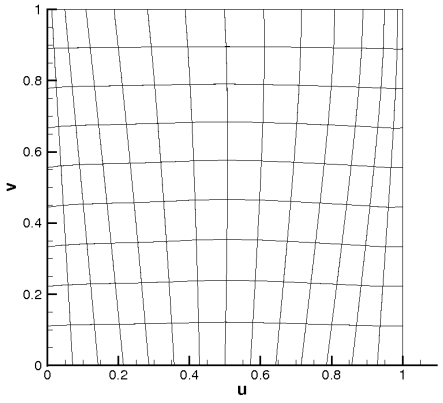
Figure 3.3 illustrates the surface mesh deformation process. Given an initial structured surface mesh, generated for example using a mesh generation method on the initial geometry, as shown in Figure 3.3(a), and a tessellation of the deformed surface, Figure 3.3(b), the surface mesh movement algorithm parameterizes them onto a unit square, shown in Figures 3.3(c) and 3.3(d), respectively. By overlapping the parameterizations, shown in Figure 3.3(e), it searches for a triangle in the tessellation that encloses each of the surface mesh points and determines the barycentric coordinates of each of the points with respect to the coordinates of the vertices of triangle. The coordinates of the new surface mesh, Figure 3.3(f), are obtained by interpolating the 3D physical coordinates of the vertices of the triangles using the corresponding barycentric coordinates. It can be seen in Figure 3.3(f) that the new surface mesh has similar features (node clustering and distribution) as the tessellation. This example demonstrates the importance of having a



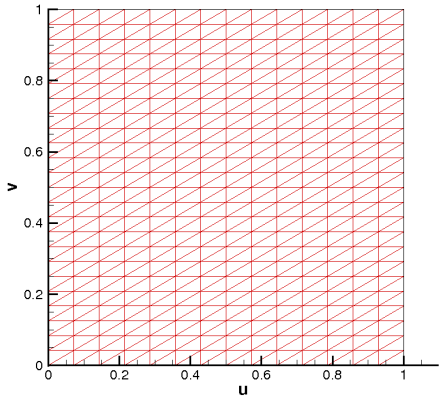
(a) Surface mesh of the initial geometry



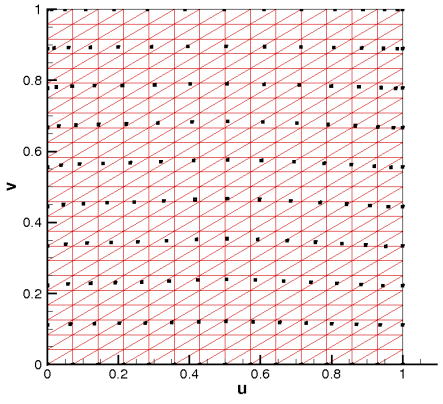
(b) Surface triangulation of the deformed geometry



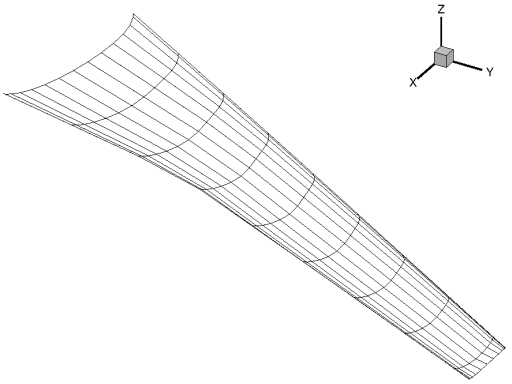
(c) Parametrization of the initial mesh



(d) Parametrization of the triangulation



(e) Overlaying of the parametrizations



(f) New surface mesh

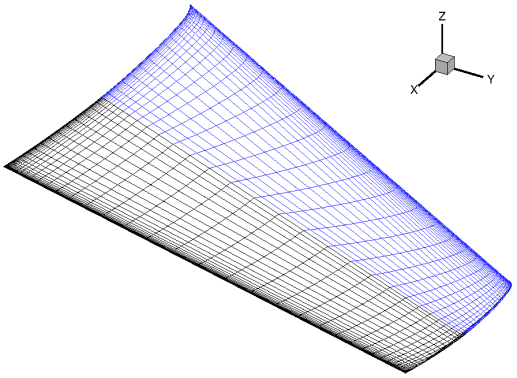
Figure 3.3: Obtaining a surface mesh from a triangulation of the new surface

uniform triangulation, as any pattern or distortion introduced by the tessellation will be transferred to the resulting surface mesh.

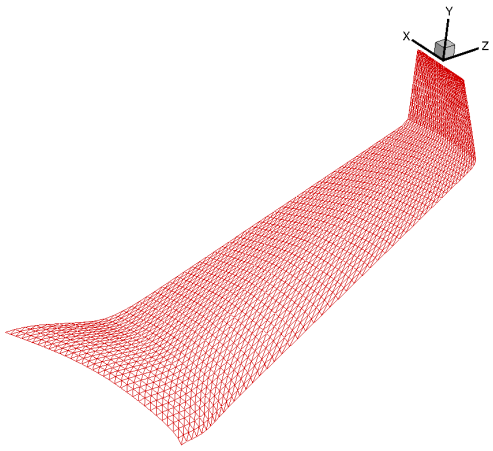
The second example, shown in Figure 3.4, involves the deformation of a dense surface mesh that is designed for viscous flow simulations. The tessellation, Figure 3.4(b), is uniform and the characteristics of the original surface mesh, Figure 3.4(a), such as the clustering around the edges and the spacing along the spanwise direction, are preserved in the deformed surface mesh, shown in Figure 3.4(d). Though, it may be hard to discern these similarities in such a drastic transformation: from a planar wing to a wing with a winglet. This example demonstrates the capability of the surface mesh movement algorithm in handling large deformation while preserving the mesh quality and characteristics of the original surface mesh.

The third example shows the mesh characteristic preservation of the algorithm more clearly, using the wing in Figure 2.2(a), and its CAPRI tessellation. Figure 3.5 shows the original and deformed mesh of the wing with the leading edge sweep reduced by 30% chord. It can be seen that the original and deformed mesh near the root, where the deformation is minimal, are almost identical. The change is smooth and gradual towards the tip to as it adapts to the new shape.

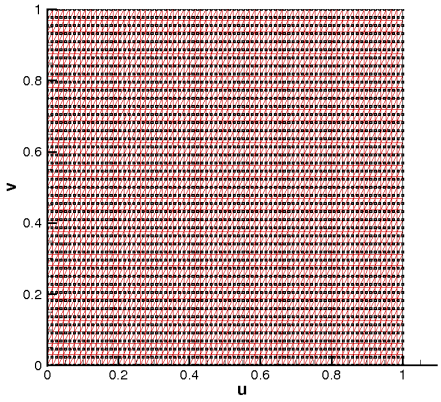
Finally, Figure 3.6 shows the same wing in the previous example experiencing a greater extent of deformation whereby the chord is increased by 30%, the wing thickness is reduced by 30% and the span is increased by 30% in addition to a decrease of 30% in the leading edge sweep. This example shows that CAPRI cannot always produce a tessellation with uniform triangles. Depending on the quality and shape of the CAD surface, there may be some distortions introduced, which affects the quality of the final surface mesh. This is seen in Figures 3.7(a) and 3.7(b), where the distortion of the CAPRI triangulation (red) results in non-smooth surface mesh (black). This necessitates the development of a smoothing algorithm that can correct such irregularities in the surface mesh. This is the subject of Chapter 4.



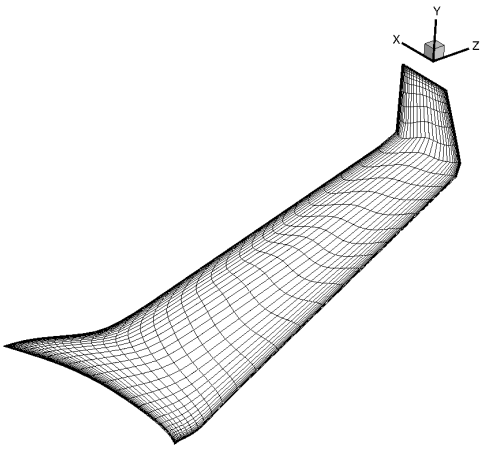
(a) Initial surface made up of 2 patches



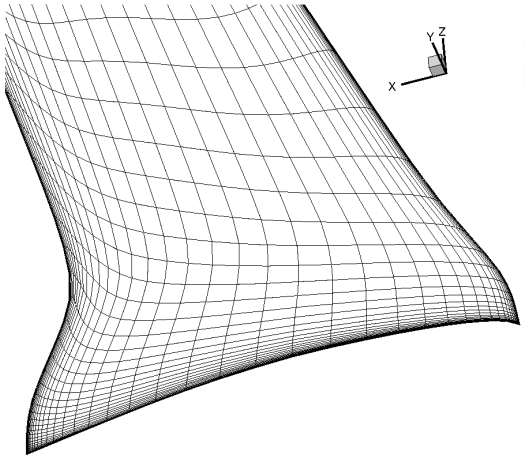
(b) Triangulation of the target surface



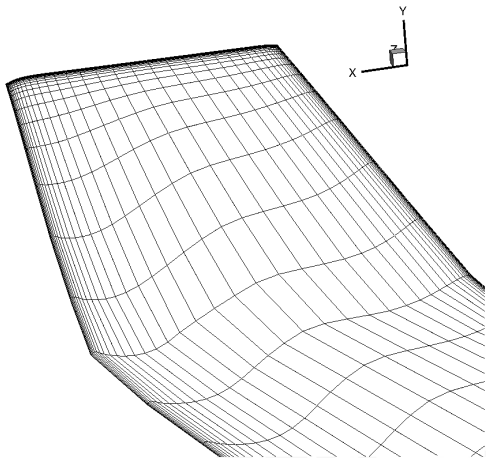
(c) Overlaying of the parametrizations



(d) New surface mesh

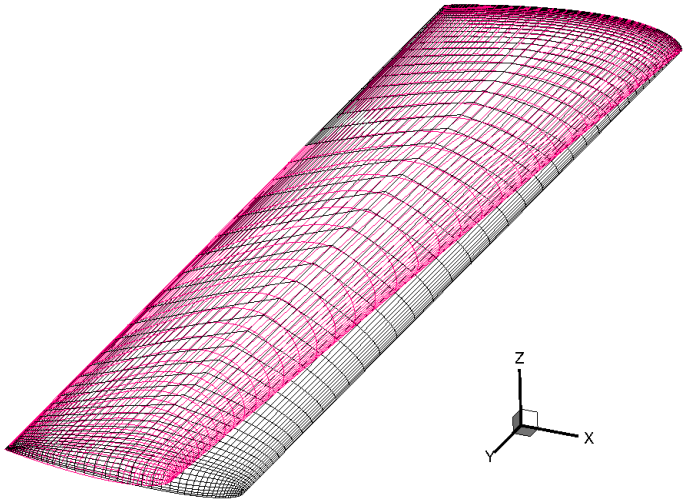


(e) New surface mesh at root

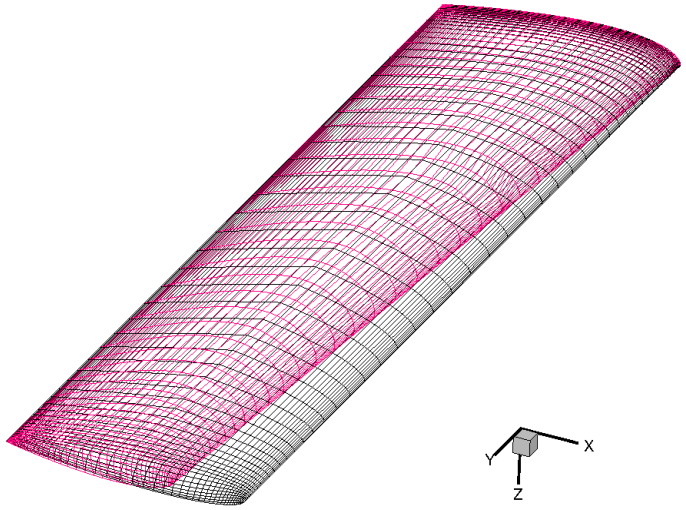


(f) New surface mesh at tip

Figure 3.4: Obtaining a surface mesh for a wing with a winglet

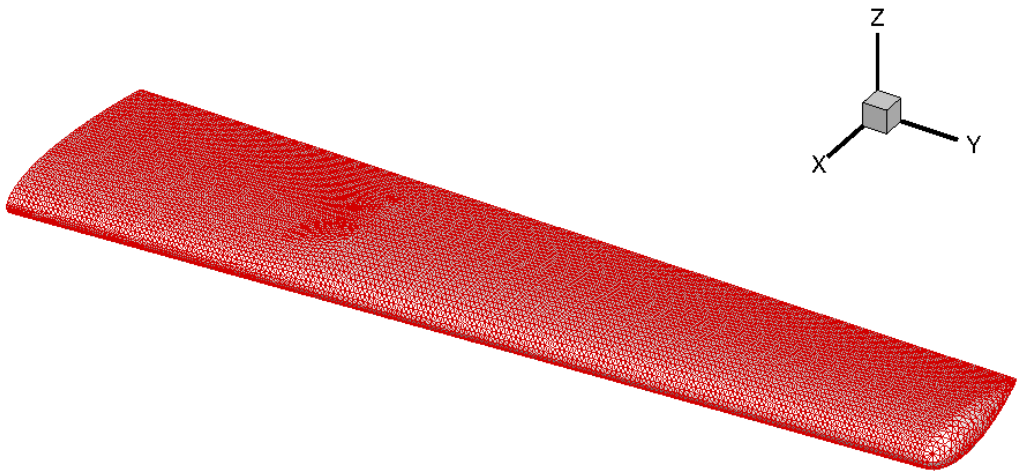


(a) Top surface

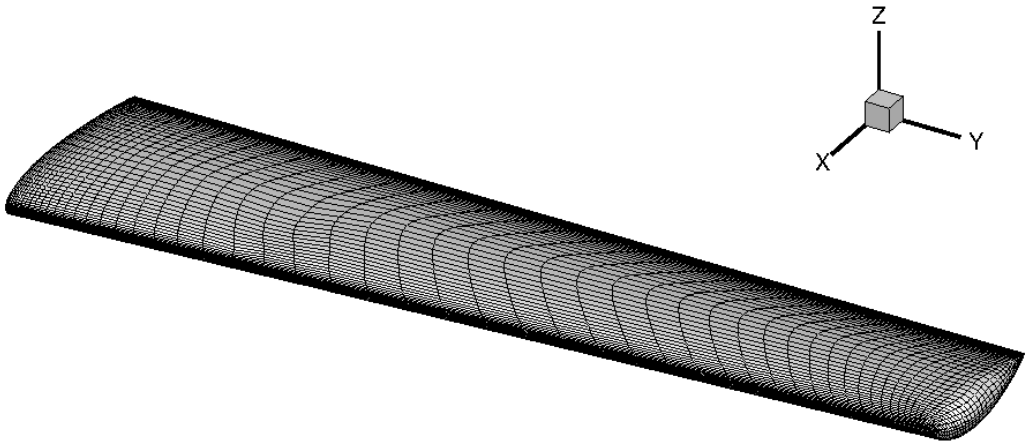


(b) Bottom surface

Figure 3.5: Comparing the mesh quality and characteristics of the original (black) and deformed (pink) meshes.

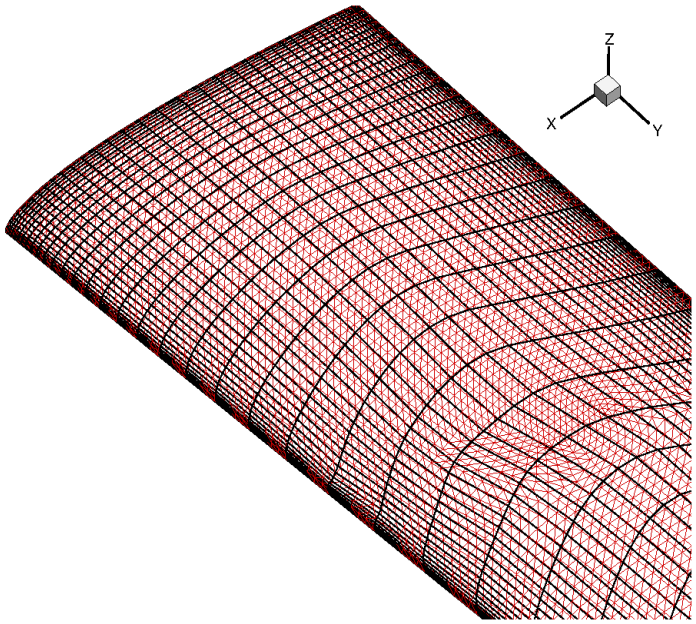


(a) CAPRI tessellation of the deformed wing

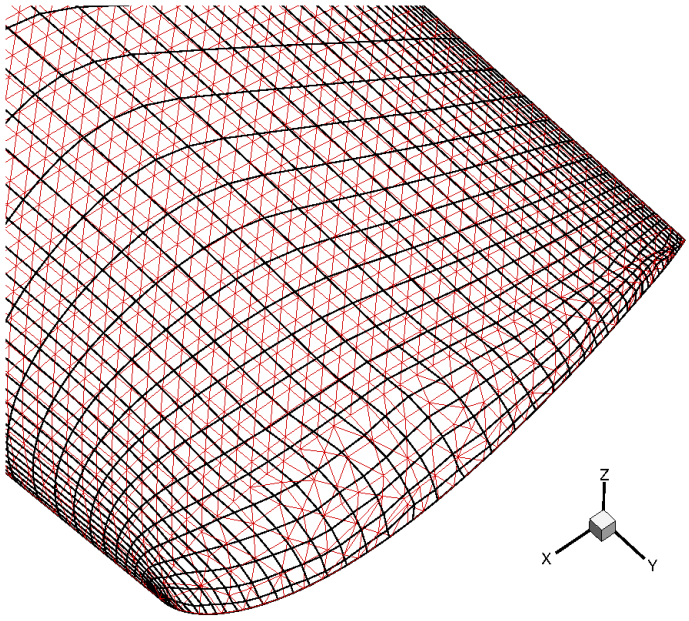


(b) Deformed surface mesh

Figure 3.6: The wing in Figure 3.5 is modified by +30% chord, −30% thickness, +30% span. Its leading edge is also swept back by 30% chord.



