

# Modeling and Sensitivity Analysis of Aircraft Geometry for Multidisciplinary Optimization Problems

by

David Sergio Lazzara

B.S., University of Southern California (2002)  
S.M., Massachusetts Institute of Technology (2004)

Submitted to the Department of Aeronautics and Astronautics  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

© David Sergio Lazzara, MMXII. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly  
paper and electronic copies of this thesis document in whole or in part.

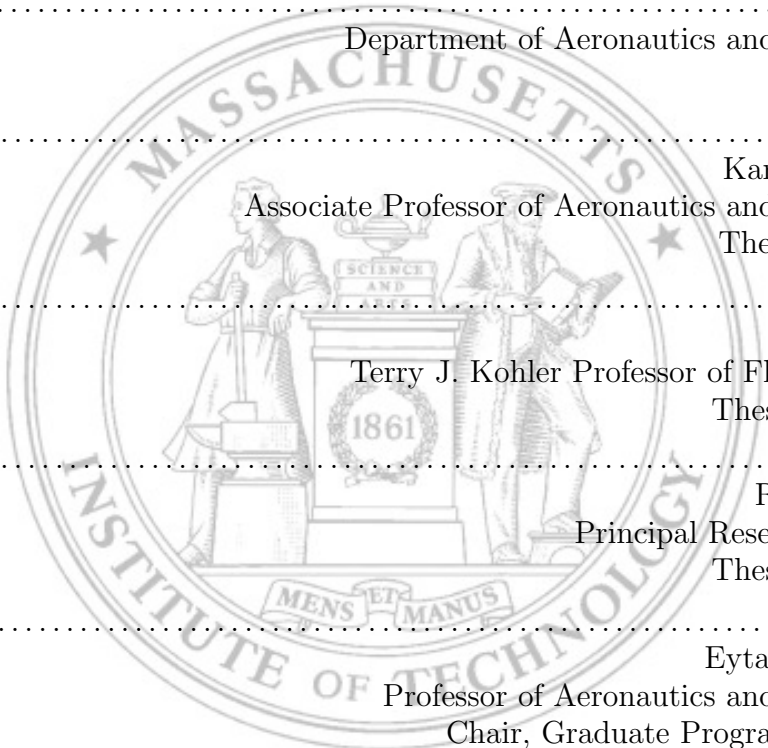
Author .....  
Department of Aeronautics and Astronautics  
May 24, 2012

Certified by.....  
Karen E. Willcox  
Associate Professor of Aeronautics and Astronautics  
Thesis Supervisor

Certified by.....  
Mark Drela  
Terry J. Kohler Professor of Fluid Dynamics  
Thesis Committee

Certified by.....  
Robert Haimes  
Principal Research Engineer  
Thesis Committee

Accepted by.....  
Eytan H. Modiano  
Professor of Aeronautics and Astronautics  
Chair, Graduate Program Committee

The seal of the Massachusetts Institute of Technology is centered on the page. It features a circular border with the text "MASSACHUSETTS INSTITUTE OF TECHNOLOGY" and "1861". Inside the seal, two figures stand on either side of a central pedestal. The figure on the left is a woman holding a scale, and the figure on the right is a man holding a book. Above the figures is a banner with the Latin motto "MENS ET MANUS".





# Modeling and Sensitivity Analysis of Aircraft Geometry for Multidisciplinary Optimization Problems

by

David Sergio Lazzara

Submitted to the Department of Aeronautics and Astronautics  
on May 24, 2012, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

## Abstract

A new geometry management paradigm for aircraft design utilizes Computer Aided Design (CAD) systems as the source for consistent geometry models across design phases and analysis tools. Yet various challenges inhibit the widespread application of CAD models in aircraft conceptual design because current CAD platforms are not designed for automated shape optimization. In particular, CAD models built with conventional methods can perform poorly in automated design frameworks and their associated CAD systems do not provide shape sensitivities. This thesis aims to remedy these concerns by bridging the computational geometry tools in CAD with aerospace design needs. A methodology for constructing CAD models is presented using concepts of multifidelity/multidisciplinary geometry and design motion. A formalized definition of design intent emerges from this approach that enables CAD models with parameterization flexibility, shape malleability and regeneration robustness for automated design settings. Analytic shape sensitivities are also presented to apply CAD models in gradient-based shape optimization. The parameterization and sensitivities for sketches, extrude, revolve and sweep features are given for mechanical design; shape sensitivities for B-spline curves and surfaces are also presented for airfoil and wing design. Furthermore, analytic methods modeling the sensitivity of intersection edges and nodes in a boundary representation (BRep) are given. Comparisons between analytic and finite-difference gradients show excellent agreement, however an error associated with the finite-difference gradient is found to exist if linearizing the support points of B-spline curves/surfaces and regenerating with a geometry kernel. This important outcome highlights a limitation of the finite-difference method when used on CAD models containing these entities. Finally, various example design problems are shown which highlight the application of the methods presented in the thesis. These include mechanical part design, inverse/forward design of airfoils and wings, and a multidisciplinary design space study. Gradient-based optimization is used in each design problem to compare the impact of analytic and finite-difference geometry gradients on the final designs obtained. With each of these contributions, the application of CAD-based geometry management is further enabled for emerging aircraft design frameworks.

Thesis Supervisor: Karen E. Willcox

Title: Associate Professor of Aeronautics and Astronautics



## Acknowledgments

I have been greatly blessed with the mentoring, friendship and expertise of so many individuals during my PhD experience at MIT that I could never fully express gratitude to all who contributed. Within the ranks of faculty, staff and students in the Department of Aeronautics & Astronautics, I have associated with wonderful individuals who impacted my personal development in ways they may never know.

I am especially grateful for the members of my Thesis Committee, namely Prof. Karen Willcox, Prof. Mark Drela and Bob Haimes. Each provided valuable feedback, guidance and a wealth of technical expertise that is truly unique in their respective fields. They provided a balance of direction and freedom which enabled me to develop a vision for the thesis work, and also taught me how to scrutinize, question and push the limits of what I thought was technically possible. Their prior work has greatly blessed my efforts to do research today. I am also amazed at how much I have personally grown thanks to the special talents they shared. I am also indebted to Marian Nemeč at NASA Ames and William Jones at NASA Langley for reading my thesis, participating in my Thesis Defense, and providing important feedback. Both Marian and Mike Aftosmis at NASA Ames enabled so much of my work thanks to their development of the Cart3D design framework; their timely support was always given and will always be remembered.

I extend a hearty thank-you to Jean Sofronas, Meghan Pepin and Beth Marois for their great support in their respective roles in the department. They truly understood that although sharing a smile does not cost anything, it is always priceless to the recipient. I am thankful to Prof. Karen Willcox, Prof. Youssef Marzouk and Prof. Alexandra Techet for allowing me to collaborate with them as a TA for various courses. I learned much from observing their role as educators and am thankful they allowed me to participate in their classes.

My friends in ACDL, both past and present, have always inspired me to learn more, grow faster and aim farther than I ever have. Each is tremendously gifted and I will always appreciate their allowing me to mingle among them. A big thanks to my cubicle-mate Andrew March for sharing so many stories, ideas and questions/answers with me. He is a great example of a loyal friend to many. Thanks to Hemant Chaurasia for sharing an office with a tremendous view and helping to clean it all—very thoroughly—before we moved in.

I extend much gratitude to all the other ACDL greats. To Masayuki Yano, Josh Krakos, Huafei Sun, Laslo Diosady, JM Modisette, Julie Andren and Eric Liu for being a part of the amazing Project-X team. To Chelsea He, Leo Ng, Rhea Liem, Cory Kays, Sergio Amaral and Beckett Zhou for being a part of the optimization group in the lab with me. Thanks also to Chad Lieberman and Doug Allaire for their friendship, as well as the UQ folks Nikhil Galagali, Patrick Conrad, Xun Huan, Alex Gorodetsky, Matthew Parno and Alessio Spantini. To the code developers David Moro, Joel Saa, Eric Dow, and Patrick Blonigan, who are gifted in so many ways. I also can't forget to express gratitude for the friendships developed with visiting European scholars Marcelo Buffoni and Laura Mainini, or post-docs Tarek Moselhy and others. There are many others who have gone before, like Garrett Barter, Todd Oliver and Krzysztof Fidkowski, who were successful in whatever they did.

Thank you also to all those I once knew from the MIT Graduate Soccer Club. We won it all by playing well. It was a pleasure to share the MIT jersey with you and lead such an international squad of great players. All the best to you in your future endeavors, and don't forget to pass the ball.

Likewise many thanks to all who participated with me in the LDSSA. Prof. Whitney Newey provided excellent counsel and was one of the nicest people I knew at MIT. We couldn't have asked for a better advisor. Thanks also to Prof. Thomas Eagar for assisting when needed, and especially helping me fund a part of my mission in Idaho by providing employment in his lab or start-up company. That help rendered much fruit.

There are also many others outside of MIT who deserve accolades for their loyal friendship. I hope to always remind them in person of how good they were to me. Thank you to Noel Bakhtian at Stanford for her Cart3D help and intense optimism. There are countless friends and mentors from the local congregations of The Church of Jesus Christ of Latter-day Saints who have been a strong source of strength and support as well. Heartfelt thanks go to Mindy McDonald, who is one of the closest examples of a ministering angel I've come to know. To my friends Brittany Baker, Amy Williams, Teppo Jouttenous, Jonathan Hopkins, Freddy Espinoza, Jeremy Jacox, Holly Greenburg, Abby Winston (Bushman), Mona Daniels, Jon Sue-Ho, Aaron Mazzeo, and Lauren Killian for taking the time to share the burdens of MIT and share the deeper things of life.

How grateful I am for amazing roommates who brought humor, fun and wonderful experiences. Living at the Dustin St. address has been fantastic for many years thanks to the

wonderful friendships of all who lived there at some point, including Allen Stoddard, Doug Rodermund, Don Mehr, Casey Reeve, Nathan Leishman, Bryan Monson, Matt Steenblich and Kelin Crane. A large amount of gratitude is owed to Dennis Villa for his friendship and mentoring; renting in his home brought countless unforgettable experiences, both during and after. Thanks to Denver Porter for introducing me to him and being so quick to share his gift of humor.

I am grateful for the inspired counsel given to me by mentors such as Clark Christian, Thomas Chapman, David Bokovoy, Bill Fogt, and Jonathan Austin at the Boston Institute of Religion. They all helped expand my mind in ways I had never experienced before in my academic career. More importantly, they transformed me. Much gratitude also goes to other mentors, such as Darrell Rigby, Roger Porter and both R. Brent Ririe and Beverly Ririe, who were truly inspiring during my mission years away from MIT. Lastly, I had the honor of working with exceptional youth who bring joy every time I see them. They are: Carlos, Andy, Anthony, Darbet, Exavier, Gismael, Kevin, Christian, Bryam, Gustavo, Jack, Kerlinks, Miles, Rolldy, Yoceiris, Josh, Roshlyn, Ray, Yancy, Mark, Chris, and others. May they continue to grow into strong and capable men that will change the world for good. A car-wash will never be the same without them. Thanks to Rich Alton and Phil Liddiard, and others, for helping me guide these youth through so many fun and memorable adventures. To anyone else I forgot to mention from the UW, LP1/2, Cambridge 3rd Spanish branch and CRW wards I ever knew: thank you for touching my life and leaving it better than you found it. It was a pleasure to serve with you.

Language cannot express the gratitude I feel for the support my parents, Susana and Sergio, and my brother Gary have given me throughout these years. Sharing Boston with my brother was a blessed experience—I will miss our dinners together. My family provided a perspective no one else could. I am especially thankful that they believed in me and encouraged me to keep my sights raised high when I thought to do otherwise. I believe their love, whether spoken or not, somehow helped me in ways I have not fully quantified. I dedicate this thesis to my parents, who structured their lives for my sake over many many years. They did so with immense sacrifice and little expectation of something in return. Whatever I have accomplished thus far is directly connected to something they taught me, to which I am forever grateful. I don't think I can ever fully repay the blessings they have shared with me.

Finally, and on an even more personal note, my acknowledgements would be incomplete if I did not give credit to my Father in Heaven for sending the inspiration behind every “light-bulb” moment I had during the course of this research. Whenever I asked Him for help, countless problems and bugs were resolved, challenges were eventually overcome, peace was granted, and tremendous strength was sent during trying times. The new ideas in this thesis were literally nuggets of light that splashed into my mind after I pondered the technical challenges I faced and researched solutions to no avail. Such has been the blessed way Heavenly Father has let me learn any universal truth. During the course of this project those moments brought real enlightenment that made this work exciting to pursue. I know my own faculties were often insufficient, yet He miraculously blessed me with the capacity to accomplish the work in this thesis. I am indebted to Him for all that He inspired herein. It is my hope that each reader experiences a “light-bulb” moment of inspiration at some point when reading this work, for it is when a revelation of truth is distilled to the mind and understood in the heart that learning occurs.

# Contents

## 1 Introduction:

<b>Geometry Management within Conventional Aircraft Design</b>	<b>41</b>
1.1 Geometry Management in Design . . . . .	41
1.2 Geometry Generation Methods . . . . .	45
1.3 Geometry Sensitivities for Shape Design Optimization . . . . .	50
1.3.1 Shape Sensitivity Analysis . . . . .	51
1.3.2 Geometry Sensitivity Methods . . . . .	52
1.4 Thesis Objectives and Contributions . . . . .	54
1.5 Thesis Outline . . . . .	56

## 2 Generating CAD Model Geometry **59**

2.1 Reasons for Focusing on Model Geometry in Design Optimization . . . . .	59
2.2 The Anatomy of CAD Model Geometry . . . . .	61
2.2.1 Modeling Geometry with CAD . . . . .	61
2.2.2 The Model Boundary Representation . . . . .	62
2.2.3 Model Topology Connectivity . . . . .	66
2.3 Model Parameterization Taxonomy . . . . .	69
2.3.1 Multifidelity Geometry Perspective . . . . .	69
2.3.2 Embedding Design Space Flexibility . . . . .	71
2.3.3 Parameterization Examples . . . . .	72
2.4 Defining a Model Design Intent . . . . .	77
2.5 Generating Multifidelity and Multidisciplinary Model Geometry . . . . .	83

2.5.1	Model Construction Methodology . . . . .	85
2.5.2	Single Discipline Examples . . . . .	88
2.5.3	Multiple Discipline Examples . . . . .	94
2.6	Summary . . . . .	105
<b>3</b>	<b>Geometry Sensitivities</b>	
	<b>for Sketches &amp; BRep Faces</b>	<b>107</b>
3.1	Geometry Gradients For Canonical Parametric Surfaces . . . . .	108
3.2	Geometry Gradients of Sketches for Sketch-Driven Surfaces . . . . .	113
3.2.1	Solving a Parameterized Sketch . . . . .	113
3.2.2	Design Velocity at Sketch Entity End-Points . . . . .	119
3.2.3	Design Velocity Along Sketch Entities . . . . .	127
3.2.4	Validation Examples . . . . .	131
3.3	Geometry Gradients of Extrude-Feature Surfaces . . . . .	142
3.4	Geometry Gradients of Revolve-Feature Surfaces . . . . .	146
3.5	Geometry Gradients of Sweep-Feature Surfaces . . . . .	151
3.6	Additional Commentary on Feature Geometry Gradients . . . . .	160
<b>4</b>	<b>Geometry Sensitivities</b>	
	<b>for BRep Edges &amp; Nodes</b>	<b>163</b>
4.1	Components of Design Velocity . . . . .	163
4.2	Closed-Form Intersection Problem . . . . .	165
4.3	Geometry Gradients on BRep Edges . . . . .	171
4.3.1	Derivation Using the Minimum Velocity Method . . . . .	171
4.3.2	Additional System Augmentation Options . . . . .	180
4.3.3	Validation and Comparison of Methods . . . . .	183
4.4	Geometry Gradients at BRep Nodes . . . . .	188
4.4.1	Extending the Trim Curve Sensitivity Derivation . . . . .	188
4.4.2	Considerations for Redundant Geometry . . . . .	192
4.4.3	Validation for BRep Nodes . . . . .	195
4.5	BRep Intersection Sensitivity Summary . . . . .	199



<b>5</b>	<b>Geometry Sensitivities for B-Spline Curves &amp; Surfaces</b>	<b>201</b>
5.1	B-Spline Curve Construction . . . . .	201
5.2	B-Spline Curve Geometry Gradient . . . . .	204
5.3	B-Spline Surface Construction . . . . .	208
5.4	B-Spline Surface Geometry Gradient . . . . .	210
5.5	B-spline Curves & Surfaces in CAD Models . . . . .	211
5.6	Examples of B-Spline Curve Geometry Gradients . . . . .	216
5.7	Examples of B-Spline Surface Geometry Gradients . . . . .	223
5.8	Scenarios Creating Inconsistent Finite-Difference Results . . . . .	228
5.8.1	Analysis for Linearized B-Spline Curves . . . . .	235
5.8.2	Analysis for Linearized B-Spline Surfaces . . . . .	242
5.8.3	Potential for Error Correction . . . . .	247
<b>6</b>	<b>Geometry Management Demonstrations</b>	<b>251</b>
6.1	Implementing CAD Model Geometry in Design Frameworks . . . . .	251
6.1.1	Overview of CAD Systems . . . . .	252
6.1.2	Inventory of Geometry Data . . . . .	254
6.1.3	Parameter Associativity to Model Topology . . . . .	255
6.1.4	Extracting Consistent Geometry Sub-models . . . . .	256
6.1.5	Impact of Problem Scaling on Geometry Management . . . . .	256
6.2	Design Framework Demonstrations . . . . .	260
6.2.1	3D Single-Discipline Mechanical Design Problem . . . . .	260
6.2.2	2D Single-Discipline Inverse-Design Problems . . . . .	264
6.2.3	2D Single-Discipline Design Problem . . . . .	274
6.2.4	3D Single-Discipline Inverse-Design Problem . . . . .	279
6.2.5	3D Single-Discipline Design Problem . . . . .	285
6.2.6	3D Multidisciplinary Design Space Exploration . . . . .	300
<b>7</b>	<b>Conclusions and Future Work</b>	<b>309</b>

<b>A CAD Model Generation</b>	<b>313</b>
A.1 Model Feature Considerations . . . . .	313
A.1.1 Perspectives on Generating Loft Features . . . . .	313
A.1.2 Mirroring Features . . . . .	316
A.1.3 Loft Self-Intersection . . . . .	316
A.2 Loft Definitions for Lifting-Surfaces . . . . .	316
A.3 Loft Definitions for Fuselage-Type Models . . . . .	326
A.4 Assembly Model Considerations . . . . .	334
A.5 Multidisciplinary Geometry Considerations . . . . .	339
A.6 Model Initialization and Automatic Generation . . . . .	345
<b>B PASS Parameterization Definitions</b>	<b>349</b>
<b>C Geometry Sensitivity Data</b>	
for BRep Faces	<b>351</b>
<b>D B-Spline Surface</b>	
Perturbation Results	<b>383</b>
<b>E “Snap” Grid</b>	
Design Velocity Results	<b>395</b>

# List of Figures

1-1	Evolution of aircraft concept representations throughout a typical “in-series” design paradigm. . . . .	43
1-2	With a new design paradigm, a single aircraft representation provides geometry information for multiple disciplines and various analysis fidelity levels. . . . .	45
2-1	The outer mold line of a model is punctured by an internal structural component at a given design point due to poor model construction. . . . .	60
2-2	A conceptual aircraft model BRep depicts 49 faces (grey) bounded by 123 edges (blue) and 82 nodes (red circles). . . . .	63
2-3	The trim curve (red) obtained at the wing-body intersection of a wing and fuselage is an edge in the model BRep. . . . .	64
2-4	The trim curve (red) from the model in Figure 2-3 is recalculated as a new trim curve (blue) when a perturbation in wing incidence angle (a driving geometry parameter) is done and the model is regenerated. Surfaces are tessellated before and after the perturbation to show which surfaces are driven by the parameter. . . . .	65
2-5	The intersection of multiple surfaces at a point becomes the node BRep entity. . . . .	65
2-6	Illustration of the node topology representation in a BRep. . . . .	66
2-7	Illustration of node perturbations where (a) no topology changes are made and where (b) topology changes are made with added trim curves and nodes. . . . .	67
2-8	An illustration of various parameterization levels that define models. Low-fidelity parameterizations can typically refer to <i>configuration</i> -level variables, whereas higher-fidelity parameterizations can refer to <i>primitive</i> -level variables. . . . .	70

2-9	An example of a low-fidelity <i>configuration</i> change is observed between (a) and (b) by modifying a <i>configuration</i> variable relating the wing and stabilizer location along the fuselage. Furthermore, higher-fidelity control of surface topology is seen between (c), (d) and (e) by varying <i>primitive</i> variables associated with the airfoil shape. . . . .	73
2-10	These SolidWorks 2010 user-guide images are two examples of employing design intent. In (a) the relative orientation and placement of geometry <i>features</i> (e.g., the drill holes) remain symmetric to the centerline and maintain a similar distribution pattern when the hinge length is changed. Figure (b) illustrates how various model dimensions (e.g., center-hole diameter, block width and height, hole height, etc.) can be changed and the <i>parts</i> maintain geometric similarity. . . . .	78
2-11	The concepts of design motion, design space, design point, design trajectory and design velocity are illustrated as a simple graph representation for a model $\mathcal{M}(\mathbf{x})$ , driven by a single design space $\mathbf{x}$ , to give greater intuition when describing a new design intent. . . . .	80
2-12	(a) Sketch entities are piecewise continuous and driven by parameters, such as the cone angle $\theta$ and height $h$ in this case. The CAD geometry kernel processes the sketch entities and generates associated surfaces for the cone <i>feature</i> . (b) Adding a cut-operator with a planar face results in a new model instance and an additional geometry parameter, $d$ . (c)-(d) The final model topology for this cone-plane <i>feature</i> shows edges $\{\mathcal{E}_1, \dots, \mathcal{E}_7\}$ (lines) and nodes $\{\mathcal{N}_1, \dots, \mathcal{N}_5\}$ (circles) bounding faces $\{\mathcal{F}_1, \dots, \mathcal{F}_4\}$ used to create the <i>feature</i> . . . . .	82
2-13	The design trajectory of the cut-cone model in Figure 2-12 is shown for a parameter range $\theta = \{25^\circ, 85^\circ\}$ at $1^\circ$ increments. Topology is unchanging on this design trajectory. BRep edges are shown for different $\theta$ values, whereas only the BRep nodes (boxes) are shown at each design point. . . . .	83

2-14	A node bifurcation is shown as the model passes through $\theta^* \approx 22^\circ$ . The initial node $\mathcal{N}_i$ bifurcates into $\mathcal{N}_{i+1,1}$ , $\mathcal{N}_{i+1,2}$ and $\mathcal{N}_{i+1,3}$ ; new edges are also created. The lower image highlights the “small” cut-plane face that exists just before the bifurcation occurs, which implies that the tolerance sphere for the initial node is substantial. . . . .	84
2-15	The automated model generation process is illustrated here: (a) depiction of global datum planes and 3D datum splines; (b) addition of <i>primitive</i> sketches as children of the datum splines; (c) final loft across the <i>primitive</i> sketches of each <i>configuration</i> component. . . . .	89
2-16	A parameterization consisting of piecewise-continuous line segments and elliptical arcs for the fuselage nose profile was defined outside of the CAD system. Classical <i>primitives</i> were used (elliptical arcs) to define the cross-sections. This parameterization determined the quadrant and center point distributions, as seen in (a) side view, and (b) top view. The distribution of elliptic <i>primitives</i> was driven in the CAD system by this parameterization, as shown in (c) side view and (d) top view. . . . .	90
2-17	A parameterization consisting of piecewise-continuous line segments was defined for the wing leading/trailing edges and quarter-chord. Reference markers for the airfoil are shown in (a) top view, (c) front view, and (e) isometric view. The distribution of spline <i>primitives</i> was driven in the CAD system by this parameterization, as seen in (b) top view, (d) front view, and (f) isometric view. . . . .	91
2-18	The automated model generation process is illustrated here for a flying-wing <i>configuration</i> : (a) depiction of 3D datum splines; (b) addition of <i>primitive</i> sketches as children of the datum splines; (c) final loft across the <i>primitive</i> sketches. . . . .	93
2-19	The airfoil stack for the generic flying-wing model is shown in this side-view, along with the span-wise twist distribution ranging from $+1^\circ$ at the root to $-3^\circ$ at the tip. . . . .	94
2-20	The airfoil stack for the generic flying-wing model is modified along the in-board 1/3 span by scaling the airfoils to create a greater center-body volume. . . . .	95

2-21	The automated model generation process is illustrated here for a lifting-body <i>configuration</i> : (a) depiction of 3D datum splines; (b) addition of <i>primitive</i> sketches as children of the datum splines; (c) final loft across the <i>primitive</i> sketches for each component. . . . .	96
2-22	The lifting-body fuselage is based on a NACA 3317 side-view profile and patched elliptical-linear sections in planform view. Elliptical <i>primitives</i> make up the cross-sections. . . . .	97
2-23	Structural components added to the generic tube-wing example followed the same geometry generation procedure as the outer mold line. . . . .	99
2-24	Spar cross-sections (red) are constructed to reference the wing airfoil definitions (dash-dot profile), thus remaining within the airfoil spline envelope. . . . .	99
2-25	Close-up view of fuselage ribs and stringers in the tapered nose section of Figure 2-23. . . . .	100
2-26	Close-up view of wing ribs and spars near the span break section of Figure 2-23. . . . .	100
2-27	Close-up view of ribs and stringers in the tapered aft section of the fuselage model in Figure 2-23. The rib cross-sections are extruded in the direction contrary to the fuselage taper direction. . . . .	101
2-28	A new design point is obtained after perturbing the fuselage datum spline, where the outer mold line loft design motion is followed by the internal structure. . . . .	102
2-29	A new design point is obtained after perturbing the wing datum spline. Internal structural components exhibit the same design motion as the outer mold line. . . . .	102
2-30	Another new design point is obtained by modifying the outer mold line loft definition via perturbation of an airfoil spline. The local internal structure follows the design motion of the wing loft as expected. . . . .	103
2-31	Internal spars and ribs are added to a flying-wing model. Each is driven by the parent wing loft, which undergoes changes in airfoil definition, twist, and datum spline deflection. . . . .	104

2-32	Surface intersections appear symmetrically across the body center-line when including internal spars and ribs in a flying-wing model. Although each is driven by the parent wing loft, which undergoes changes in airfoil definition, twist, and datum spline deflection, modifications of the spar offset distance is needed at the highlighted locations to remove surface intersections there.	104
2-33	The lifting-body model is given internal structural components which follow parameterization changes made to the tube-and-wing model.	105
3-1	(a) Sketch entities are piecewise continuous and driven by parameters, such as the cone angle $\theta$ and height $h$ in this case. The CAD geometry kernel processes the sketch entities and generates associated surfaces for the cone <i>feature</i> . (b) Adding a cut-operator with a planar face results in a new model instance and an additional geometry parameter, $d$ . (c)-(d) The final model topology for this cone-plane <i>feature</i> shows edges $\{\mathcal{E}_1, \dots, \mathcal{E}_7\}$ (lines) and nodes $\{\mathcal{N}_1, \dots, \mathcal{N}_5\}$ (circles) bounding faces $\{\mathcal{F}_1, \dots, \mathcal{F}_4\}$ used to create the <i>feature</i> .	109
3-2	An example model containing two extrusion <i>features</i> , $\Omega_1$ and $\Omega_2$ , consisting of simple and complex sketch-driving geometry.	113
3-3	(a) An under-defined sketch of two line segments, constrained vertically, that are coincident to two line segments, constrained horizontally, with corresponding dimensions. (b) A fully-defined version of the sketch in (a) due to the addition of two additional dimensions ( $d_{13}$ and $d_{14}$ ).	114
3-4	The fully-defined driving sketch for <i>feature</i> $\Omega_2$ in Figure 3-2.	116
3-5	The fully-defined driving sketch containing elliptical and arc sketch entities added to the sketch in Figure 3-4.	117
3-6	The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.	122
3-7	The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.	123
3-8	The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.	124

3-9	The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch. . . . .	125
3-10	The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch. . . . .	126
3-11	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{21}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	132
3-12	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{22}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	133
3-13	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{23}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	134
3-14	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{25}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	135
3-15	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{26}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	136
3-16	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{27}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	137
3-17	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{28}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	138
3-18	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{29}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	139
3-19	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{30}$ (step-size $h = 1.0 \times 10^{-4}(\pi/180)$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.	140



3-20	(a) Design velocity vectors of active sketch <i>primitives</i> for $\mathcal{P} = d_{31}$ (step-size $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method. . . . .	141
3-21	Face labels for the extruded sketch in Figure 3-5 are shown in (a) with no extrude-direction dependence of the design velocity field in (b). . . . .	142
3-22	The maximum difference between the analytic and finite-difference geometry sensitivity (across all <i>feature</i> faces) is shown against the parameters in Figure 3-5. . . . .	144
3-23	The addition of geometry “noise” increases the maximum difference between analytic and finite-difference design velocity. . . . .	145
3-24	Face labels for the revolved sketch in Figure 3-5. . . . .	146
3-25	The design velocity with respect to $d_{21}$ is decoupled to the revolve-direction in (a), whereas with respect to $d_{22}$ the design velocity field is coupled from the revolve-direction in (b). . . . .	147
3-26	The maximum difference between the analytic and finite-difference geometry sensitivity (across all <i>feature</i> faces) is shown against the parameters in Figure 3-5. . . . .	149
3-27	The addition of geometry “noise” increases the maximum difference between analytic and finite-difference design velocity. . . . .	150
3-28	Face labels resulting from a sweep <i>feature</i> of sketch 3-5. . . . .	152
3-29	The design velocity field in (a) preserves the sketch normal-direction, whereas the field shown in (b) has a normal-following effect on the $v$ -isoparameter line sketch gradients. Neither exhibit a dependence on the sweep path compared to the case in (c). . . . .	153
3-30	The maximum difference between the analytic and finite-difference geometry sensitivity (across all <i>feature</i> faces) is shown against the parameters in Figure 3-5. . . . .	154
3-31	Comparison of the analytic (blue) and finite-difference (red) design velocity vectors shows a design motion discrepancy between the pure circular-arc <i>primitive</i> and the rational B-spline representation on isoparameter lines. . .	155