(a) Wing root



(b) Wing tip

Figure 3.7: Zoomed-in view of the root and tip region on the top surface of the wing with the surface mesh (black) overlaying the CAPRI triangulation (red). The surface mesh is slightly distorted where the triangulation is irregular.

4

Surface Mesh Smoothing

Since the quality of the deformed surface mesh depends on the quality of the triangulation, any irregularity in the angle or size of the triangles will introduce distortion in the mapping and consequently distortion in the final surface mesh, shown in Figures 4.1 and 4.2. To alleviate this problem, the final mesh is smoothed with an elliptic smoother in the parametric (2D) space before it is mapped back into the physical (3D) space. Smoothing in 2D is extremely efficient. Usually, no more than 4 iterations are required to achieve a smooth mesh.

4.1 Survey of Mesh Smoothing Techniques

There are a number of smoothing algorithms, mainly applied to triangulated meshes. The most computationally efficient and easy to implement method is the Laplacian scheme where a node is moved to the centroid of its neighboring vertices [33]. This method operates heuristically, and has no control on mesh quality and can produce distortion, shrinkage, and inverted or invalid elements. Many improved variants [96, 109] of the Laplacian smoothing technique are available that address the shortcomings of the original method. For example, Vollmer *et al.* [163] introduced a method to prevent the effect of shrinkage by pushing the vertices of the smoothed mesh back towards their previous locations. Shontz *et al.* [150] used local weights based on element lengths to improve the mesh quality. "Smart" Laplacian methods [14] have element distortion metrics that are checked before the node is moved. This method is effective in avoiding inverted elements, but at the expense of increased computational cost. Optimization-based smoothing methods use the quality measures to define cost functions [41]. These methods guarantee an

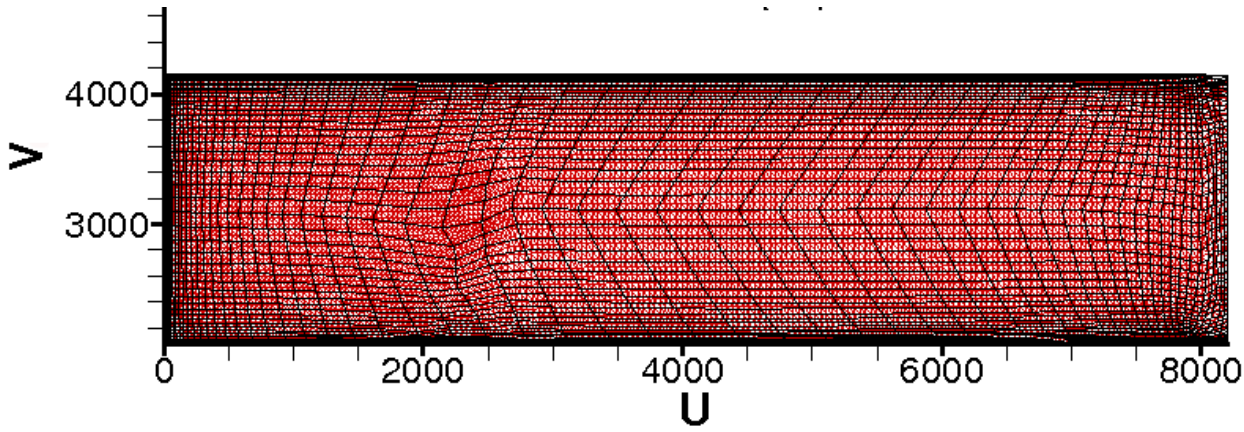
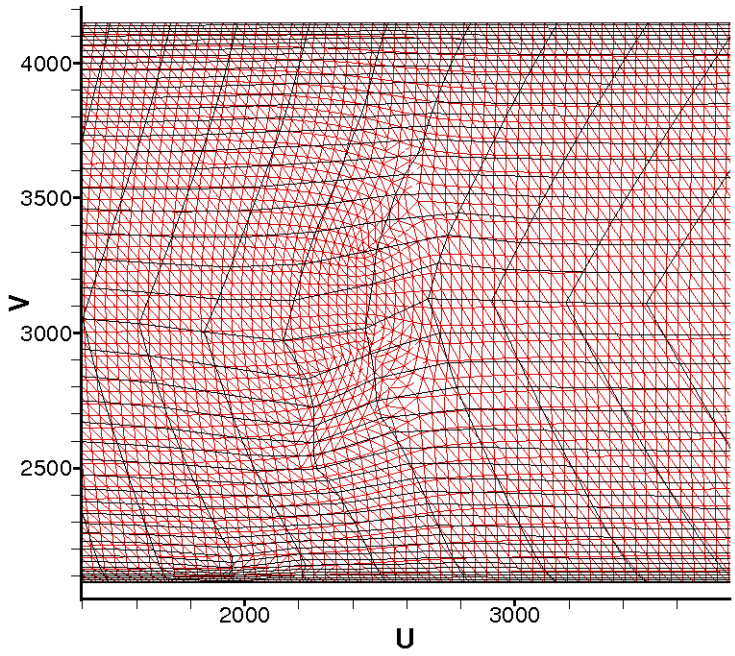


Figure 4.1: Surface mesh (black) overlaying the CAPRI triangulation(red), in the transformed CAD parametric space, of the wing in Figure 3.6(b) whose LE has been swept back by 30% chord. The surface mesh is slightly distorted where the triangulation is irregular.

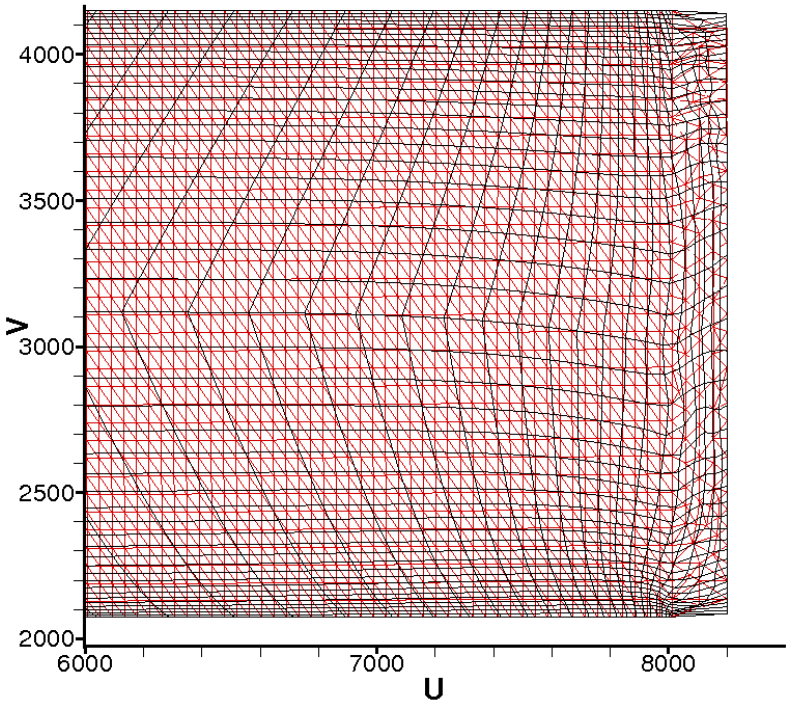
improvement in mesh quality. However, the computational cost is much higher. For example, for a 2D triangular mesh, the cost of an optimization-based smoothing method is about five times that of a "smart" Laplacian, and 30-40 times that of the Laplacian [14]. Hybrid algorithms combining smart Laplacian methods with optimization-based techniques have also been developed [14, 18, 40] but with limited success.

Physics-based smoothing methods treat smoothing as solving a physics problem that models a torsion-spring system [167, 175], electrical system [105], or force-based system [11, 29]. The force-directed method assumes that on each vertex a certain force is applied that moves the vertex relative to its neighbors so that the shapes of its incident elements are improved. The final stable configuration often corresponds to a graph with good global properties.

In structured meshing, the Winslow elliptic smoother, first proposed in 1967 by Winslow [165], is widely used because it offers a high degree of control over grid point spacing and grid line angularity, and guarantees invertibility in two dimensions [83]. It has been incorporated in many commercial mesh-generation packages, including Gridgen [154]. The Winslow equations are derived from Poisson's Equation for a parameter distribution over a region. Since a mapping from the physical domain to the parametric domain, where the smoothing equations are solved, does not exist for unstructured meshes, the Winslow equations have to be solved using a finite-element method or fi-



(a) Region near the wing root



(b) Wing tip region

Figure 4.2: Close-up view of the distorted regions in the mesh shown in Figure 4.1

nite differences [89], Karman *et al.* [83] later extended the method to 3D. The Winslow smoothing algorithm is chosen here for its numerical efficiency and robustness. Its derivation is given in Section 4.2.

In order to confine the mesh points to the CAD surface during smoothing, this operation is performed in the parametric space. Since each CAD face has its own parameterization, smoothing in the CAD parametric space to is limited to node movements within each face. Thus, we have to transform the CAD faces onto one coordinate system or use a different parametric space.

4.2 Winslow Smoothing Algorithm

The Winslow equations are derived from a Laplacian operator applied to the computational coordinates, shown in (4.1).

$$\nabla^2 \xi = \xi_{xx} + \xi_{yy} = 0 \tag{4.1}$$

$$\nabla^2 \eta = \eta_{xx} + \eta_{yy} = 0 \tag{4.2}$$

The equations describe a smooth distribution of computational coordinates (ξ, η) in physical space (x, y) [80]. The Laplace equations satisfy the max-min property, which states that the parameter on the interior domain will not exceed the values on the boundary, i.e the grid lines will not cross. The Winslow equations (4.3) result from the transformation of the unknown variables x and y to known variables ξ and η in the computational space:

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0 \tag{4.3}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0 \tag{4.4}$$

$$\alpha = x_\eta^2 + y_\eta^2 \tag{4.5}$$

$$\beta = x_\xi x_\eta + x_\xi x_\eta \tag{4.6}$$

$$\gamma = x_\xi^2 + y_\xi^2 \tag{4.7}$$

For a 2D structured mesh with mesh coordinates at the integer indices (i, j), (4.1) becomes

$$\alpha_{\xi,\eta} \frac{\partial^2 f}{\partial \xi^2} - 2\beta_{\xi,\eta} \frac{\partial^2 f}{\partial \xi \partial \eta} + \gamma_{\xi,\eta} \frac{\partial^2 f}{\partial \eta^2} = 0 \tag{4.8}$$

which can be discretized as

$$\alpha(f_{i+1,j} - 2f_{i,j} + f_{i-1,j}) - \frac{\beta}{2}(f_{i+1,j+1} - f_{i-1,j+1} - f_{i+1,j-1} + f_{i-1,j-1}) + \gamma(f_{i,j+1} - 2f_{i,j} + f_{i,j-1}) = 0 \tag{4.9}$$

where

$$\alpha = \left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2 \tag{4.10}$$

$$\approx \frac{1}{4}\left[(x_{i,j+1} - x_{i,j-1})^2 + (y_{i,j+1} - y_{i,j-1})^2\right] \tag{4.11}$$

$$\beta = \frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} \tag{4.12}$$

$$\approx \frac{1}{4}\left[(x_{i+1,j} - x_{i-1,j})(x_{i,j+1} - x_{i,j-1}) + (y_{i+1,j} - y_{i-1,j})(y_{i,j+1} - y_{i,j-1})\right] \tag{4.13}$$

$$\gamma = \left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2 \tag{4.14}$$

$$\approx \frac{1}{4}\left[(x_{i+1,j} - x_{i-1,j})^2 + (y_{i+1,j} - y_{i-1,j})^2\right] \tag{4.15}$$

and f is either x or y. The solution to the coordinate position at (i, j) can be obtained by solving (4.9) for $f_{i,j}$. Smoothing is applied to the mesh in the CAD uv-space, where each CAD surface patch exists in a coordinate system that is independent of the other patches. During smoothing, mesh points are restricted to move within the patch boundary to ensure that they remain on the CAD surface. Figures 4.3 shows the CAPRI tessellation of a wing model (Figure 2.2(a)) in CAD parametric coordinates, showing different CAD faces existing as independent entities (i.e. not necessarily connected at the boundary where the connection exists in physical space). In order to smooth the mesh across patch boundaries, it is necessary to consolidate all of the patches in different coordinate systems into one. This process is called coordinate transformation, which is described in Section 4.3.

4.3 Coordinate Transformation

Coordinate transformation is the process of determining and applying the relationship between two sets of points on different coordinate systems [52]. It is one of the fundamental operations in computer graphics [39], and has applications in photogrammetry [97, 103] and geographic information system [52] and others. To consolidate all the CAD surface patches, a transformation can be applied to the faces so that they all exist in one coordinate system. This is possible because neighboring patches share a common boundary. Along this boundary, CAPRI supplies two sets of overlapping nodes belonging to each of the neighbors. By applying a coordinate transformation, we can establish a relationship between the two systems to fit one set of coordinates into the other.

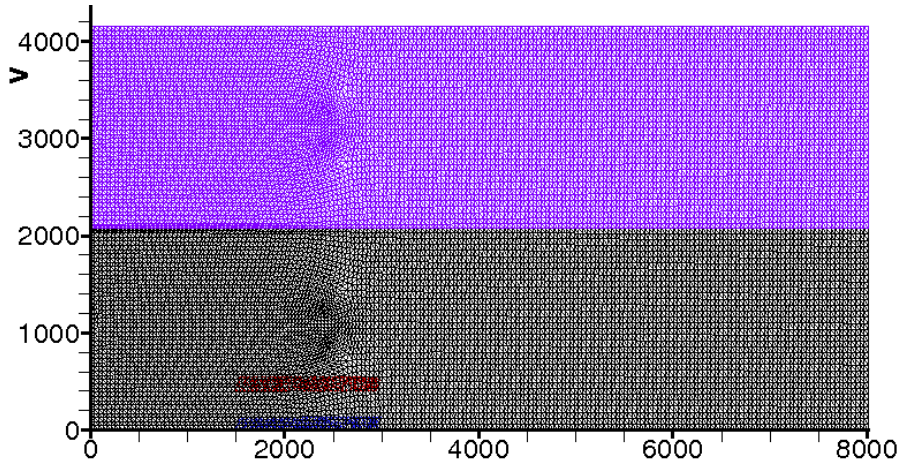


Figure 4.3: CAPRI tessellation of the CAD model, Figure 2.2(a), in CAD uv-space, color-coded the same way as its corresponding tessellation in physical space shown in Figure 2.2(b). Each CAD surface has its own coordinate system and the faces are not necessary connected at the common boundary in physical space.

In general, there are six parameters that characterize a two-dimensional coordinate system. These parameters are:

1. a_0 = translation of the origin in the x direction
2. b_0 = translation of the origin in the y direction
3. φ = rotation of the axes of one coordinate system with respect to the other
4. s_x = scale change in the x axis
5. s_y = scale change in the y axis
6. ε = nonorthorgonality angle between the axes of the two coordinate systems

Basic coordinate transformation methods can be classified based on the number of parameters involved. The most simple and (rigid) transformation is translation and rotation, involving only three parameters: $a_0, b_0, \varphi, s_x = s_y = 1$, where the coordinates in the new system, (X, Y), can be calculated from the coordinates in the old system, (x, y), as follows:

$$\begin{aligned} X &= a_0 + x \cdot \cos\varphi - y \cdot \sin\varphi \\ Y &= b_0 + x \cdot \sin\varphi + y \cdot \cos\varphi \end{aligned} \tag{4.16}$$

A similarity transformation, also known as a conformal or isogonal transformation, involves two systems that may have different scales, and is defined by four parameters: $a_0, b_0, \varphi, s_x = s_y$:

$$\begin{aligned} X &= a_0 + x \cdot \cos\varphi - s_x \cdot y \cdot \sin\varphi \\ Y &= b_0 + x \cdot \sin\varphi + s_y \cdot y \cdot \cos\varphi \end{aligned} \tag{4.17}$$

Equation (4.17) can be expressed in four linear terms (a_0, b_0, a_1, b_1):

$$\begin{aligned} X &= a_0 + a_1 x - b_1 y \\ Y &= b_0 + b_1 x + a_1 y \end{aligned} \tag{4.18}$$

When one system has a different scales in X and Y, the transformation involves five parameters $a_0, b_0, \varphi, s_x, s_y$ and is called an orthogonal affine transformation:

$$\begin{aligned} X &= a_0 + s_x \cdot x \cdot \cos\varphi - s_y \cdot y \cdot \sin\varphi \\ Y &= b_0 + s_x \cdot x \cdot \sin\varphi + s_y \cdot y \cdot \cos\varphi \end{aligned} \tag{4.19}$$

Finally, when one system has unknown characteristics, all six parameters are used in the transformation, and the process is called an affine transformation:

$$\begin{aligned} X &= a_0 + s_x \cdot x \cdot \cos\varphi - s_y \cdot y \cdot (\sin\varphi + \sin\varepsilon \cdot \cos\varphi) \\ Y &= b_0 + s_x \cdot x \cdot \sin\varphi + s_y \cdot y \cdot (\cos\varphi - \sin\varepsilon \cdot \sin\varphi) \end{aligned} \tag{4.20}$$

An affine transformation does not preserve orthogonality but it preserves parallelism, i.e. parallel lines remain parallel lines. It is also a linear transformation because (4.20) can be expressed in six linear terms $(a_0, b_0, a_1, b_1, a_2, b_2)$:

$$\begin{aligned} X &= a_0 + a_1 x + a_2 y \\ Y &= b_0 + b_1 x + b_2 y \end{aligned} \tag{4.21}$$

Since four- and six-parameter transformations can be modified to become sets of linear equations, the formulas for computing a least-squares-based transformation are simple. For the three- and five-parameter transformations, the least squares solution becomes nonlinear and requires iterations until the solution converges. After experimenting with the linear transformations (conformal and affine), obtained using all of the common points between neighboring patches, it was determined that the patches can be transformed using conformal transformation. In many instances, a coordinate transformation brings the neighboring patch to the target coordinate system but the position of the nodes needs

to be flipped about the line of attachment between the two patches so that the patches do not fold onto each other. The reflection about an arbitrary line is described in Section 4.4. Coordinate transformation is usually carried out using least squares because it provides a best fit between the coordinate systems by analyzing simultaneously all the common points. The method of least squares is described in Section 4.5. The result of applying the above transformations is shown in Figure 4.1. The original CAD parametrization can be seen in Figure 4.3.

4.4 Reflection About an Arbitrary Line

The equation of the line about which the reflection is calculated is obtained using a least-squares method. The transformation matrix for reflection in either $x-$ or $y-$axis is:

$$\text{Refl}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Refl}_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.22}$$

In general, the reflection about an arbitrary line is obtained by transforming the line to one of the axes, reflecting in that axis, and then taking the inverse of the first transformation. More specifically, the reflection is accomplished in five steps, as follows:

1. Translate the line to intersect the origin. For an arbitrary line $Ax + By + C = 0$, the translation that maps the y-intersection, $(0, -C/B)$, to the origin is:

$$\text{Transl}_{(A,B,C)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -C/B \\ 0 & 0 & 1 \end{bmatrix} \tag{4.23}$$

2. Rotate the line about the origin through an angle -θ to coincide with the x-axis. The transformation matrix that describes this transformation is:

$$\text{Rot}_{(A,B,C)} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.24}$$

3. Apply a reflection in the x-axis.

4. Rotate about the origin by θ

5. Translate by $(0, C/B)$

The above transformations can be concatenated into one reflection matrix

$$\text{Refl}_{(A,B,C)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -C/B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & C/B \\ 0 & 0 & 1 \end{bmatrix} \tag{4.25}$$

$$\text{Refl}_{(A,B,C)} = \begin{bmatrix} \cos^2\theta - \sin^2\theta & 2\sin\theta\cos\theta & \frac{2C}{B}\sin\theta\cos\theta \\ 2\sin\theta\cos\theta & \sin^2\theta - \cos^2\theta & -\frac{2C}{B}\cos^2\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

Since $\tan\theta = -A/B$, it follows that $\cos^2\theta = 1/(1+\tan^2\theta) = B^2/(A^2+B^2)$ and $\sin^2\theta = 1 - \cos^2\theta = A^2/(A^2+B^2)$, $\sin\theta\cos\theta = \tan\theta\cos^2\theta = -AB/(A^2+B^2)$. Substituting these expressions into (4.26) yields

$$\text{Refl}_{(A,B,C)} = \begin{bmatrix} \frac{B^2-A^2}{A^2+B^2} & -\frac{2AB}{A^2+B^2} & -\frac{2AC}{A^2+B^2} \\ -\frac{2AB}{A^2+B^2} & \frac{A^2-B^2}{A^2+B^2} & -\frac{2BC}{A^2+B^2} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.27}$$

The above matrix can be scaled by a factor of A^2+B^2 to remove all the denominators in the entries to yield

$$\text{Refl}_{(A,B,C)} = \begin{bmatrix} B^2-A^2 & -2AB & -2AC \\ -2AB & A^2-B^2 & -2BC \\ 0 & 0 & A^2+B^2 \end{bmatrix} \tag{4.28}$$

4.5 Least Squares Method

Least squares techniques are applied in instances where there are more data than the minimum necessary to obtain a unique solution. With i common points, a matrix representation of (4.18) is

$$\mathbf{Y} = \mathbf{AX} \tag{4.29}$$

where

$$\mathbf{X} = \begin{bmatrix} a_0 \\ b_0 \\ a_1 \\ b_1 \end{bmatrix}; \qquad \mathbf{A} = \begin{bmatrix} 1 & 0 & x_1 & -y_1 \\ 0 & 1 & x_1 & y_1 \\ 1 & 0 & x_2 & -y_2 \\ 0 & 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_i & -y_i \\ 0 & 1 & x_i & -y_i \end{bmatrix}; \qquad \mathbf{Y} = \begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ \vdots \\ X_i \\ Y_i \end{bmatrix}; \tag{4.30}$$

The solution is found by minimizing the squared norm of the residual, $r = \mathbf{Y} - \mathbf{AX}$

$$\begin{aligned} \|r\|_2^2 = r^T r &= (\mathbf{Y} - \mathbf{AX})^T (\mathbf{Y} - \mathbf{AX}) \\ &= \mathbf{Y}^T\mathbf{Y} - (\mathbf{AX})^T\mathbf{Y} - \mathbf{Y}^T\mathbf{AX} + \mathbf{X}^T\mathbf{A}^T\mathbf{AX} \\ &= \mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{AX} + \mathbf{X}^T\mathbf{A}^T\mathbf{AX} \end{aligned} \tag{4.31}$$

Minimizing the residual with respect to the unknown parameters requires

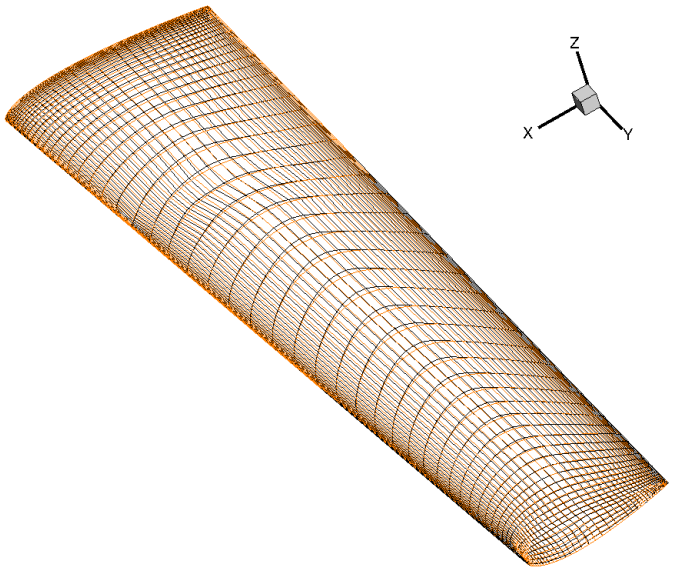
$$\frac{\partial r}{\partial \mathbf{X}} = -2\mathbf{A}^T\mathbf{Y} + 2\mathbf{A}^T\mathbf{AX} = 0 \tag{4.32}$$

or

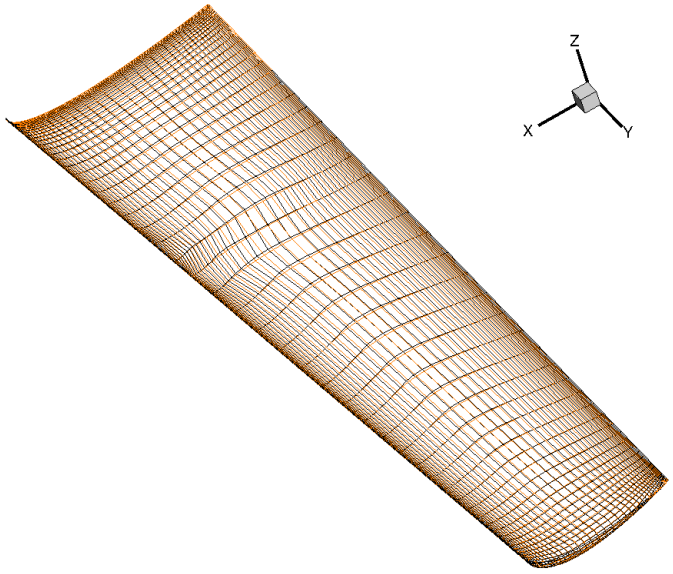
$$\mathbf{X} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \tag{4.33}$$

4.6 Surface Mesh Smoothing Example

Figures 4.4 and 4.5 compare the original and smoothed mesh after 2 iterations. In this case, the leading edge of the wing shown in Figure 2.2(a) has been swept back by 30% chord.

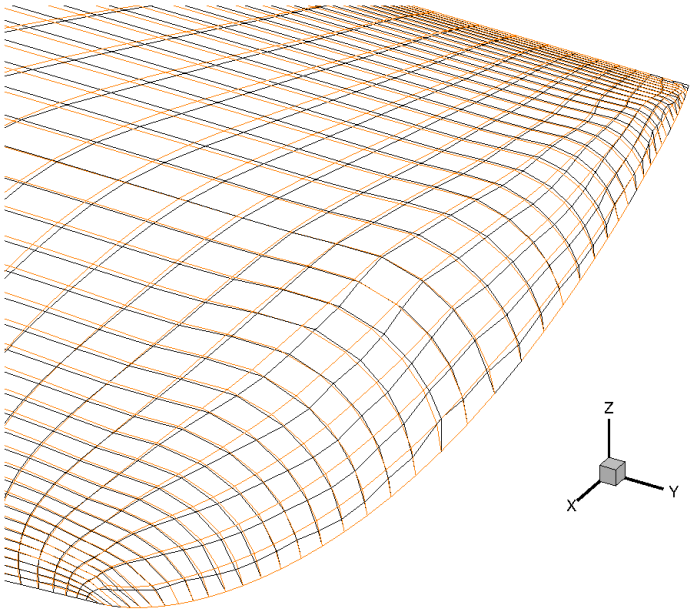


(a) Top surface

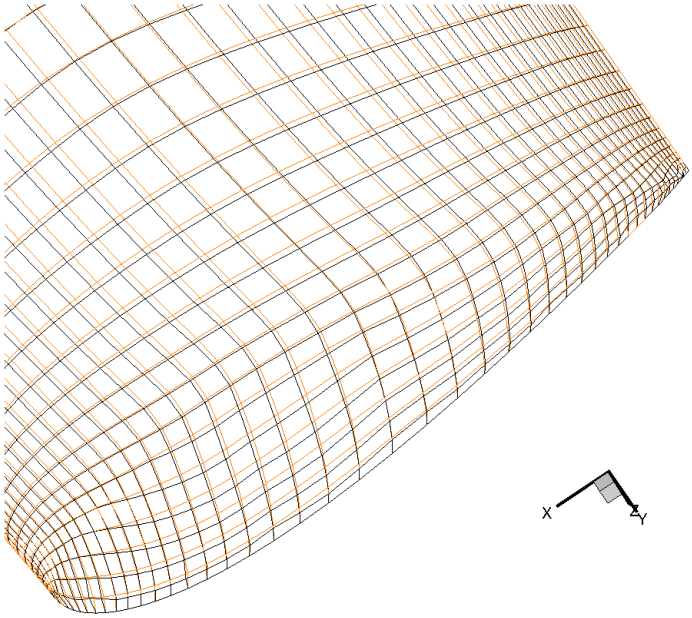


(b) Bottom surface

Figure 4.4: Smoothed surface mesh (orange) overlaying the unsmoothed mesh (black) shown in Figure 3.6(b)



(a) Top surface



(b) Bottom surface

Figure 4.5: Zoomed-in view of the tip region of the smoothed surface mesh (orange) overlaying the unsmoothed mesh (black) shown in Figure 3.6(b)

5 Gradient Calculations

5.1 Aerodynamic Shape Optimization

The aerodynamic shape optimization problem begins with a geometry whose shape can be controlled by a set of parameters called design variables. These geometric design variables are usually a subset of parameters used to create the geometry. Additional design variables such as angle of attack and Mach number can be specified to fully describe the design space. It is important to include a sufficient number of design variables from which an optimal design can be obtained. The goal of the optimization procedure is to obtain a final set of design variables such that the objective is minimized while all constraints are satisfied.

There are a number of optimization algorithms available [19, 67, 75, 118]. They can be divided into two broad categories: gradient-based and gradient-free methods. The choice of a suitable algorithm depends on the number of design variables, number of objectives and constraints, and the smoothness of the design space [176]. Gradient-free methods, such as genetic algorithms [111], do not require gradient information and are particularly suitable for problems with discrete design variables and problems with discontinuous and noisy design space. Gradient-based optimization algorithms require not only the value of the objective function, but also its derivatives with respect to the design parameters. Gradient-based optimization methods interpret first and sometimes second derivatives to take steps in the design space that will lead to the optimum [100]. They are effective in reaching the optimum with a relatively small number of function evaluations [76]. Unfortunately, only convergence to a local minimum is guaranteed. Some researchers have devised and successfully applied methods that combine elements of gradient-free

and gradient based methods [8, 19, 107] in order to ensure a global optimum is found efficiently. In this thesis, we use the gradient-based optimization algorithm SNOPT [48], as it enables the use of large number of design variables which is typically required in aircraft design, especially towards the end of the preliminary design stage [100].

In gradient-based optimization, the derivative calculation is often the most costly step in the optimization cycle. Hence, it is important to use a method that is both accurate and efficient for sensitivity analysis. A clear choice is the adjoint method [6, 12, 47, 55, 64, 76, 87, 100, 102, 108, 110, 114, 115, 117, 120, 130–133, 159] which is particularly suitable for aerodynamic shape optimization since the cost of computing the gradient of a given objective is independent of the number of design variables. Nielsen and Park [110] introduced the augmented adjoint formulation which explicitly includes the mesh sensitivities in the adjoint equations. Using this method, the total cost to obtain the gradient amounts to the cost of one flow and mesh movement solutions and two linear systems of adjoint equations. The augmented adjoint formulation is used in this thesis and its derivation is given in Section 5.2.

5.2 Augmented Adjoint Formulation

The optimization problem can be posed as minimizing the design objective, $\mathcal{J}$, with respect to the design variables, $\mathcal{X}$, the grid, G, and the conservative flow field variables, Q, within a feasible region of the design space Ω, subject to the constraint that the flow solver and grid movement equations must have converged (i.e the flow residual, $\mathcal{R}$, and the grid residual, r, are zero):

$$\begin{aligned} \min_{X} \quad & \mathcal{J}(\mathcal{X}, Q, G) \\ \text{s.t.} \quad & \mathcal{R}(\mathcal{X}, Q, G) = 0 \qquad \forall \quad \mathcal{X} \in \Omega \\ & r(G, G_{surf}(\mathcal{X})) = 0 \end{aligned} \tag{5.1}$$

where G_{surf} is the surface mesh of the solid boundary.

In general, there may be other constraints, $\mathcal{C}(\mathcal{X}, Q)$, associated with the design variables or flow conditions; these are considered separately by the optimizer, and are not shown here as part of this augmented adjoint formulation. To enforce the constraints in (5.1), let the Lagrangian, $\mathcal{L}$, be defined as follows:

$$\mathcal{L}(\mathcal{X}, Q, G, \psi, \lambda) = \mathcal{J}(\mathcal{X}, Q, G) + \psi^{\mathrm{T}}\mathcal{R}(\mathcal{X}, Q, G) + \lambda^{\mathrm{T}} r(G, G_{surf}(\mathcal{X})) \tag{5.2}$$

where λ and ψ are Lagrange multipliers. Setting each of the partial derivatives of the Lagrangian to zero then provides optimality conditions for the objective function:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 = r \tag{5.3}$$

$$\frac{\partial \mathcal{L}}{\partial \psi} = 0 = \mathcal{R} \tag{5.4}$$

$$\frac{\partial \mathcal{L}}{\partial Q} = 0 = \frac{\partial \mathcal{J}}{\partial Q} + \psi^{\mathrm{T}} \frac{\partial \mathcal{R}}{\partial Q} \tag{5.5}$$

$$\frac{\partial \mathcal{L}}{\partial G} = 0 = \frac{\partial \mathcal{J}}{\partial G} + \lambda^{\mathrm{T}} \frac{\partial r}{\partial G} + \psi^{\mathrm{T}} \frac{\partial \mathcal{R}}{\partial G} \tag{5.6}$$

$$\frac{\partial \mathcal{L}}{\partial \mathcal{X}} = \frac{\partial \mathcal{J}}{\partial \mathcal{X}} + \psi^{\mathrm{T}} \frac{\partial \mathcal{R}}{\partial \mathcal{X}} + \lambda^{\mathrm{T}} \frac{\partial r}{\partial G_{surf}} \frac{\partial G_{surf}}{\partial \mathcal{X}} \tag{5.7}$$

Equations (5.3) and (5.4) represent the residuals of the linear elasticity mesh movement algorithm and the flow, respectively, while (5.5) and (5.6) represent the discrete adjoint equations for the flow simulation and mesh movement, respectively.

A number of approaches can be taken to find a solution to these optimality conditions to minimize the Lagrangian, including possibilities such as solving for all of the variables simultaneously—the approach that defines simultaneous analysis and design (SAND) [119]. Using this one-shot approach, a large-scale solver sets the entire gradient of the Lagrangian to zero. The sequential approach taken here delegates the computation to the existing specialized solvers. For a given $\mathcal{X}$, (5.3) can be solved using the mesh movement code to yield G. Using that solution, the flow solver can be used to solve (5.4) to yield Q. The linear systems in (5.5)–(5.6) can then be solved in the order they appear to give ψ and each λ. The size of these linear systems is independent of the number of design variables. In cases where nonlinear aerodynamic constraints, such as lift or pitching moment are used, additional adjoint gradient computation has to be performed for each of the constraints.