3-32	The design velocity vectors in (a) correctly follow the expected normal-preserving orientation and show no sweep-trajectory dependence. In (b), however, the “snap” grid yields non-normal-preserving design velocity results with a spurious sweep-trajectory dependence. These design velocities reflect finite-differencing with the parameter $\mathcal{P} = d_{21}$ and step-size $h = 1.0 \times 10^{-4}$ .	157
3-33	The design velocity vectors in (a) correctly display a continuous derivative along the $u$ -isoparameter line (as seen in Figure 3-11). In (b), however, the “snap” grid yields incorrectly discontinuous design velocities along the $u$ -isoparameter line. These design velocities reflect finite-differencing with the parameter $\mathcal{P} = d_{21}$ and step-size $h = 1.0 \times 10^{-4}$ .	158
3-34	The addition of geometry “noise” increases the maximum difference between analytic and finite-difference design velocity.	159
4-1	The intersection of two cylinders and a plane at a node.	166
4-2	The norm of design motion offsets are plotted across $t = [0, 1]$ for seven methods with respect to an analytic solution.	184
4-3	The design velocities of seven Methods (labeled 1 through 7) were used to determine design motion by a linear perturbation of the cone-plane parameter $\delta d = 0.0004$ . The analytic result is labeled 0; the initial lower curve (blue) overlaps with a tangent vector (green) and the upper curve (red) is the perturbed curve.	185
4-4	The value of $\tilde{\nu}$ , or relative design velocity on the curve, as determined by methods 3, 6 and 7.	186
4-5	The error between design velocity results when geometry tolerances do not match (geometry “noise”) are shown for both Method 7 and Method 5.	187
4-6	Node validation cases tested on (a) a two-surface, four-face node intersection, (b) a one-surface, four-face intersection, and (c) a five-surface intersection.	194
4-7	Application of the node algorithm (equivalent to Method #1 for edges) on (a) a trim curve where $\partial\mathbf{r}/\partial u$ and/or $\partial\mathbf{r}/\partial v$ are aligned for both surfaces, and (b) an edge where both surfaces have $\partial\mathbf{r}/\partial\mathcal{P} = \mathbf{0}$ .	195

4-8	Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the three surface, closed-form intersection formulation in Figure 4-1. . . .	196
4-9	Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-6(a). . . . .	197
4-10	Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-7(a). . . . .	197
4-11	Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-6(c). . . . .	198
4-12	Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-6(b). . . . .	198
4-13	Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-7(b). . . . .	199
5-1	(a) Overall design motion of surface control points for a simple loft after perturbing a single support point ( $y$ -coordinate). (b) Zoomed view of the control point design trajectories. . . . .	213
5-2	(a) The variation in number of $U$ knots during design motion and (b) the design motion for the $U$ knots. . . . .	214
5-3	Example B-spline curves interpolated through support points (blue circles) with a control polygon (red circles) determined by the method in Section 5.1.	217
5-4	(a) Design velocity field with respect to $\mathbf{Q}_{k,y}$ for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero). . . . .	218

5-5	(a) Design velocity field with respect to $\mathbf{Q}_{k,x}$ for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero). . . . .	219
5-6	(a) Design velocity field with respect to $\mathbf{Q}_{k,y}$ for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero). . . . .	220
5-7	(a) Design velocity field with respect to $\mathbf{Q}_{k,x}$ for an example cubic B-spline curve. The green vector denotes the perturbation direction (unit magnitude). (b) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero). . . . .	221
5-8	(a) Design velocity field with respect to $\mathbf{Q}_{k,y}$ for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero). . . . .	222
5-9	(a) A SolidWorks model of a generic, solid loft <i>feature</i> with five cross-section B-spline curves, (b) contour regions denote the gradient magnitude in the $z$ -coordinate direction with respect to a variation in $z$ at a single support point, and (c) an overlay of design velocity vectors for the same case. . . . .	225
5-10	(a) A SolidWorks model of a generic, solid loft <i>feature</i> constructed with additional linearly-scaled cross-section B-spline curves, (b) contour regions denote the gradient magnitude in the $z$ -coordinate direction with respect to a variation in $z$ at a single support point, and (c) an overlay of design velocity vectors for the same case. . . . .	226
5-11	(a) A SolidWorks model of a generic, solid loft <i>feature</i> with various cross-section B-spline curves, (b) contour regions denote the gradient magnitude in the $y$ -coordinate direction with respect to a variation in $y$ at a single support point, and (c) an overlay of design velocity vectors for the same case. . . . .	227

5-12 (a) The magnitude of design velocity with respect to a support point $y$ -component with (b) associated design velocity vectors. . . . .	228
5-13 (a)–(c) Comparison of geometry gradient offset from finite-difference (knot-preserving) and Section 5.5 approaches. The $u$ - and $w$ -component offset are exactly zero in this case. . . . .	230
5-14 (a)–(c) Comparison of geometry gradient offset from finite-difference (non-preserving knots) and Section 5.5 approaches. . . . .	231
5-15 (a) A profile view of the design velocity vectors from the knot non-preserving central-difference and analytic method results of Figure 5-14. (b) A detailed view of the design velocity discrepancy. Analytic gradient results are in blue and finite-difference results (knot non-preserving) are in red. . . . .	232
5-16 (a)–(c) Comparison of geometry gradient offset between finite-difference results and the Section 5.5 approach on a CAD model geometry. . . . .	236
5-17 The impact of control point perturbations on knot vector reparameterization was done using this SolidWorks model geometry of a Blended-Wing-Body aircraft. . . . .	248
6-1 A design framework flow-chart where the geometry management paradigm has a secondary role. . . . .	253
6-2 A design framework flow-chart where the geometry management paradigm has a primary role. . . . .	254
6-3 Timing tests for automated model construction of a loft <i>feature</i> with increasing number of cross-section definitions. . . . .	257
6-4 A computational expense study for model regeneration as the number of model sketches increases (a) with dimensions and (b) without dimensions. . . . .	259
6-5 Timing tests for the geometry gradients of a loft <i>feature</i> with increasing number of cross-section definitions. Only one design variable per sketch is considered for differentiation. . . . .	260
6-6 (a) A pulley generated as a revolve <i>feature</i> using the sketch <i>primitives</i> parameterized in (b). . . . .	262

6-7	(a) Objective function history as influenced by analytic and finite-difference geometry gradients for the pulley design problem. (b) A comparison of the initial and final cross-section geometry for the pulley after optimization. . .	263
6-8	Comparison between a developed linear vortex panel code and XFOIL results with (a) 20 panels (top and bottom) on a NACA 0012 airfoil input geometry and (b) $C_p$ results. . . . .	265
6-9	Comparison between a developed linear vortex panel code and XFOIL results with (a) 100 panels (top and bottom) on a NACA 0012 airfoil input geometry and (b) $C_p$ results. . . . .	266
6-10	Simple 3D wing lofts were constructed in the SolidWorks CAD system to represent the (a) NACA 0012 and (b) RAE 2822 airfoil shapes. . . . .	267
6-11	(a)–(b) Inverse-design problem for the NACA 0012 airfoil depicting initial and final design points. . . . .	270
6-12	(a) Comparison between the objective function history results using the analytic and finite-difference design velocity methods for the inverse-design problem in Figure 6-11. (b) Comparison between the resulting design trajectory obtained with both methods. . . . .	271
6-13	(a)–(b) Inverse-design problem for the RAE 2822 airfoil depicting initial and final design points. . . . .	272
6-14	(a) Comparison between the objective function history results using the analytic and finite-difference design velocity methods for the inverse-design problem in Figure 6-13. (b) Comparison between the resulting design trajectory obtained with both methods. . . . .	273
6-15	In using the MSES analysis tool, (a) the initial design point for a NACA 0012 sub-model is driven to a (b) final design point. The airfoil $y$ -coordinates are scaled to show greater curvature detail. . . . .	275
6-16	(a) Comparison between the objective function history results using the analytic and finite-difference design velocity methods for the inverse-design problem in Figure 6-15. (b) Comparison between the resulting design trajectory obtained with both methods. . . . .	276

6-17 Comparison of objective function and design variable history between an analytic and finite-difference design velocity method for a drag minimization problem. . . . .	278
6-18 Comparison of $C_p$ distribution between the initial and final design point results using an analytic and finite-difference design velocity method in a drag minimization problem. . . . .	279
6-19 Initial and final design points for the 3D NACA 0012 wing via inverse design with the Cart3D design framework. . . . .	281
6-20 (a) The objective function history for the 3D NACA 0012 inverse design problem comparing the analytic and finite-difference gradient methods; (b) the design trajectory comparison between the two methods. . . . .	282
6-21 (a) Initial 3D NACA 0012 wing with spline point perturbation in a 2D analysis, followed by (b) the final design point obtained via inverse design with the Cart3D design framework. . . . .	283
6-22 (a) The objective function history for the 3D NACA 0012 inverse design problem comparing the analytic and finite-difference gradient methods; (b) the design trajectory comparison between the two methods. . . . .	284
6-23 Model geometry created using SolidWorks to represent a (a) RAE 2822 airfoil and (b) Blended-Wing-Body aircraft outer mold line. . . . .	286
6-24 Isometric contours of $C_p$ showing the initial and final distributions obtained with the unconstrained-lift BWB design problem. . . . .	291
6-25 Select cross-section $C_p$ distributions are shown from across the BWB semi-span to compare the impact of initial and final geometry obtained in the unconstrained-lift design problem. Blue denotes results stemming from analytic geometry gradients and Red denotes results using finite-differences. Non-smoothness in the inboard $C_p$ distributions reflect a locally more coarse mesh spacing. . . . .	292
6-26 Isometric contours of $C_p$ showing the initial and final distributions obtained with the constrained-lift BWB design problem. . . . .	293

6-27	Select cross-section $C_p$ distributions are shown from across the BWB semi-span to compare the impact of initial and final geometry obtained in the constrained-lift design problem. Blue denotes results stemming from analytic geometry gradients and Red denotes results using finite-differences. Non-smoothness in the inboard $C_p$ distributions reflect a locally more coarse mesh spacing. . . . .	294
6-28	Comparison of objective function history results stemming from analytic and finite-difference design velocity methods. . . . .	295
6-29	The design trajectory of design variable #1 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the unconstrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences.	296
6-30	The design trajectory of design variable #2 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the unconstrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences.	297
6-31	The design trajectory of design variable #1 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the constrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences. .	298
6-32	The design trajectory of design variable #2 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the constrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences. .	299
6-33	The 3D source geometry is tessellated to create (a) a complete quad mesh of the 3D structure surfaces and (b) a sub-model geometry consisting of quad elements from certain faces on the 3D surface. The mesh representing the wing skin is not shown. . . . .	301
6-34	The <i>assembly</i> connectivity embedded in the full 3D model, seen in (a), is by necessity conserved in (b) the sub-model <i>assembly</i> seen by the structural analysis. . . . .	303



6-35	An unloaded view of a structural sub-model (dark shade) is overlaid with a deflected sub-model (light shade) under loading to depict proper <i>assembly</i> connectivity. Improper connectivity would appear with displacements that violate first-principles. . . . .	304
6-36	Aero-structural coupling is captured by mapping a wing pressure distribution from an aerodynamic sub-model to a structural sub-model. The aerodynamics sub-model (dark shade) shows a wing pressure distribution (Mach 0.8, 0.4 lift-coefficient); the deflected sub-model (light shade) is the static structural response to the pressure loading. . . . .	304
6-37	Scatter plots of the design space obtained by Latin Hypercube sampling. The labeled points correspond to the structural layouts in Figure 6-38. . . . .	305
6-38	(a) A baseline structural layout is compared to various design points (b)–(h) obtained by automated Latin Hypercube sampling. . . . .	307
A-1	A comparison of two methods used to create cross-section sketch planes along a loft span. Figure (a) shows sketch planes constrained parallel to a global Cartesian coordinate plane. Figure (b) shows sketch planes constrained normal to a 3D spline reference datum at its support points; in this case the cross-sections exhibit out-of-plane design motion. . . . .	321
A-2	A comparison between continuous and piecewise continuous (discontinuous derivatives) datum splines. Figures (a)-(b) show sketch planes normal to a continuous spline at its spline points. Figures (c)-(d) show points of discontinuous derivatives on the piecewise spline, where two sketch planes pass through each spline point normal to a tangent vector on each side of the spline point. . . . .	322
A-3	With the exception of (a), these loft construction approaches illustrate limited usefulness for automated surface tessellation downstream of model construction. In (a) there are no splinter trim curves or faces; however, (b) contains splinter trim curves and (c) results in splinter faces. Clearly (d) highlights poor loft construction because a single closed volume is not obtained as desired.	323

A-4	In (a) the loft can represent asymmetric geometry modes; however, (b) a loft generated using a mirror feature cannot model asymmetric geometry (a skeleton of the asymmetric loft is also portrayed for comparison). The loft construction must embed support for each geometry mode needed in a model.	324
A-5	A loft is hollowed-out to demonstrate a region of self-intersection (grey) that results from a model design point that spreads two cross-sections too far apart. The large discrepancy between cross-section thickness also exacerbated this design motion for the loft. . . . .	324
A-6	A planform view ( $xy$ plane) showing the application of the anchor point mapping in (a), where the reference line and anchor points are on the wing leading edge. . . . .	325
A-7	A front view ( $yz$ plane) showing the application of the anchor point mapping in (a), where the reference line and anchor points are on the wing leading edge.	326
A-8	Perspective views showing the application of the anchor point mapping in (a), where the reference line and anchor points are on the wing leading edge.	327
A-9	A three-view of a wing with constrained datum spline and NACA 0012 airfoil cross-sections approximated by spline <i>primitives</i> , each of which are independent of all other components in the model. . . . .	328
A-10	(a) A side-view of a simple fuselage concept with a rounded nose cap (accomplished using guide curves along the fuselage top/bottom and sides). The top guide curve is shown in red to demonstrate inflection-points that occur after regenerating with smaller aft cross-sections. This occurs because the tangency vectors at the mid and aft spline points maintain their original orientation. (b) A perspective view for a simple fuselage concept with a sharp nose cap, as seen in supersonic aircraft. In this case the elliptical cross-sections are visible and centered on a datum spline with fixed spacing between sketch planes. . . . .	331
A-11	A three-view of a fuselage depicting unconstrained cross-sections made of elliptical <i>primitives</i> that are solely constrained to a reference datum spline. Each are independent of all other components in the model. . . . .	333

A-12	A perspective view of a simple fuselage concept assembled with a wing. A “path mate” is utilized, where the wing longitudinal position along the fuselage datum spline (shown in grey) is set by the central anchor point at the wing leading edge (shown in green). A relative vertical wing design motion is not possible in this setup. . . . .	334
A-13	A three-view of the reference datum splines that serve as the “skeleton” for subsequent cross-section sketches in a generic tube-and-wing model. . . . .	336
A-14	A three-view of the independent cross-section geometry <i>primitives</i> initialized after placing sketch planes along each reference datum in Figure A-13. . . . .	337
A-15	A three-view of the generic tube-and-wing model after lofting across each set of cross-section <i>primitives</i> from Figure A-14. . . . .	338
A-16	Two 2D splines sharing the same support points, but not the same span, cannot generate the same space-curve. . . . .	339
A-17	The red loft (spar structure) passes through the same cross-section planes as the grey loft (wing outer mold line), but does not share the same span. Hence, the spar loft cannot remain inside the envelope of the wing loft. . . . .	340
A-18	The design point for a wing-tip deflection exacerbates discrepancies between the design motion of the red loft (spar structure) and grey loft (wing outer mold line), leading to surface intersections that may not have occurred earlier on the model design trajectory. . . . .	341
A-19	The red loft (spar structure) passes through the same cross-section planes as the grey loft (wing outer mold line) and shares the same span. However, the spar loft still penetrates the envelope of the wing loft due to its construction method. . . . .	342
A-20	The spar is constructed using the wing airfoil profile as a constraint in the top figure; however, unless it is constructed with an <i>offset</i> from the wing contour as in the bottom image, the spar loft will more easily create a surface intersection. . . . .	342
A-21	The effect of approximating a loft swath with another loft is shown, where the wing upper face is approximated by the caps of a thin and box spar loft. As the spar width increases, the approximation improves and the CAD system cannot determine the intersection trim curve. . . . .	344

C-1	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for the extrusion length parameter. . . . .	352
C-2	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{21}$ . The $w$ -component data is exactly zero in this case. . . . .	353
C-3	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{22}$ . . . . .	354
C-4	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{23}$ . The $w$ -component data is exactly zero in this case. . . . .	355
C-5	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{25}$ . . . . .	356
C-6	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{26}$ . . . . .	357
C-7	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{27}$ . . . . .	358
C-8	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{28}$ . . . . .	359
C-9	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{29}$ . . . . .	360
C-10	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{30}$ . The $w$ -component data is exactly zero in this case. . . . .	361
C-11	Relative offset between the extrude <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{31}$ . The $w$ -component data is exactly zero in this case. . . . .	362
C-12	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{21}$ . . . . .	363
C-13	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{22}$ . . . . .	364
C-14	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{23}$ . . . . .	365

C-15	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{25}$ . . . . .	366
C-16	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{26}$ . . . . .	367
C-17	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{27}$ . . . . .	368
C-18	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{28}$ . . . . .	369
C-19	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{29}$ . . . . .	370
C-20	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{30}$ . . . . .	371
C-21	Relative offset between the revolve <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{31}$ . . . . .	372
C-22	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{21}$ . . . . .	373
C-23	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{22}$ . . . . .	374
C-24	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{23}$ . . . . .	375
C-25	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{25}$ . . . . .	376
C-26	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{26}$ . . . . .	377
C-27	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{27}$ . . . . .	378
C-28	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{28}$ . . . . .	379
C-29	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{29}$ . . . . .	380
C-30	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{30}$ . . . . .	381

C-31	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing for parameter $\mathcal{P} = d_{31}$ . . . . .	382
E-1	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{21}$ . . . . .	396
E-2	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{22}$ . . . . .	397
E-3	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{23}$ . . . . .	398
E-4	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{25}$ . . . . .	399
E-5	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{26}$ . . . . .	400
E-6	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{27}$ . . . . .	401
E-7	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{28}$ . . . . .	402
E-8	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{29}$ . . . . .	403
E-9	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{30}$ . . . . .	404
E-10	Relative offset between the sweep <i>feature</i> differentiation method and centered-differencing on a “snap” grid for parameter $\mathcal{P} = d_{31}$ . . . . .	405
E-11	The design velocity vectors in (a) correctly display a continuous derivative along the $u$ -isocline (as seen in Figure 3-12). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the $u$ -isocline. These design velocities reflect finite-differencing with the parameter $\mathcal{P} = d_{22}$ and step-size $h = 1.0 \times 10^{-4}$ . . . . .	406

- E-12 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-13). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{23}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 407
- E-13 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-14). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{25}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 408
- E-14 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-15). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{26}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 409
- E-15 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-16). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{27}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 410
- E-16 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-17). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{28}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 411
- E-17 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-18). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{29}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 412

E-18 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-19). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{30}$  and step-size  $h = 1.0 \times 10^{-2}(\pi/180)$ . A larger step-size (greater than  $1.0 \times 10^{-3}$ ) was needed to ensure a non-zero design velocity in this case. . . . . 413

E-19 The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-20). In (b), however, the “snap” grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{31}$  and step-size  $h = 1.0 \times 10^{-4}$ . . . . . 414



# List of Tables

1.1	Distinguishing features in traditional aircraft design phases. . . . .	42
2.1	Different hierarchy levels of BRep connectivity among the BRep entities. . .	68
2.2	Definitions for the various parameterization levels possible in a model. . . .	69
2.3	These PASS geometry parameters (see Appendix B) are categorized with respect to geometry level of fidelity for a monoplane aircraft <i>class</i> . Low-fidelity design variables on the left of the table are separated from high-fidelity design variables on the right. Italicized parameters with an asterisk are necessary to create a CAD model, yet are not explicit inputs as PASS parameters. . . . .	74
2.4	A parameterization for fuselage models utilized in the PASS aircraft design system. . . . .	75
2.5	A planform parameterization for lifting-surface models. . . . .	75
2.6	These derived variables are a function of the PASS design variables and provide further information needed to create a wing CAD model. . . . .	76
2.7	The application of conventional design intent ideas is appropriate for particular CAD-based model applications; however, all design scenarios do not benefit as well from this approach. . . . .	79
2.8	The conventional and proposed model construction methodologies yield distinct model attributes that better serve different post-application uses. . . .	87
3.1	The connectivity hierarchy for the cut-cone model in Figure 3-1. . . . .	110
3.2	Analytic surface parameterizations for the vertical plane and cone generated in Figure 3-1. . . . .	110

3.3	Analytic representation of partial derivatives for the plane and cone surface parameterizations in Table 3.2. . . . .	111
4.1	Summary of sensitivity information needed to determine the analytic gradient at node $\mathcal{N}$ for the cylinder-cylinder-plane intersection problem. . . . .	170
4.2	Comparison between the closed-form method, central-difference approximation and an analytic solution for the node sensitivity to $d_1$ . Underlined digits denote mismatches with the analytic solution. . . . .	171
4.3	Comparison between the closed-form method, central-difference approximation and an analytic solution for the node sensitivity to $R_2$ . Underlined digits denote mismatches with the analytic solution. . . . .	171
4.4	Comparison between the closed-form method, central-difference approximation and an analytic solution for the node sensitivity to $R_3$ . Underlined digits denote mismatches with the analytic solution. . . . .	172
4.5	Advantages and disadvantages of each trim curve sensitivity method. . . . .	181
4.6	Validation results for the node sensitivity algorithm on the geometry cases in Figures 4-6 and 4-7. . . . .	196
5.1	A comparison between the $U$ -direction knot vector for the B-spline surfaces generated in the central-difference problem of Figure 5-14. Underlined digits denote a variation from the baseline value. The step-size used in this case was $h = 10^{-5}$ . . . . .	234
5.2	The offset between the $i$ th B-spline curve knot vectors $U_{+h}^i - U_{-h}^i$ ( $i = 1, \dots, 5$ ) obtained in central-differencing for Figure 5-14 are shown for the non-preserving case. Non-zero offsets exist where the geometry gradient was computed ( $h = 10^{-5}$ ). . . . .	234
5.3	A comparison between the $U$ knot vector for the B-spline surfaces generated by finite-differencing the CAD model from Figure 5-16. The step-size used in this case is $h = 10^{-5}$ , which resulted in no change of the knot vectors. . .	237
5.4	A comparison between the $V$ knot vector for the B-spline surfaces generated by finite-differencing the CAD model from Figure 5-16. The step-size used in this case is $h = 10^{-5}$ , which resulted in a minor change of the knot vectors.	237

5.5	The offset between the $i$ th B-spline curve knot vectors $U_{+h}^i - U_{-h}^i$ ( $i = 1, \dots, 5$ ) obtained in finite-differencing the CAD model in Figure 5-16 are shown. Non-zero offsets exist where the geometry gradient was computed ( $h = 10^{-5}$ ). . . . .	238
5.6	Perturbation of a B-spline curve control point preserves its knot vector and support point parameterization. Perturbation size is $h = 10^{-5}$ in the $y$ -component of a control point in a cross-section B-spline curve of Figure 5-17.	249
5.7	Perturbation of a B-spline curve support point leads to knot vector and support parameterization changes. Perturbation size is $h = 10^{-5}$ in the $y$ -component of a support point in a cross-section B-spline curve of Figure 5-17. Underlined digits reflect deviations from the baseline case. . . . .	250
6.1	A summary of results for the pulley design problem. The reported computation time is an average over all geometry gradient computations in a design run. Underlined digits correspond to deviations from the target solution. . .	264
6.2	Setup data for the inverse design problems on simple NACA 0012 and RAE 2822 model geometry. The differentiated panel code is used for each problem except the NACA 0012 with the MSES designation. . . . .	269
6.3	Results from a design problem for wave-drag minimization wherein analytic and finite-difference design velocity are used. . . . .	277
6.4	Setup data for the inverse design problems on a simple 3D NACA 0012 model geometry and 2D cross-section case. The design problems are both executed using the Cart3D design framework. . . . .	280
6.5	Results comparison when using the analytic and finite-difference geometry gradients for an unconstrained-lift and constrained-lift design problem. Surface mesh size is about 196000 elements and average volume mesh size is about 935500 cells. Computation time is reported for the full geometry gradient. . . . .	288

6.6	Results comparison when using the analytic and finite-difference geometry gradients for an unconstrained-lift and constrained-lift design problem. Surface mesh size is about 596000 elements and average volume mesh size is about 935500 cells. Computation time is reported for the full geometry gradient. . . . .	289
6.7	A listing of rib, spar and wing skin thickness for the design point layouts in Figure 6-38. . . . .	306
A.1	A comparison between the B-spline surface parameters of the four faces highlighted in Figure A-21. . . . .	343
B.1	Wing parameterization used in PASS. . . . .	349
B.2	Horizontal tail parameterization used in PASS. . . . .	350
B.3	Vertical tail parameterization used in PASS. . . . .	350
B.4	Fuselage parameterization used in PASS. . . . .	350
D.1	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{21}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	384
D.2	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{22}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	385
D.3	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{23}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	386
D.4	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{25}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	387
D.5	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{26}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	388

D.6	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{27}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	389
D.7	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{28}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	390
D.8	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{29}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	391
D.9	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{30}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	392
D.10	Changing $U$ knot vectors on Faces $\mathcal{F}_4$ (semi-ellipse) and $\mathcal{F}_6$ (circular-arc) when finite-differencing with parameter $\mathcal{P} = d_{31}$ and step-size $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values. . . . .	393

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 1

## Introduction:

# Geometry Management within Conventional Aircraft Design

Challenges in geometry management within contemporary aircraft design motivate the research objectives and contributions presented in this dissertation. This introductory chapter summarizes the current technology and approaches typically used in geometry management for aircraft design. This includes the implementation of geometry in design frameworks, how the geometry is generated, and how geometry sensitivities are determined for optimization. Discussion of these topics begins with the evolution of geometry management in design. Secondly, conventional methods for generating model geometry are considered to facilitate geometry management in automated design settings. Finally, to accommodate gradient-based optimization of geometry, common methods for geometry sensitivities are introduced to identify areas for improvement. This modern state of geometry management provides the foundation for the research objectives that follow and bound the scope of this dissertation.

## 1.1 Geometry Management in Design

Geometry management in aircraft design has evolved throughout the last 100 years with the advent of improved design tools. As the analysis sophistication evolved, so did the

required geometry representation. Trial-and-error experimentation was the norm in the Wright Brothers era using manufactured (often by hand) prototype models. Early multidisciplinary design optimization (MDO) suites contained empirical analysis tools, which required only a small parameter space to describe an aircraft. Design methodologies utilizing high-fidelity computational fluid dynamics (CFD) with closed, 3D surface tessellations of geometry have since followed. Geometry definition has advanced in a similar fashion from manual drafting, to computer-aided drafting, to three-dimensional CAD model geometry.

Modern design paradigms mature an aircraft design through a sequence of “in-series” design phases, summarized in Table 1.1, yet give geometry management a secondary role in the design process. Figure 1-1 depicts this process with geometry evolving from initial sketches into wireframe and 3D model geometry for medium- and high-fidelity analysis, respectively. Analysis codes usually dictate the aircraft geometry representation needed in each phase. For example, low-fidelity codes require only planform geometry information, while high-fidelity codes require fully-defined 3D model geometry. Initially, since little geometry information is available, Raymer [66] and others [37, 31, 11] suggest the use of simple drafting methods, or artist renderings, as the means to cast the parameter-view of model geometry into a recognizable *configuration*<sup>1</sup>. The remaining preliminary and detailed design phases then commence with the addition of more specific geometry information.

<b>Design Phase</b>	<b>Design Scope</b>	<b>Analysis Fidelity</b>	<b>Geometry Representation Model</b>
<i>Conceptual</i>	Define the initial aircraft planform	Low	Planform parameters
<i>Preliminary</i>	Disciplinary trade studies on a fixed planform	Low to High	Planform parameters, idealized wireframe model
<i>Detailed</i>	Define all specifications for every aircraft component	High	3D, closed surfaces

Table 1.1: Distinguishing features in traditional aircraft design phases.

Since the different aircraft geometry representations are independently defined in the various design phases, there arises the potential for inconsistencies as the design progresses.

<sup>1</sup>Emphasis is given to the terms *configuration*, *assembly*, *part*, *feature* and *primitive* to distinguish components and parameter-levels found in a CAD model geometry. These are explained further in Chapter 2.



This is possible, for example, when information is lost through the recasting of higher-order geometry to a lower-order geometry (e.g., a fifth-order B-spline interpolated curve recast as a third-order B-spline). Since the aircraft geometry representation and analysis fidelity are coupled, it is unclear how the results for one geometry/analysis pairing should influence design decisions above another pairing. Ideally the high-fidelity results should impact design decisions the most. However, the “in-series” paradigm tends to leave high-fidelity analysis out of early design decisions. Furthermore, the model geometry achieved in the final design phase can be viewed as an object receiving input sizing-data without influencing (via feedback of high-fidelity results) the upstream design decisions that provided those inputs. This means that the 3D aircraft model geometry depends mostly on design decisions based on low-fidelity tools and simple early geometry representations.

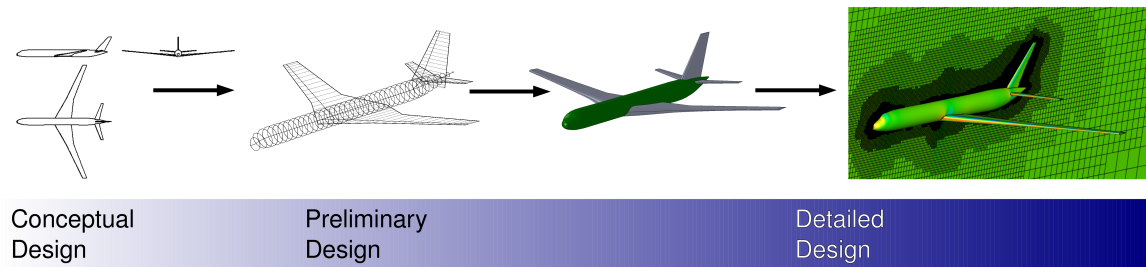


Figure 1-1: Evolution of aircraft concept representations throughout a typical “in-series” design paradigm.

Emerging conventional design processes with more stringent scheduling constraints and shorter schedules, will magnify inherent weaknesses in the design paradigms themselves. For example, Roskam [68] originally reported that design knowledge is inversely proportional to committed aircraft life-cycle program costs, where 65% of the program costs are often committed by the time a concept is selected at the end of conceptual design [40, 75, 63, 4]; with current design methods, it is clear that low-fidelity analysis tools are typically used to establish that cost commitment. However, the empirical nature of low-fidelity analysis limits its applicability to previous design types alone (e.g., tube-and-wing *configurations*); extrapolation of low-fidelity trends to account for new designs or technologies is generally less effective as well. This is manifest in later stages of the design paradigm, where high-fidelity analysis or testing informs designers that earlier design decisions were problematic as a result of insufficient analysis fidelity or geometry information. Therefore, advancing high-fidelity analysis into the conceptual design phase will likely eliminate these issues and

reduce the future cost of implementing engineering work-arounds [10, 28].

It is difficult to implement high-fidelity analysis early in current design paradigms. The difficulty arises because a fully-realizable 3D model of the aircraft is usually undefined and unavailable at the onset. These circumstances result in a distinction between the geometry representation of one design phase to another, as listed in Table 1.1, which reflects a different design scope, analysis fidelity and design methodology between design phases as well. In other words, the inconsistency in geometry representation leads to a *methodological gap* between each design phase. This results in a disjointed design process instead of the intended seamless one.

Even though an emerging CAD-based geometry management paradigm is beginning to gain emphasis (at least within academic circles), the introduction of high-fidelity analysis in an automated conceptual design setting still requires further research to improve geometry management. Bowcutt indicated in [10] that many advancements in design technologies were needed, especially in the generation of parametric geometry. One particular case of unconventional geometry management is in DEE (Design and Engineering Engine), which was developed to experiment with a Knowledge Based engineering approach to conceptual design [47]. This design framework is founded on the idea of having a central aircraft model geometry contain the data transmitted to all design phases and analysis disciplines, in contrast to the design phase executed “in-series” seen in Figure 1-1. Crawford et al. use a similar methodology in the design of a wind turbine [18]. By using these methods a single aircraft model geometry serves as the primary source of any geometry information which an analysis code requires (both across multiple disciplines and multiple levels of fidelity), as shown in Figure 1-2. Compared to the “in-series” paradigm of Figure 1-1, this approach utilizes a central “hub” of geometry information for all analyses and matures with evolving design changes.

Amadori et al. [2, 1] and Jounnet et al. [39] are part of a large development in improving conceptual design methodology via a new web-services, CAD-based framework for multidisciplinary optimization that employs some of the ideas in Figure 1-2. Their work encompasses the notion that empirical or statistical methods are appropriate in sizing aircraft, yet should not be the sole means of providing performance results. They point out that such methods cannot be extrapolated to include unconventional *configurations*, or those dissimilar to those used to create the empirical methods. Amadori et al. also recognize, as do

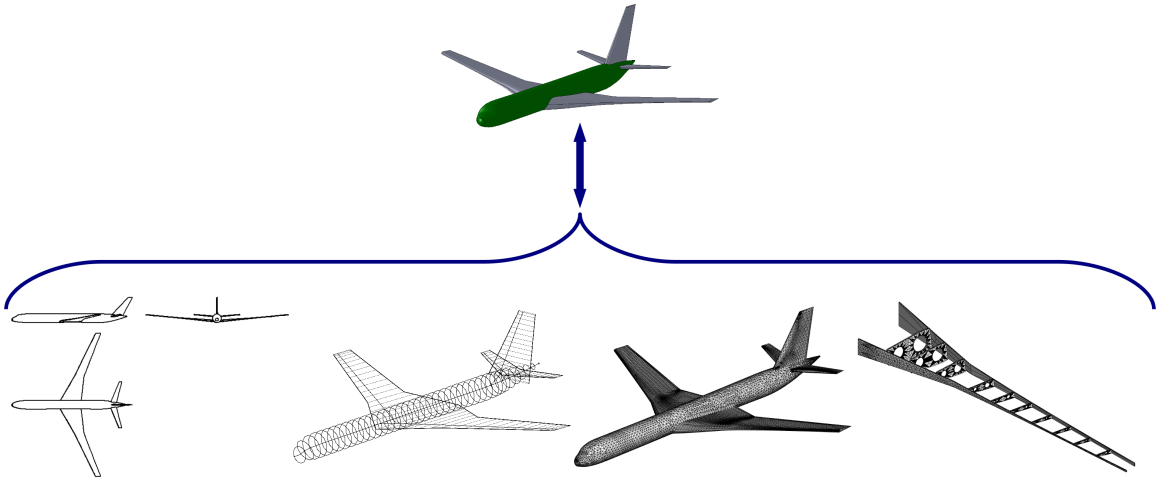


Figure 1-2: With a new design paradigm, a single aircraft representation provides geometry information for multiple disciplines and various analysis fidelity levels.

others [18], the need to have parameterized 3D model geometry that regenerates successfully over a continuous range of parameters. In their work, automating the generation of CAD model geometry is important in order to avoid the tedious manual process. Although their CAD-generated model geometry contains parameterized data used by the optimizer, their panel code aerodynamics module utilized a separate surface mesh until further development permitted tessellating the CAD model directly [1]. As in the case of [76], their work also utilizes heuristic methods instead of gradient-based optimizers because sensitivities are not required from the CAD-generated model.

## 1.2 Geometry Generation Methods

An important aspect of geometry management with CAD is the creation of geometry itself. A community of CAD-users have developed methods that serve to create a wide variety of mechanical devices within existing design processes. The shape optimization community has also developed analytic methods for representing models outside of conventional CAD systems, thereby permitting fine surface shape control and geometry differentiation. Each methodology serves as a useful tool within its respective design process; however, individually these methods may not fare well in an aircraft design setting where geometry management supports multidisciplinary optimization or multifidelity tools. Understanding both realms of model generation leads to an improved method that enables using CAD models early in design.

## Geometry Modeling

A 3D CAD-generated solid or model geometry consists of a manifold *Boundary Representation* (BRep) that is “closed” within a certain tolerance. To define a solid, groups of surfaces separate the “inside” of the solid from the “outside.” The computational geometry engine that generates geometry surfaces, executes intersection algorithms, solves geometry constraint relations coupled with parameters and outputs a model geometry description in some standard format (e.g., STEP format, or various BRep representations) is called a *geometry kernel*. Further details about the anatomy of CAD-generated models is found in Chapter 2.

There are at least two methods of working with BReps in practice. First, a *Parametric CAD* approach uses an embedded geometry kernel wrapped by a graphical user interface (GUI) and BRep management tools to manage an analytic description of geometry. Together, this is known as a *CAD system*, which is intended to simplify manual user-generation of model geometry by hiding the computational geometry details handled by the geometry kernel. Users/designers access model geometry via a GUI and conduct 3D solid operations (i.e., the addition/removal of surfaces) without directly accessing and modifying the analytic description. Instead, the CAD system maps the user inputs to the BRep and manages its modifications internally. The second method, a *Direct Geometry Construction* approach, provides a set of low-level construction *primitives* that are ultimately used to build a BRep by direct use of some geometry kernel (either proprietary or in-house code).

## Constructing Geometry with Design Intent

Model geometry is traditionally developed manually (or also via automated scripts) using a suite of “best-practices” and CAD system tools that users learn through experience and training. Many commercial CAD systems are available and system-specific instructions for “best-practices” are provided among numerous publications [48, 77, 78, 53, 52, 71]. In particular, these practices invoke the concept of *design intent*, which has steadily evolved from the realm of 2D CAD to 3D CAD. Many researchers focusing on design frameworks propose different definitions for design intent, as found by Iyer and Mills [32]. In their findings the design intent definitions range from the historical list of decisions throughout the model construction process to the complete set of model *features* and their attributes.

CAD vendors and CAD users also maintain individual views of design intent, typically as the selection of *features* and constraints to geometry that make up a model and create a desired appearance or behavior when design parameters are changed. Since CAD systems are often used to design mechanical components, users naturally identify design intent as the steps needed to develop a model into its final form for manufacture.

### **CAD-based Approach**

Parametric, or *feature*-based, CAD systems have evolved substantially in recent years and are well suited for design. These systems provide a variety of GUI setups for users to control model geometry construction. Users manually create a sequence of *features*, usually by sketching 2D *primitive* geometry and applying a 3D operation to the *primitive* sketches. The resulting ordered list of construction operations produces a *feature-tree* that, when executed (or regenerated), results in an instance of a *master-model*, which is an ordered collection of *features* and driving parameters that represents a high-level construction recipe for a model geometry BRep. *Features* can be added to the end of the feature-tree sequence as an independent entity or as an appendage to existing *features*, which typically means that previous *features* are modified (e.g., adding a fillet *feature* to a prior extrusion). Thus, the feature-tree perspective produces a high-level parametric abstraction for geometry construction handled by a geometry kernel.

CAD experience indicates that there is no unique feature-tree that creates a given solid model. Thus, multiple permutations of a feature-tree can create the “same” solid model, in the sense that the solid model has the same final shape, yet the underlying boundary representation topologies may be different. More importantly, the topology may not change in the same manner when perturbing driving design variables. Differences between model feature-trees often reflect a more elegant generation sequence that minimizes the need for a complex reference datum layout or explicit coupling between various *features* across the feature-tree. Furthermore, a difference in regeneration robustness is dependent on the datum referencing layout and the manner in which *primitives* are constrained and dimensioned. In some cases, many combinations of dimension values cannot regenerate into a new model instance [10]. If these potential differences are acknowledged, it becomes apparent that the final application of the model geometry begins to drive the feature-tree development towards fulfilling some intended purpose. This contrasts with the notion that

*only* the final shape of the model geometry matters. Lastly, understanding the CAD system strengths and weaknesses are paramount to ensure that models regenerate properly throughout a continuous range of driving dimensions [12].

### **Direct Construction Approach**

In contrast to the CAD approach, low-level geometry construction is traditionally employed throughout the early phases of design to represent simplified 2D “idealized” objects (i.e., simple, exact geometry cross-sections defined with classical *primitives*). Surfaces are also represented by many types of analytic geometry representations. Modification of such analytic definitions provides geometry control used by the shape optimization community. A survey of extensively-used parameterization methods for analytic model geometry is presented by Samareh [70]. Of particular interest are the parameterizations for airfoil shapes in 2D and wing surfaces in 3D. Kulfan provided a parameterization method using Bezier curves and surfaces for airfoils, bodies of revolution and typical surface forms seen in aerospace applications (e.g., nacelles, wings, and fuselage), with a small set of driving variables [45, 46]. Other formulations include NURBS curves and surfaces [64], parametric lines and arcs, cubic B-splines (and other spline types) and Bezier curves, all of which provide added local surface control. Even with a large variety of analytic approaches to define geometry, Chen and Tortorelli [14] point out some advantages and disadvantages in using analytical methods for model construction, which are discussed next.

The benefits of using direct construction include complete control over the shape and the potential to get analytic geometry derivatives. This is important for any gradient-based optimization which require sensitivities of the resultant geometry with respect to the parameters driving the design. By controlling the model construction at the lowest level, the resulting geometry is consistent with the designer’s intended construction and easily understood (i.e., a simple isolated wing may always have an intended four surfaces: root, tip, lower and upper). In this manner, each geometry entity is a form of *meta-data* for subsequent analysis or sensitivity calculation.

The disadvantages of using direct construction methods arise from the inherent fixed nature of the resulting geometry definition and the increased coding required to add functionality or geometry *features*. At the extreme end of high complexity, the direct construction method ends up recreating CAD systems, while at the opposite end it only yields simplis-

tic shapes that are clearly insufficient for complex geometries. Regardless of complexity, however, this construction approach must have the capability to generate water-tight solid models. Such a model definition is necessary in order to automate the tessellation of its geometry. The alternative is to close the BRep manually before tessellation is possible. With manual intervention the automation of design optimization is hindered, although approaches that utilize Constructive Solid Geometry (CSG) concepts, or a solids-based geometry kernel, can be used to circumvent the issue<sup>2</sup>.

Other challenges arise in the application of models generated with direct construction methods. Since full control of the geometry parameterization is available, there is potential to have a very large design space when attempting free-form shape design. For free-form curves or surfaces, the parameters may be control points or interpolation points which can yield “wavy-like” appearances if perturbed individually. Adjoint methods that utilize smoothing may circumvent this issue, as in [36], yet there remains a disconnect between the free-form geometry and traditional parameterizations like chord, sweep, aspect ratio, leading edge radius, camber and others. Free-form parameterizations may be too unconventional for designers and lead to difficulty in interpreting optimization results as well.

Finally, importing analytic geometry into a CAD-generated model (as is usually done later in design phases) may also introduce errors and inconsistencies. For example, importing non-native geometry *primitives* (such as higher-order splines that the CAD system does not support) clearly creates an immediate discrepancy between the optimized analytic geometry and the new CAD-generated model. It is unclear how the optimization results are interpreted for a distinct geometry representation in this case, nor how later design phases should incorporate those results when evolving the model. In the end, since manufacturing will often rely on a CAD-generated model, it is uncertain how its performance will compare to the optimized results stemming from an original analytic model since both model geometries are inconsistent in definition.

## Geometry Parameterizations

Many parameterization options have been studied and applied for shape design optimization. Kulfan [46, 45], besides providing parameterization for common aircraft components,

---

<sup>2</sup>Applying CSG with the Direct Construction perspective introduces complexity, though, because Boolean operations on solids result in surface shapes and trim curves that are no longer simple analytics.

points out that the chosen parameterization greatly impacts the MDO scheme utilized, depending on the number of design variables, geometry representation and possibility of design smoothness or irregularity throughout the design space. Many applications of design optimization in the literature represent varied parameterizations that are unique to the problem at hand, as found in [65, 16, 26, 39, 55]. The shape optimization community also provides a large number of parameterization choices. Engineering parameters, airfoil shape functions, NURBS surfaces, lines, conic sections, splines, algebraic curves, four-point curves, Bezier curves, and B-spline curves all contain different parameterization options. Various examples of parameterization in shape optimization are found in these references: [69, 14, 81, 6, 57, 58, 34, 42, 61, 9, 29, 70, 74, 15].

In CAD-generated model geometry, parameterization exists at different levels of transparency to the user. For example, the user may dimension geometry in a model via the GUI and define the dimensioning scheme as the model parameterization, or have such dimensions driven by external engineering parameters. Internally, though, a CAD system geometry kernel uses proprietary logic to parameterize the surfaces in a non-transparent manner (e.g., through surface generation and trimming algorithms) in order to fully construct a closed volume. Regardless of how the model geometry is driven, it is expected to regenerate within a predefined, bounded and continuous parameter set. Crawford et al. [18] accomplished this with wind turbine design. Amadori et al. automated the generation of 3D models in CAD [1, 2] using specific parameter sets that fixed the design space of the model. Vandenbrande et al. [79] at Boeing created the General Geometry Generator (GGG) to automate the construction of analytic models and have full control of its geometry parameterization.

### 1.3 Geometry Sensitivities for Shape Design Optimization

Once geometry exists for shape design, the calculation of geometry sensitivities becomes the next important part of geometry management. Zang et al. pointed out that (at the time of their publication) there was no analytic method for extracting surface sensitivities of CAD-generated models with respect to the model parameters [82]. Geometry sensitivities for a parameterized domain remains a topic of current research, much of which is conducted in the structures community for finite element meshes. Since CAD systems do not support shape sensitivities, a short summary of analytic, CAD-free and CAD-based geometry gradient



methods are presented from the literature.

### 1.3.1 Shape Sensitivity Analysis

Differentiation approaches used in other settings are often applied to CAD models with varying degrees of success. For example, automatic differentiation (as available using codes such as ADIFOR [7] and ADIC [8]) of entire CAD systems is not possible without its source code. Some researchers [41] prefer developing an in-house CAD system in order to fully differentiate the geometry kernel. However, such development remains limited in scope, unless constant scope expansion accommodates for greater design flexibility.

Other researchers utilize variants of shape sensitivity analysis to bypass the need for a differentiated CAD system. Choi and Kim [15] and Chen et al. [13] present a derivation of shape sensitivity analysis that is often used for discretizations of CAD geometry. Their approach introduces the concept of *design time* as a range of parameter values  $\mathcal{P}$  that evolve a geometry from an initial instance to a final instance. Through regularity assumptions, they show a linear mapping  $\mathbf{T}(\mathbf{r}, \mathcal{P})$  between one geometry instance  $\mathbf{r}$  and another  $\mathbf{r}'$  as

$$\mathbf{r}'(\mathcal{P}) = \mathbf{T}(\mathbf{r}, \mathcal{P}) = \mathbf{r} + \mathbf{V}(\mathbf{r}, \mathcal{P})\delta\mathcal{P},$$

for parameter perturbations  $\mathcal{O}\left(\delta\mathcal{P}^2\frac{\partial^2\mathbf{r}(\mathcal{P})}{\partial\mathcal{P}^2}\right) \ll 1$ . The term  $\mathbf{V}(\mathbf{r}, \mathcal{P})$  is designated the *design velocity*

$$\mathbf{V}(\mathbf{r}, \mathcal{P}) = \frac{\partial\mathbf{T}}{\partial\mathcal{P}} + \frac{\partial\mathbf{T}}{\partial\mathbf{r}} \frac{\partial\mathbf{r}}{\partial\mathcal{P}}.$$

It is the determination of design velocity  $\mathbf{V}(\mathbf{r}, \mathcal{P})$  that is of crucial importance in shape design. Choi and Kim [15], Chen et al. [13] and Hardee et al. [29] each show the derivation of the objective functional sensitivity to design parameters and include the necessity of finding the design velocity. Belegundu et al. used a similar derivation while using natural design variables, where the magnitude of the design variables pertained to a set of fictitious loads applied to a structural domain [6].

Objective functionals of the form  $F(\mathcal{P}) = \int_{\Omega(\mathcal{P})} f(\mathbf{r}(\mathcal{P})) d\Omega$  (where  $f(\mathbf{r}(\mathcal{P}))$  can be a scalar, vector or tensor) evaluated on a geometry domain  $\Omega(\mathcal{P})$  with boundary  $\partial\Omega(\mathcal{P})$  are representative of many performance metrics in structural analysis. For these Choi and Kim

[15] derive their sensitivity as:

$$\frac{dF}{d\mathcal{P}} = \int_{\Omega(\mathcal{P})} \frac{\partial f}{\partial \mathbf{r}} d\Omega + \int_{\partial\Omega(\mathcal{P})} f(\mathbf{V} \cdot \mathbf{n}) d\Gamma, \quad (1.1)$$

where  $\mathbf{n}$  is the local normal on the geometry. Only the design velocity on the *boundary* is required (*not* in the interior volume) to determine  $dF/d\mathcal{P}$ , assuming that the partial derivative for  $\partial f/\partial \mathbf{r}$  can be found (either analytically or numerically). Also, only the normal component  $\mathbf{V} \cdot \mathbf{n}$  of design velocity is required on  $\partial\Omega(\mathcal{P})$ . Chen et al. [13] further show that the design velocity is needed only on the *active* portions of the boundary, meaning boundaries directly influenced by the parameter  $\mathcal{P}$ , which saves computational expense. They also promoted the use of an implicit analytic function,  $\Phi$ , which required derivation of the parameter relationship to *primitives* used in constructing the solid model. This representation changed the sensitivity outcome to

$$\frac{dF}{d\mathcal{P}} = \int_{\Omega(\mathcal{P})} \frac{\partial f}{\partial \mathbf{r}} d\Omega + \int_{\partial\Omega(\mathcal{P})} \frac{f}{|\nabla\Phi|} \frac{\partial\Phi}{\partial\mathcal{P}} d\Gamma. \quad (1.2)$$

An advantage of using this method is that an implicit representation is only required for *primitives* in the model, and topology changes do not violate the calculation. However, defining  $\Phi$  may be very difficult in many cases.

Armstrong et al. and Robinson et al. in [3, 67], respectively, followed a similar procedure as outlined by Choi and Kim and Chen et al. In contrast to the use of an implicit representation by Chen et al. Robinson et al. utilized finite-differencing of triangulation element centroids to find the design velocity of the entire domain boundary. Issues with mesh vertices found on intersecting faces are avoided by using element centroids, yet this representation of the geometry is of lower-order since a single point is chosen to represent a planar mesh element. Lastly, Armstrong et al. relied on the Boundary Method for Design Sensitivity (described in Choi and Kim [15]) by combining adjoint sensitivity maps with the design velocity field to calculate performance sensitivity in a single analysis.

### 1.3.2 Geometry Sensitivity Methods

In the simplest scenarios, geometry sensitivity is easily calculated if an analytic expression exists between an objective functional and geometry. However, this is generally not the case for most types of geometry representation. Fudge et al. [22] provides a comparison of

advantages and disadvantages surrounding both direct construction approaches (also called CAD-free) and CAD-based methods in gradient-based optimization. In CAD-free scenarios, where a majority of shape optimization research currently exists, analytic representations of a domain or a surface tessellation are used with Free-Form Deformation (FFD) techniques, finite-differencing or adjoint methods to determine geometry sensitivities.

Samareh [70] and others in the shape design community utilize Free-Form Deformation (FFD) techniques to perturb and deform surface meshes. Chen and Tortorelli [14] provide geometry sensitivities by using variational geometry on analytic surfaces. Cosentino and Holst utilized finite-differences on cubic spline support points for airfoil optimization as well [17]; others utilize spline control points instead to limit potential undulations of the airfoil shape when finite-differencing. Shape design using adjoint methods with CFD exists for 2D and 3D geometry created without CAD [38, 25, 35, 34, 54]. In these instances a discretized geometry is the main geometry representation used in analysis, obtained from analytic surface definitions or tessellated CAD surfaces (however, the CAD model itself is not part of the optimization procedure). The geometry sensitivity becomes necessary by considering the sensitivity of an objective functional to design parameters. For example, consider a scalar objective functional  $G = G(\mathbf{P}, \mathbf{U}(\mathbf{P}))$  with parameter set  $\mathbf{P} \in \{\mathcal{P}\}$  and CFD flow solution  $\mathbf{U}$  that implicitly depends on  $\mathbf{P}$ . Here the analysis tool is a CFD flow solver that generates a volume mesh once given a surface tessellation. The sensitivity  $dG/d\mathbf{P}$  can be decomposed via the chain rule [82]:

$$\frac{dG}{d\mathbf{P}} = \left( \frac{\partial G}{\partial \mathbf{V}} \frac{\partial \mathbf{V}}{\partial \mathbf{S}} \frac{\partial \mathbf{S}}{\partial \mathbf{P}} \right)_{\mathbf{U}} + \left( \frac{\partial G}{\partial \mathbf{U}} \right)_{\mathbf{P}} \frac{d\mathbf{U}}{d\mathbf{P}}. \quad (1.3)$$

The first term is computed with fixed CFD solution  $\mathbf{U}$ , whereas the second term contains a partial derivative computed at fixed parameters  $\mathbf{P}$ . At fixed  $\mathbf{U}$ , the chain rule of terms begins with the sensitivity of the objective functional,  $G$ , to the volume mesh,  $\mathbf{V}$ . Secondly, the sensitivity of the volume mesh  $\mathbf{V}$  to the surface mesh,  $\mathbf{S}$ , follows. Thirdly, the sensitivity of the surface mesh to the parameters is required; this term couples the model geometry to the objective function. With the exception of the geometry sensitivity, the remaining terms can be computed using an adjoint in an adjoint-based CFD code ([60], [57], [59]). If the surface mesh is the only geometry representation of interest, then the term  $\partial \mathbf{S} / \partial \mathbf{P}$  is not needed because the mesh coordinates are the parameters in the problem. However, if

an underlying CAD geometry drives the surface mesh, then the geometry sensitivity term must be obtained by differentiating the CAD geometry itself.

Finite-differencing is commonly used to differentiate CAD-free and CAD-based model geometry [34, 28, 57, 58, 9, 41, 42, 59, 17]. This method is often appealing because it does not require access to CAD source code nor an understanding of the geometry kernel. Otherwise, it becomes necessary to “reverse-engineering” how a model is constructed.

One approach with finite-differences avoids meshing multiple geometry instances by utilizing a “snap grid” approach [57]; in this case the nearest point on the perturbed surfaces is queried using the  $(u, v)$  locations of the baseline. A more expensive approach is commonly used whereby the meshes corresponding to each perturbed geometry instance are differenced. The sensitivity of the nodes to the parameter perturbation is determined by differencing the new and old  $(x, y, z)$  coordinates of the vertices projected on the associated face or edge, at fixed normalized  $(u, v)$  values. An important assumption in this method is that the model topology does not change (i.e., the number of BRep faces does not change with perturbations). In either finite-difference method the step-size selection for each design variable is an important consideration. As the number of design variables increases, limitations on available CAD licenses become an issue since multiple regenerations may be distributed across numerous CAD installations [60].

## 1.4 Thesis Objectives and Contributions

The preceding discussion highlights various challenges that arise when conventional CAD systems are used for aircraft conceptual design. Implementing CAD within automated geometry management requires bridging computational geometry tools and aerospace design needs because these systems were not designed for these specific applications. By addressing the particular challenges surrounding model construction methods and shape sensitivities, this thesis aims to further enable the application of CAD systems in aircraft conceptual design.

Three main research objectives provide the scope for work presented in this thesis. The first objective is to automatically generate CAD-based model geometry suitable for conceptual design. This includes finding a new approach to CAD-specific construction methodologies that reforms traditional notions of “design intent” to utilize concepts of “de-

sign motion.” The second objective is to develop an analytic formulation for differentiating CAD models. This includes differentiating commonly used sketch entities, *features* (such as extrusions and lofts) and the model boundary representation. Finally, the third objective is to demonstrate the application of these tools when managing CAD models in design frameworks.

The first thesis objective is fulfilled by developing a construction methodology for CAD models that specifically targets their usage in automated conceptual design. The methodology stems from a new perspective on “design intent” driven by principles of “design motion” and geometry fidelity. Model construction tips are also given to avoid potential challenges that arise when CAD models are not designed for automated shape optimization.

The second thesis objective is fulfilled by formulating analytic geometry gradients of common CAD components. This includes various methods for calculating sensitivities on the model boundary representation (i.e., faces, edges, nodes), sketches, extrusions, revolve, sweep and loft *features*. These methods collectively yield geometry gradients for use on discretized geometry, necessary for optimization with high-fidelity analyses. Furthermore, sources of gradient error are analyzed to prescribe when a finite-difference approach is, or is not, suitable for a given CAD model. The analysis shows how a gradient error arises when CAD systems regenerate B-spline curves and surfaces for finite-differencing with respect to their support points.

The third thesis objective is accomplished by implementing CAD models in various inverse/forward design problems and a multidisciplinary design space exploration study. Each example implements the tools developed with the other thesis objectives. In particular, insight is gained on the usefulness of the new model construction methodology and the impact of analytic geometry gradients. These examples also portray how the thesis contributions further enable using CAD within geometry management in aircraft design optimization.

## 1.5 Thesis Outline

Each chapter in the thesis builds on prior material and is organized as follows:

### Chapter 2

The aspects of geometry management related to generating CAD models are discussed. Relevant fundamentals in computational geometry are presented to provide consistent notation and reference material for subsequent chapters. Perspectives are drawn from “design motion” in the shape-optimization community and a taxonomy for geometry fidelity to develop a new “design intent” definition. These principles are used in a proposed methodology for generating parametric CAD models suitable for automated aircraft design optimization.

### Chapter 3

The major role of geometry sensitivities in geometry management is discussed. A formulation of geometry gradients for sketches is presented and extended to the surfaces on extrusion, revolve and sweep *features*.

### Chapter 4

The geometry sensitivity methods are augmented further to include surface intersections in this chapter. A formulation for gradients on both trim curve and node entities of a boundary representation is presented. Comparison is made with a three-surface intersection formulation (a special case) and validation results against finite-differencing are shown.

### Chapter 5

Specific attention to B-spline curve and surface sensitivities is given in this chapter for aerospace applications. The fundamentals of B-spline curve and surface development are presented as a foundation for their geometry gradient formulation afterward. Analysis and examples are shown for gradient errors that arise when CAD systems regenerate B-spline curves and surfaces for finite-differencing with respect to their support points. This marks an important limitation to this method if applied on many CAD models.

## **Chapter 6**

Various practical considerations regarding the geometry management of CAD-generated models in design frameworks are presented, including the effect of scaling the number of parameters in a model. Implementation of tools presented earlier in the thesis are done with a mechanical part design, inverse/forward design problems of 2D airfoils and 3D wings, and a multidisciplinary design space exploration study. The effect of geometry gradient quality on final designs is also assessed.

## **Chapter 7**

A concluding discussion is presented that summarizes the thesis contributions. Suggestions are also given for future research that furthers the feasibility of applying CAD in the geometry management of automated aircraft design optimization.

THIS PAGE INTENTIONALLY LEFT BLANK



## Chapter 2

# Generating CAD Model Geometry

Geometry management begins with a methodology for constructing geometry representations. When managing CAD-generated models this becomes a non-trivial task if the resulting model must be suitable for automated design optimization. For completeness, a thorough presentation of CAD model geometry is presented in this chapter. The anatomy of a CAD-generated model is first presented to introduce terminology. The various parameterization levels for such a model are then defined, and a new formal definition for *design intent* is given. The focus then shifts to a proposed set of CAD-based model construction techniques which embed particular attributes requisite in design optimization frameworks. Various CAD-generated model geometries are explored to discuss the application of these new concepts.

### 2.1 Reasons for Focusing on Model Geometry in Design Optimization

The process of design optimization needs a greater focus on model geometry from the onset. Compared to research done in other aspects of design optimization (e.g., optimization algorithms, framework data management, simulation tools, etc.), there is little in the design optimization literature on the importance of geometry management, including how model geometries are constructed for implementation in a design framework. The CAD geometry community focuses on computational geometry and digital geometry representation. The shape design community focuses on methods for geometry sensitivity calculation, both for

CAD models and analytic geometry. The CAD user community typically focuses on the construction of CAD models for manufacturing. However, a bridge between all these CAD research activities *in the context* of design optimization is lacking.

In general, model geometries may hinder a design optimization framework if poorly managed for various reasons. First, if a model geometry cannot regenerate everywhere within the feasible design space, an automated optimization system will fail (unless the system is programmed to handle such exceptions). Secondly, automated analysis tools will fail or give spurious results if the input geometry is corrupted due to poor construction. This is evident, for example, if wing surfaces are “punctured” by internal structure after a new geometry is regenerated. As depicted in Figure 2-1, a high-fidelity aerodynamic analysis will then “see” a wing surface with sharp corners protruding the previously smooth airfoil contours and give spurious results. Again, the design system must be tolerant of these situations to avoid issues such as erroneous finite-difference calculations (otherwise the optimizer will struggle to determine an appropriate step direction). Thirdly, if the design space of the geometry is fixed, then the design system cannot explore outside of this design space. Hence, a completely different model must be constructed for each design space of interest, which results in added expense in the early time-limited phases of design. Fourth, if the model geometry is not constructed to morph in *expected* ways, then impractical geometry designs may result that cause an automated design system to bifurcate its design path away from the *intended* geometry shapes.

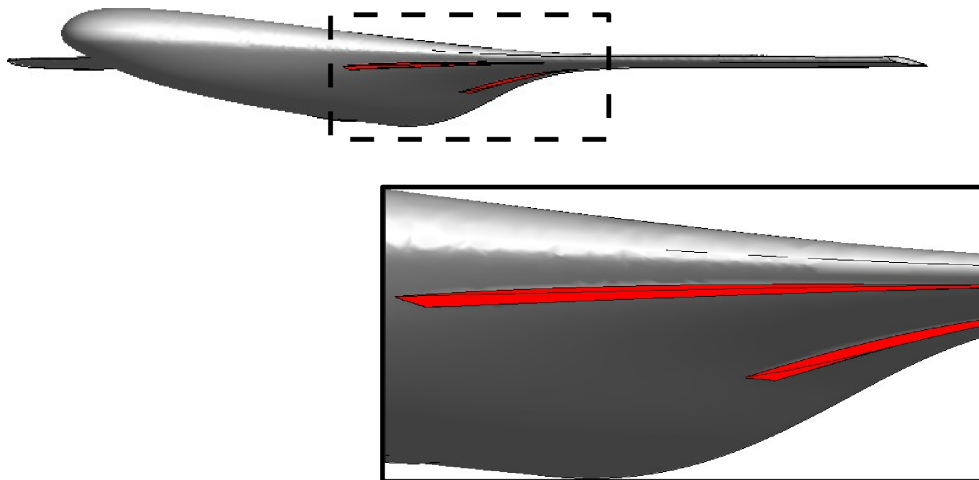


Figure 2-1: The outer mold line of a model is punctured by an internal structural component at a given design point due to poor model construction.