The evaluation of $\frac{\partial G_{surf}}{\partial \mathcal{X}}$ in (5.7), which is the sensitivity of the surface grid with respect to the design variables, can be performed analytically if the source code of the CAD system is available. However, this is not the case for commercial CAD systems. Thus, it has to be determined using finite differences or semi-analytically, as discussed in Section 5.3.

5.3 Survey of Methods for Calculating the Surface Sensitivity Derivatives

The surface sensitivity derivatives can be obtained by differentiating the CAD modeler if the source code is available, such as in the case of the open-source CAD engine OpenCASCADE [121]. However, differentiating the source code is not an easy task due to its size and complexity [79]. At present, OpenCASCADE contains over $14,000$ classes, making differentiation a quite difficult undertaking. Thus far, only Kleinveld *et al.* [88] were successful in differentiating their in-house CAD modeler, Ganimede (Geometry ANd Inherent MEsh DEformation), to obtain analytical sensitivity derivatives.

Yu *et al.* [170] propose an alternate representation of CAD models using NURBS (Non-uniform rational B-splines), a form which CAD systems use to export their geometries. The derivatives of the surface with respect to the design variables in any given location on the surface are obtained by applying automatic differentiation (AD) in reverse mode to a generic NURBS implementation. Xu *et al.* [149] generalized this approach to 3D geometries consisting of multiple NURBS patches. They introduced constraints for geometric continuity across NURBS patch interfaces to maintain a desired level of continuity of tangency and curvature between adjacent NURBS patches when a control point on or near a patch interface is displaced. Jones *et al.* [79] infer the parameter sensitivity by tracing the evolution of the hierarchical associativity of CAD features. Robinson *et al.* [133] use the design velocity field approach to evaluate the surface displacement over a faceted coarse surface mesh and then interpolate on the boundary of the fine computational mesh. The use of design velocity for optimization is well established. It can be computed for meshes of the boundary [157, 174], or directly from the geometric CAD model [16, 61]. The design velocity, V_n, is a measure of the normal displacement of the model boundary caused by a modification of the design parameter. The change in objective function caused by the perturbation of the design parameter can be predicted using the boundary method expressed in terms of design sensitivity and is described by Choi and Kim [20, 21]. This method assumes that the change in performance is of first order, which is valid for small boundary movements that are continuous over the boundary [133]. If the relationship between the design variables and nodal coordinates is linear, then the velocity field needs to be calculated once, whereas if the relation is nonlinear, then the velocity field must be updated at each design iteration [17].

More recently, Haimes and Dannenhoffer [59] created a browser-based geometry construction and manipulation tool called Engineering Sketch Pad, which, in many ways, mirrors the functionality of modern parametric commercial CAD systems, as it is built upon OpenCSM (which is the open-source constructive solid modeler that is in turn built upon EGADS - the Engineering Geometry Aerospace Design System, and OpenCASCADE), that is able to provide analytic parameter sensitivity. However, at the moment, the tool is still in its early stages of development.

Typically, the effect of each design parameter is determined using the finite-difference approach [3, 22, 114–116, 166]. Using this approach, the number of required geometry and surface mesh deformation scales proportionally to the number of design variables. Furthermore, in a Sequential Quadratic Programming optimizer such as SNOPT, during a line search, the geometry is generally perturbed 3 more times to construct a quadratic fit along the direction of the gradient. The computational cost can become excessive for problems with a large numbers of design variables. Moreover, choosing an optimal step size to avoid truncation or round-off errors can be challenging as it varies with each design variable and with each design cycle [135]. The finite-difference approach also requires the topology of the model to remain constant, which can be hard to achieve in some cases. Despite these issues, this approach is general and relatively straightforward to implement.

5.4 Computation of Surface Sensivity

In this work, we choose the method of Nemec and Aftosmis [113, 114] for its generality and ease of implementation. The procedure involves the generation of a surface grid for the CAD model to reflect the current values of the design variables, thereby obtaining a baseline model. For each CAD face, the CAD parameterized (u, v) coordinates of the surface grid are normalized such that $u, v \in [0, 1]$. Then two additional model regenerations and surface mesh movements are obtained for each design variable, which correspond to the plus and minus perturbations. The normalized (u, v) values on the perturbed model are re-scaled by their new (u, v) range from which the corresponding physical coordinates (x, y, z) can be obtained from the CAD system. This procedure is efficient since the query is done in the parameter space, assuming that the face topologies of baseline and perturbed models are the same. This is a reasonable assumption considering the size of the perturbation.

The size of the perturbation is chosen based on trial and error. It is 0.002% for most design variables except control point and sectional displacement design variables where a value of 0.0002 mm is used. The accuracy of the derivative with respect to the latter design variables seems to be more sensitive to the step size.

5.5 Verification of Gradient Accuracy

The total gradient calculated using the adjoint method, i.e. $\frac{\partial \mathcal{L}}{\partial \mathcal{X}}$ in Eq. (5.7), is compared with that calculated using central differencing to ensure that it is sufficiently accurate for gradient-based optimization algorithms. The gradient accuracy verification test problem is a lift-constrained drag-minimization of a rectangular NACA 0018 wing with a root chord of 2.0 m and span of 8.0 m (which are non-dimensionalized by the root chord in the flow solver). The Mach number is $M = 0.50$. The design variables are the linear twist, τ, of the wing and the angle of attack, ω. This problem is formulated as:

$$\begin{aligned} \min_{\omega,\tau} \quad & C_D S \\ \text{s.t.} \quad & C_L S/(C_L S)_{ref} = 1 \end{aligned} \tag{5.8}$$

where C_L and C_D are the coefficients of lift and drag, respectively, and S the wetted planform area of the wing. The reference or target lift, $(C_L S)_{ref}$, is set to the initial value of 1.0153, and $S_{ref} = 4.249$ units squared.

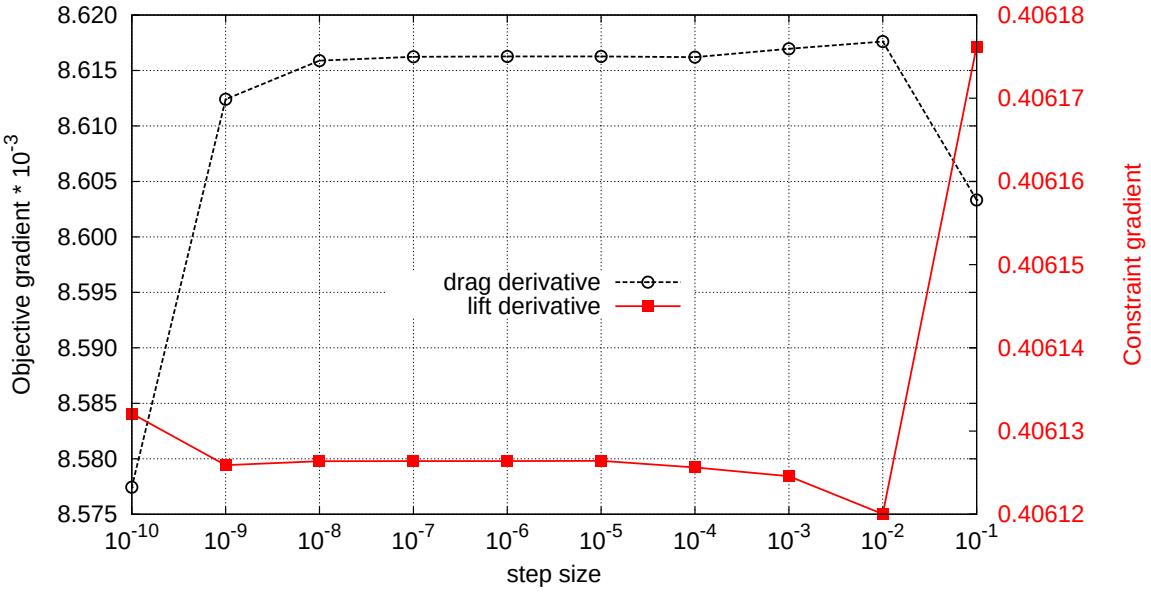
The second-order finite-difference approximation of a given objective is

$$\frac{d\mathcal{J}}{d\mathcal{X}} = \frac{\mathcal{J}(\mathcal{X} + \epsilon\mathcal{X}) - \mathcal{J}(\mathcal{X} - \epsilon\mathcal{X})}{2\epsilon} + \mathcal{O}(\epsilon^2) \tag{5.9}$$

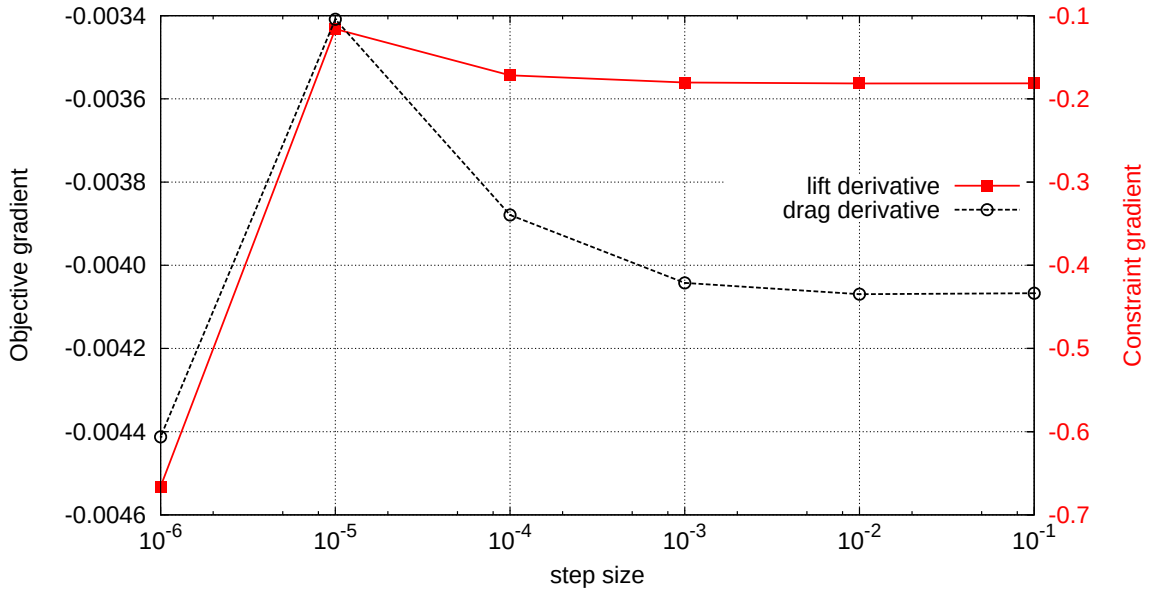
where ϵ is the finite-difference step size.

A step size study was conducted to determine the optimum step size for each design variable using the initial conditions, where the twist is $-0.5°$ and angle of attack is $2.75°$. Figure 5.1 shows the total gradient for the objective and constraint as a function of the step size used. Based on this study, step sizes of 10^{-2} and 10^{-6} are chosen for the twist and angle of attack design variables, respectively, to obtain the finite-difference gradients.

The objective function and constraint accuracy are presented in Table 5.1 for the first iteration. The agreement between the adjoint and centered-difference gradient values is excellent. Small differences of approximately 1% for both design variables are attributed to numerical errors. Figure 5.2 compares the optimization convergence histories for the



(a) Angle of attack



(b) Tip twist

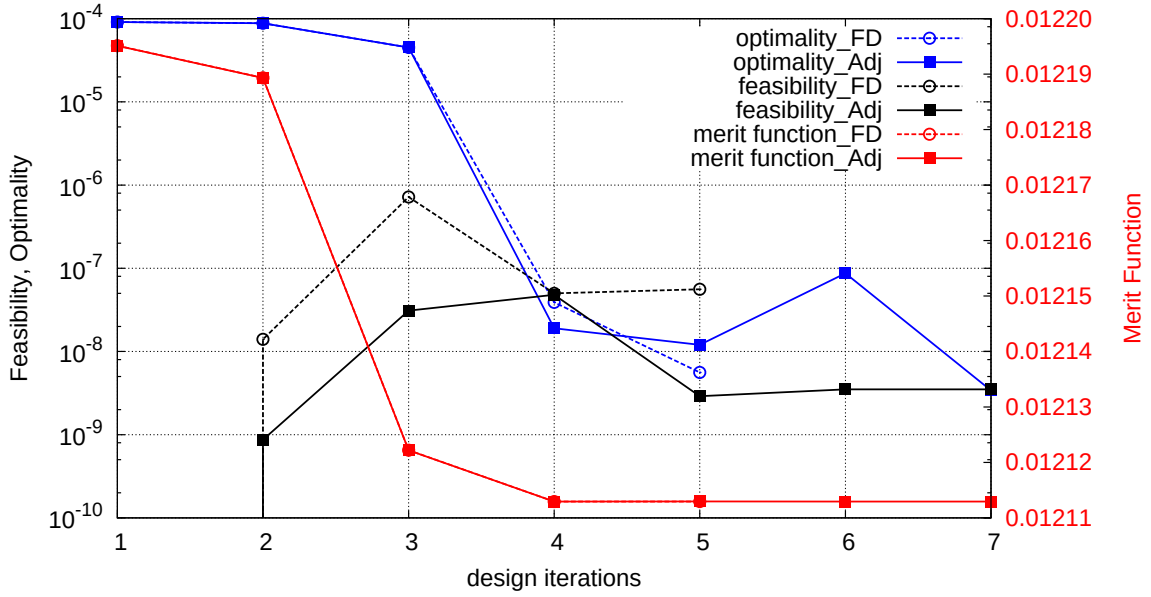
Figure 5.1: Objective and constraint gradients as a function of step size for the angle of attack and tip twist design variables.

Table 5.1: Gradient accuracy for the lift-constrained drag-minimization problem.

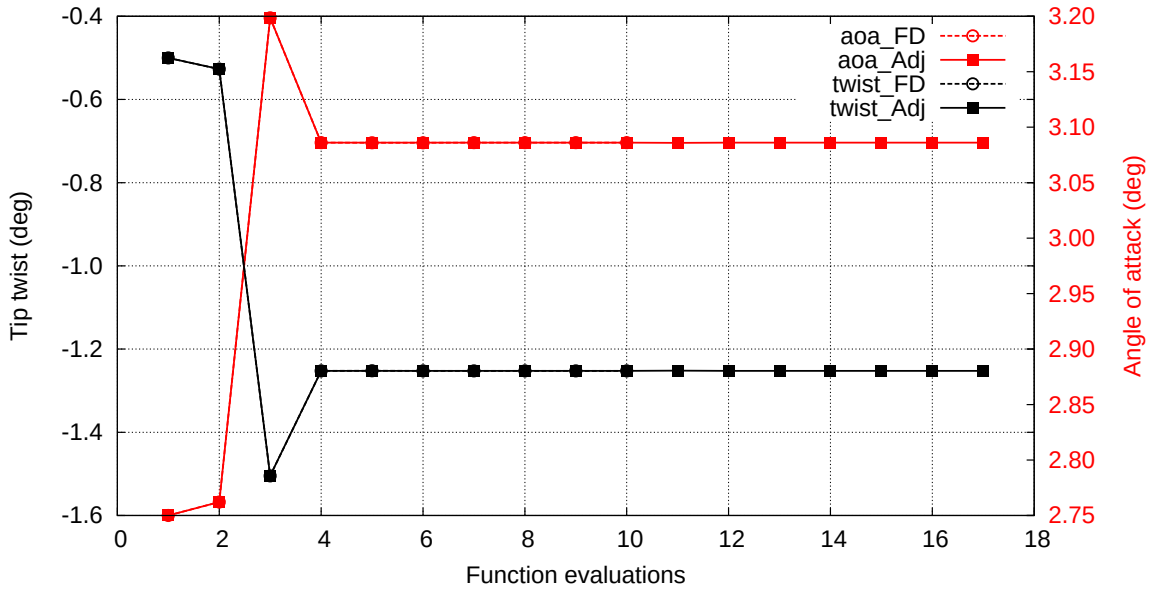
Design variable	Method	Objective gradient	Constraint gradient
Angle of attack	Adjoint	-0.0040671	-0.1786727
	Finite difference	-0.0040655	-0.1812989
Tip twist	Adjoint	0.0086162	0.4000044
	Finite difference	0.0086163	0.4061264

finite-difference and adjoint methods. Both have approximately the same degree of convergence and yield nearly identical results: The final angle of attack of 3.09° is higher than the initial in order to maintain the lift while a slightly lower negative final twist angle of 1.25° is added to minimize the induced drag. These results are consistent with those published in the literature [63, 98]. Note that the optimality reported here is a measure of the gradient convergence provided by SNOPT which takes constraint gradients into account, while the feasibility describes how well the optimizer is able to meet the constraint. Finally, the merit function, also reported by SNOPT, is an augmented Lagrangian merit function. When the constraints are satisfied, the merit function is equal to the objective. For example, for the objective function defined in Eq. (5.8), the merit function will equal to the coefficient of drag times the wetted wing planform area at convergence.

Finally, the objective function for different values of the twist angle with the angle of attack chosen to satisfy the above lift constraint is calculated so as to map out the feasible region of the design space, shown in Figure 5.3. It can be seen that the minimum drag is at $\tau = -1.25°$, which confirms the optimization results shown in Figure 5.2(b) and verifies that they are indeed the optimum.