In order to address the issues presented here, a focus on CAD-generated model construction is provided to reduce or eliminate such problems. The intent here is not to develop a single, “perfect” CAD model geometry that will serve every design scenario. Instead, the intent is to create guiding principles for development of simple, adaptable models that are adequate for most design studies. Especially at the onset of design, minimal geometry is sufficient (e.g., an aircraft outer mold line and simple structural *features*) since more detailed geometry can be added later. This is also adequate for the low-order models used early in the design cycle.

Additional reasons for considering model geometry arise in finding geometry gradients of CAD models and casting non-CAD geometry into CAD models. Without insight into how CAD systems construct geometry, geometry sensitivities are often obtained through finite-differences. This gets difficult when changing model topology, or may get too expensive as the number of design variables increases. Furthermore, since design processes also inevitably lead to a manufactured model, the multiple geometry representations used early in design will eventually need to be translated into CAD geometry. This often leads to a loss of geometry information that makes the resultant CAD model inconsistent with the optimized geometry coming from prior analysis.

## 2.2 The Anatomy of CAD Model Geometry

CAD systems describe model geometry by various useful approaches depending on the level of interaction desired with the underlying geometry. From the perspective of the user and GUI, a high-level representation of the model is available; for greater access to the computational geometry representation of the model, the boundary representation with topology are accessible. These notions of model geometry are discussed here.

### 2.2.1 Modeling Geometry with CAD

When a designer uses a *feature*-based CAD system to construct a closed, 3D object, a complete build procedure must be created in a “top-down” manner. First, the abstract object is planned as a model containing a set of *features*  $\{\Omega_i\}$  that intersect via geometry operations (e.g., using Constructive Solid Geometry Boolean operations). The ordered construction of these *features* makes a build-recipe called the *feature-tree*, which generally

is non-unique for generating a given model, that is driven by a set of design parameters. The coupled feature-tree and parameter space are embodied in a *master-model* that defines an instance of the model representing the original abstract idea. The CAD system geometry kernel processes the feature-tree and parameters in the master-model, conducts the specified geometry operations on the *features* (i.e., generates appropriate surfaces and calculates their intersections) and subsequently defines a boundary representation (BRep) for the model. This BRep then becomes the computational geometry representation of the abstract object, whereas the master-model contains the construction and design information that bridges the abstract idea to a model and BRep.

Each *feature* in the model is also constructed by a non-unique build-recipe that correlates with the intended role of the *feature* in the model. This construction procedure consists of multiple 2D or 3D cross-section sketches which drive various geometry operators (e.g., extrusions, revolutions, cuts, blends, etc.) when applied. These sketches consist of numerous *sketch entities* (e.g., lines, arcs, conic sections, splines, etc.) connected in a piecewise continuous manner as a closed loop. The individual sketch entities may be driven by parameters and constraints; in some cases multiple sketch entities may be driven by the same parameter and constraints. Any parameter may be a stand-alone value driving a sketch entity, whereas in other cases the parameter may be driven by an expression in the master-model equation set or an equation outside of the master-model.

### 2.2.2 The Model Boundary Representation

The CAD geometry kernel will generate a *feature* by processing its appropriately defined sketches and applying geometry operators. This begins by “solving” the sketches, meaning the end-points of sketch entities (which are typically connected in order to create closed-volumes) are located in Euclidean space in terms of driving dimensions and constraints. The geometry operators reference the sketches when creating surfaces. Depending on the sketch complexity, these surfaces may have simple analytic representations (such as a cylinder or cone) or require NURBS definitions. Since the sketch may result in various surfaces, the geometry kernel executes surface-intersection algorithms, as needed<sup>1</sup>, and bounds the extent of the surfaces according to the underlying sketch end-points. The new *feature* is

---

<sup>1</sup>Typically the resulting surfaces are defined to their full extent per the sketch *primitives*; however, in some scenarios trimming is done for planar surfaces, which are of infinite extent and often serve as an “end-cap” surface to close a volume.

also intersected with the model surfaces on previously defined *features*. The resulting set of bounded, or trimmed, surfaces are thus labeled *faces*  $\mathcal{F}$ . These faces are bounded by loops of piecewise continuous *edges*  $\mathcal{E}$  that correlate with sketch *primitive* end-points or result from intersection algorithms. All edges are subsequently trimmed and bounded by *nodes*  $\mathcal{N}$  at the intersection with other edges. These entities define the model boundary representation.

The BRep entities also make up the model *topology*, which adds the connectivity information between faces, loops, edges and nodes. An example BRep of a conceptual aircraft model is shown in Figure 2-2, consisting of 49 faces bounded by 123 edges and 82 nodes. In general, the topology must be manifold and “closed” (i.e., obeying the Euler rules) within a tolerance in order for the model to be designated a *solid model*. When this is the case, the BRep and model topology separate the “outside” of the model from the “inside” volume. Model topology is also a consequence of the feature-tree and underlying construction of the model. These entities can be implicitly or explicitly driven by a set of parameters in the master-model.

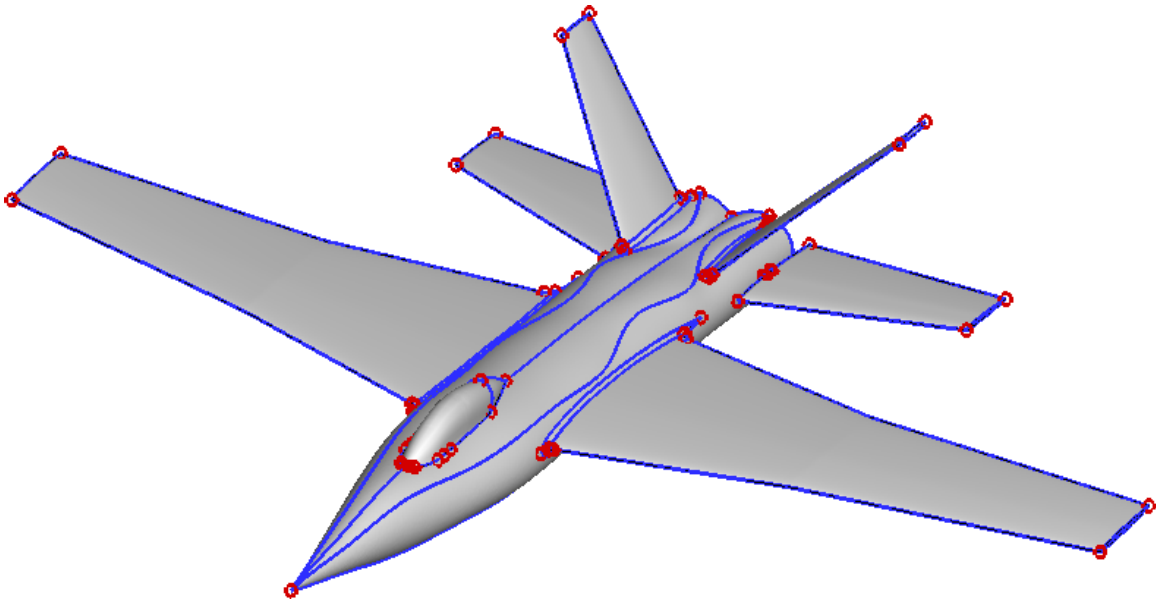


Figure 2-2: A conceptual aircraft model BRep depicts 49 faces (grey) bounded by 123 edges (blue) and 82 nodes (red circles).

A trim curve is generally a B-spline curve that approximates the true intersection space-curve between surfaces. These result from various trimming algorithms in a geometry kernel that find points of intersection between surfaces to interpolate with B-splines [56]. Figure

2-3 illustrates an example trim curve between a wing and fuselage intersection. Whenever a driving parameter is changed, the geometry kernel regenerates the model by recomputing the surfaces and their intersections to create a new model instance. For example, Figure 2-4 illustrates how a new trim curve is defined after changing wing incidence angle for the model in Figure 2-3. Surface tessellation of the wing and fuselage are also shown to indicate which surfaces change with this parameter.

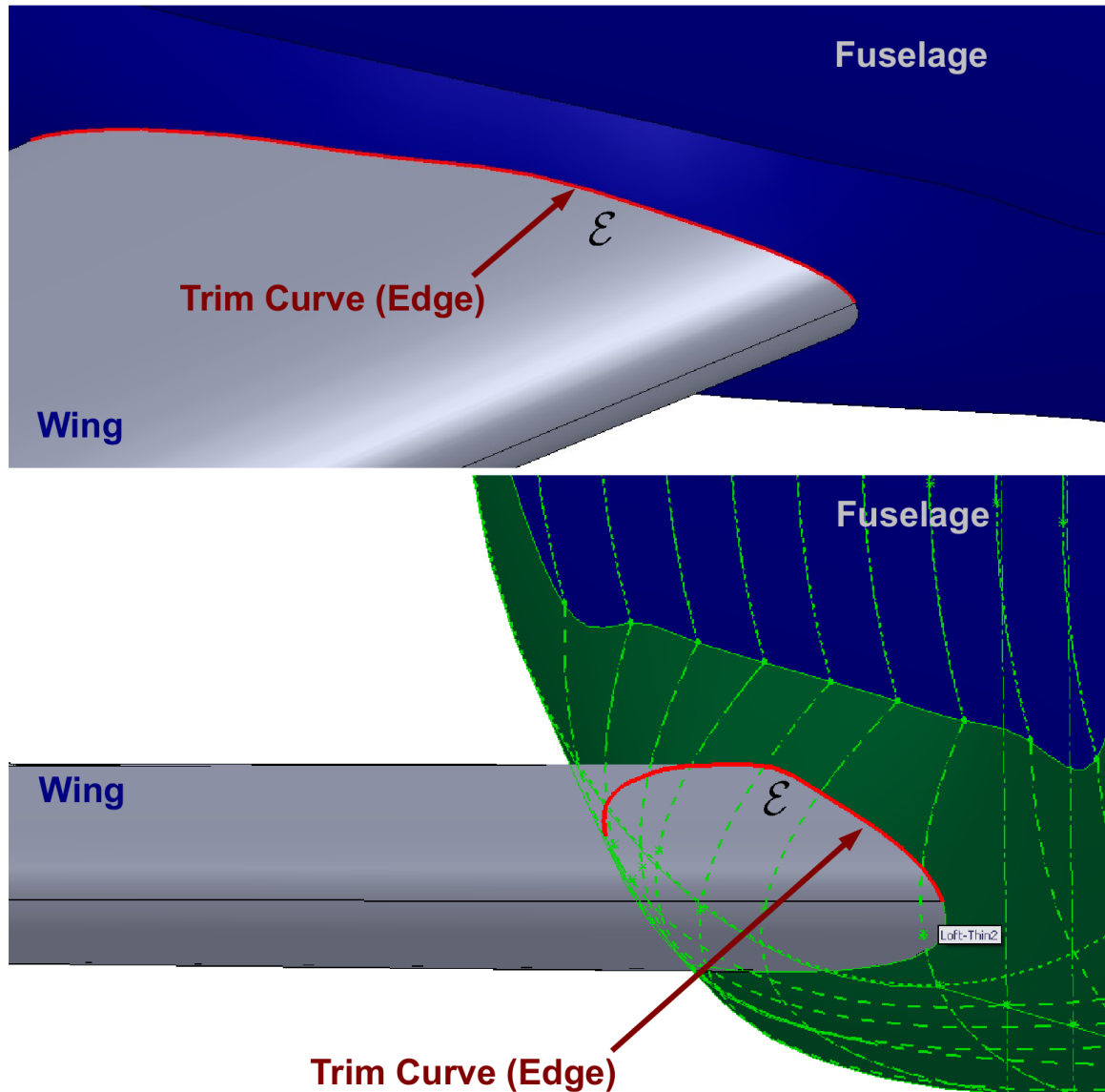


Figure 2-3: The trim curve (red) obtained at the wing-body intersection of a wing and fuselage is an edge in the model BRep.

The intersection at a node, as seen in Figure 2-5, is represented by a single point in  $\mathbb{R}^3$  by a geometry kernel. These are created by collapsing the end-points of intersecting edges

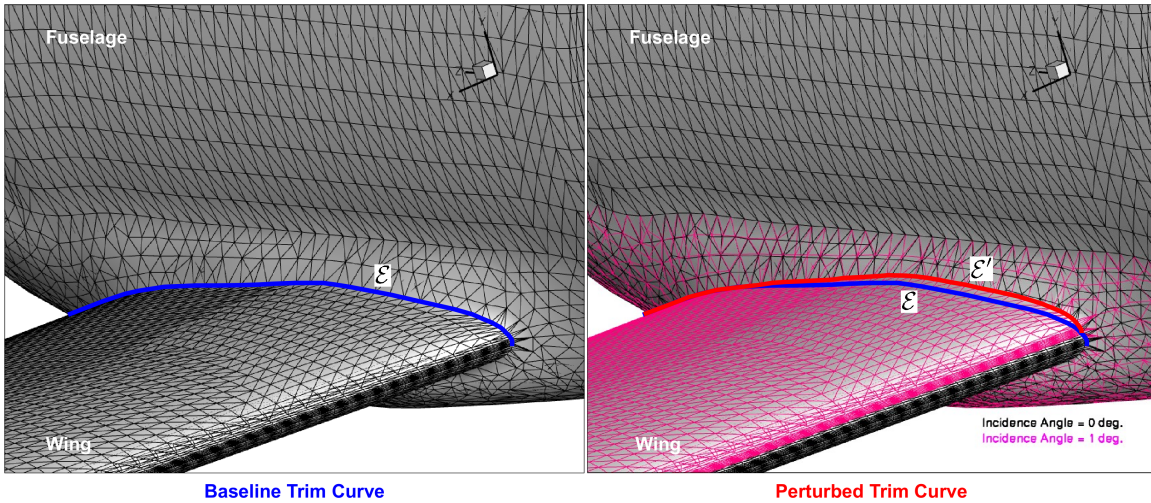


Figure 2-4: The trim curve (red) from the model in Figure 2-3 is recalculated as a new trim curve (blue) when a perturbation in wing incidence angle (a driving geometry parameter) is done and the model is regenerated. Surfaces are tessellated before and after the perturbation to show which surfaces are driven by the parameter.

to a single point, as seen in Figure 2-6. Such end-points are only used if they are within some relative tolerance sphere of each other. The specifics of these intersection algorithms and their logic are typically non-transparent to the CAD user.

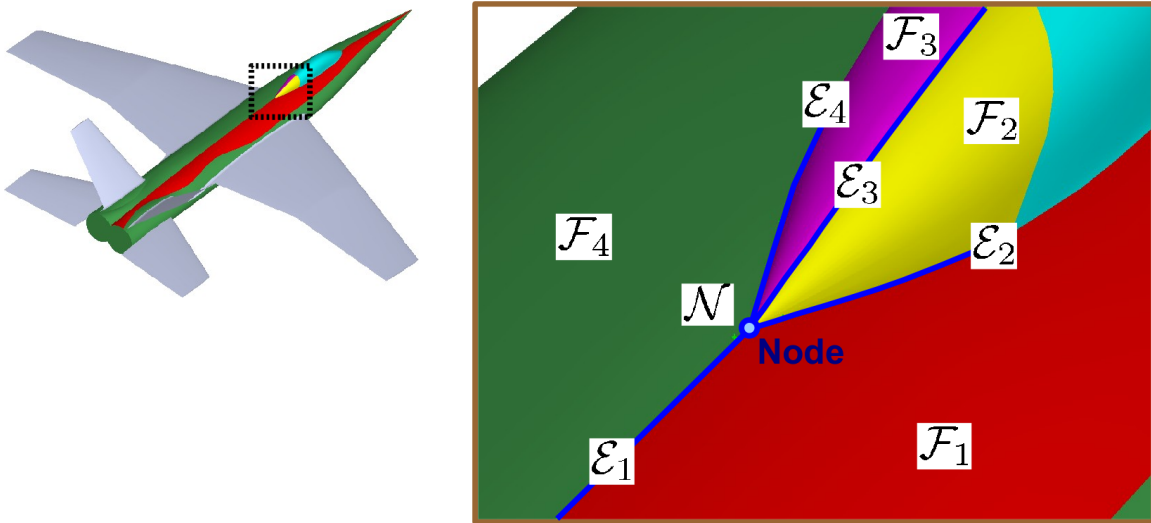


Figure 2-5: The intersection of multiple surfaces at a point becomes the node BRep entity.

When model parameters are changed, it is possible for the topology at the intersection nodes to not change. In this case, the parameter change may cause the trim curve end-points to move in tandem and maintain a relative spacing within a tolerance sphere of each other, as illustrated in Figure 2-7(a). This new intersection node is based on the

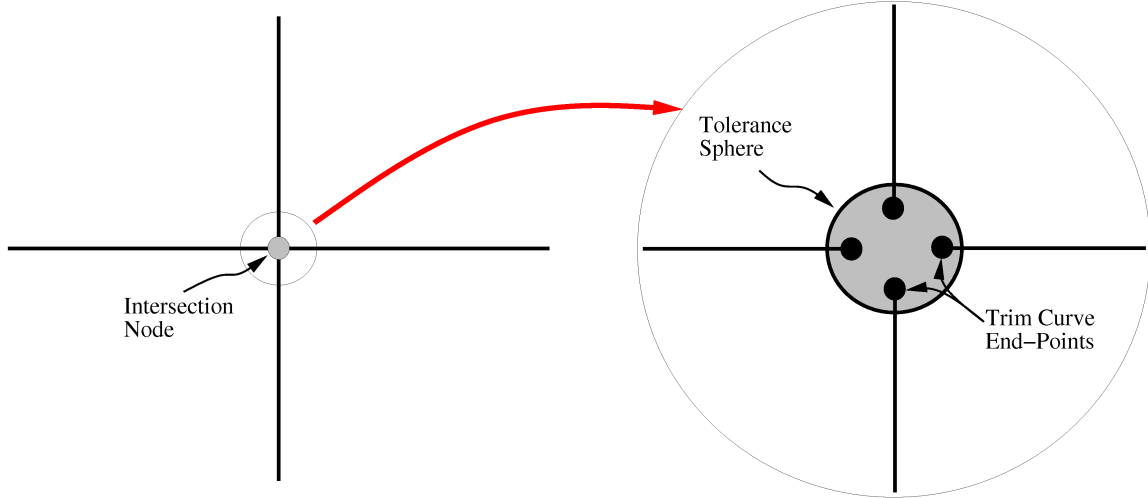


Figure 2-6: Illustration of the node topology representation in a BRep.

same initial trim curve end-points. The original topology is conserved in this case because no *additional* intersection nodes, trim curves or faces are created. In some instances, the model may also be constructed to preserve the single intersection node, thus disallowing the trim curve end-points from moving outside of their relative tolerance sphere. The case of three intersecting surfaces is a special case where the intersection node is always preserved; however, this preservation is less likely as the number of intersecting surfaces increases.

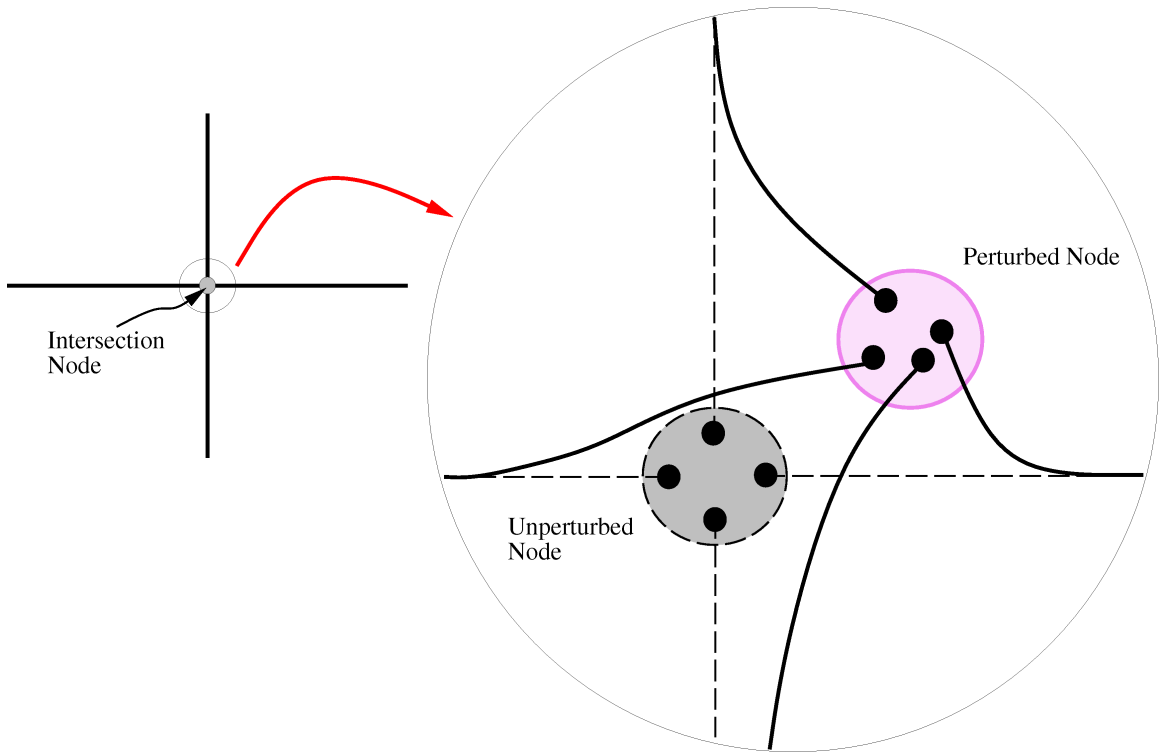
It is more likely, though, for parameter changes to cause a topology change at a node. This occurs when the new trim curve end-points move beyond their relative tolerance sphere and new intersection nodes and trim curves are created to close the volume. This scenario is depicted in Figure 2-7(b). As a result of two end-points being perturbed beyond the tolerance sphere surrounding the two remaining end-points, two additional nodes and trim curves, shown in blue, are created to close the topology (thus ensuring “watertightness”).

### 2.2.3 Model Topology Connectivity

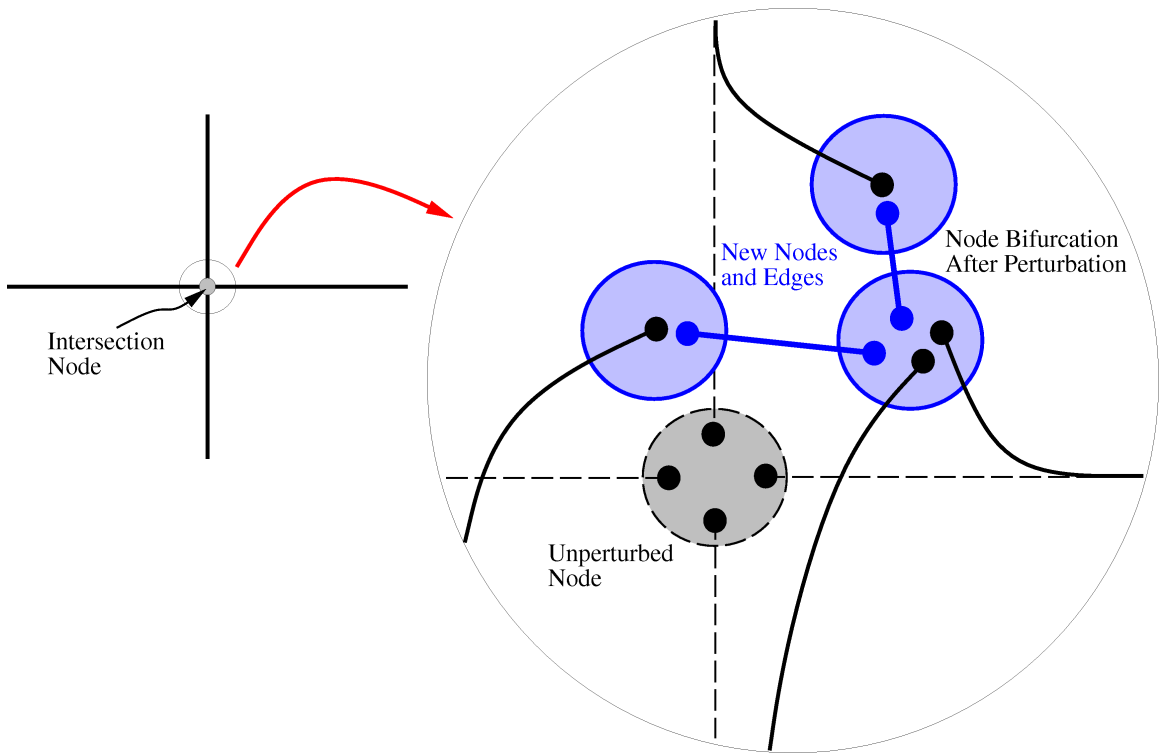
A geometry kernel determines the BRep topology connectivity to ensure a closed-volume from a computational geometry perspective. This connectivity information is also useful to have if shape sensitivities are desired for a CAD model. As discussed in Chapter 3, knowing how the BRep entities associate with the master-model information allows a designer to find how the model geometry is driven by parameters.

A set of intersected *features*  $\Omega = \{\Omega_i\}$  ( $i = 1, \dots, n_\Omega$ ) in a model is represented with a





(a)



(b)

Figure 2-7: Illustration of node perturbations where (a) no topology changes are made and where (b) topology changes are made with added trim curves and nodes.

BRep topology, wherein each *feature* points to the global listing of faces in the model BRep. For *feature*  $\Omega_i$ , the associated topology tree exists in hierarchical fashion from shells, faces, loops, edges and finally nodes. Sets of face, edge and node indices contain connectivity information as well. Table 2.1 shows the hierarchy for different connectivity levels, where the sets of topological elements are indexed as

$$\begin{aligned}
 \mathbf{F} &= \{\mathcal{F}_j\} & j &= 1, \dots, n_F \\
 \mathbf{E} &= \{\mathcal{E}_k\} & k &= 1, \dots, n_E \\
 \mathbf{N} &= \{\mathcal{N}_l\} & l &= 1, \dots, n_N.
 \end{aligned} \tag{2.1}$$

When looping over the *feature* set  $\mathbf{\Omega}$ , the topology tree is traversed by looping over the subset of faces  $\{\mathcal{F}\} \in \mathbf{F}$  pertaining<sup>2</sup> to  $\Omega_i$ . Each face branches to loops  $\{\mathcal{L}\} \in \mathbf{L}$ ; each loop index also branches to a subset of edges  $\{\mathcal{E}\} \in \mathbf{E}$ ; all edge indices then branch to a subset of nodes  $\{\mathcal{N}\} \in \mathbf{N}$ . Unless otherwise specified by the geometry kernel, each face has  $n_L$  loops, which is different for each face.

Entity	BRep Connectivity			
Features $\mathbf{\Omega}$	→ Shells $\{\mathcal{S}\}$ → Faces $\{\mathcal{F}\}$	→ Loops $\{\mathcal{L}\}$	→ Edges $\{\mathcal{E}\}$	→ Nodes $\{\mathcal{N}\}$
Faces $\mathbf{F}$	→ Feature $\in \{\mathbf{\Omega}\}$ → Shell $\in \{\mathcal{S}\}$ → Loops $\{\mathcal{L}\}$	→ Edges $\{\mathcal{E}\}$	→ Nodes $\{\mathcal{N}\}$	
Edges $\mathbf{E}$	→ Face $\in \{\mathcal{F}\}$ → Loop $\in \{\mathcal{L}\}$ → Nodes $\{\mathcal{N}\}$			
Nodes $\mathbf{N}$	→ Face $\in \{\mathcal{F}\}$ → Loop $\in \{\mathcal{L}\}$ → Edge $\in \{\mathcal{E}\}$			

Table 2.1: Different hierarchy levels of BRep connectivity among the BRep entities.

Looping over the sets  $\mathbf{F}$ ,  $\mathbf{E}$  or  $\mathbf{N}$  yields connectivity information in a different manner. Each face  $\mathcal{F}_j$  points to a single generating *feature* index,  $\{\Omega\} \in \mathbf{\Omega}$ , and has a topology tree that expands as described for the *feature* set above. From the view of edges, the faces that

<sup>2</sup>Groups of faces may also be catalogued as a *shell* and multiple shells  $\{\mathcal{S}\}$  may be attributed to a *feature*.

meet along edge  $\mathcal{E}_k$  are a subset  $\{\mathcal{F}\} \in \mathbf{F}$  and the end-point nodes are a subset  $\{\mathcal{N}\} \in \mathbf{N}$  (both sets have a cardinality of 2). Finally, from the perspective of nodes, the faces and edges that intersect at node  $\mathcal{N}_l$  are a subset  $\{\mathcal{F}\} \in \mathbf{F}$  and  $\{\mathcal{E}\} \in \mathbf{E}$ , respectively (the cardinality of which are both equal to  $\chi_l$ ). It is obvious that each node in the BRep is constructed from a different number of intersecting faces/edges.

## 2.3 Model Parameterization Taxonomy

The perspective used to categorize parameters in a model geometry is an important consideration when planning the model construction process in a CAD system. The notions of multifidelity geometry described here are useful to employ when a CAD model is meant to support multifidelity analysis in a design optimization setting.

### 2.3.1 Multifidelity Geometry Perspective

There are a number of ways to parameterize a solid model generated in a CAD system. In the context of aircraft conceptual design, though, a natural choice of parameterization hierarchy is obtained by starting from a planform-view of the model to a more detailed view. Using this perspective, a general taxonomy of parameterization levels are identified as: *class*, *configuration*, *assembly*, *part*, *feature* and *primitive* levels. This taxonomy is helpful in the context of design frameworks because each level represents a different *geometry fidelity* that corresponds with the fidelity of analysis that utilizes such information. Figure 2-8 captures the essence of a general model parameterization with the levels defined in Table 2.2.

Parameter Level	Definition
Class	Distinguish <i>configuration</i> layouts
Configuration	Distribute and orient a set of <i>assemblies</i>
Assembly	Distribute and orient a set of <i>parts</i>
Part	Relate the distribution and orientation of a set of <i>features</i>
Feature	Dimension a group of <i>primitives</i>
Primitive	Dimension properties of a classical or spline <i>primitive</i>

Table 2.2: Definitions for the various parameterization levels possible in a model.

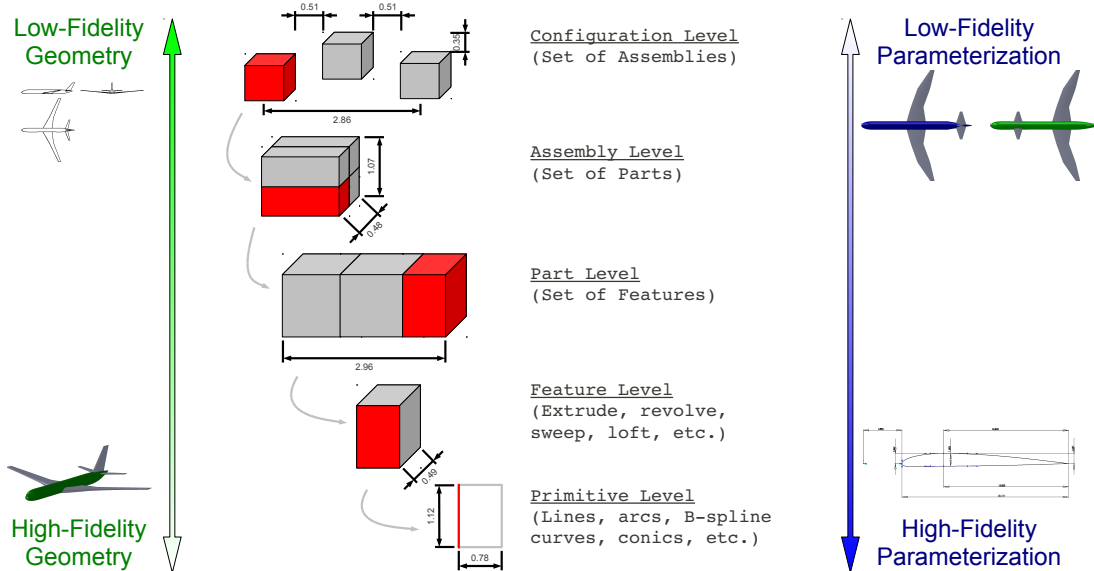


Figure 2-8: An illustration of various parameterization levels that define models. Low-fidelity parameterizations can typically refer to *configuration*-level variables, whereas higher-fidelity parameterizations can refer to *primitive*-level variables.

The *class* level is not shown in Figure 2-8 because it serves to distinguish between *configurations* such as monoplane, bi-plane, blended-wing body and other complex geometries. Each *class* embodies the design space illustrated in Figure 2-8, which is of greater interest here. All red entities in Figure 2-8 become the focus for the subsequent level of parameterization. For example, one of the three *assembly* blocks in the *configuration* level is chosen to highlight its four component *parts*. One of the four *parts* is then selected to view the three *features* (shown as blocks) used to generate it. By choosing one of the *features*, the *primitive* level geometry and variables are displayed.

This taxonomy of geometry parameterization provides a perspective to view levels of geometry information. It also assists the designer in planning where driving parameters are defined within the model. If Figure 2-8 is viewed as a hierarchy of geometry information in a model, the *configuration* and *assembly* variables can be designated *low-fidelity geometry variables* because they are typically associated with planform descriptors. At the other end, the *part*, *feature* and *primitive* variables resolve surfaces in the model and may be designated *high-fidelity geometry variables*. This labeling distinction references the amount of geometry information available at each level. Low-fidelity variables inherently describe a

*low-fidelity geometry representation* which does not provide surface information. The high-fidelity variables thus describe a *high-fidelity geometry representation* that does give surface information. By generating a model that embodies both low- and high-fidelity geometry, the model is appropriately designated a *multifidelity geometry representation*. As shown in subsequent sections, it is interesting to note that the model generation process flows in the direction of high- to low-fidelity variables; however, in aircraft conceptual design the amount of available geometry information flows in the direction of low- to high-fidelity variables.

### 2.3.2 Embedding Design Space Flexibility

Careful consideration of each parameter is needed when planning an intended geometry morphology for a model. For example, consider a wing that is designated the geometry level of a *part*, where its feature-tree may consist of a single loft *feature* between root and tip airfoil profiles. The design variable that denotes wing sweep thus becomes a *part* variable that is constant across the wing span. Consequently, a future design change that warrants a break in the wing, resulting in two wing segments with different wing sweep, will be impossible to represent within the current design space of the model. An entirely new model would be necessary with an expanded design space consisting of two sweep variables.

However, if the potential for non-constant wing sweep were considered at the start of model construction, the sweep could be categorized as a *feature* variable. The wing construction recipe could potentially be designed with two or more loft *features*, as follows: (1) A loft between the root and first break airfoil profiles; (2) a loft between the first and second break airfoil profiles (repeated for subsequent break locations); (3) a loft between the final break and tip airfoil profiles. Each loft would be driven by a separate sweep variable with this plan. As a result, the new model design space obtains an increase in degrees of freedom while able to represent the smaller design space (proven by setting each sweep *feature* variable to the same value).

This trivial example highlights the importance of carefully categorizing the design variables in the context of how the model may be used after construction. By contemplating if a design variable has the potential for spanning multiple geometry *features* during a portion of the design iteration process, an opportunity exists to embed more degrees of freedom in a model. Identifying all of the design variables for a complete solid model may not be a trivial task, though. This obviously depends on the complexity and level of detail a designer wishes

to embed into the model for design-space exploration in conceptual design. The initial cost of planning the model will pay off downstream in design with greater parameterization flexibility.

### 2.3.3 Parameterization Examples

Simple examples illustrate the nature of multifidelity parameterizations in models. For instance, the location of the wing and horizontal stabilizer along a fuselage can be defined as a *configuration* variable<sup>3</sup>. Its value causes a change in the aircraft *configuration* (aft tail versus canard *configuration*) within the monoplane *class*, as seen in Figures 2-9(a) and (b), without modifying the lifting-surface definitions. Only the trimming of the fuselage and lifting-surfaces are modified by the geometry kernel. However, the higher-fidelity parameterization of a spline *primitive*, which represents the lifting-surface airfoil, allows local surface control for shape optimization, as observed in Figures 2-9(c)–(e). Changing these values causes finer control of local surface characteristics in the resulting *feature* (or *part*) without changing the wing *assembly*, aircraft *configuration* or *class*.

There are various schemes for parameterizing an aircraft planform. The Program for Aircraft Synthesis Studies (PASS) [44] parameterizes a monoplane aircraft<sup>4</sup> using the input geometry parameters in Table 2.3. The fuselage contains additional parameters shown in Table 2.4. This code is usually designated a low-fidelity analysis due to its usage of empirical formulations, thus its geometry representation is limited to primarily planform parameters. An extension of the wing geometry parameters is made to create a new parameterization in Table 2.5, which allows defining various planform wing panels. In each of these examples the parameters correspond to the *configuration*, *assembly* and *part* levels of parameterization for a model.

If the PASS wing parameterization is chosen, for example, additional *part* and *feature* variables must be derived in terms of the given PASS parameters when planning a wing design. The parameters in the wing portion of Table 2.3 are also given classical notation in Table 2.6, and also define derived variables needed to construct a wing model. Other *primitive* airfoil-shape parameters are needed to define the wing outer mold line necessary

---

<sup>3</sup>Model regeneration robustness is improved if this parameter is defined as a percentage of fuselage length rather than using absolute model coordinates. The former ensures the wing or stabilizer will always reside within the fuselage length, whereas the latter has no such guarantee.

<sup>4</sup>See Appendix B for a description of PASS geometry parameters

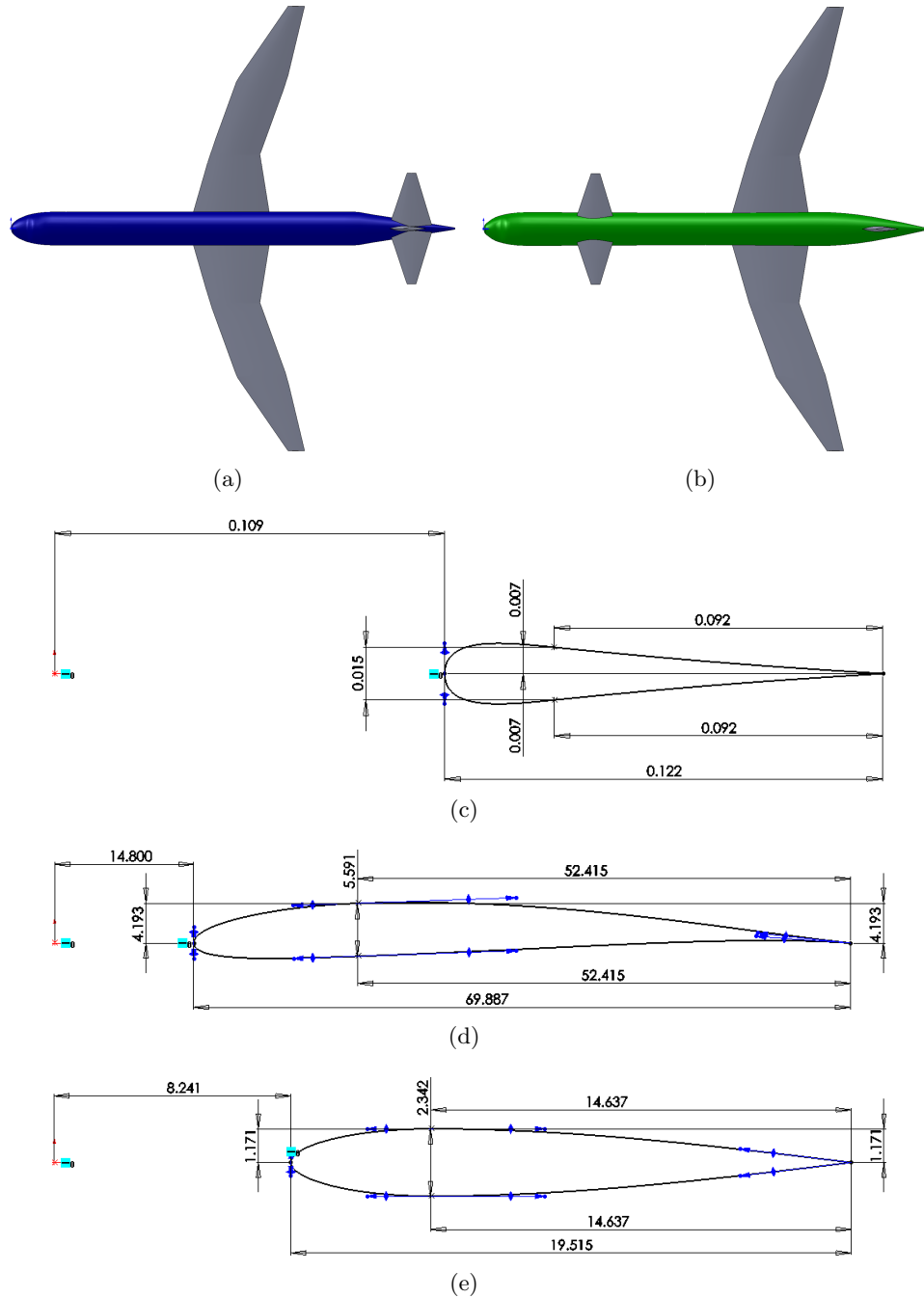


Figure 2-9: An example of a low-fidelity *configuration* change is observed between (a) and (b) by modifying a *configuration* variable relating the wing and stabilizer location along the fuselage. Furthermore, higher-fidelity control of surface topology is seen between (c), (d) and (e) by varying *primitive* variables associated with the airfoil shape.

	Configuration	Assembly	Part	Feature	Primitive
<b>Wing</b>	wingxposition wingheight	wingdihedral	taperw arw sref	chordextspan sweepw	tovercw x/ctransition lex tex <i>Airfoil*</i>
<b>Horizontal Tail</b>	<i>Tail Arm*</i> <i>Tail Height*</i>	dihedralh	taperh arh sh/sref	sweepv	toverch <i>Airfoil*</i>
<b>Vertical Tail</b>	ttail <i>Tail Arm*</i>	–	taperv arv sv/sref	sweepv	tovercv <i>Airfoil*</i>
<b>Fuselage</b>	–	–	fuseh/w	nosefineness tailfineness fwdspace aftspace	pilotlength windshieldht <i>Nose Shape*</i> <i>Wing-Body Junction*</i> <i>Fuselage Lobe Shape*</i>

Table 2.3: These PASS geometry parameters (see Appendix B) are categorized with respect to geometry level of fidelity for a monoplane aircraft *class*. Low-fidelity design variables on the left of the table are separated from high-fidelity design variables on the right. Italicized parameters with an asterisk are necessary to create a CAD model, yet are not explicit inputs as PASS parameters.



---

<b>Fuselage</b>	
FuseL	Fuselage length
FuseHW	Fuselage height-to-width ratio (constant cross-section)
FuseH	Fuselage height
NoseFineness	Nose fineness ratio (Nose length-to-fuselage width ratio)
Lpilot	Longitudinal distance to pilot station from origin
Lradome	Longitudinal distance to radome wall from origin
Lextrafwd	Longitudinal length of additional constant cross-section added after the nose (separate from constant cross-section region)
Lextraaft	Same as <b>Lextrafwd</b> , yet applied aft of the constant section of fuselage
Lwindow	Longitudinal length of pilot window as seen from side-view
TailFineness	Tailcone fineness ratio (tailcone length-to-fuselage width ratio)
Hradome	Vertical distance from the top of the radome to the global $z = 0$ line
Hwindow	Vertical length of the pilot window as seen from side-view
Hpilot	Vertical distance from the start of the pilot station from the $z = 0$ line
Hnosedroop	Vertical length of nose displacement below the centerline of the constant-section fuselage
Htailconeend	Vertical length of the tailcone end-face
Htailcone	Vertical distance from tailcone tip (bottom portion) to the bottom of the fuselage

---

Table 2.4: A parameterization for fuselage models utilized in the PASS aircraft design system.

---

<b>Lifting Surfaces</b>	
btotal	Total wing span
Stotal	Total wing area
b()	Vector of partial-span wing segments
S()	Vector of partial wing areas corresponding to each wing segment
Sweep()	Vector of sweep angles
Dihedral()	Vector of dihedral angles
N()	Vector containing the number of cross-sections per wing segment
Taper	Total taper ratio, similar to that for a trapezoidal reference area (tip chord to root chord ratio)
WingPositionX	Longitudinal distance of the root leading edge to the global origin
WingPositionZ	Vertical distance of the root leading edge to the $z = 0$ line
refline	Percentage of wing chord where an orienting reference line passes through an airfoil profile (e.g., the quarter-chord line) for defining sweep/dihedral

---

Table 2.5: A planform parameterization for lifting-surface models.

for high-fidelity analysis. Different parameterizations exist for airfoils and one must be selected *a priori* (e.g., leading edge radius, boat-tail angle, location of maximum thickness, or airfoil shape functions). For the tail surfaces, a PASS parameterization similar to that of the wing is used (excluding the  $c_{lex}$  and  $c_{tex}$  variables). Although the fuselage planform is described by parameters in Table 2.4, the nose contours, fuselage-lobe shape and wing-body junction surface are not specified in detail. These aspects of the fuselage are important to properly capture the total vehicle aerodynamic performance.

Design Variables	Derived Variables	Equation
taperw	$t_w$	Ref. Root Chord $c_{1_{ref}} = \left( \frac{1}{1 + t_w} \right) \sqrt{\frac{S_{ref}}{\mathcal{R}}}$
chordextspan	$b_{ext}$	Ref. Tip Chord $c_{n_{ref}} = \left( \frac{t_w}{1 + t_w} \right) \sqrt{\frac{S_{ref}}{\mathcal{R}}}$
arw	$\mathcal{R}$	Ref. Break Chord $c_{b_{ref}} = (1 - b_{ext}) c_{1_{ref}} + b_{ext} c_{n_{ref}}$
sweepw	$\Lambda$	Root Chord $c_{root} = c_{lex} + c_{1_{ref}} + c_{tex}$
sref	$S_{ref}$	
lex	$c_{lex}$	
tex	$c_{tex}$	

Table 2.6: These derived variables are a function of the PASS design variables and provide further information needed to create a wing CAD model.

A sensitivity analysis of the solid model shape (with respect to a performance metric) will indicate which initial surface shapes require greater attention for redesign later. In this light, a designer can initially design surfaces using their expertise for the sake of creating a model that captures the entire aircraft *configuration*. This may entail using a default wing airfoil, for example, that is compatible with the cruise Mach number for the vehicle. If multiple *configurations* utilize that same wing airfoil, then comparison can be isolated to *configuration*, *assembly* and *part* parameters. Otherwise, poor initial airfoil shapes will likely cause adverse effects on the overall performance that may negate the positive aspects of the *configuration* itself. Since shape optimization is not the main focus of the conceptual design phase, certainly later design phases can conduct more detailed shape optimization on *primitive* and *feature* variables to further improve performance.

## 2.4 Defining a Model Design Intent

Creation of CAD-generated models typically consists of combining geometry entities and operations in a particular manner until the model appears as the designer intended. This suffices for new users learning to use CAD software. However, as experience is gained it becomes very apparent to the user that a reasonable visual appearance of the model does not ensure a good design. The nature of any design process is to iterate the design parameters until an objective optimum is reached to within some degree of satisfaction. Therefore, the CAD-generated model must have the capacity to appropriately represent design variations, which means it maintains a designer's intended shape-change over a spectrum of driving parameter values.

This abstract notion of model construction is coarsely defined as *design intent* and may take on different meanings for different designers. The CAD user community views design intent as the way in which sketch dimensions/constraints, plus *feature* selection, combine to create a digital shape rendering of an abstract object. Others view design intent as the manner in which model entities move when driving dimensions are modified. For example, the SolidWorks CAD system user's-guide states: "Design intent is how your model behaves when dimensions are modified." A survey of design intent research provides another perspective: "The interpretation of design intent in general range from treating it as a historical record of analyses and decisions that led to the choice of the particular artifact or feature in question to treating it as the sum of the features (functional, geometric, constraint etc.) and the attributes of the features" [32]. Some researchers aim to understand design decisions and quantify design intent through algorithms that analyze *features* and geometry operations used on a model. Since those decisions are often influenced by unquantifiable metrics, such as aesthetics, prior experience in manufacturing, etc., the nature of design intent becomes very vague indeed.

Each definition of design intent is made in the context of how a CAD model will be used once it exists. By observing the views on design intent already mentioned, coupled with the training practices given to CAD users for learning model construction, it becomes clear that conventional design intent is primarily geared for constructing models to digitally represent a final, static abstract object, or family of objects that are geometrically similar, for manufacture. Figure 2-10 illustrates two examples from the SolidWorks 2010 user-guide

that support this view of design intent. Both examples are models wherein holes reference the dimensions of a base extrusion. Either the spacing of holes is driven by the extrusion dimensions, or a hole is constrained to not be driven by those dimensions. In 2-10(a) the orientation and spacing of drill holes is maintained even after the hinge length is increased. In 2-10(b) a family of *parts* with similar geometric *features* is possible by modifying various driving dimensions and leaving others unchanged. In these examples, the models are intended to represent manufacturable objects that must maintain a specific overall shape description without representing different geometry modes or other parameterizations. This is acceptable for the end-use of such CAD models and particular design processes. However, this does not apply to all possible design scenarios, especially aircraft design in early conceptual phases.

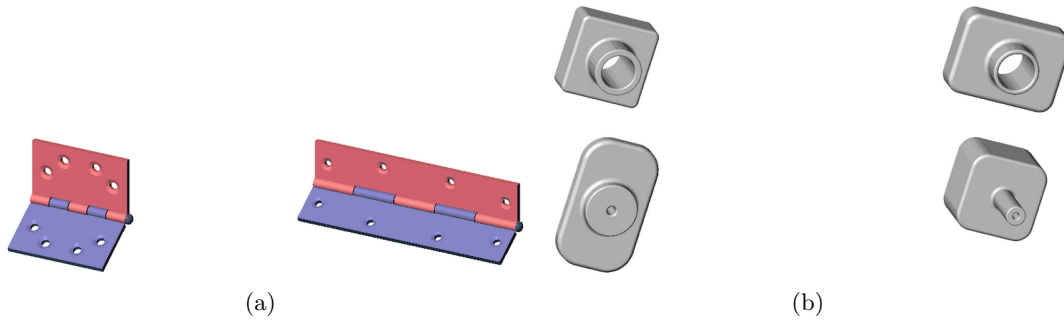


Figure 2-10: These SolidWorks 2010 user-guide images are two examples of employing design intent. In (a) the relative orientation and placement of geometry *features* (e.g., the drill holes) remain symmetric to the centerline and maintain a similar distribution pattern when the hinge length is changed. Figure (b) illustrates how various model dimensions (e.g., center-hole diameter, block width and height, hole height, etc.) can be changed and the *parts* maintain geometric similarity.

As shown in Table 2.7, the pros and cons of conventional design intent perspectives highlight the need for a new perspective that especially applies to aircraft conceptual design. Even though modern CAD systems are designed to implement current design intent perspectives, it is possible to still utilize such systems with a new design perspective in mind. In particular, the application of CAD in aircraft conceptual design brings the need for greater flexibility in defining the aircraft model via multiple parameterizations. Shape design must be supported, hence the selection of geometry *features* require special consideration for malleability. Finally, the desire to automate the design optimization process entails a need for greater regeneration robustness, or a lower likelihood of violating geometry construction

rules within a geometry kernel as a result of conflicting dimensions/constraints. With these needs in mind, a CAD model used in aircraft conceptual design should not be *intended* to represent a final design for manufacture. Instead, the model *intent* should be multifaceted, possibly driven by various parameterizations, and capable of handling planform or shape-design changes. This perspective invites designers to consider that

$$\text{Design Intent} = f(\text{a posteriori model application}),$$

meaning that some post-construction usage of a CAD model drives the initial design intent embedded in the model. Usage may refer to “representation of static geometry,” or “amorphous shape design in an automated design framework containing specific analysis tools.” In practice, designers need not foresee all possible applications of the CAD model downstream in their design process because the new perspective allows for model adaptation when additional geometry and analysis information is provided. A model is sufficient even if it only properly functions within the design circumstances it is exposed to.

<b>Conventional Design Intent Perspective</b>	
<b>Pros</b>	<b>Cons</b>
<ul style="list-style-type: none"> <li>— Represent objects with a fixed parameterization</li> <li>— Objects remain geometrically similar when changing driving dimensions</li> </ul>	<ul style="list-style-type: none"> <li>— Cannot represent the same object using multiple parameterizations</li> <li>— Shape design is limited to the embedded parameterization and Feature selection</li> <li>— Greater combinations of dimensions/constraints may lead to model regeneration problems</li> </ul>

Table 2.7: The application of conventional design intent ideas is appropriate for particular CAD-based model applications; however, all design scenarios do not benefit as well from this approach.

A more formal definition of design intent is proposed here that is suited to meet the needs inherent to aircraft conceptual design. In order to fully understand this definition, certain principles from the shape-design optimization community are employed in this context of CAD models. Using the illustration in Figure 2-11 as a guide, the concept of *design motion* for a manifold, closed model  $\mathcal{M} = \mathcal{M}(\mathbf{x}_j)$  is defined here to correspond with the movement of

the model geometry surfaces, and their intersections, when the  $j^{\text{th}}$  set of driving parameters  $\mathbf{x}_j$  (each corresponding to a different chosen parameterization) are modified and a new model instance is generated. The *design space*,  $\mathbf{x}_j = \{P_i\}$ , is the set of parameters  $P_i \in \mathbb{R}$  used to drive the model geometry. A *design time* is also considered as  $P_i \in [P_{i,\text{initial}}, P_{i,\text{final}}]$ , which is the value of a parameter between an initial value  $P_{i,\text{initial}}$  and a final value  $P_{i,\text{final}}$ . Since the model  $\mathcal{M}(\mathbf{x}_j)$  is a function of the design space, the model geometry and BRep topology are instantiated at a *design point* by fixed parameter values. This leads to the notion of a *design trajectory*,  $\{\mathcal{M}(\mathbf{x}_{j,\text{initial}}), \mathcal{M}(\mathbf{x}_{j,\text{final}})\}$ , which is the range of design points representing model geometry found between an initial and final design time. Finally, the *design velocity*,  $d\mathcal{M}(\mathbf{x}_j)/d\mathbf{x}_j$ , is the design point rate-of-change in Euclidean space of an instance of model geometry with respect to changing parameters (i.e., design sensitivity)<sup>5</sup>.

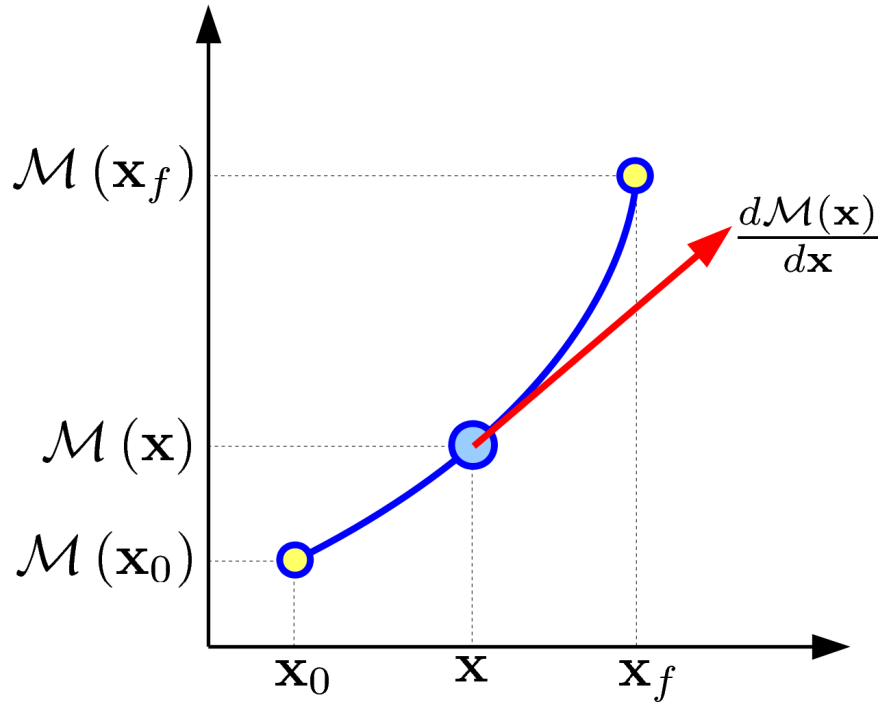


Figure 2-11: The concepts of design motion, design space, design point, design trajectory and design velocity are illustrated as a simple graph representation for a model  $\mathcal{M}(\mathbf{x})$ , driven by a single design space  $\mathbf{x}$ , to give greater intuition when describing a new design intent.

Therefore, the manner in which a model morphs is best quantified by the design motion of its geometry and BRep. That design motion, whether intended or not in the context

<sup>5</sup>The shape-design community sometimes defines design velocity as the normal component of the geometry gradient (as projected onto the local surface normal). Throughout this dissertation, however, a broader definition is used where design velocity refers to the complete gradient.

of other qualitative considerations, is the embedded design intent of the model. With these definitions for design motion and the guiding principles behind multifidelity geometry parameterizations, the proposed definition for design intent can be stated as:

**Definition 2.4.1.** Given a model  $\mathcal{M} = \mathcal{M}(\mathbf{x}_j)$  with an embedded  $j^{\text{th}}$  design space of driving model parameters  $\mathbf{x}_j = \{P_i\}$ , where  $P_i \in \mathbb{R}$  varies between a span of design time  $P_i \in [P_{i,\text{initial}}, P_{i,\text{final}}]$ , the design intent of a CAD-generated model is the set of design trajectories  $\{\mathcal{M}(\mathbf{x}_{j,\text{initial}}), \mathcal{M}(\mathbf{x}_{j,\text{final}})\}$  over which  $\mathcal{M}(\mathbf{x}_j)$  successfully regenerates and exhibits design motion.

It is clear from Definition 2.4.1 that the proposed design intent emphasizes the geometry response to driving parameters as its main focus. This formal definition of design intent provides a lens through which a designer can evaluate the a posteriori usage of their CAD model and its construction methodology, especially if automated design space exploration is expected. Otherwise, an ambiguous definition of design intent provides a very limited means to evaluate the effectiveness of a CAD model construction methodology for subsequent application. An example design trajectory for the cut-cone model in Figure 2-12 is shown in Figure 2-13 to identify its embedded design intent. The parameter set is  $\{\theta, h, d\}$  for this model geometry, which set the cone half-angle, the cone height and the distance of a vertical cut-plane from the cone axis, respectively. The design points are instantiated across the design space  $\theta = \{25^\circ, 85^\circ\}$  at  $1^\circ$  increments. Only the BRep edges and nodes are depicted for  $\theta = \{25^\circ, 30^\circ, 55^\circ, 70^\circ\}$ , but only BRep nodes are shown for all other design points. The resulting design trajectory for each BRep entity is intuitive for variations in  $\theta$  because the cone is “flattened” with increasing  $\theta$ . The cone is still cut by the vertical plane at high  $\theta$  values because it has infinite extent in all directions. Though when  $\theta = \arctan(d) = \theta^*$  (where  $\theta^* \approx 22^\circ$  for  $d = 0.4$ ), the cone and vertical plane create a difficult situation for the geometry kernel because both *features* are coincident solely at a single point. The cut-operation cannot be accomplished, thus the geometry kernel circumvents that scenario by modifying the BRep topology accordingly, as described in Section 2.2.2, via a node bifurcation. If  $\theta \rightarrow \theta^*$  from above, then the three nodes  $\mathcal{N}_3, \mathcal{N}_4$  and  $\mathcal{N}_5$  (see Figure 3-1) are collapsed into  $\mathcal{N}'$  and edges  $\mathcal{E}_3, \mathcal{E}_4$  and  $\mathcal{E}_7$  are removed. If  $\theta \rightarrow \theta^*$  from below, then the initial node  $\mathcal{N}'$  bifurcates into  $\mathcal{N}_3, \mathcal{N}_4$  and  $\mathcal{N}_5$  and edges  $\mathcal{E}_3, \mathcal{E}_4$  and  $\mathcal{E}_7$  are created as a result of applying the cut-operation. By observing such design motion, the potential for

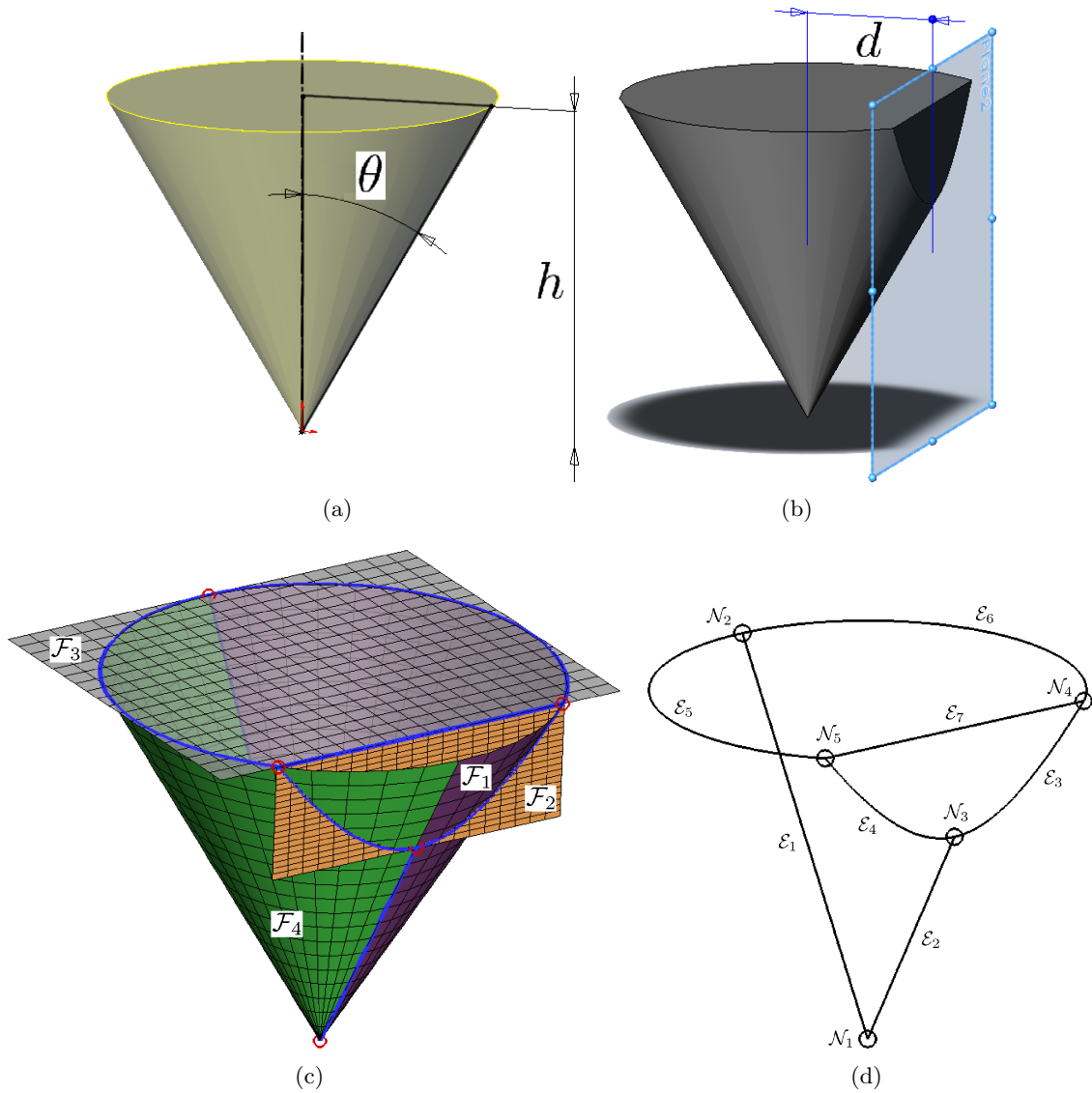


Figure 2-12: (a) Sketch entities are piecewise continuous and driven by parameters, such as the cone angle  $\theta$  and height  $h$  in this case. The CAD geometry kernel processes the sketch entities and generates associated surfaces for the cone feature. (b) Adding a cut-operator with a planar face results in a new model instance and an additional geometry parameter,  $d$ . (c)-(d) The final model topology for this cone-plane feature shows edges  $\{\mathcal{E}_1, \dots, \mathcal{E}_7\}$  (lines) and nodes  $\{\mathcal{N}_1, \dots, \mathcal{N}_5\}$  (circles) bounding faces  $\{\mathcal{F}_1, \dots, \mathcal{F}_4\}$  used to create the feature.