(a) Optimizer



(b) Design variables

Figure 5.2: Convergence history for the finite difference (FD) and adjoint (Adj) methods.

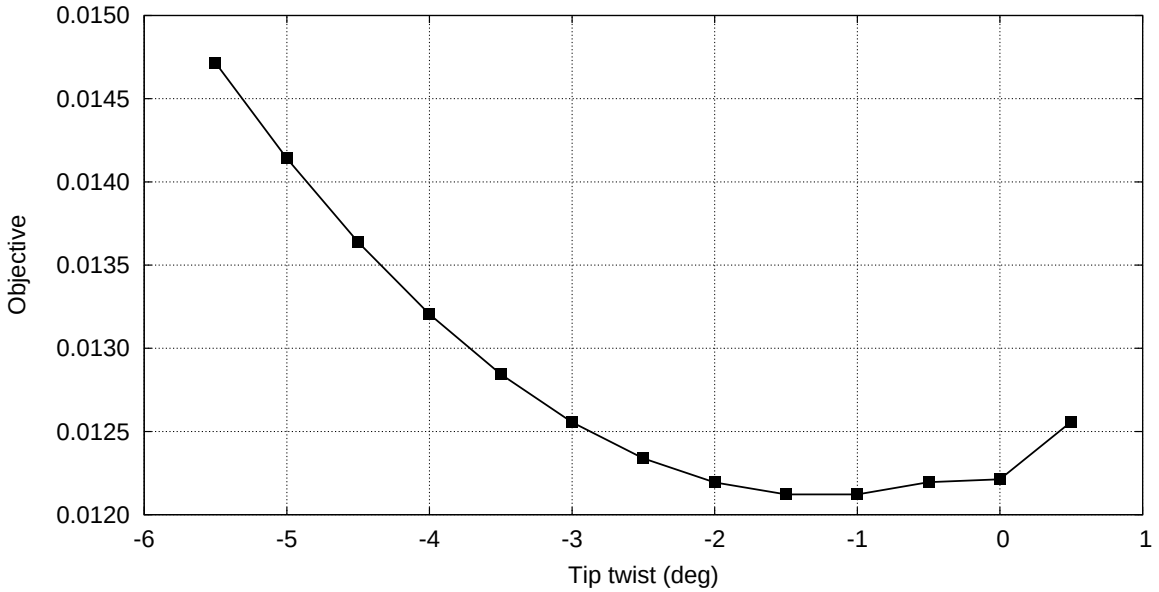


Figure 5.3: Objective function value for different twist angles.

6 Design Optimization Results

This chapter presents optimization results obtained using the CAD-based geometry parameterization tool described in Chapters 2 to 5 integrated with the flow solver [63], volume grid mover [158] and optimizer [49]. The optimization cases serve to demonstrate the accuracy and effectiveness of the CAD-based aerodynamic shape optimization algorithm. In particular, the induced drag minimization case is used to validate the methodology [65], while the wave drag reduction case both validates and verifies the applicability of the CAD-based geometry parameterization to problems with large shape changes. Finally, the inverse design cases demonstrate the use of the algorithm for a variety of design variables.

The baseline wing used for all cases is a straight, rectangular wing with a NACA0018 airfoil profile. The wing has a half span of eight meters and a chord of two meters. Note that the wing is non-dimensionalized by the root chord in the flow solver. All of the optimization results presented are computed on a $1,040,000$ node mesh with an H-topology consisting of 16 blocks. The farfield boundary is approximately 20 chords from the wing, and the off-wall spacing of the mesh is 10^{-3} m. Although the same mesh is used for all optimization cases, the CAD geometry is slightly different in the number of spanwise sections, as required by the problem.

6.1 Induced Drag Minimization

This test case is based on the induced drag validation case performed by Hicken and Zingg [65]. They showed that one can recover the elliptical lift distribution given by lifting line theory by twisting the spanwise sections about the trailing edge, thus minimizing

the induced drag. In particular, twisting about the straight trailing edge minimizes nonplanar effects in the wake. The design variables are the spanwise sectional twist angles at midspan and tip and the angle of attack. The lift, $(C_L S)_{ref}$, is constrained to the initial value.

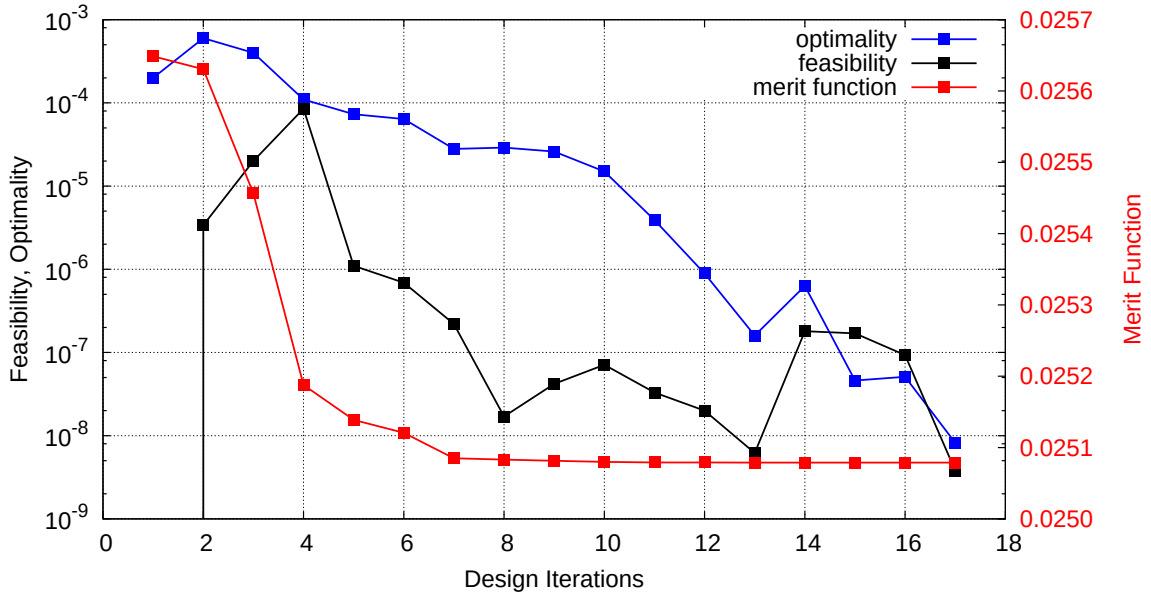
The convergence history for the twist optimization is shown in Figure 6.1. After 25 function evaluations the optimality measure has been reduced by 5 orders of magnitude and the absolute constraint violation has been reduced below 10^{-8}. The drag has been improved by 2% from 0.02565 to 0.02508. Figure 6.2 shows the lift distributions for the initial and optimized designs. It shows that the optimizer was able to improve the lift distribution to more closely match that of the elliptical distribution.

6.2 Wave Drag Reduction

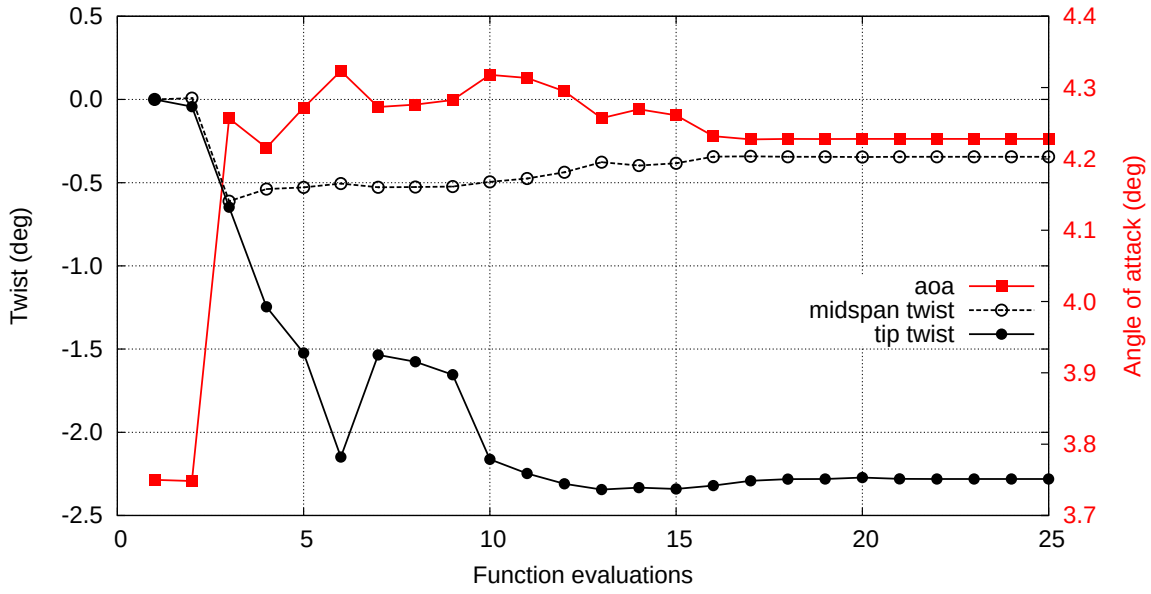
This test case validates the use of sweep angle, and airfoil thickness by the optimizer to reduce the wave drag that is present in the original unswept wing in transonic flow conditions with $M = 0.76$. In addition to sweep angle and airfoil thickness, other design variables include tip twist and angle of attack. Table 6.1 shows the upper and lower limits set for each of the design variables. The objective is to minimize the drag while maintaining a lift of $(C_L S)_{ref} = 2.125$.

Figures 6.3 and 6.4 show the Mach number contours and sectional coefficient of pressure distribution over the initial and optimized wings. It can be seen that the optimizer has swept the wing back to the limit set at the beginning of the optimization in an attempt to reduce the wave drag. A strong shock was present over much of the initial wing, but is significantly weakened in the optimized wing. The thickness is also reduced to the minimum allowable value in order to reduce the wave drag, while a negative twist angle of 3.1° is added to reduce the induced drag. Finally, the angle of attack is increased to 6.03° to maintain the required lift. Note also how, other than a slight distortion at the tip, the characteristics of the initial surface mesh are mostly preserved in the final mesh, as seen in Figure 6.3.

Figure 6.5 shows the convergence histories for the wave drag reduction case. The optimizer is able to reduce the optimality by about 5 orders of magnitude, achieving an overall reduction of drag of 47% from an initial value of 0.1946 to the final value of 0.1029, while achieving the specified lift.



(a) Optimizer



(b) Design variables

Figure 6.1: Convergence history for the twist optimization case with 3 design variables.

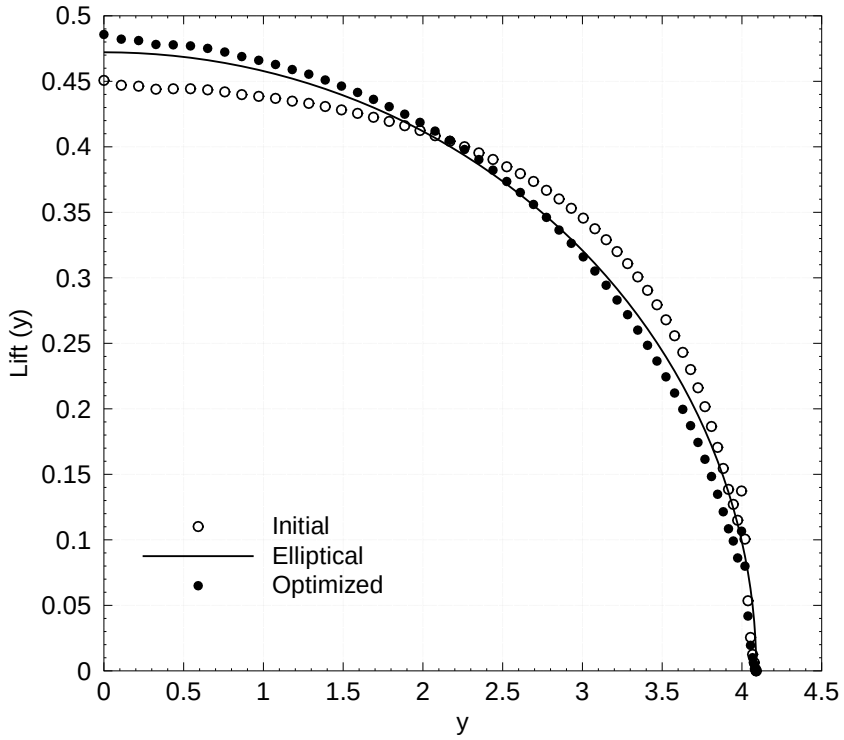


Figure 6.2: Lift distributions for the initial and optimized designs.

Table 6.1: Upper and lower limits for the design variables in the wave drag reduction problem

Design variable	Lower limit	Initial Value	Upper limit	Units
Sweep	-25	0	25	° (rel. to the init. sweep)
Tip twist	-15	0	15	° (rel. to the init. twist)
Airfoil thickness	-33	0	10	% (initial thickness)
Angle of attack	-5.25	3.75	8.75	°

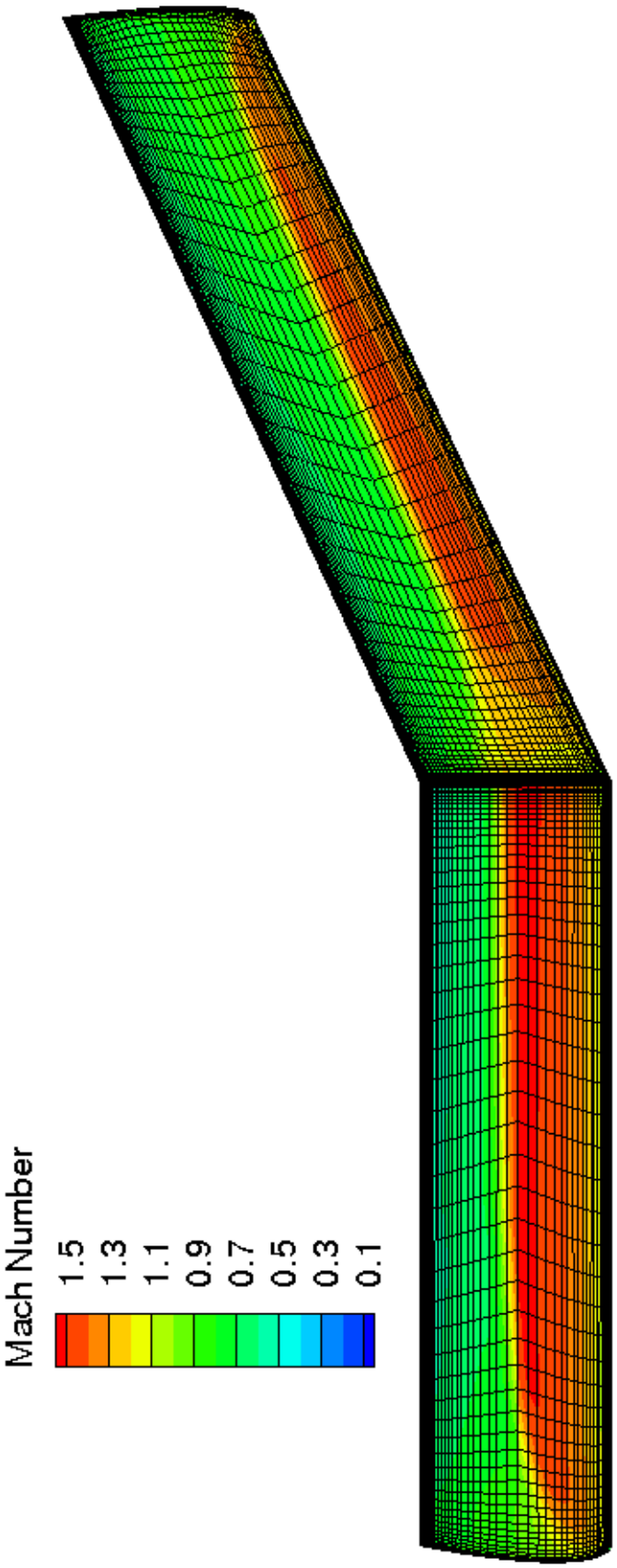


Figure 6.3: Contours of Mach number on the surface of the initial (left) and optimized (right) wing.