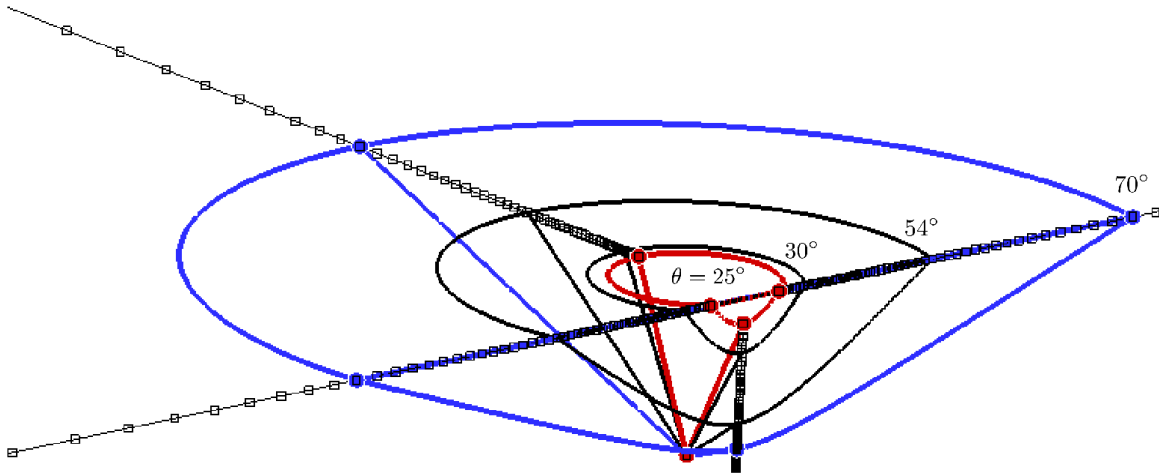


Figure 2-13: The design trajectory of the cut-cone model in Figure 2-12 is shown for a parameter range  $\theta = \{25^\circ, 85^\circ\}$  at  $1^\circ$  increments. Topology is unchanging on this design trajectory. BRep edges are shown for different  $\theta$  values, whereas only the BRep nodes (boxes) are shown at each design point.

BRep topology changes can be found along a design trajectory.

In the context of the parameter  $\theta$ , it is clear that the design intent for the cut-cone model consists of two piecewise continuous design trajectories. One design trajectory exists between  $0 < \theta < \theta^*$  and the other between  $\theta^* < \theta < \pi/2$ . There is no unique design velocity at the end-points of each span of design time because the cone geometry is undefined at  $\theta = 0, \pi/2$  as well. The complete design intent for the cut-cone model is obtained by conducting a similar design motion analysis for parameters  $d$  and  $h$  and identifying the feasible design trajectories that result.

## 2.5 Generating Multifidelity and Multidisciplinary Model Geometry

Since conventional approaches in constructing model geometry are often tailored for purposes other than automated design optimization, a new perspective is necessary to generate models that are suitable for such use. The following discussion explains a proposed approach with various example cases.

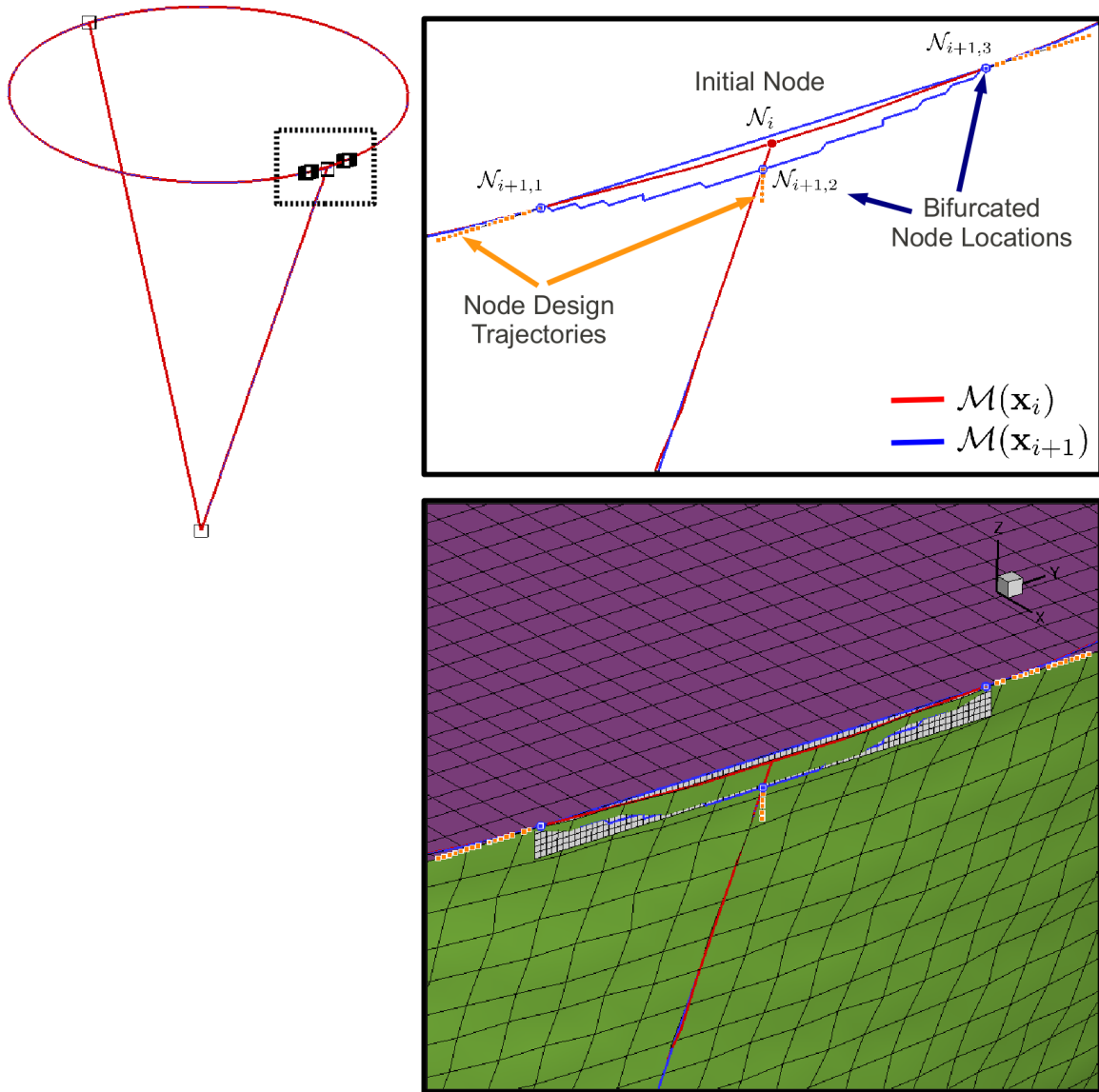


Figure 2-14: A node bifurcation is shown as the model passes through  $\theta^* \approx 22^\circ$ . The initial node  $\mathcal{N}_i$  bifurcates into  $\mathcal{N}_{i+1,1}$ ,  $\mathcal{N}_{i+1,2}$  and  $\mathcal{N}_{i+1,3}$ ; new edges are also created. The lower image highlights the “small” cut-plane face that exists just before the bifurcation occurs, which implies that the tolerance sphere for the initial node is substantial.

### 2.5.1 Model Construction Methodology

Once an understanding of CAD model *features* (see Appendix A.4), parameterization levels, design intent and construction approaches are obtained, a general model construction methodology can be created that is unique for aircraft models implemented in conceptual design settings. This is accomplished by *combining* particular aspects of CAD-based and Direct Construction approaches (see section 1.2). The useful elements of both construction approaches are summarized here.

#### CAD-based Elements

For simplicity, utilize classical *primitives* to define cross-section characteristics of non-lift-generating bodies (an exception may be a lifting-fuselage body, though) and spline *primitives* for the cross-sections of lift-generating surfaces. Natural geometry constraints (stemming from geometric similarity) accompany classical *primitives* if a specific body shape is presumed to remain consistent throughout design. Otherwise, it is possible to use spline *primitives* for cross-sections when the “best” final surface is unknown for the application. These sketches are constrained/dimensioned based on the intended design motion rigidity: less rigidity provides greater shape control than a fully constrained and dimensioned *primitive*. In addition, utilize loft *features* for a greater range of design motion. Use a global datum reference layout with minimal *primitive-to-primitive* or *primitive-to-feature* referencing.

#### Direct Construction Elements

Choose a parameterization that will drive the outer mold line of the aircraft. For example, a fuselage body profile (planform view, or side view) can be parameterized with conic sections, linear segments or splines (e.g., see Figure A-11). This parameterization will *not* be explicitly defined in the model sketches. Instead, it will serve to orient and distribute the cross-section *primitives* in  $\mathbb{R}^3$  space with respect to the global origin. In the case of a fuselage body, for example, the coordinates of quadrant points on an elliptical cross-section *primitive* will be driven by the outer mold line parameterization and remain independent of any other *feature* in the model (except the global coordinate system). The same will hold for lofted wing cross-sections (e.g., see Figure A-9), which can be driven by multiple airfoil parameterizations that position spline points in a sketch. This approach allows the designer

the option to *change* the outer mold line parameterization and afterwards map it to driving dimensions in the *same* 3D model. This is generally possible if a mapping between the selected parameterization and the model dimensioning scheme exists<sup>6</sup>. Through this approach differences in parameterization can represent differences in the concept design spaces with the same underlying CAD model. Without this approach, added reference datums, dimensions and constraints will be needed to restrict the design motion of *primitives*.

## A Combined Approach

A combination of the CAD-based and Direct Construction methods results in a scheme for model building that is *malleable*, *robust*<sup>7</sup> and *flexible* for a general design trajectory. Malleability stems from control of loft cross-sections, which permits a form of local surface design. Regeneration robustness is embedded by reducing the coupling of datum references and geometry to a minimum, the limit of which is sole utilization of the global datum planes, origin and potentially a reference curve for certain *features* in the model. Minimal dimensioning between geometry *primitives* and other *features* also reduces the potential for regeneration hysteresis sometimes seen with CAD systems [79]. Finally, flexibility is obtained by having any parameterization drive the distribution and orientation of independent *primitives* in the model.

Automated model generation greatly simplifies the manual and tedious process of creating complicated models by hand with a CAD system GUI. Although an initial development cost is required, the automated process reaps large time-savings in the long-term when multiple models are desired. A scripted procedure also simplifies the global distribution, orientation and attributes of *primitive* sketches via parameter distributions (e.g., chord, airfoil thickness, and radii distributions). Models with greater shape control are possible by including more *primitive* sketches along an outer mold line profile, or a minimum number of sketches is possible to define the 3D geometry in a coarse manner. Such model characteristics are easily adapted via automated scripts.

Lastly, multidisciplinary 3D modeling becomes feasible by including additional *features*

---

<sup>6</sup>If the CAD system enables scripting a parameterization definition within its master-model, then the parameterization remains “internal” to the model geometry. However, if the CAD system does not have this capability, an “external” parameterization can drive the model when an interface to the geometry kernel exists. This approach is less ideal since the model geometry depends on additional software external to its definition.

<sup>7</sup>Throughout this thesis the usage of “robust” implies regeneration robustness, not the more expansive topic of design robustness that is beyond the scope of this document.

or *parts* to the model that represent other disciplines. For example, the methodology explained above is applicable for creating internal structures. A distribution of structural attributes, such as those related to a wing spar, can be parameterized within the context of the wing outer mold line that already exists. This results in an internal structure that remains within the outer mold line envelope. In this manner the attributes of malleability, robustness and flexibility are given to these structural components because they are created in the same fashion as the outer mold line. It is also possible to develop a library of structural components that are driven by the outer mold line and also a separate structural parameterization. The resulting attributes for the *assembly* of structural components and outer mold line surfaces are the same as those of the individual components, meaning the assembly inherits malleability, robustness and flexibility from its parts.

Table 2.8 summarizes the benefits of the model construction methodology and design intent presented thus far. Using these principles, a CAD model is more enabled for automated design optimization than through conventional approaches.

<b>Model Construction Approaches</b>	
<b>Conventional</b>	<b>Proposed</b>
— Fixed parameterization only reflects needed dimensioning to generate desired shape	— The same shape can be driven by multiple parameterizations
— Geometry information in the model is not coupled with any analysis geometry requirements	— Analysis geometry requirements are embedded in the CAD model
— Datum referencing, dimensioning/constraints are sufficient to restrict design motion (thus, greater design rigidity) and preserve geometric similarity	— Robustness is achieved with minimal datum referencing, dimensioning/constraints for greater design motion
— Single geometry mode is representative of final design shape	— Malleability for shape design is possible and various geometry modes can be represented
— Primarily manual construction process via a GUI	— Minimal surface quality issues (sliver edges/faces) via Feature-selection — Primarily automated geometry generation

Table 2.8: The conventional and proposed model construction methodologies yield distinct model attributes that better serve different post-application uses.

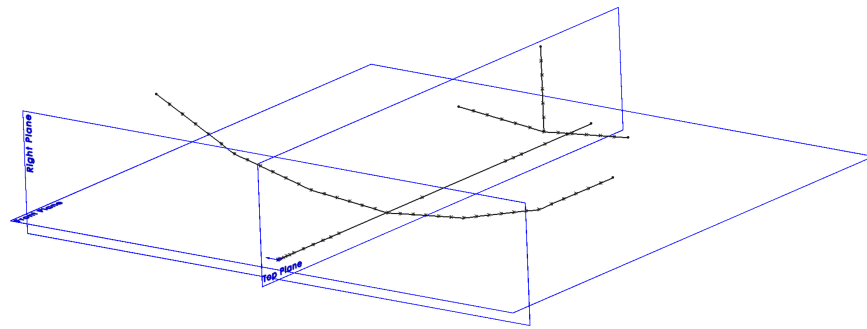
## 2.5.2 Single Discipline Examples

Three examples are given to demonstrate the model construction methodology summarized in Section 2.5. First the reference datum layout is explained, followed by the manner in which sketch *primitives* are created. Although any parametric CAD system would suffice, the SolidWorks CAD system [19] was selected for model generation and Visual Basic code was developed to automate the generation process through the SolidWorks API. Without the automation, though, the same 3D models could be created manually using the GUI at a greater time expense.

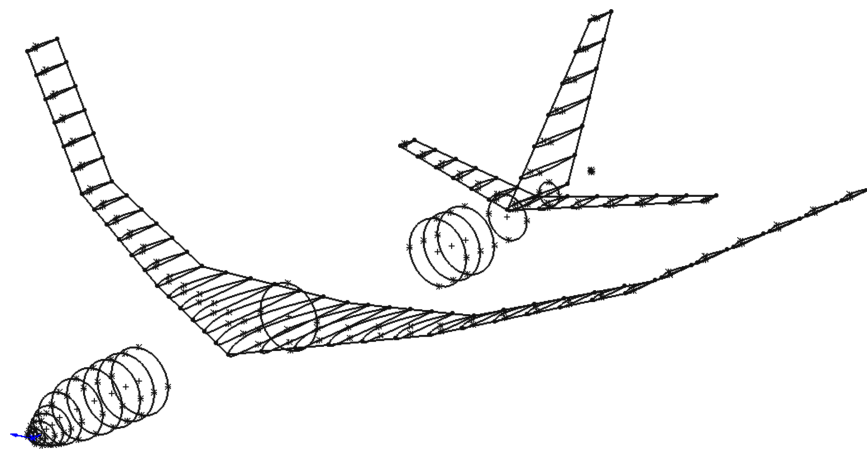
### Tube-and-Wing Model

The reference datum layout establishes how rigid (in a design motion sense) and robust the model construction is in a design space. Datum planes and lines are often arbitrarily established in order to create a model that fits a final shape. This can unintentionally lead to a more rigid design motion for the model. To maximize robustness and model applicability throughout diverse design trajectories, a minimal datum reference layout is best. This methodology also has shorter regeneration times and a more compact feature-tree sequence. Upon initialization of the CAD system, a global coordinate system is created with associated global datum planes. Sketch planes with greater orientation flexibility are defined by constraining their origin to the support points of a 3D datum spline *primitive* (which is not constrained to a single sketch plane) and constraining their normal vector to the datum curve tangent. The distribution of support points sets the distribution of sketch planes, as seen in Figure 2-15(a). The 3D datum spline thus becomes the only parent *feature* to the sketch plane. As the tangent vector of the datum spline changes at a support point, the constrained sketch plane will reorient accordingly.

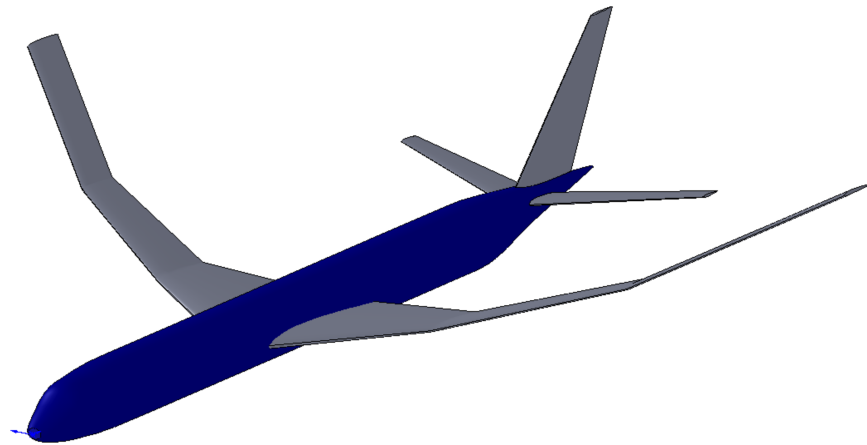
Once 3D datum splines are created for the fuselage, wing, and empennage, with corresponding child sketch planes, geometry *primitives* are added to the sketch planes, as observed in Figure 2-15(b). Since each sketch plane has its origin defined at the reference spline point, a globally-defined parameterization for a cross-section *primitive* must be mapped to the local sketch coordinate system. Once all sketch planes are populated with *primitives*, a single loft is generated through the appropriate sketches for each component, as seen in Figure 2-15(c). Guide curves may be necessary to constrain the loft, as in the



(a)



(b)



(c)

Figure 2-15: The automated model generation process is illustrated here: (a) depiction of global datum planes and 3D datum splines; (b) addition of *primitive* sketches as children of the datum splines; (c) final loft across the *primitive* sketches of each *configuration* component.

case of explicitly defining the leading/trailing edge of a lifting-surface. A single loft is advantageous over multiple lofts across a component, as discussed in Section A.1.

The resulting model contains independent components defined by independent *primitive* sketches. Designers must inspect the loft quality in order to determine what number of cross-section *primitives* is sufficient for a component. Loft overshoots/undershoots may occur in regions where an abrupt change in surface derivatives occur, requiring greater refinement with added sketches. Conversely, regions with constant surface derivatives may warrant fewer sketches.

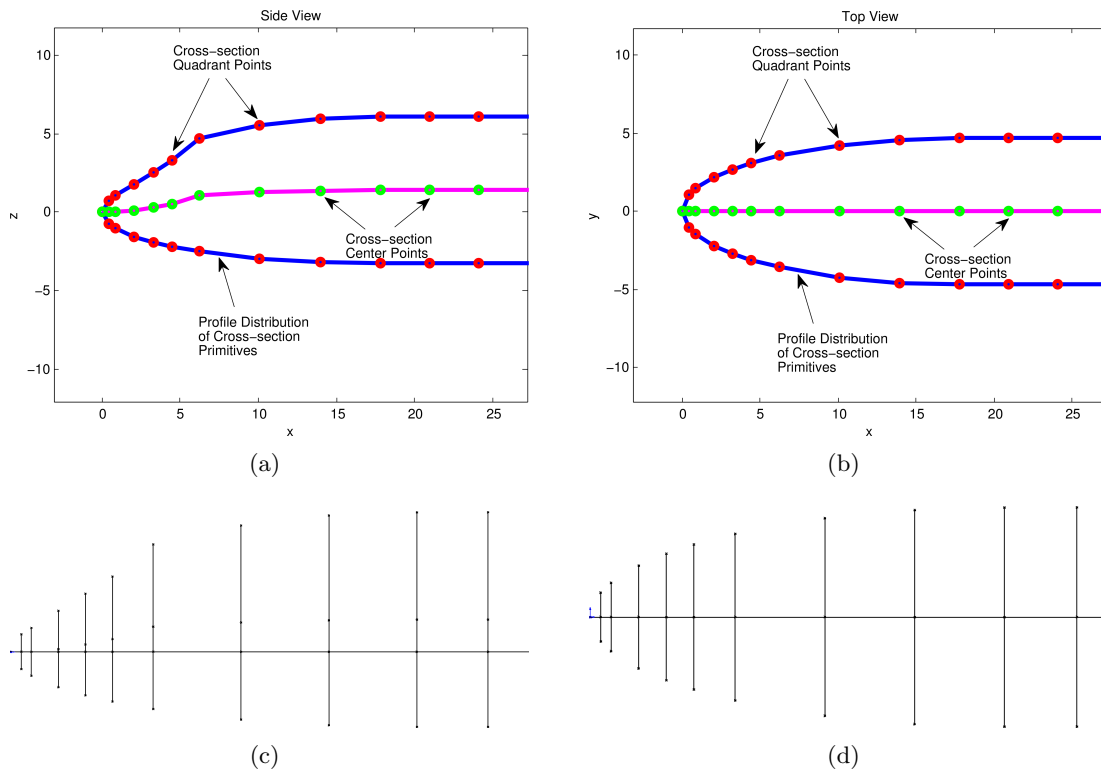


Figure 2-16: A parameterization consisting of piecewise-continuous line segments and elliptical arcs for the fuselage nose profile was defined outside of the CAD system. Classical *primitives* were used (elliptical arcs) to define the cross-sections. This parameterization determined the quadrant and center point distributions, as seen in (a) side view, and (b) top view. The distribution of elliptic *primitives* was driven in the CAD system by this parameterization, as shown in (c) side view and (d) top view.

The fuselage and wing profiles were modeled as piecewise-linear segments patched with conic sections, as illustrated in Figures 2-16(a)–(d) and 2-17(a)–(f), respectively. These profiles were also driven by the parameters in Tables 2.3 through 2.5. The resulting loft surfaces could be morphed in numerous ways by manipulating the datum reference curve



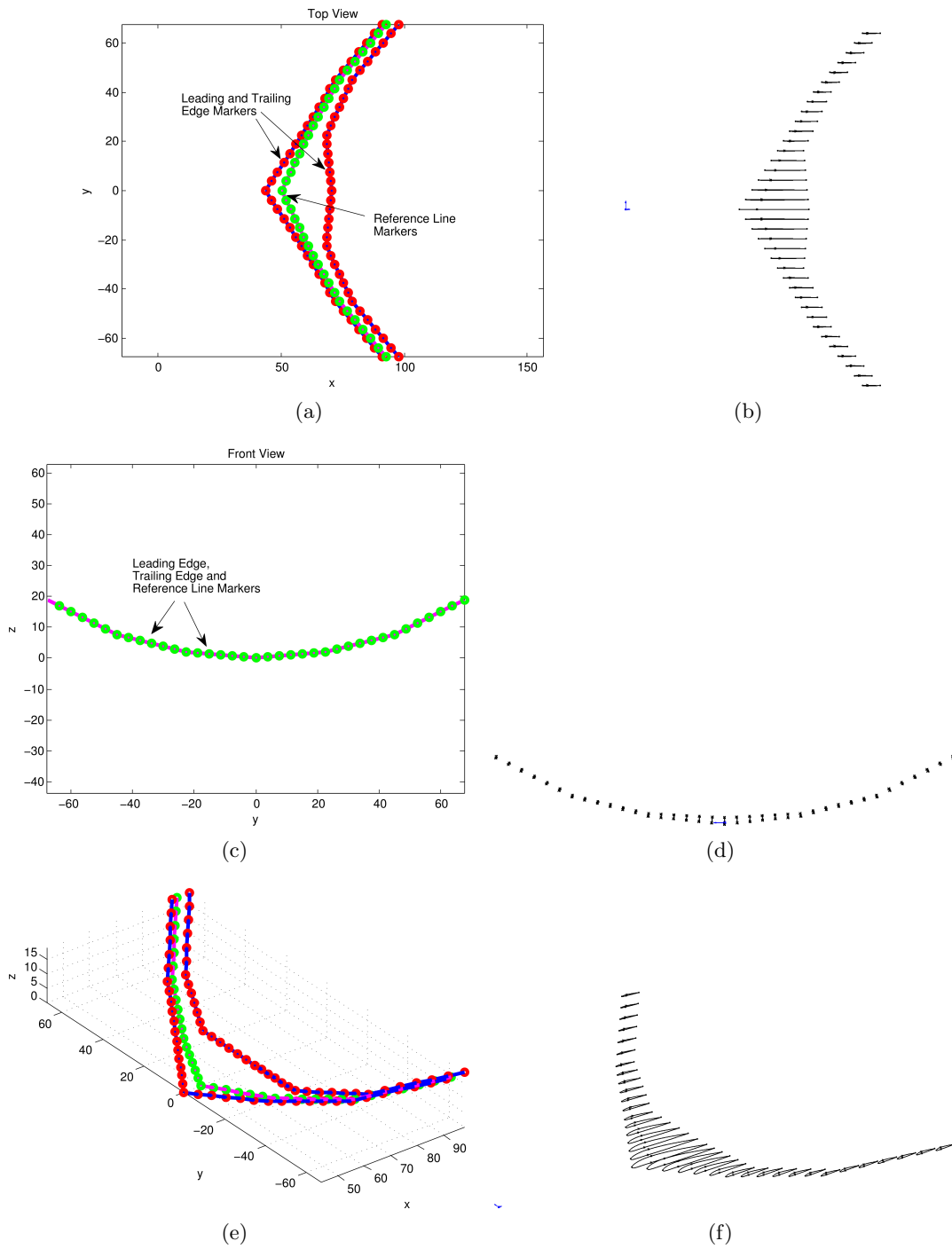


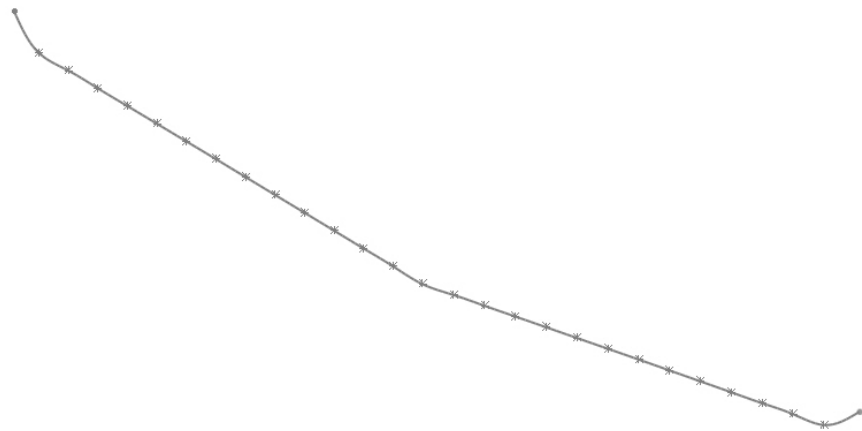
Figure 2-17: A parameterization consisting of piecewise-continuous line segments was defined for the wing leading/trailing edges and quarter-chord. Reference markers for the airfoil are shown in (a) top view, (c) front view, and (e) isometric view. The distribution of spline *primitives* was driven in the CAD system by this parameterization, as seen in (b) top view, (d) front view, and (f) isometric view.

and/or changing the distribution of sizing parameters among their cross-section sketches. These parameterization choices permit the shape control possible in free-form design methods. When assuming that components will maintain a particular profile shape, though, classical *primitives* available in the CAD system provide a natural geometry shape constraint for body cross-sections.