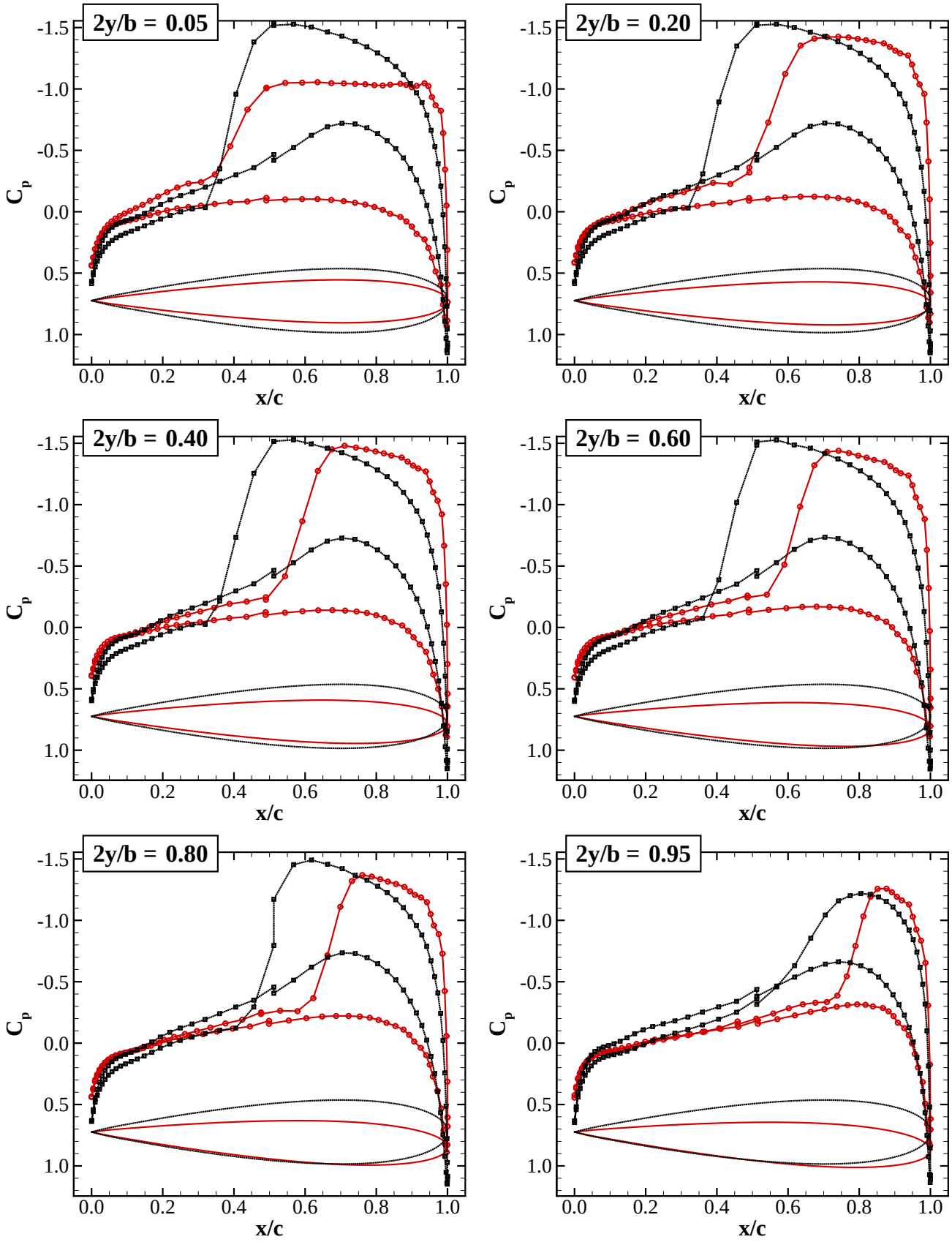
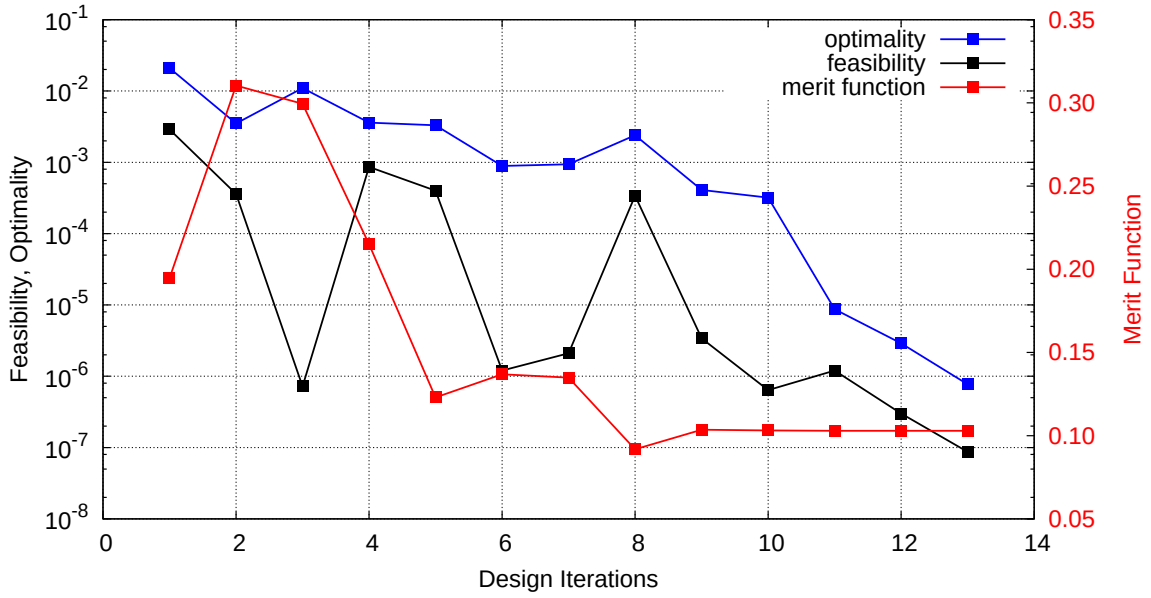
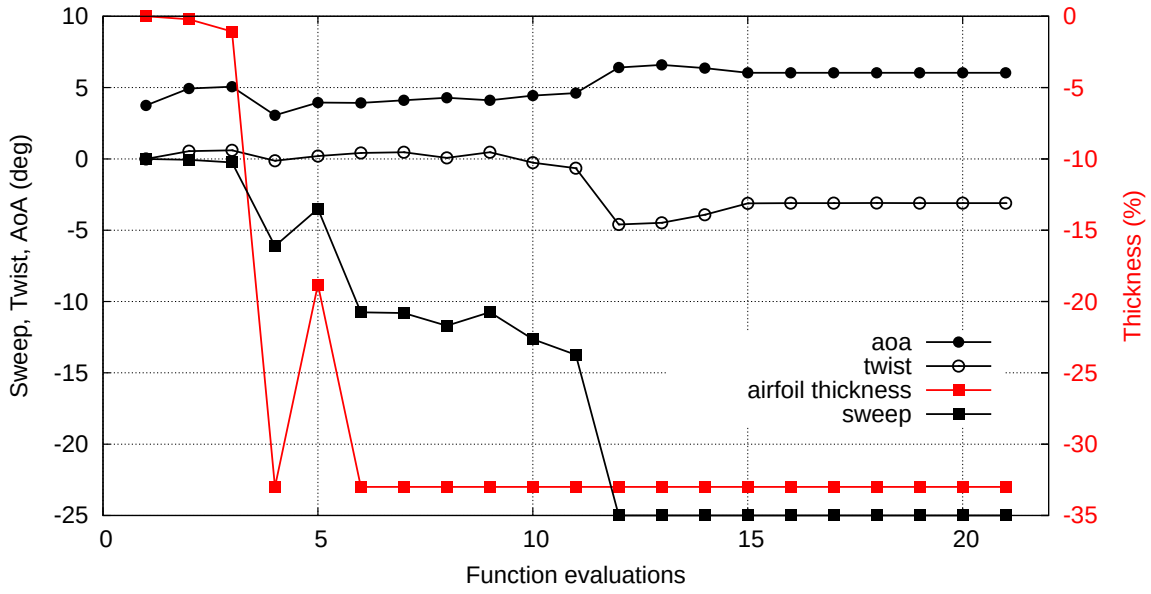


Figure 6.4: Sectional C_p distribution over the initial (black) and optimized (red) wing.



(a) Optimizer



(b) Design Variables

Figure 6.5: Convergence history for the wave drag reduction case with 4 design variables.

6.3 Inverse Optimization

The inverse problems involve a transonic flow where the Mach number is 0.74, and angle attack 2.5°. An inverse design based on the pressure distribution on the wing is considered. A target geometry is created by perturbing the values of the design variables. The optimization starts with the design variables at the specified initial values, and the goal is to recover the perturbed values based on the target pressure distribution. The objective is defined as:

$$\mathcal{J} = \frac{1}{2} \sum_{i=1}^{N_{surf}} (p_i - p_{i,targ})^2 \Delta A_i \tag{6.1}$$

where N_{surf} is the total number of surface nodes, ΔA_i is the surface area element at node i; and p_i and $p_{i,targ}$ are the pressure and target pressure at node i, respectively.

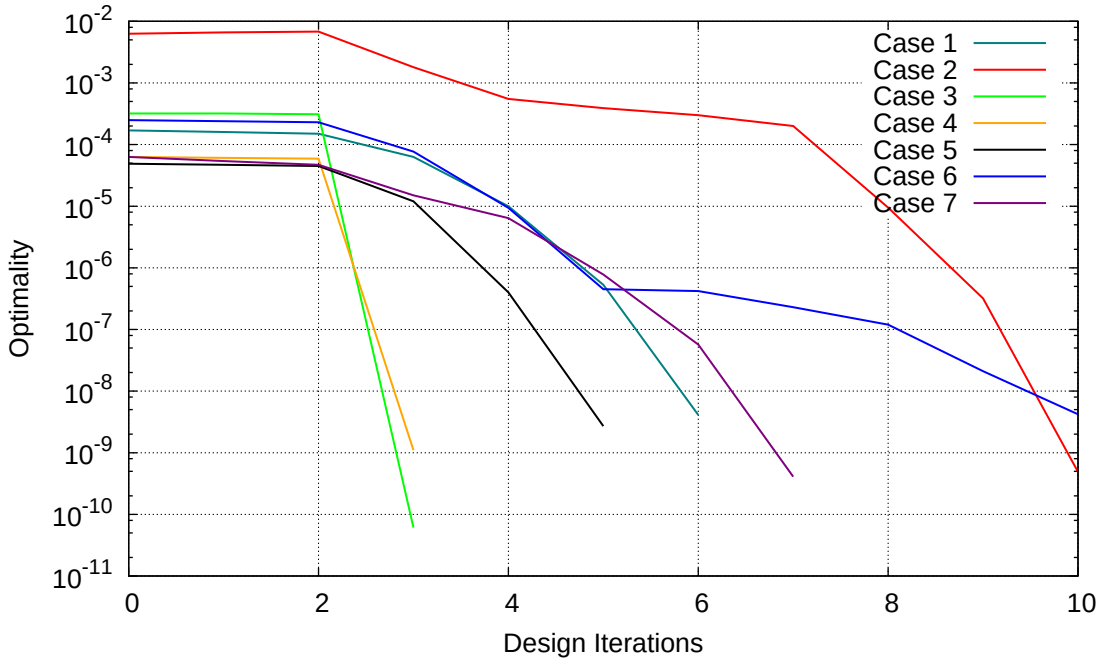
Table 6.2 lists the inverse optimization cases and Figure 6.6 shows the corresponding convergence histories for each of the cases. Note that the magnitude of a geometric design variable corresponds to the change from the initial value so that the initial shape is recovered when all the values are zero.

Table 6.2: Case description for the inverse optimization problems

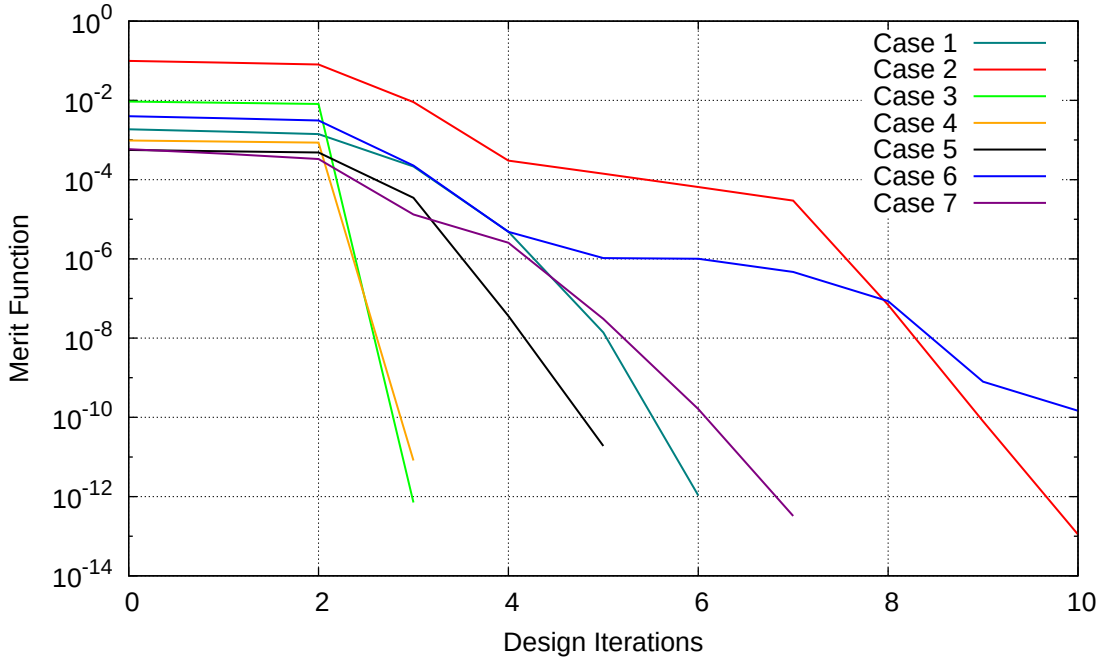
Case	Design Variables	Initial Value	Target Value	Units
1	Thickness taper	5.0	-10.0	% (initial thickness)
2	Tip twist	15.5	0.5	° (rel. to the init. twist)
3	Leading edge sweep	15.0	-15.0	% (initial chord)
4	Span	15.0	-15.0	% (initial span)
5	Span	17.0	-10.0	% (initial span)
6	Leading edge sweep	-0.5	14.5	% (initial chord)
	Span	-5.0	5.0	% (initial span)
7	Control point[1]	8.0	2.0	mm (rel. to the init. pos)
	Control point[2]	9.0	3.0	mm (rel. to the init. pos)

[1] at approximately midspan and midchord on the upper surface of the wing

[2] at approximately midspan and midchord on the lower surface of the wing



(a) Optimality



(b) Merit function

Figure 6.6: Convergence histories for the inverse optimization cases described in Table 6.2.

7 Conclusions and Recommendations

7.1 Conclusions

A CAD-based geometry parameterization framework has been developed for 3D aerodynamic shape optimization. A number of original contributions have been made in the development of this famework. They include:

1. A geometry control tool to aid in the automatic construction and manipulation of the CATIA geometry. It uses inputs such as section profile, section position, scaling and orientation to generate a script that can be run in CATIA (see Appendix A). This minimizes manual labour and time in the geometry construction phase. The method of construction gives the resulting geometry a great deal of flexibility and capability to change throughout the optimization cycle.

2. An efficient method to obtain a structured surface mesh from CAD geometries based on an initial surface mesh:

   The new surface mesh is guaranteed to lie on the CAD geometry and have the same characteristics (mesh density and spacing) as the original surface mesh even for large shape changes. The surface mesh movement has the following unique components:

   (a) a clustering algorithm to control changes in CAD topology during regeneration;

   (b) a parameterization formulation to preserve the characteristics of the original mesh;

(c) an inverse mapping procedure to guarantee the new mesh points to be on the CAD geometry. The procedure is efficient in that once the new geometry is created and a corresponding tessellation obtained, no subsequent communication with the CAD description is required to determine the surface mesh points in CAD parametric space. All calculations are done in two dimensions.

3. A global smoothing procedure that allows migration of surface mesh points across the boundary of a CAD surface:

   This is achieved by transforming the coordinate systems of the different CAD faces into one system where the Winslow mesh smoothing algorithm is performed. Smoothing in two dimensions is efficient in that only a few iterations (less than 4) are required to attain a smooth mesh.

4. An augmented adjoint gradient computation approach that includes CAD surface sensitivities and provides accurate functional derivatives for aerodynamic shape optimization:

   The discrete adjoint equations have been augmented to include grid and CAD surface sensitivities, ensuring efficiency and accuracy in the gradient computations. The CAD surface sensitivity derivatives are obtained using centered-difference approximations where the deformed surface is obtained for the positive and negative perturbations of the geometry and the difference calculated. The three-dimensional perturbed surface meshes are determined from their scaled uv-coordinates. This method is straightforward and efficient, as it incurs no additional calculations by the CAD software in going from parametric to three-dimensional space.

5. A demonstration of the validity and accuracy of the algorithm for three-dimensional aerodynamic shape optimization with CAD geometries on multiblock structured grids through a number of verification and design cases. In particular, it has been demonstrated that the surface mesh movement algorithm can handle very large changes in the geometry while maintaining the initial mesh quality and geometric fidelity.

7.2 Recommendations

Recommendations for future work include:

1. Improvement of the surface mesh quality:

   The quality of final surface mesh in regions of high curvature such as the tip of the wing can be improved further with better quality triangulation. The CAPRI triangulation can be smoothed and refined to achieve this.

2. Development of a dynamic face clustering algorithm:

   The current static CAD face clustering algorithm can be extended to dynamically group CAD faces based on the blocking criteria set out at the beginning of the optimization. This is required in more complex design cases where there are intersecting faces, such as in a wing-body configuration, or complex deformations of the surfaces involving movements of the airfoil control points.

3. Extension of the current algorithm to handle multiple volumes:

   For the optimization of more complex wing configurations with slat and flap, or more complete aircraft configurations with wing, fuselage, tail etc., multiple volumes have to be constructed to represent the different parts.

4. Parallelization of the mesh movement and mesh adjoint for multiple increments:

   This is required for cases involving large, out-of-plane deformations which require multiple mesh increments to minimize the distortion of the mesh. Since the computational cost of the linear elasticity mesh movement algorithm is about 30% of that of a flow solve, and varies depending on the amount of movement involved, paralellization would be very beneficial. This includes the full parallelization of the current mesh movement algorithm to update the stiffness matrix on the left hand side of the mesh movement equation and calculation of its derivatives.

References

[1] M. J. Aftosmis, *On the use of CAD-native predicates and geometry in surface meshing*, NASA/TM-1999-208782, NASA Ames Research Center, Moffett Field, California, August 1999.

[2] M. Alexa, *Recent advances in mesh morphing*, Computer Graphics Forum, 21 (2002), pp. 173–198.