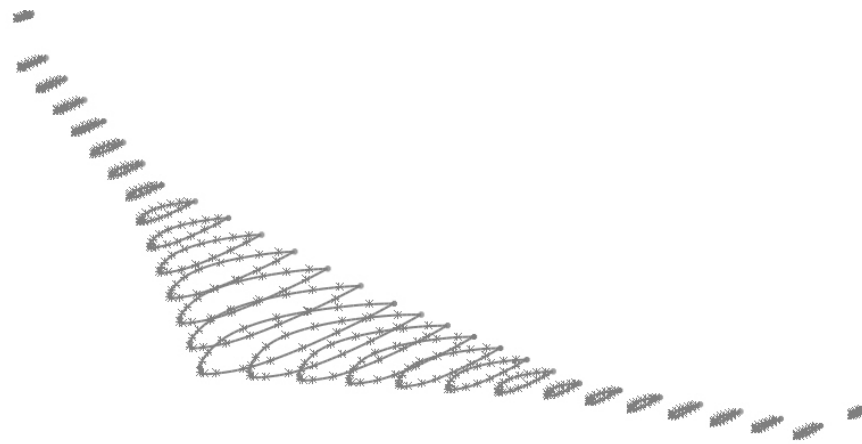
Depending on the extent of shape design needed, an additional compromise is made in the decision of which *primitives* to use when creating cross-sections. Spline *primitives* can be defined with numerous degrees of freedom (either via support or control points) to better represent an intended curve shape. However, enriching the design space in this manner can increase the likelihood of reaching additional local extrema in design optimization, especially if aerodynamics analysis is involved. Fine control of surfaces can also lead to undesirable variations in curvature if support points are changed one at a time. On the other hand, foregoing fine surface control allows for classical *primitives* that represent certain shapes with fewer design variables. These avoid unwanted curvature variations since the CAD geometry kernel constrains these *primitives* with geometric similarity (e.g., elliptical *primitives* maintain a particular curvature distribution profile regardless of perturbations in the quadrant point locations). A designer must decide which components benefit from classical *primitives* (such as a constant-radii fuselage section) or splines (better suited for fuselage-nose *features* and airfoils) while considering problem tractability for optimization.

### **Flying-wing Model**

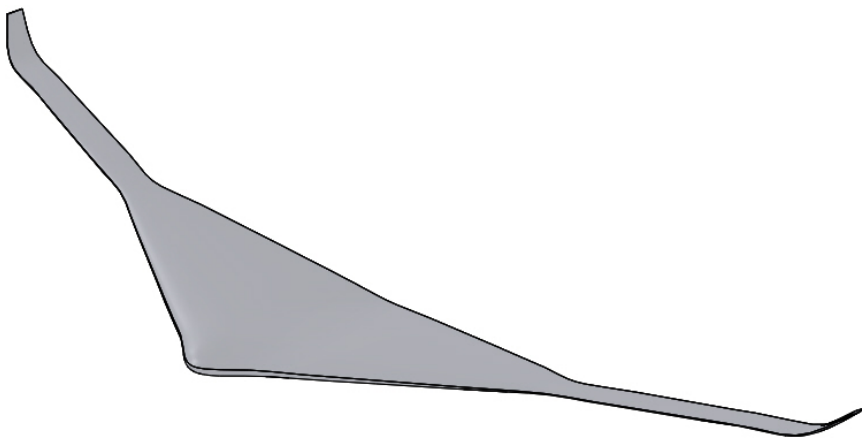
The main steps in constructing a generic flying-wing *configuration* are shown in Figure 2-18. Although seemingly similar to the wing construction for the tube-and-wing model, this model exemplifies a different wing parameterization. The 3D datum spline in Figure 2-18(a) serves as the parent to all cross-section sketches defined normal to it. The end-points of the datum spline are modified to provide sufficient dihedral for modeling winglets. The cross-sections in Figure 2-18(b) are oriented according to the datum spline and distributed using the planform parameterization in Table 2.6. In contrast to the main wing in the tube-and-wing model, the airfoil cross-sections have a twist distribution linearly ranging from  $+1^\circ$  at the root,  $0^\circ$  at the span-break and  $-3^\circ$  at the tips. Furthermore, the airfoil stack consists of three standard airfoil profiles, as shown in side-view in Figure 2-19. The inboard  $1/3$  semi-span contains a NACA 23015 shape, followed by the NACA 3317 airfoil in



(a)



(b)



(c)

Figure 2-18: The automated model generation process is illustrated here for a flying-wing configuration: (a) depiction of 3D datum splines; (b) addition of *primitive* sketches as children of the datum splines; (c) final loft across the *primitive* sketches.

the next 1/3 semi-span and the NACA 0012 in the final 1/3 outboard panel. The option to modify the inboard airfoils via scaling is also shown in a three-view portrayal in Figure 2-20, where the airfoils along the inboard 1/3 span were lengthened to extend the center-body volume aft.

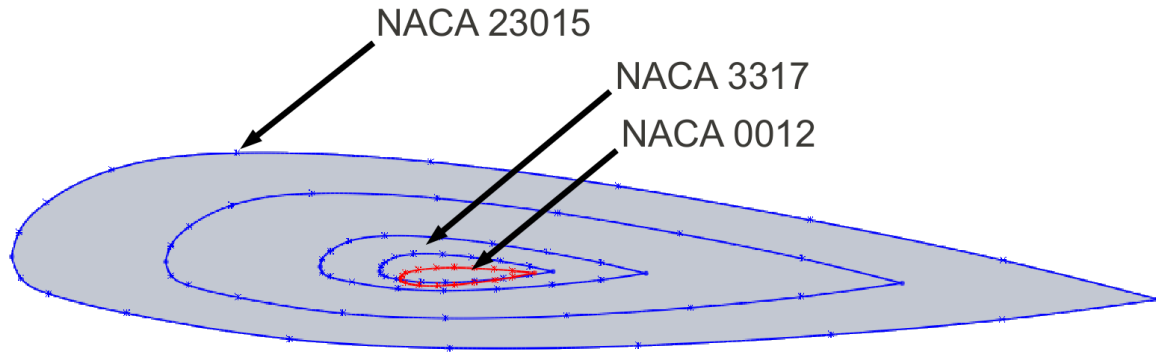


Figure 2-19: The airfoil stack for the generic flying-wing model is shown in this side-view, along with the span-wise twist distribution ranging from  $+1^\circ$  at the root to  $-3^\circ$  at the tip.

### Lifting-body Model

The construction of a generic lifting-body model is shown in Figure 2-21. Compared to the tube-and-wing model, this model utilizes a very different fuselage parameterization and empennage. Instead of using a cross-section distribution based on linear segments patched with conics, the fuselage side-view profile is driven by a NACA 3317 airfoil section, as seen in the three-view of Figure 2-22. Fuselage cross-sections are elliptical *primitives* distributed with higher density near the fuselage nose to capture the higher surface curvature expected there. The planform (top-view) extent of the cross-sections are based on an elliptical nose section followed by a straight linear section. The horizontal tail component is adjusted for a “T-tail” to avoid flying in the fuselage wake at high angles of attack. This is accomplished by simply relocating its datum spline to the desired position, as shown in Figure 2-21(a). In this case all lifting-surfaces have NACA 0012 airfoil sections with zero twist.

### 2.5.3 Multiple Discipline Examples

The traditional CAD concept of top-down design applies to generating structural components and layouts. An outer mold line acts as a parent component that constrains the subsequent child components added internally or externally. These constraints are expressed

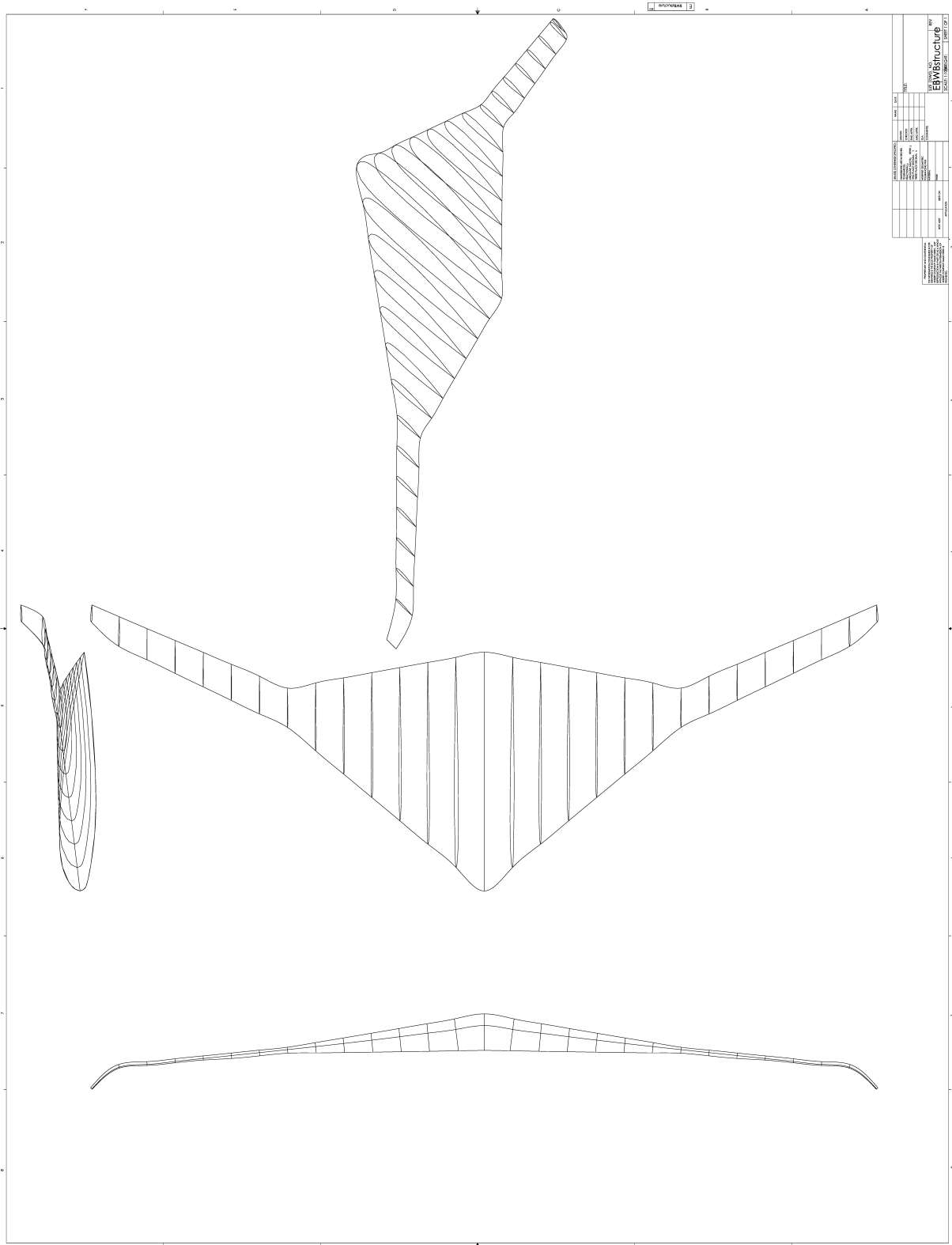
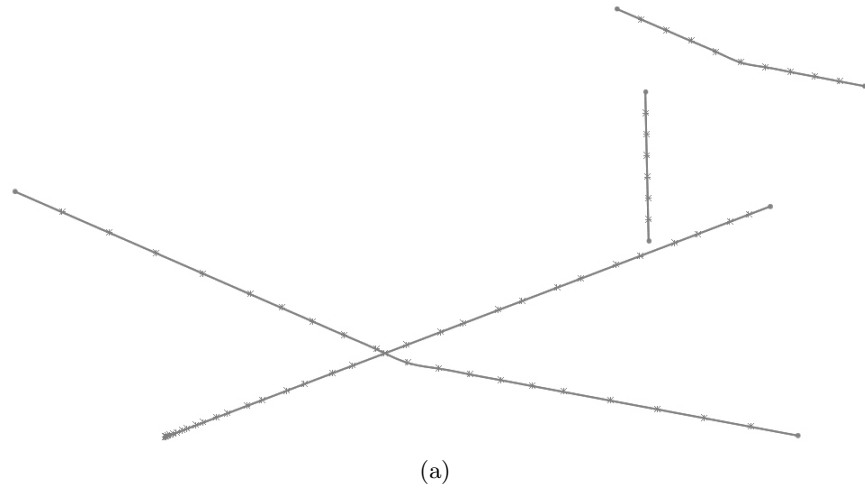
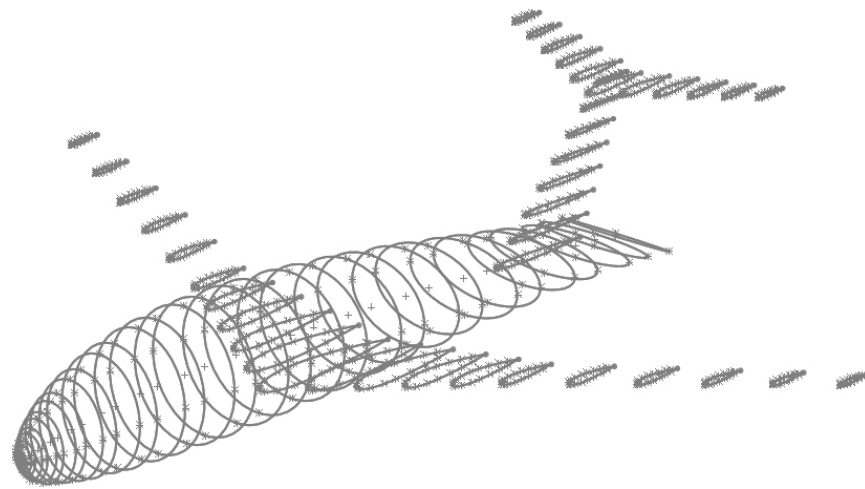


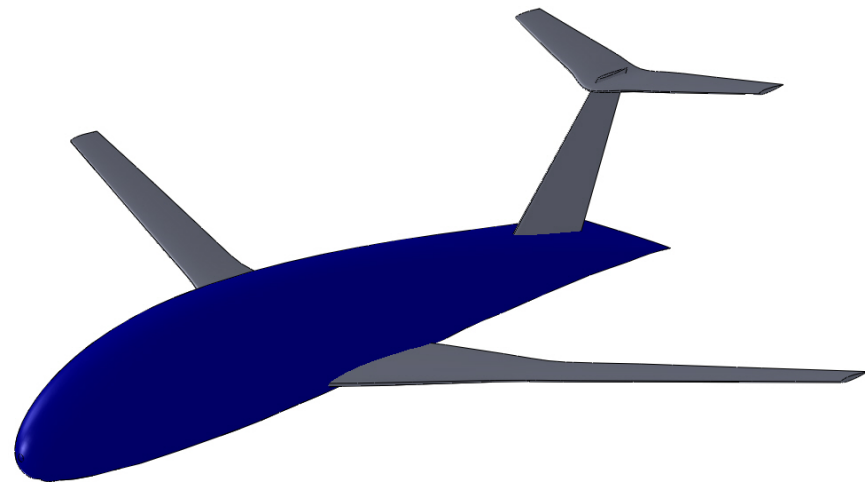
Figure 2-20: The airfoil stack for the generic flying-wing model is modified along the inboard 1/3 span by scaling the airfoils to create a greater center-body volume.



(a)



(b)



(c)

Figure 2-21: The automated model generation process is illustrated here for a lifting-body configuration: (a) depiction of 3D datum splines; (b) addition of *primitive* sketches as children of the datum splines; (c) final loft across the *primitive* sketches for each component.

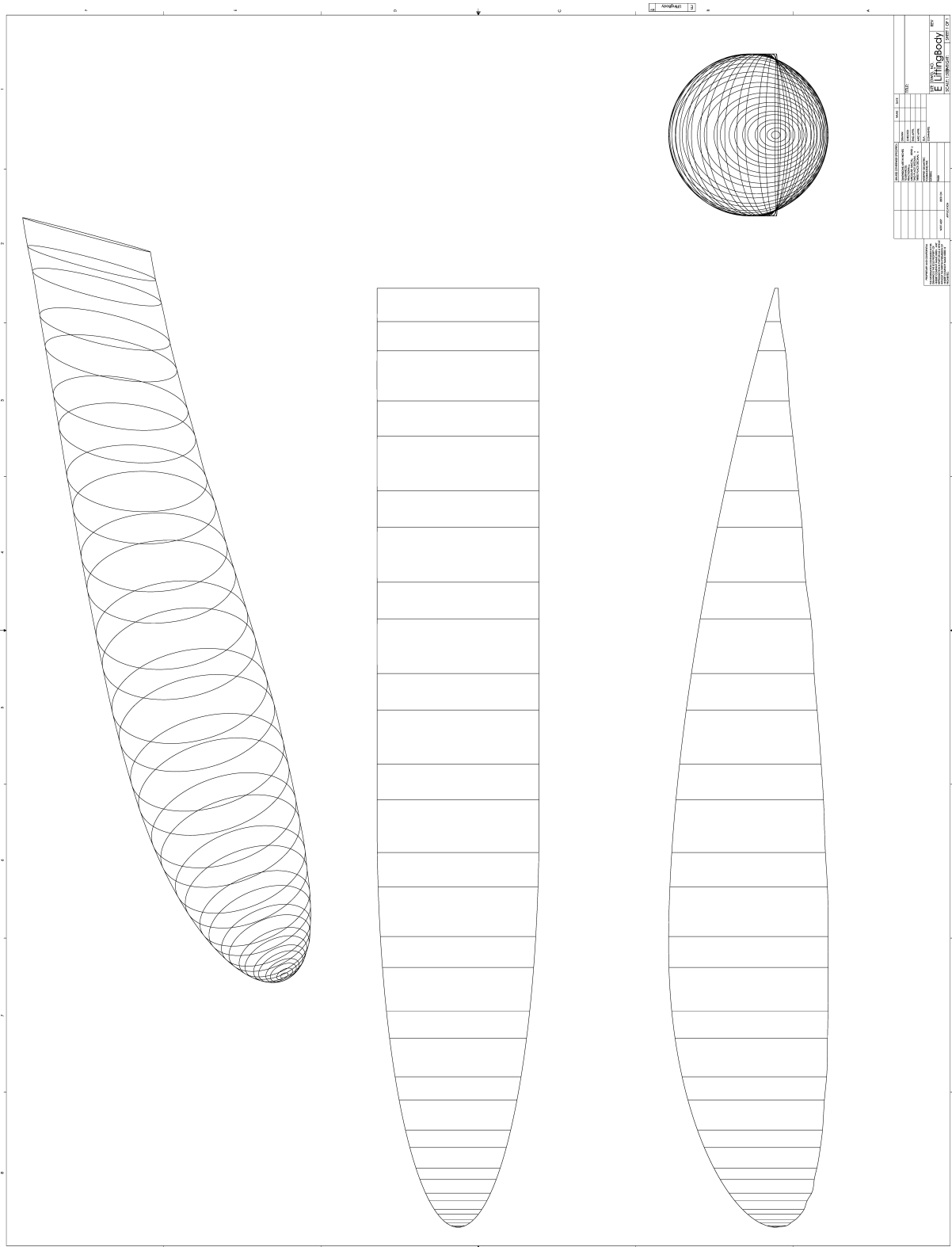


Figure 2-22: The lifting-body fuselage is based on a NACA 3317 side-view profile and patched elliptical-linear sections in planform view. Elliptical *primitives* make up the cross-sections.

as a maximum volume, for example, or as geometry *features* that provide a constraint of design motion (either in  $\mathbb{R}^2$  or  $\mathbb{R}^3$ ). Generating additional components within the context of existing parent components is possible in most CAD systems. An assembly-mode perspective permits utilization of existing *features*, such as datum references and constraining surfaces, to define new child components. Regeneration of the model is thereafter successful by maintaining these component-coupling references intact.

Generating the structural components in a manner similar to the outer mold line generation allows for particular *a priori* knowledge to become available (see Section A.4 for further discussion on this). For example, knowing that certain geometry *primitives* are used to define the leading and trailing edge of a wing may serve for datum referencing. Also, knowing the sequence of airfoil sketches used to create the wing loft can simplify the structural datum referencing layout. Such information is key in order for the outer mold line and internal structural components to exhibit similar design motion. Without utilization of such information, the design space of the *assembly* can become, in a design motion sense, restricted by the most rigid component. Structural components are added to the three models presented in Section 2.5.2 to discuss the implementation of these concepts in different scenarios. The added *parts* are “merged,” or added in a Boolean sense, to the existing outer mold line geometry in order to determine if surface intersections occur (a more in-depth discussion on surface intersections is found in Section A.5).

### **Tube-and-Wing Model**

Structural components were added to the generic tube-wing example, shown in Figure 2-23. Both forward and aft box-spars were added to lifting-surfaces in addition to ribs. Stringers were added along the fuselage length with ribs as well. These are seen in Figures 2-25 and 2-26, respectively.

Since the construction of the model outer mold line is known (see Section 2.5.2), the structural components are created in like manner. For example, the spar cross-sections were constrained with an offset from the airfoil splines and lofted across the entire wing span. This was done for the spar loft to remain within the envelope of the wing loft. From a cross-section view in Figure 2-24, the top/bottom of the spar are always within the airfoil spline envelope. Since the spar upper/lower surfaces approximate the wing surface, the design motion will also be similar.



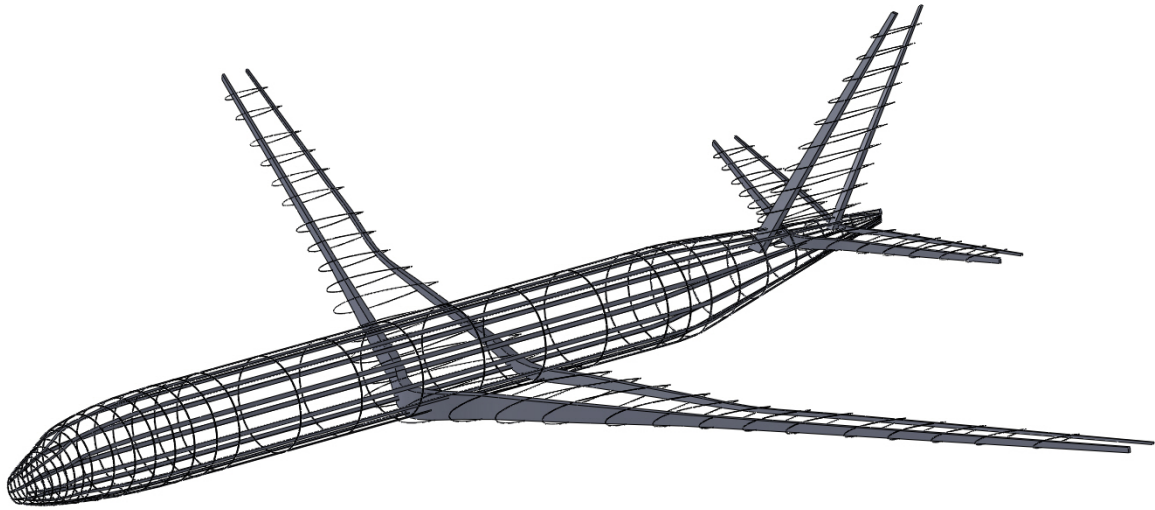


Figure 2-23: Structural components added to the generic tube-wing example followed the same geometry generation procedure as the outer mold line.

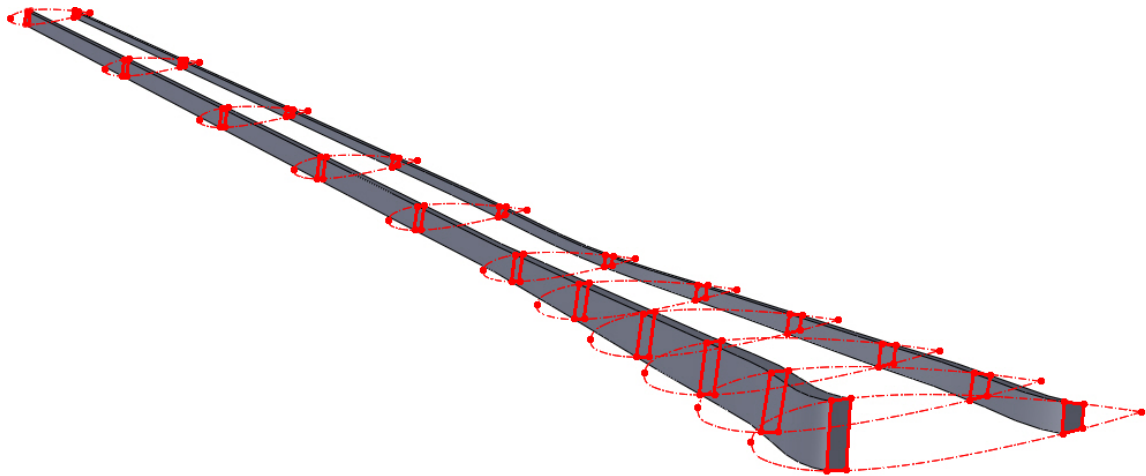


Figure 2-24: Spar cross-sections (red) are constructed to reference the wing airfoil definitions (dash-dot profile), thus remaining within the airfoil spline envelope.

The rib structural components were also created with similar considerations in mind. Figure 2-25 depicts ribs in the fuselage nose section and Figure 2-26 highlights ribs on a lifting-surface. The ribs in both cases fall within the envelope of the parent lofts. In order to avoid surface intersections, rib cross-sections were also constrained with an offset from outer mold line loft and extruded away from any taper direction, as seen in Figure 2-27.

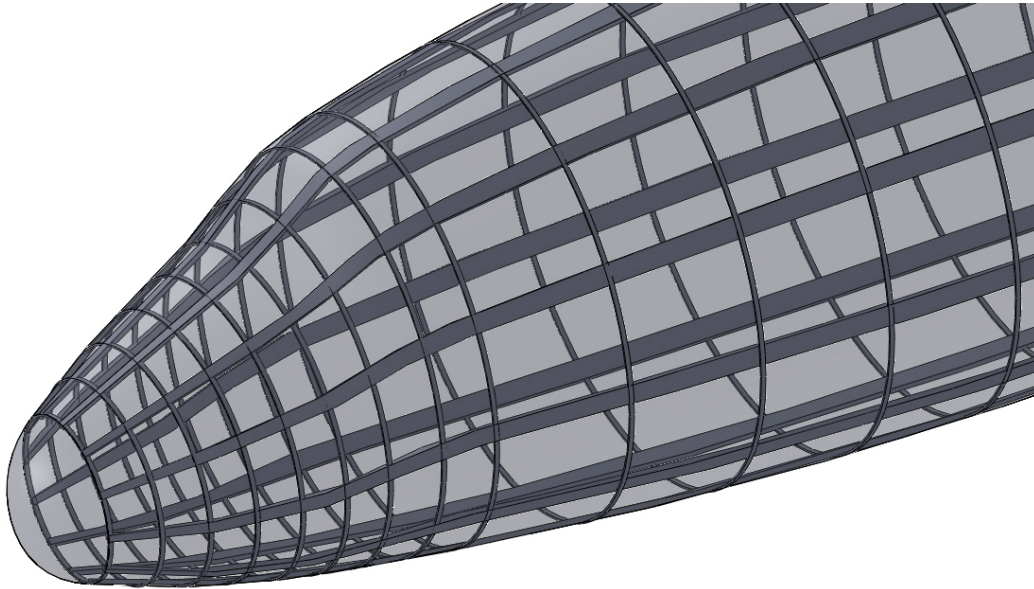


Figure 2-25: Close-up view of fuselage ribs and stringers in the tapered nose section of Figure 2-23.

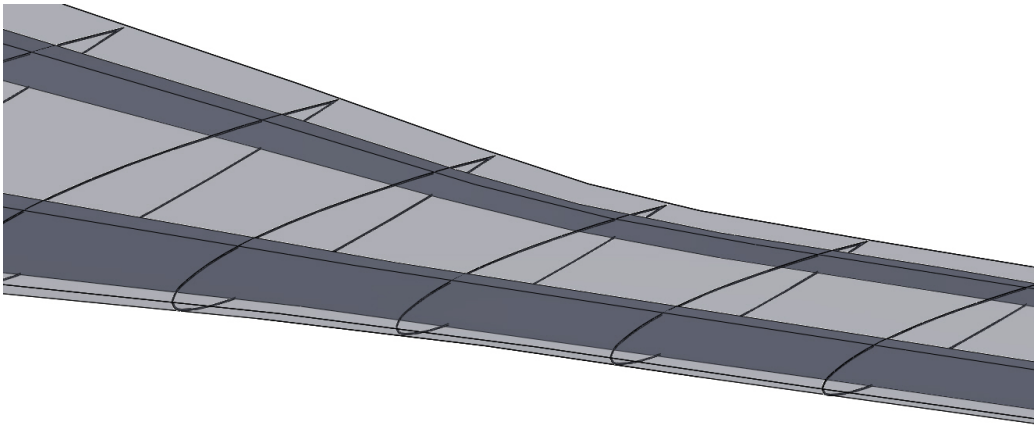


Figure 2-26: Close-up view of wing ribs and spars near the span break section of Figure 2-23.

Figure 2-28 depicts a new design point for the fuselage, where a point on the fuselage datum spline is perturbed in the  $+z$ -direction. The fuselage cross-sections respond to this

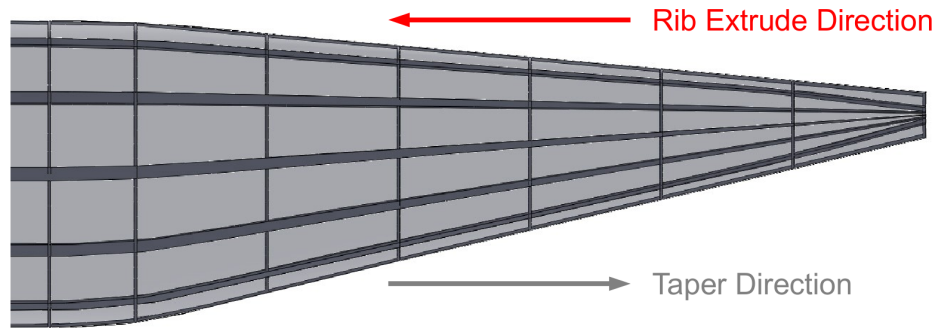


Figure 2-27: Close-up view of ribs and stringers in the tapered aft section of the fuselage model in Figure 2-23. The rib cross-sections are extruded in the direction contrary to the fuselage taper direction.

perturbation by moving to a new point on the model design trajectory. The internal structure follows suit because their design motion is constrained to that of the outer mold line loft. By referencing the datum spline, the expected out-of-plane design motion of each rib is also observed. The same is seen in Figure 2-29 for a  $+z$  perturbation on the wing datum spline. In this case the perturbation occurs between two rib locations and the appropriate design motion occurs without surface intersections. Sometimes the offset values must be larger in regions of high curvature, unless more cross-sections are used. Finally, Figure 2-30 depicts the design motion of internal structural components when an airfoil spline definition (which drives the outer mold line loft) is perturbed in the  $+z$ -direction. The neighboring ribs follow the outer mold line design motion locally, as does the forward spar, by regenerating appropriately.