[3] J. J. Alonso, J. R. R. A. Martins, J. J. Reuther, and R. Haimes, *High-fidelity aero-structural design using a parametric CAD-based model*, AIAA Paper 2003-3429, 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL, June 2003.

[4] G. R. Anderson, M. J. Aftosmis, and M. Nemec, *Constraint-based shape parameterization for aerodynamic design*, tech. rep., International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, Hawaii, July 2012.

[5] G. R. Anderson, M. J. Aftosmis, and M. Nemec, *Parametric deformation of discrete geometry for aerodynamic shape design*, AIAA Paper 2012–0965, 50th AIAA ASM Aerospace Sciences Meeting and Exhibit, Nashville, TN, Jan. 2012.

[6] W. K. Anderson and V. Venkatakrishnan, *Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation*, Computers & Fluids, 28 (1999), pp. 443–480.

[7] N. E. Antoine and I. M. Kroo, *Framework for aircraft conceptual design and environmental performance studies*, AIAA Journal, 43 (2005), pp. 2100–2109.

[8] H. A. Bashir and R. S. Neville, *Hybrid evolutionary computation for continuous optimization*, CoRR, abs/1303.3469 (2013).

[9] M. W. Beall, J. Walsh, and M. S. Shepphard, *Accessing CAD geometry for mesh generation*, in Proceedings of the 12th International Meshing Roundtable, Santa Fe, NM, September 2003, Sandia National Laboratories, pp. 33–42.

[10] T. L. Benyo, *Project Integration Architecture (PIA) and Computational Analysis Programming Interface (CAPRI) for accessing geometry data from CAD files*, NASA/TM-2002-211358, NASA Glenn Research Center, Cleveland, Ohio, January 2002.

[11] S. Bhowmick and S. M. Shontz, *Towards high-quality, untangled meshes via force-directed graph embedding approach*, tech. rep., International Conference on Computational Science (ICCS 2010), 2010.

[12] W. E. Brock, C. Burdyshaw, S. L. Karman, V. C. Betro, C. B. Hilbert, W. K. Anderson, and R. Haimes, *Adjoint-based design optimization using CAD parameterization through CAPRI*, AIAA Paper 2012–0968, 50th AIAA Aerospace Sciences Meeting and Exhibit, Nashville, Tennessee, January 2012.

[13] G. Butlin and C. Stops, *CAD data repair*, in Proceedings of the 5th International Meshing Roundtable, Pittsburgh, Pennsylvania, October 1996, Sandia National Laboratories, pp. 7–12.

[14] S. A. Canann, J. R. Tristano, and M. L. Staten, *An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes*, in International Meshing Roundtable, 1998, pp. 479–494.

[15] *CATIA V5R21*, 2012. [Online; accessed 20-April-2012].

[16] J. Chen, M. Freytag, and V. Shapiro, *Shape sensitivity of constructively represented geometric models*, Computer Aided Geometric Design, 25 (2008), pp. 470–488.

[17] S. Chen and D. A. Tortorelli, *Three-dimensional shape optimization with variational geometry*, Structural Optimization, 13 (1997), pp. 81–94.

[18] Z. Chen, J. R. Tristano, and W. Kwok, *Combined Laplacian and optimization-based smoothing for quadratic mixed surface meshes.*

[19] O. Chernukhin and D. W. Zingg, *Multimodality and global optimization in aerodynamic design*, AIAA Journal, 51 (2013), pp. 1342–1354.

[20] J. H. Choi, J. H. Won, and J. M. Yoon, *Boundary method for shape design sensitivity aanlysis in the optimization of three-dimensional elastostatics*, in 6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, 30 May - 03 June 2005.

[21] K. K. Choi and K.-H. Chang, *A study of design velocity field computation for shape optimal design*, Finite Elements in Analysis and Design, 15 (1994), pp. 317–541.

[22] S. Choi, J. J. Alonso, and I. M. Kroo, *Multifidelity design optimization of low-boom supersonic jets*, Journal of Aircraft, 45 (2008), pp. 106–118.

[23] F. Cirak, M. Ortiz, and Schroder, *Subdivision surfaces: a new paradigm for thin shell finite-element analysis*, International Journal for Numerical Methods in Engineering, 47 (2000), pp. 2039–2072.

[24] U. Clarenz, N. Litke, and M. Rumpf, *Axioms and variational problems in surface parameterization*, Computer Aided Geometric Design, 21 (2004), pp. 727–749.

[25] J. F. Dannenhoffer III and R. Haimes, *Quilts: A technique for improving boundary representations for CFD*, AIAA Paper 2003–4131, 16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida, June 2003.

[26] P. Degener, J. Meseth, and R. Klein, *An adaptable surface parameterization method*, in Proceedings of the 12th International Meshing Roundtable, 2003, pp. 201–213.

[27] Y. Deremaux, K. Willcox, and R. Haimes, *Physically-based, real-time visualization and constraint analysis in multidisciplinary design optimization*, AIAA Paper 2003–3876, 15th Computational Fluid Dynamics Conference, Orlando, Florida, June 2003.

[28] M. Desbrun, M. Meyer, and P. Alliez, *Intrinsic Parameterizations of Surface Meshes*, Computer Graphics Forum, 21 (2002).

[29] H. Djidjev, *Force-directed methods for smoothing unstructured triangular and tetrahedral meshes*, in In Proceedings of the 9th International Meshing Roundtable, 2000, pp. 395–406.

[30] *DLR-F6 geometry*, 2012. [Online; accessed 20-April-2012].

[31] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, *Multiresolution analysis of arbitrary meshes*, in Proceedings of the 22nd Annual Conference on Computer Graphics and Iteractive Techniques, SIGGRAPH '95, New York, NY, USA, 1995, ACM, pp. 173–182.

[32] R. T. Farouki, *Closing the gap between CAD model and downstream application*, SIAM News, (1999).

[33] D. Field, *Laplacian smoothing and Delaunay triangulations*, Communications in Applied Numerical Methods, 4 (1988), pp. 709–712.

[34] M. S. Floater, *Parametrization and smooth approximation of surface triangulations.*, Computer Aided Geometric Design, (1997), pp. 231–250.

[35] M. S. Floater, *Mean value coordinates*, Computer Aided Geometric Design, 20 (2003), pp. 19–27.

[36] M. S. Floater and K. Hormann, *Surface parameterization: a tutorial and survey*, in Advances in Multiresolution for Geometric Modelling, N. A. Dodgson, M. S. Floater, and M. A. Sabin, eds., Springer Verlag, 2005, pp. 157–186.

[37] M. S. Floater, K. Hormann, and G. Kós, *A general construction of barycentric coordinates over convex polygons*, Advances in Computational Mathematics, 24 (2006), pp. 311–331.

[38] M. S. Floater and V. Pham-Trong, *Convex combination maps over triangulation, tilings, and tetrahedral meshes*, Advances in Computational Mathematics, 25 (2006), pp. 347–356.

[39] J. D. Foley and A. Van Dam, *Fundamentals of interactive computer graphics*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1982.

[40] L. A. Freitag, *On combining Laplacian and optimization-based mesh smoothing techniques*, in Trends in Unstructured Mesh Generation, 1997, pp. 37–43.

[41] L. A. Freitag and P. M. Knupp, *Tetrahedral element shape optimization via the jacobian determinant and condition number*, in Proceedings of the 8th International Meshing Roundtable, 1999, pp. 247–258.

[42] D. Fudge, *A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization*, Master's thesis, University of Toronto, 2004.

[43] D. M. Fudge, D. W. Zingg, and R. Haimes, *A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization*, AIAA Paper 2005–0451, 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2005.

[44] H. Gagnon and D. W. Zingg, *Two-level free-form deformation for high-fidelity aerodynamic shape optimization*, AIAA Paper 2012–5447, 12th AIAA Aviation Technology, Integration and Operations ATIO Conference and 14th AIAA/ISSM, Indianapolis, Indiana, September 2012.

[45] J. Gain and D. Bechmann, *A survey of spatial deformation from a user-centered perspective*, ACM Trans. Graph., 27 (2008), pp. 107:1–107:21.

[46] S. Gerbino, *Tools for the interoperability among CAD systems*, tech. rep., XIII ADM International Conference on Tools and Methods Evolution in Engineering Design, Napoli, Italy, June 2003.

[47] M. B. Giles and N. A. Pierce, *An introduction to the adjoint approach to design*, Flow, Turbulence and Combustion, 65 (2000), pp. 393–415.

[48] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Journal on Optimization, 12 (1997), pp. 979–1006.

[49] ——, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Review, 47 (2005), pp. 99–131.

[50] S. Gopalsamy, D. H. Ross, and A. M. Shih, *API for grid generation over topological models*, in Proceedings of the 13th International Meshing Roundtable, Williamsburg, VA, September 2004, Sandia National Laboratories, pp. 221–230.

[51] C. Gotsman, X. Gu, and A. Sheffer, *Fundamentals of spherical parameterization for 3D meshes*, 2003.

[52] J. S. Greenfeld, *Least squares weighted coordinate transformation formulas and their applications*, Journal of Surveying Engineering, 123 (1997), pp. 147–161.

[53] G. Greiner and K. Hormann, *Parameterizing meshes with applications*. IMA Workshop Presentation, 2001.

[54] X. Gu and S.-T. Yau, *Global conformal surface parameterization*, in Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP '03, Aire-la-Ville, Switzerland, Switzerland, 2003, Eurographics Association, pp. 127–137.

[55] M. D. Gunzburger, *Adjoint equation-based methods for control problems in incompressible, viscous flows*, Flow, Turbulence and Combustion, 65 (2000), pp. 249–272.

[56] R. Haimes, *CAPRI: Computational Analysis Programming Interface, A Solid Modeling Based Infrastructure for Engineering Analysis and Design*, 2.10 ed., December 2004.

[57] R. Haimes and M. J. Aftosmis, *Watertight anisotropic surface meshing using quadrilateral patches*, in Proceedings of the 13th International Meshing Roundtable, Williamsburg, VA, September 2004, Sandia National Laboratories, pp. 311–322.

[58] R. Haimes and C. Crawford, *Unified geometry access for analysis and design*, in Proceedings of the 12th International Meshing Roundtable, Santa Fe, NM, September 2003, Sandia National Laboratories, pp. 21–31.

[59] R. Haimes and J. F. Dannenhoffer, *The engineering sketch pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry*, AIAA Paper 2013–3073, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 2013.

[60] R. Haimes and J. F. Dannenhoffer III, *Control of boundary representation topology in multidisciplinary analysis and design*, AIAA Paper 2010–1504, 48th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 2010.

[61] E. Hardee, K.-H. Chang, J. Tu, K. K. Choi, I. Grindeanu, and X. Yu, *A CAD-based design parameterization for shape optimization of elastic solids*, Advances in Engineering Software, 30 (1999), pp. 185–199.

[62] J. E. Hicken and D. W. Zingg, *A parallel Newton-Krylov flow solver for the Euler equations on multi-block grids*, AIAA Paper 2007–4333, 18th AIAA Computational Fluid Dynamics Conference, Miami, Florida, USA, June 2007.

[63] ——, *A parallel newton-krylov solver for the euler equations discretized using simultaneous approximation terms*, AIAA Journal, 46 (2008), p. 2773–2786.

[64] ——, *Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement*, AIAA Journal, 48 (2010), p. 400–413.

[65] ——, *Induced-drag minimization of nonplanar geometries based on the Euler equations*, AIAA Journal, 48 (2010), p. 2564–2575.

[66] S. M. Hitzel, L. Nardin, K. Sorensen, and H. Rieger, *Aerodynamic optimization of aircraft configurations with multidisciplinary aspects*, ICAS 2008, International Congress of the Aeronautical Sciences, Bristol, U.K., 2008.

[67] T. L. Holst and T. H. Pulliam, *Aerodynamic shape optimization using a real-number-encoded genetic algorithm*, AIAA Paper 2001–2473, Anaheim, CA, June 2001.

[68] K. Hormann, *Barycentric mappings*, June 2007.

[69] K. Hormann and G. Greiner, *MIPS: An efficient global parametrization method*, in Curve and Surface Design: Saint-Malo 1999, P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, eds., Innovations in Applied Mathematics, Vanderbilt University Press, Nashville, TN, 2000, pp. 153–162.

[70] W. M. Hsu, J. F. Hughes, and H. Kaufman, *Direct manipulation of free-form deformations*, Computer Graphics, 26 (1992), pp. 177–184.

[71] *SDRC*, 2012. [Online; accessed 20-April-2012].

[72] D. S. Inc., *Stereolithography Interface Format Specification*, 1988.

[73] K. Inoue, T. Itoh, A. Yamada, and T. Furuhata, *Clustering a large number of faces for 2-Dimensional mesh generation*, in Proceedings of the 8th International Meshing Roundtable, South Lake Tahoe, CA, October 1999, Sandia National Laboratories, pp. 281–292.

[74] K. Inoue, T. Itoh, A. Yamada, T. Furuhata, and K. Shimada, *Face clustering of a large-scale CAD model for surface mesh generation*, Computer-Aided Design, 33 (2001), pp. 251–261.

[75] A. Jameson, *Aerodynamic design via control theory*, Journal of Scientific Computing, 3 (1988), pp. 233–260. Also ICASE report 88–64.

[76] A. Jameson, *Aerodynamic shape optimization using the adjoint method*, in VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute of Fluid Dynamics, Rhode St Genese, February 6, 2003, pp. 3–7.

[77] A. Jameson, Sriram, L. Martinelli, and R. Haimes, *Aerodynamic shape optimization of complete aircraft configurations using unstructured grids*, AIAA Paper 2004–0533, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2004.

[78] W. T. Jones, *Toward a global parameterization for quilted CAD entities*, AIAA Paper 2004–0611, 42rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2004.

[79] W. T. Jones, D. Lazzara, and R. Haimes, *Evolution of geometric sensitivity derivatives from computer aided design models*, AIAA Paper 2010-9128, 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, Texas, September 2010.

[80] S. L. K. Jr., *Virtual control volumes for two-dimensional unstructured elliptic smoothing*, in Proceedings of the 19th International Meshing Roundtable, Chattanooga, TN, 2010, pp. 121–142.

[81] M. Kahrs, *The heart of IGES*, Software: Practice and Experience, 25 (1995), pp. 935–946.

[82] T. Kanai, H. Suzuki, and F. Kimura, *3D geometric metamorphosis based on harmonic maps*, in Proceedings of Pacific Graphics '97, 1997, pp. 97–104.

[83] S. Karman, W. K. Anderson, and M. Sahasrabudhe, *Mesh generation using unstructured computational meshes and elliptic partial differential equation smoothing*, AIAA Journal, 44 (2006), pp. 1277–1286.

[84] Z. Karni, C. Gotsman, and S. J. Gortler, *Free-boundary linear parameterization of 3D meshes in the presence of constraints.*, in SMI'05, 2005, pp. 268–277.

[85] G. K. W. Kenway, G. J. Kennedy, and J. R. R. A. Martins, *A CAD-free approach to high-fidelity aerostructural optimization*, AIAA Paper 2010-9231, 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, Texas, September 2010.

[86] E. Kesseler, *Advancing the state-of-the-art in the civil aircraft design: A knowledge-based multidisciplinary engineering approach*, tech. rep., European Conference on Computational Fluid Dynamics (ECCOMAS CDF), Delft, The Netherland, 2006.

[87] H.-J. Kim, D. Sasaki, S. Obayashi, and K. Nakahashi, *Aerodynamic optimization of supersonic transport wing using unstructured adjoint method*, AIAA Journal, 39 (2001), pp. 1011–1020.

[88] S. Kleinveld, G. Roge, L. Daumas, and Q. Dinh, *Differentiated parametric CAD used within the context of automatic aerodynamic design optimization*, AIAA Paper 2008-5952, 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, September 2008.

[89] P. M. Knupp, *Winslow smoothing on two-dimensional unstructured meshes*, in Proceedings of the Seventh International Meshing Roundtable, Park City, UT, 1998, pp. 449–457.

[90] P. B. Laval, *Mathematics for computer graphics - barycentric coordinates*, 2003.

[91] C. K. Lee, *Automatic metric 3-D surface mesh generation using subdivision surface geometry model. Part 1: Construction of underlying geometric model*, International Journal for Numerical Methods in Engineering, 56 (2003), pp. 1593–1614.

[92] T.-Y. Lee and S.-W. Yen, *Texture mapping on 3D surfaces using clustering-based cutting paths*, International Journal for Computational Science and Engineering, 3 (2007), pp. 71–79.

[93] Y. Lee, H. S. Kim, and S. Lee, *Mesh parameterization with a virtual boundary*, Computers & Graphics, 26 (2002), pp. 677–686.

[94] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, *Least squares conformal maps for automatic texture atlas generation*, in SIGGRAPH 02 Conference Proceedings, 2002, pp. 362–371.

[95] L. Liu, C. Zhang, and F. F. Cheng, *Parameterization of quadrilateral meshes*, in Computational Science - ICCS 2007, Y. Shi, G. Albada, J. Dongarra, and P. M. A. Sloot, eds., vol. 4488 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 17–24.

[96] X. Liu, H. Bao, H.-Y. Shum, and Q. Peng, *A novel volume constrained smoothing method for meshes*, Graphical Models, 64 (2002), pp. 169–182.

[97] D. Lowe, *Object recognition from local scale-invariant features*, in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 2, 1999, pp. 1150 –1157 vol.2.

[98] C. A. Mader and J. R. R. A. Martins, *Optimal flying wings: A numerical optimization study*, in 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, HI, April 2012. AIAA 2012-1758.

[99] J. Maillot, H. Yahia, and A. Verroust, *Interactive texture mapping*, in SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1993, ACM, pp. 27–34.

[100] J. R. R. A. Martins, *A Coupled-Adjoint Method for High-Fidelity Aero-Structural Optimization*, PhD thesis, Stanford University, 2002.

[101] J. R. R. A. Martins, J. J. Alonso, and J. Reuther, *Aero-structural wing design optimization using high-fidelity sensitivity analysis*, in In Proceedings ̆2014 CEAS Conference on Multidisciplinary Aircraft Design Optimization, 2001, pp. 211–226.

[102] D. J. Mavriplis, *A discrete adjoint-based approach for optimization problems on three-dimensional unstructured meshes*, AIAA Journal, 45 (2007), pp. 741–750.

[103] A. Meid, *Object based determination of comparator coordinates in blurred images*, International Archives of Photogrammetry and Remote Sensing, 29 (1993), pp. 79–86.

[104] S. Merazzi, E. A. Gerteinsen, and A. A. Mezentsev, *A generic CAD-mesh interface*, in Proceedings of the 9th International Meshing Roundtable, New Orleans, Louisiana, October 2000, Sandia National Laboratories, pp. 361–370.

[105] A. A. Mezentsev, *A generalized graph-theoretic mesh optimization model*, in IMR, 2004, pp. 255–264.

[106] T. Michikawa, T. Kanai, M. Fujita, and H. Chiyokura, *Multiresolution interpolation meshes*, in Proceedings of Pacific Graphics, 2001, pp. 60–69.

[107] F. Muyl, L. Dumas, and V. Herbert, *Hybrid method for aerodynamic shape optimization in automotive industry*, Computers & Fluids, 33 (2004), pp. 849–858.

[108] S. Nadarajah and A. Jameson, *Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization*, in AIAA 2001-2530, 2001.

[109] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, *Laplacian mesh optimization*, in Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, GRAPHITE '06, New York, NY, USA, 2006, ACM, pp. 381–389.

[110] E. J. Neilsen and M. A. Park, *Using an adjoint approach to eliminate mesh sensitivities in computational design*, AIAA Journal, 44 (2006), pp. 948–953.

[111] V. Nejati and K. Matsuuchi, *Aerodynamics design and genetic algorithms for optimization of airship bodies*, JSME International Journal Series B Fluids and Thermal Engineering, 46 (2003), pp. 610–617.

[112] M. Nemec, *Optimal Shape Design of Aerodynamic Configurations: A Newton-Krylov Approach*, PhD thesis, University of Toronto, 2003.

[113] M. Nemec and M. J. Aftosmis, *Adjoint algorithms for CAD-based shape optimization using a cartesian method*, NAS technical report nas-05-014, NASA Ames Research Center, Moffett Field, CA, October 2005.

[114] ——, *Aerodynamic shape optimization using a cartesian adjoint method and CAD geometry*, AIAA Paper 2006–3456, 24th AIAA Applied Aerodynamics Conference, San Francisco, CA, June 2006.

[115] ——, *Adjoint sensitivity computations for an embedded-boundary cartesian mesh method*, Journal of Computational Physics, 227 (2008), pp. 2724–2742.

[116] ——, *Parallel adjoint framework for aerodynamic shape optimization of component-based geometry*, AIAA Paper 2011–1249, 49th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, FL, January 2011.

[117] M. Nemec and D. W. Zingg, *Aerodynamic shape optimization using the discrete adjoint method*, in 48th Annual CASI Conference, Toronto, ON, 2001, pp. 203–213.

[118] M. Nemec, D. W. Zingg, and T. H. Pulliam, *Multi-point and multi-objective aerodynamic shape optimization*, AIAA Paper 2002–5548, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, September 2002.

[119] J. C. Newman III, A. C. Taylor III, R. W. Barnwell, P. A. Newman, and G. J.-W. Hou, *Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations*, Journal of Aircraft, 36 (1999), pp. 87–96.

[120] E. J. Nielsen and W. L. Kleb, *Efficient construction of discrete adjoint operators on unstructured grids using complex variables*, AIAA Journal, 44 (2006), pp. 827–836.

[121] *OpenCASCADE*, 2012. [Online; accessed 20-April-2012].

[122] A. Oyama, *Multidisciplinary optimization of transonic wing design based on evolutionary algorithms coupled with CFD solver*, in SOLVER, Proceedings of European Congress on Computational Methods in Applied Sciences and Engineering, 2000.

[123] *Parasolid: 3D geometric modeling engine*, 2012. [Online; accessed 20-April-2012].

[124] S. Pierret, H. Kato, R. F. Coelho, and A. Merchant, *Aero–mechanical optimization method with direct CAD access–application to counter rotating fan design*, in Proceedings of ASME TURBO EXPO, Barcelona, Spain, May 8–11 2006, DRAFT-GT2006-90505.

[125] U. Pinkall, S. D. Juni, and K. Polthier, *Computing discrete minimal surfaces and their conjugates*, Experimental Mathematics, 2 (1993), pp. 15–36.

[126] *Creo Elements/Pro*, 2012. [Online; accessed 20-April-2012].

[127] D. H. Quynh, A. Yezzi, and G. Turk, *Texture transfer during shape transformation*, ACM Transactions on Graphics, 24 (2005), pp. 289–310.

[128] M. Ramasubramanian and A. Mittal, *Three-dimensional metamorphosis using multiplanar representation*, in Proceedings of the IEEE International Conference on Multimedia Computing and Systems - Volume 2, ICMCS '99, Washington, DC, USA, 1999, IEEE Computer Society, pp. 9270–.

[129] K. Reed, *The Initial Graphics Exchange Specification (IGES)*, September 1991.

[130] J. J. Reuther, J. J. Alonso, M. J. Rimlinger, and A. Jameson, *Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on distributed memory parallel computers*, Computers & Fluids, 28 (1999), pp. 675–700.

[131] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders, *Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1*, Journal of Aircraft, 36 (1999), pp. 51–60.

[132] ——, *Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2*, Journal of Aircraft, 36 (1999), pp. 61–74.

[133] T. T. Robinson, C. G. Armstrong, H. S. Chua, C. Othmer, and T. Grahs, *Optimizing parameterized CAD geometries using sensitivities based on adjoint functions*, Computer-Aided Design and Applications, 9 (2012), pp. 253–268.

[134] J. A. Samareh, *Use of CAD geometry in MDO*, in The 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1996.

[135] ——, *Survey of shape parametrization techniques for high-fidelity multidisciplinary shape optimization*, AIAA Journal, 39 (2001), pp. 877–883.

[136] ——, *Aerodynamic shape optimization based on free-form deformation*, in Collection of Technical Papers - 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, vol. 6, Multidisciplinary Optimization Branch, NASA Langley Research Center, Hampton, VA 23681, 2004, AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, pp. 3672–3683.

[137] ——, *Geometry and grid/mesh generation issues for CFD and CSM shape optimization*, Optimization and Engineering, 6 (2005), pp. 21–32.

[138] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, *Texture mapping progressive meshes*, in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, New York, NY, USA, 2001, ACM, pp. 409–416.

[139] E. Schwartz and I. M. Kroo, *Aircraft design: Trading cost and climate impact*, AIAA Paper 2009–1261, 47th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 2009.

[140] *STEP application handbook: ISO 10303*, June 2006.

[141] A. Sheffer, *Model simplification for meshing using face clustering*, Computer-Aided Design, 33 (2001), pp. 925–934.

[142] A. Sheffer, T. Blacker, J. Clements, and M. Bercovier, *Virtual topology construction and applications*, in Theory and Practice of Geometric Modeling, Blaubeuren II, Springer-Verlag, 1996, pp. 247–259.

[143] ——, *Virtual topology operators for meshing*, International Journal of Computational Geometry and Applications, 10 (2000), pp. 309–331.

[144] A. Sheffer and E. de Sturler, *Parameterization of faceted surfaces for meshing using Angle-Based flattening*, Engineering with Computers, 17 (2001), pp. 326–337.

[145] ——, *Parameterization of faceted surfaces for meshing using angle based flattening*, Engineering with Computers, 17 (2001), pp. 326–337.

[146] ——, *Smoothing an overlay grid to minimize linear distortion in texture mapping*, ACM Transactions on Graphics, 21 (2002), pp. 874–890.

[147] A. Sheffer, B. Lévy, I. Lorraine, M. Mogilnitsky, and E. Bogomyakov, *ABF++: Fast and robust angle based flattening*, ACM Transactions on Graphics, 24 (2005), pp. 311–330.

[148] A. Sheffer, E. Praun, and K. Rose, *Mesh parameterization methods and their applications*, Foundations and Trends in Computer Graphics and Vision, 2 (2006), pp. 105–171.

[149] W. J. Shenren Xu and J.-D. Müller, *CAD-based shape optimization with CFD using a discrete adjoint*, International Journal for Numerical Methods in Fluids, 74 (2014), pp. 153–168.

[150] S. M. Shontz and S. A. Vavasis, *A mesh warping algorithm based on weighted Laplacian smoothing*, in IMR, 2003, pp. 147–158.

[151] H. Sobieczsky, *Aerodynamic design and optimization tools accelerated by parametric geometry preprocessing*, in European Congress on Computational Methods in Applied Sciences and Engineering, Barcelona, Spain, 2000.

[152] *Solidworks*, 2012. [Online; accessed 20-April-2012].

[153] O. Sorkine, D. Cohen-or, R. Goldenthal, and D. Lischinski, *Bounded-distortion piecewise mesh parameterization*, in IEEE Visualization, 2002, pp. 355–362.

[154] J. P. Steinbrenner and J. Chawner, *Gridgen's implementation of partial differential equation based structured grid generation methods*, in Proceedings, 8th International Meshing Roundtable, South Lake Tahoe, CA, October 1999, Sandia National Laboratories, pp. 143–152.

[155] K. Stephenson and K. Stephenson, *The approximation of conformal structures via circle packing*, in Computational Methods and Function Theory 1997, Proceedings of the Third CMFT Conference, World Scientific, 1997, pp. 551–582.

[156] T. J. Tautges, *The Common Geometry Module (CGM): A generic, extensible geometry interface*, in Proceedings of the 9th International Meshing Roundtable, New Orleans, Louisiana, October 2000, Sandia National Laboratories, pp. 337–359.

[157] J. I. Toivanen and J. Martikainen, *A new method for creating sparse design velocity fields*, Computer Methods in Applied Mechanics and Engineering, 196 (2006), pp. 528–537.

[158] A. H. Truong, *Development of grid movement algorithms suitable for aerodynamic shape optimization*, Master's thesis, University of Toronto, 2005.

[159] A. H. Truong, C. A. Oldfield, and D. W. Zingg, *Mesh movement for a discrete-adjoint Newton-Krylov algorithm for aerodynamic optimization*, AIAA Journal, 46 (2008), pp. 1695–1704.

[160] W. T. Tutte, *How to draw a graph*, Proceedings of The London Mathematical Society, 13 (1963), pp. 743–767.

[161] *NX (Unigraphics)*, 2012. [Online; accessed 20-April-2012].

[162] *VDAFS Processor V1.7.1 for CATIA V5: User Manual*, 1.7.1 ed., April 2008.

[163] J. Vollmer, R. Mencl, and H. Müller, *Improved Laplacian smoothing of noisy surface meshes*, Comput. Graph. Forum, 18 (1999), pp. 131–138.

[164] *Current State of the Art on Multidisciplinary Design Optimization (MDO)*, American Institute of Aeronautics and Astronautics, 1991.

[165] A. M. Winslow, *Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh*, Journal of Computational Physics, 1 (1966), pp. 149 – 172.

[166] A. Wood, C. Sainmont, A. Merchant, D. Leblond, and E. Laurendeau, *CAD-based parametric design optimization with structured multi-block grid re-generation*, in 58th Annual CASI Conference, Montreal, QC, 2011.

[167] H. Xu and T. S. Newman, *An angle-based optimization approach for 2d finite element mesh smoothing*, Finite Elem. Anal. Des., 42 (2006), pp. 1150–1164.

[168] Y.-L. Yang, J. Kim, F. Luo, S.-M. Hu, and X. Gu, *Optimal surface parameterization using inverse curvature map.*, IEEE Trans. Vis. Comput. Graph., (2008), pp. 1054–1066.

[169] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, *A fast and simple stretch-minimizing mesh parameterization*, in Proceedings of the Shape Modeling International 2004, SMI '04, Washington, DC, USA, 2004, IEEE Computer Society, pp. 200–208.

[170] G. Yu, J.-D. Müller, D. Jones, and F. Christakopoulos, *CAD-based shape optimisation using adjoint sensitivities*, Computers and Fluids, 46 (2011), pp. 512–516.

[171] R. Zayer, B. Lévy, and H.-P. Seidel, *Linear Angle Based Parameterization*, in Fifth Eurographics Symposium on Geometry Processing - SGP 2007, M. G. Alexander Belyaev, ed., Barcelone, Spain, 2007, Eurographics Association, pp. 135–141.

[172] R. Zayer, C. Rössl, and H.-P. Seidel, *Convex boundary angle-based flattening.*, in VMV'03, 2003, pp. 281–288.

[173] E. Zhang, K. Mischaikow, and G. Turk, *Feature-based surface parameterization and texture mapping*, ACM Transactions on Graphics, 24 (2005), pp. 1–27.

[174] W.-H. Zhang, P. Beckers, and C. Fleury, *A unified parametric design approach to structural shape optimization*, International Journal for Numerical Methods in Engineering, 38 (1995), pp. 2283–2292.

[175] T. Zhou and K. Shimada, *An angle-based approach to two-dimensional mesh smoothing*, in Proceedings, 9th International Meshing Roundtable, 2000, pp. 373–384.

[176] D. W. Zingg, M. Nemec, and T. H. Pulliam, *A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization*, Revue Europeenne de mecanique Numerique, (2008), pp. 103–126.

APPENDIX

A

Geometry Generation

A.1 Geometry Generator Inputs

The following three sets of files are required:

1. The settings file (geometry_generator.input), which contains
   - the number of control points, which is only used if the number of points in adjacent sections differs. To extract the same number of points at equivalent positions along the sections, the section with the lower number of points is splined. If necessary, the geometry generator shall modify these settings.
   - the order of the approximating curve (order = degree + 1, max = 20)
   - the debug option, and
   - the Tecplot output option
2. The section settings file (sections.dat) is used to indicate
   - the number of sections
   - the sections filename
   - the scaling of the coordinates
   - the number of dimensions, scaling, position, rotation, and
   - the number of sections to be interpolated between the defining sections. An increased number of sections gives the GCS greater control during optimization.
3. The section coordinates file contains the coordinates of the defining section.

The coordinates may be 2D or 3D but it must be stated in the section settings file. If the coordinates are 2D, it is assumed that they lie in the xz-plane. All translations and rotations are made about the first point in the file. The section must be closed by repeating the reference point at the end of the file. If the first and last points are not coincident, the last point will be overwritten with the first.

The output is the MicroSoft Visual Basic script file 'loft.catvbs' that can be run as a macro in CATIAv5. Once the geometry is generated, each of the sketch has to be changed to type "positioned" and anchored to the points defining the plane which contains it. Additionally, if spline control points are used as design variables, the sections coupling has to be of mode "Tangency then curvature" to allow greater surface flexibility. This is accomplished by double clicking on the part and in the "Multi-Sections Solid Definition" dialog box that pops up, clicking on the "coupling" tab and making sure that the "Tangency then curvature" option is selected.

A.2 Planform Variables

The geometry generation process outlined above allows great flexibility for control and modification during optimization. For wing optimization, one can change the control points defining individual airfoils or the planform shape: span, area, thickness, taper, sweep, dihedral and twist. The methodology is described below for some design variables which require some processing of the CAD data before updating them.

- Span

  The span design variable is the percentage that is applied to each section along the original span. This delta is linearly interpolated for each section from the full value at the tip to zero at the root. These interpolated values are then added to the y-coordinate of each corresponding plane defining the section.

- Twist

  The twist of an airfoil is the angle of attack of the airfoil minus the angle of attack of the root. The twist design variable is a delta that is added to the angle of attack of each airfoil section with the full value added to the tip and linearly interpolated to zero at the root. The twist of each section is defined as the rotation of the plane origin about the global y-axis. Since the sketch origin is "positioned" or constrained to the plane origin, it will rotate with the plane. Since each plane is defined by

3 points, given a rotation angle, we can use a rotation matrix to determine the rotation angle that has to be applied to these points in order to get their new positions. A rotation of ψ, θ, and ϕ degrees about each of the $x-$, $y-$, and $z-$axis, respectively, is defined as

$$\mathrm{R}_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{A.1}$$

$$\mathrm{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{A.2}$$

$$\mathrm{R}_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

The angles ψ, θ, and ϕ are known as Euler angles. A general rotation matrix is used to describe the rotation of the plane. It is a sequence of three rotations, one about each principal axis, represented as the matrix product:

$$\mathrm{R} = \mathrm{R}_z(\phi)\mathrm{R}_y(\theta)\mathrm{R}_x(\psi) \tag{A.4}$$

$$\mathrm{R} = \begin{bmatrix} \cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi \\ -\sin\theta & \sin\psi\cos\theta & \cos\psi\cos\theta \end{bmatrix} \tag{A.5}$$

$$\mathrm{R} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23} \\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33} \end{bmatrix} \tag{A.6}$$

Since matrix multiplication does not commute, the order of the rotation will affect the result. In this thesis, the rotation is done first about the $x-$axis, then the $y-$axis, and finally the $z-$axis.

- Chord

  The chord of an airfoil is the x-coordinate of the trailing edge minus the leading edge. The chord design variable is percentage that is applied to each of the original chords. This delta is linearly interpolated for each control point from the full value

at the trailing edge to zero at the leading edge. These interpolated values are then added to the x-coordinate of each corresponding control point.