### Flying-wing Model

The flying-wing model is given internal structural components in a similar manner to the tube-and-wing model, as seen in Figure 2-31. In this case the distribution of spar cross-sections and ribs are also driven by the wing loft, meaning that the variation in airfoil profile, twist and dihedral (which models the winglets) are also manifested in the structure. Compatibility in design motion is equivalent to that of the wing and structure in the tube-and-wing model. However, care must be taken near the wing tips since a quick change in span-wise curvature makes the lofts prone to surface intersections (i.e. the top/bottom face of the wing or spar may “pinch” and intersect with the internal structure surfaces). This aspect of the model may constrain the extent of its design trajectory. Figure 2-32 shows

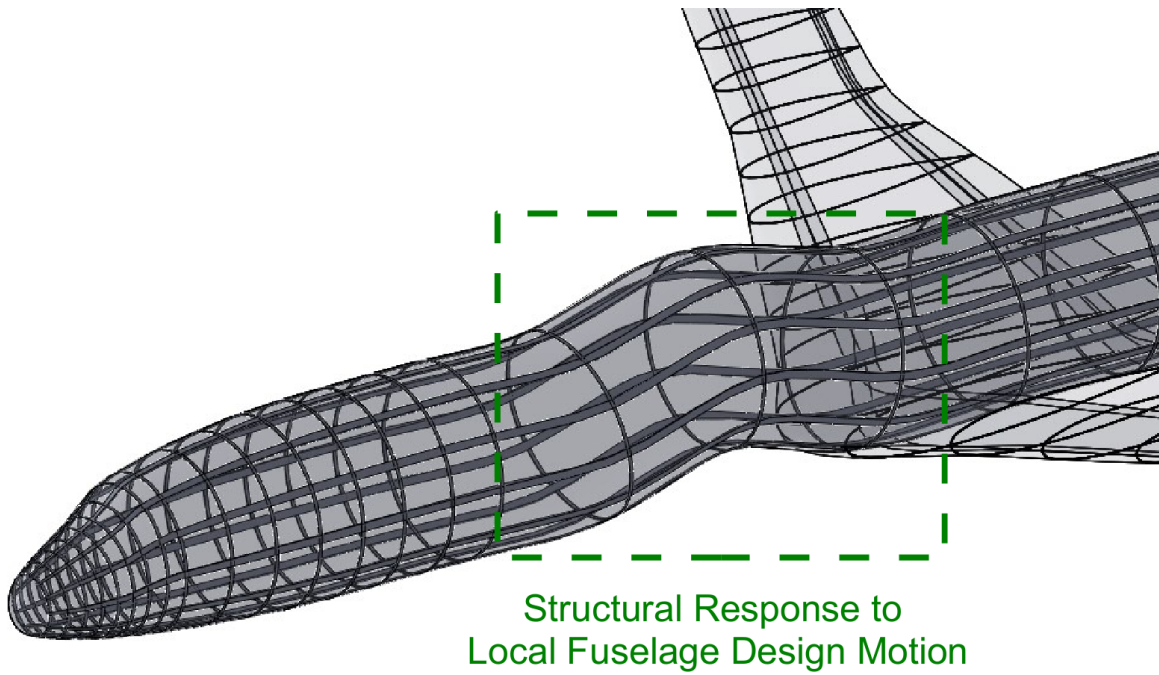


Figure 2-28: A new design point is obtained after perturbing the fuselage datum spline, where the outer mold line loft design motion is followed by the internal structure.

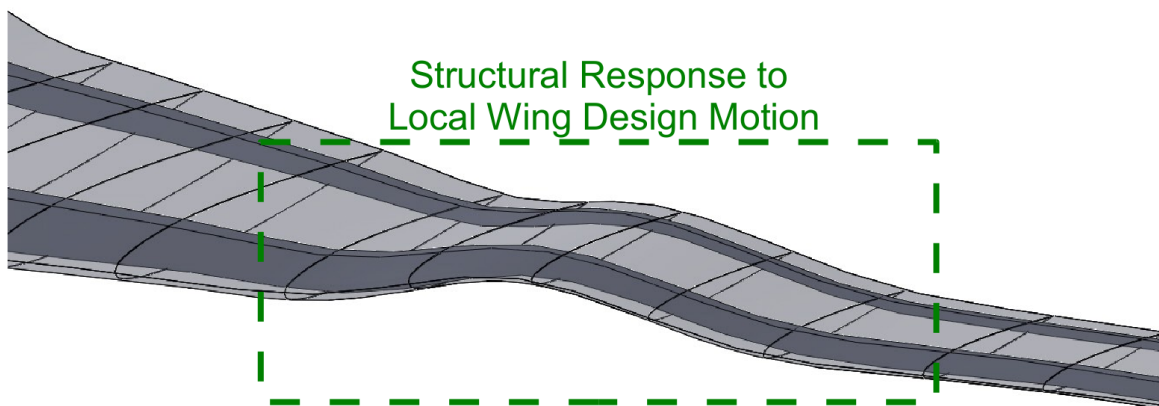


Figure 2-29: A new design point is obtained after perturbing the wing datum spline. Internal structural components exhibit the same design motion as the outer mold line.

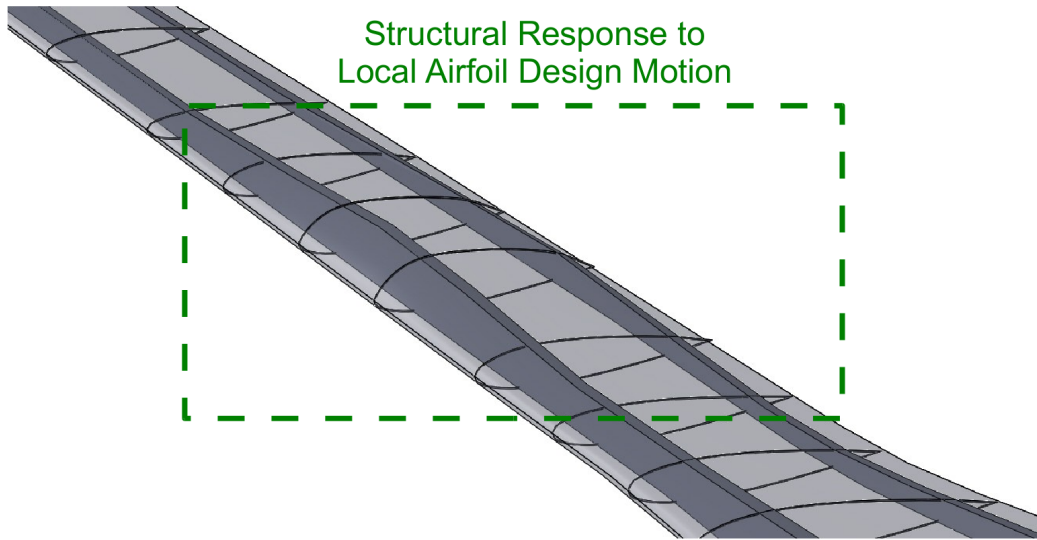


Figure 2-30: Another new design point is obtained by modifying the outer mold line loft definition via perturbation of an airfoil spline. The local internal structure follows the design motion of the wing loft as expected.

surface intersections that appear when a single offset value is used between the spar surface and local wing loft. An increased offset is required at the highlighted spar stations shown in Figure 2-32 to remove the surface intersections in those regions (equivalently, mirroring the modifications to the spar sections at the other wing-tip removes the remaining intersections in the figure).

### Lifting-body Model

The lifting-body model is also given internal structural components, as shown in Figure 2-33. It is no surprise that the structure in the horizontal tail is repositioned within its outer mold line in a “T-tail” as well. The fuselage structure, though, faithfully follows the planform (NACA 3317) and side-view distribution of cross-sections. The circumferential distribution of stringers is maintained and their width is scaled according to the local cross-section sizing. Modification of the stringer offset from the local fuselage loft is needed near the nose and tail to avoid surface intersections due to quickly changing surface curvature. Adding cross-sections to these areas can also accomplish this. Lastly, the distribution of integer-number stringers around the fuselage are parameterized by the relative angle from horizontal of their cross-section. This is the case at each cross-section, however at the trailing edge Figure 2-33 incorrectly suggests the stringers collapse to a point. Close inspection shows that the high

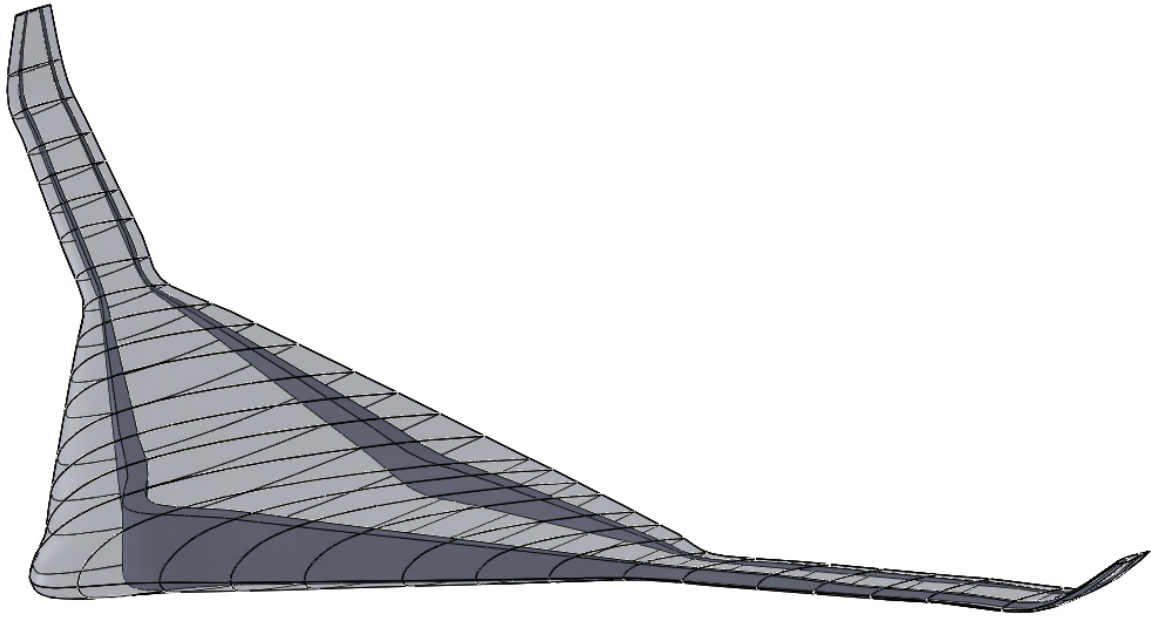


Figure 2-31: Internal spars and ribs are added to a flying-wing model. Each is driven by the parent wing loft, which undergoes changes in airfoil definition, twist, and datum spline deflection.

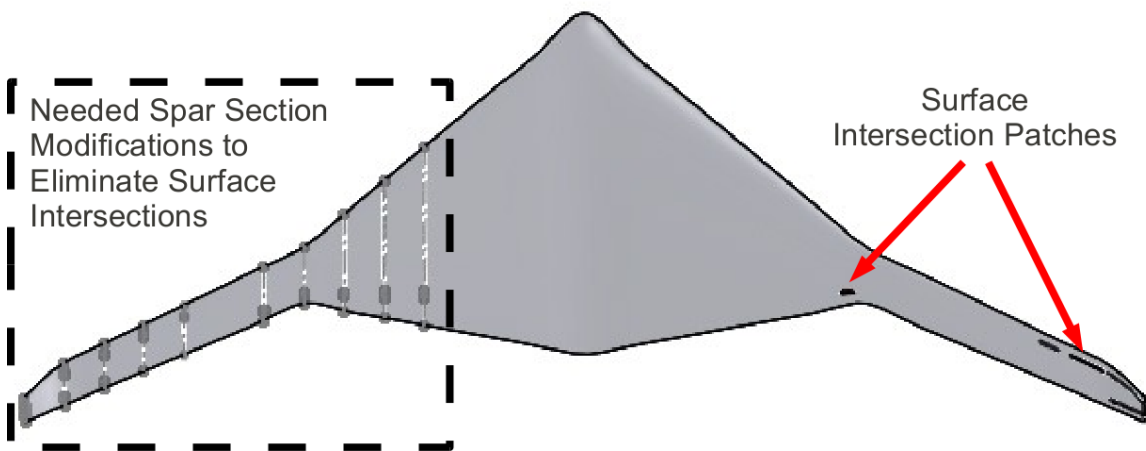


Figure 2-32: Surface intersections appear symmetrically across the body center-line when including internal spars and ribs in a flying-wing model. Although each is driven by the parent wing loft, which undergoes changes in airfoil definition, twist, and datum spline deflection, modifications of the spar offset distance is needed at the highlighted locations to remove surface intersections there.



aspect ratio cross-section maintains the appropriate relative-angular parameterization.

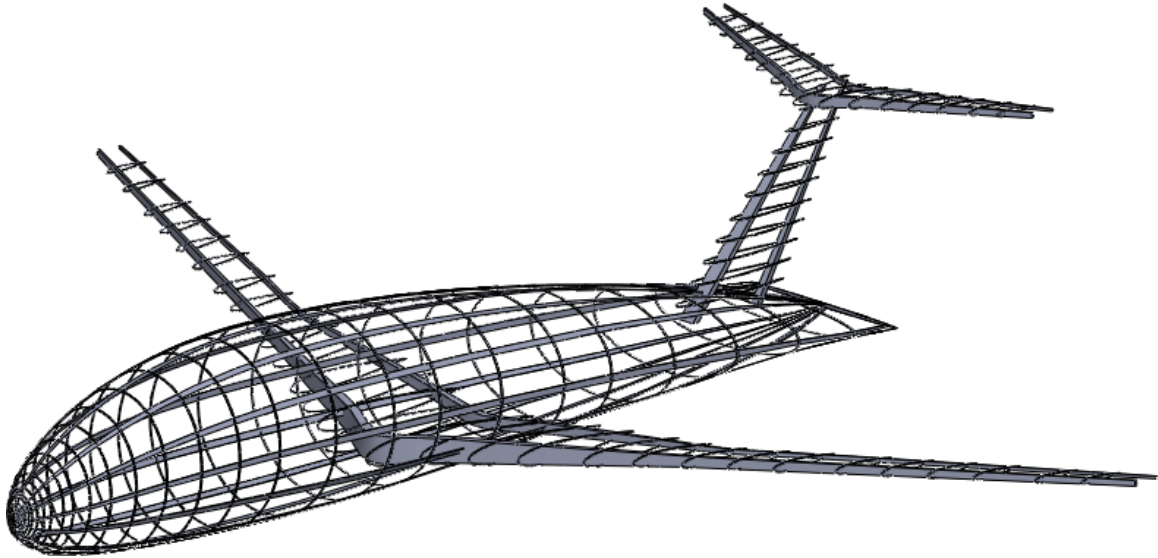


Figure 2-33: The lifting-body model is given internal structural components which follow parameterization changes made to the tube-and-wing model.

## 2.6 Summary

This chapter presents a new perspective on constructing CAD-based model geometry for specific usage in an automated design framework. Notions of multifidelity and multidisciplinary geometry are proposed, along with ideas of design motion, which lead to a formal definition of design intent. With that perspective, examples of model geometry are reviewed with embedded characteristics that make them sufficiently suitable for automated design settings. Using the guiding principles presented herein, CAD models simplify geometry management by circumventing problematic design motion. Geometry gradients become the next focus when gradient-based optimization is desired with such models.

THIS PAGE INTENTIONALLY LEFT BLANK



## Chapter 3

# Geometry Sensitivities for Sketches & BRep Faces

A major component of geometry management in gradient-based optimization is the computation of geometry sensitivities. Doing so with CAD model geometry in an automated fashion via finite-differencing requires additional work to avoid potential complications. The alternative of obtaining analytic geometry derivatives requires some knowledge of how a geometry kernel calculates the final surfaces. Each step of surface generation must be differentiated in order to achieve gradients with respect to *any* driving parameter. This chapter focuses on the “reverse-engineering” efforts needed to obtain such analytic geometry sensitivities for CAD models.<sup>1</sup> Without these contributions, the subsequent chapters focusing on BRep sensitivities could not be fully implemented. This chapter is organized by first considering the solution of non-linear geometry-constraint systems for sketches. Secondly, the gradient of sketch entities to driving parameters is presented. Lastly, the extension of sketch derivatives to geometry gradients on sketch-driven surfaces are discussed for extrude, revolve and sweep *features*.

---

<sup>1</sup>In this thesis the term “reverse engineering” implies inferring the algorithm model for surface-generation in a CAD system geometry kernel. This contrasts with a particular usage of the term in the CAD community, where it is often described as recasting an already existing physical object (via 3D imaging, for example) into a digital model using a CAD system.

### 3.1 Geometry Gradients For Canonical Parametric Surfaces

Since the faces, edges and nodes in the BRep essentially “sit on” each geometry surface used to construct the model (within a tolerance), it is clear that the BRep sensitivity will depend on the sensitivity of the underlying surfaces. Therefore, each surface in the model must be differentiated with respect to parameters in order to obtain the desired geometry gradients. As seen in the shape sensitivity analysis for Equation (1.1), only the design velocity on active boundaries is required. This is convenient for shape design with CAD geometry because BRep information is readily available and its associativity with surfaces can be established. To demonstrate this, a simple scenario is first considered for a cone-plane model geometry by determining its topology connectivity and associating its driving parameters to its BRep for differentiation.

In Figure 3-1(a) there are four sketch entities (depicted as thick lines) used to create a cone *feature* (shown as shaded faces): a vertical line segment, vertical center-line, horizontal line segment and a diagonal line segment. The driving parameters for the horizontal and diagonal line segments are the angle  $\theta$  and height  $h$ . When a revolve geometry operator is applied to this sketch, the CAD geometry kernel determines that a cone surface and a planar surface are the resulting geometry needed to create the cone *feature*, as shown by the shaded cone in Figure 3-1(a). The CAD geometry kernel trim the planar “cap” surface with the parametric cone, which is also bounded by the sketch entities. The planar surface in Figure 3-1(a) is associated with the sketched horizontal line segment and the cone surface is associated with the sketched diagonal line segment. The parameters  $\theta$  and  $h$  are associated with both surfaces through this sketch-surface associativity. When an additional *feature*, a plane-cut, is introduced in the feature-tree, the model in Figure 3-1(b) results after regeneration. An additional distance parameter,  $d$ , drives the new cut *feature* and the resulting planar surface alone. At this point, the master-model contains two *features* (a cone and plane-cut) in the feature-tree and three parameters. The resulting BRep topology for the new model is shown in Figures 3-1(c) and 3-1(d) (note that periodic and symmetric edges are split with a node), containing faces  $\{\mathcal{F}_1, \dots, \mathcal{F}_4\}$ , edges  $\{\mathcal{E}_1, \dots, \mathcal{E}_7\}$  and nodes  $\{\mathcal{N}_1, \dots, \mathcal{N}_5\}$ . In this final form the edges  $\{\mathcal{E}_3 \dots \mathcal{E}_7\}$  are trim curves since two planar surfaces and the cone surface are trimmed. The connectivity hierarchy between these BRep entities is shown in Table 3.1.

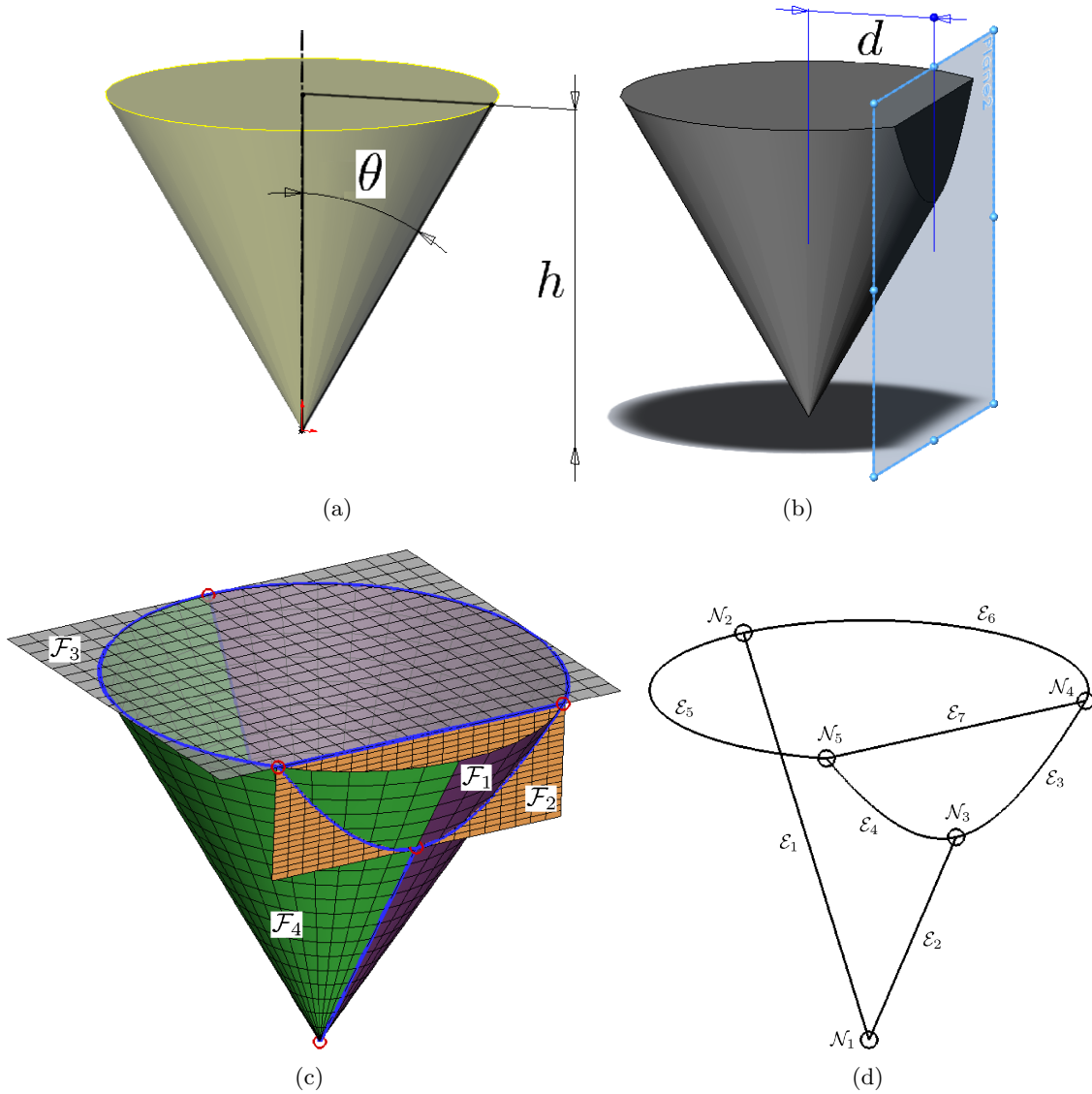


Figure 3-1: (a) Sketch entities are piecewise continuous and driven by parameters, such as the cone angle  $\theta$  and height  $h$  in this case. The CAD geometry kernel processes the sketch entities and generates associated surfaces for the cone feature. (b) Adding a cut-operator with a planar face results in a new model instance and an additional geometry parameter,  $d$ . (c)-(d) The final model topology for this cone-plane feature shows edges  $\{\mathcal{E}_1, \dots, \mathcal{E}_7\}$  (lines) and nodes  $\{\mathcal{N}_1, \dots, \mathcal{N}_5\}$  (circles) bounding faces  $\{\mathcal{F}_1, \dots, \mathcal{F}_4\}$  used to create the feature.

Feature Name	BRep Connectivity						
	Features	Faces	Loops	Edges	Nodes		
Cone	$\Omega_1$	$\mathcal{F}_1$	1	$\mathcal{E}_1$	$\{\mathcal{N}_1, \mathcal{N}_2\}$		
				$\mathcal{E}_2$	$\{\mathcal{N}_1, \mathcal{N}_3\}$		
				$\mathcal{E}_3$	$\{\mathcal{N}_3, \mathcal{N}_4\}$		
				$\mathcal{E}_6$	$\{\mathcal{N}_2, \mathcal{N}_4\}$		
				$\mathcal{F}_3$	1	$\mathcal{E}_5$	$\{\mathcal{N}_2, \mathcal{N}_5\}$
						$\mathcal{E}_6$	$\{\mathcal{N}_2, \mathcal{N}_4\}$
						$\mathcal{E}_7$	$\{\mathcal{N}_4, \mathcal{N}_5\}$
		$\mathcal{F}_4$	1	$\mathcal{E}_1$	$\{\mathcal{N}_1, \mathcal{N}_2\}$		
				$\mathcal{E}_2$	$\{\mathcal{N}_1, \mathcal{N}_3\}$		
				$\mathcal{E}_4$	$\{\mathcal{N}_3, \mathcal{N}_5\}$		
				$\mathcal{E}_5$	$\{\mathcal{N}_2, \mathcal{N}_5\}$		
		Plane-Cut	$\Omega_2$	$\mathcal{F}_2$	1	$\mathcal{E}_3$	$\{\mathcal{N}_3, \mathcal{N}_4\}$
						$\mathcal{E}_4$	$\{\mathcal{N}_3, \mathcal{N}_5\}$
						$\mathcal{E}_7$	$\{\mathcal{N}_4, \mathcal{N}_5\}$

Table 3.1: The connectivity hierarchy for the cut-cone model in Figure 3-1.

An additional associativity connection is made between the edges and nodes bounding each face to the underlying surface for that face. In the case of Figure 3-1, the parameters  $\theta$  and  $h$  are associated with faces  $\{\mathcal{F}_1, \mathcal{F}_3, \mathcal{F}_4\}$ . The parameter  $d$  only drives face  $\mathcal{F}_2$ . In terms of edges and nodes, the  $d$  and  $\theta$  parameters both drive the trim curves  $\{\mathcal{E}_3, \dots, \mathcal{E}_7\}$  and nodes  $\{\mathcal{N}_2, \dots, \mathcal{N}_5\}$ , whereas the  $h$  parameter only drives edges  $\{\mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_7\}$  and nodes  $\{\mathcal{N}_3, \mathcal{N}_4, \mathcal{N}_5\}$ . Furthermore, the underlying canonical planar and cone surfaces ( $\mathbf{r}_1 = \mathbf{r}_1(u_1, v_1; d)$  and  $\mathbf{r}_2 = \mathbf{r}_2(u_2, v_2; \theta, h)$ , respectively) are defined using the classical surface parameterizations in Table 3.2. The parameterizations in Table 3.2 have  $\mathbf{O}_1 = [d, 0, 0]^T \in \mathbb{R}^3$  as the relative

Vertical Plane	Cone
$\mathbf{r}_1 = \mathbf{O}_1 + \begin{bmatrix} 0 \\ v_1 - 1/2 \\ u_1 \end{bmatrix}$	$\mathbf{r}_2 = \mathbf{O}_2 + \begin{bmatrix} v_2 \tan(\theta) \cos(u_2) \\ v_2 \tan(\theta) \sin(u_2) \\ hv_2 \end{bmatrix}$

Table 3.2: Analytic surface parameterizations for the vertical plane and cone generated in Figure 3-1.

plane origin with  $d \in \mathbb{R}$  as the distance parameter.  $\mathbf{O}_2 = [0, 0, 0]^T \in \mathbb{R}^3$  is the relative cone origin.  $\theta \in \mathbb{R}$  is the cone semi-angle and  $h \in \mathbb{R}$  is the cone height. A global reference frame is also presumed with an origin at  $\mathbf{O}_0 = [0, 0, 0]^T \in \mathbb{R}^3$ . In this example each surface is defined

in its own domain space,  $W_1 \in \mathbb{R}^2$  and  $W_2 \in \mathbb{R}^2$ , where  $W_1 = U_1 \times V_1$  and  $W_2 = U_2 \times V_2$  with  $U_1 = [u_{1,\min}, u_{1,\max}] \in \mathbb{R}$ ,  $V_1 = [v_{1,\min}, v_{1,\max}] \in \mathbb{R}$ ,  $U_2 = [u_{2,\min}, u_{2,\max}] \in \mathbb{R}$  and  $V_2 = [v_{2,\min}, v_{2,\max}] \in \mathbb{R}$  and coordinates  $u_1 \in U_1$ ,  $v_1 \in V_1$ ,  $u_2 \in U_2$  and  $v_2 \in V_2$ . The parameters, which in the case of Figure 3-1 are  $\theta = 30^\circ$ ,  $d = 0.4$  and  $h = 1$ , were associated with the BRep topology and surface parameterizations by inspection. Ideally these parameters must be associated to the appropriate topology and surface equation terms in an automated fashion. Once this is established, a complete associativity chain is created that links driving parameters from a sketch to the resulting faces in every *feature* of the model.

Vertical Plane	Cone
$\frac{\partial \mathbf{r}_1}{\partial u_1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\frac{\partial \mathbf{r}_2}{\partial u_2} = \begin{bmatrix} -v \tan(\theta) \sin(u) \\ v \tan(\theta) \cos(u) \\ 0 \end{bmatrix}$
$\frac{\partial \mathbf{r}_1}{\partial v_1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\frac{\partial \mathbf{r}_2}{\partial v_2} = \begin{bmatrix} \tan(\theta) \cos(u) \\ \tan(\theta) \sin(u) \\ h \end{bmatrix}$
$\frac{\partial \mathbf{r}_1}{\partial d} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\frac{\partial \mathbf{r}_2}{\partial \theta} = \begin{bmatrix} v \sec^2(\theta) \cos(u) \\ v \sec^2(\theta) \sin(u) \\ 0 \end{bmatrix}$
	$\frac{\partial \mathbf{r}_2}{\partial h} = \begin{bmatrix} 0 \\ 0 \\ v \end{bmatrix}$

Table 3.3: Analytic representation of partial derivatives for the plane and cone surface parameterizations in Table 3.2.

The geometry gradients of  $\mathbf{r}_1$  or  $\mathbf{r}_2$  with respect to any parameter in  $\{\theta, h, d\}$  are easily computed using the formulas in Table 3.3. In an automated geometry management setting, these relations can be hard-coded and passed the requisite  $(u, v)$  and parameter values to return a partial derivative. The same is true when a model is constructed of *features* using any classical geometry surface with canonical parameterizations (these parameterizations are sometimes made available by the CAD system vendor, such as Pro/ENGINEER [62], or need to be inferred). If the geometry parameters themselves are defined as explicit functions of other parameters, say  $\{\theta(a), h(b), d(c)\}$ , by the designer, then the partial derivatives in

Table 3.3 are augmented via the chain-rule as follows:

$$\frac{\partial \mathbf{r}_1}{\partial c} = \frac{\partial \mathbf{r}_1}{\partial d} \frac{\partial d}{\partial c}, \quad \frac{\partial \mathbf{r}_2}{\partial a} = \frac{\partial \mathbf{r}_2}{\partial \theta} \frac{\partial \theta}{\partial a}, \quad \frac{\partial \mathbf{r}_2}{\partial b} = \frac{\partial \mathbf{r}_2}{\partial h} \frac{\partial h}{\partial b}.$$

If the designer creating the model defines the functions  $\theta(a)$ ,  $h(b)$  and  $d(c)$ , their partial derivatives can be easily hard-coded and passed requisite data to return the desired derivative. It is important to note that in these circumstances the *only* parameters driving a surface (and hence BRep faces) are those found explicitly in the surface parameterization or in added designer-defined functions. No other geometry parameter in a model will impact the geometry gradient of these surfaces.

A more difficult scenario arises when geometry parameters do not explicitly drive surfaces through a canonical parameterization. In particular, these parameters consist of model dimensions often placed on the end-points of sketch entities that contribute to bounds on *feature* surfaces. Without visual inspection of these dimensions, constraints and the sketch entities defining a *feature*, it is potentially ambiguous for an automated geometry management system to determine how these particular dimensions drive surfaces. The ambiguity stems first from the fact that the geometry kernel may utilize various rules to define the nonlinear system of equations incorporating all dimensions and geometry constraints in a sketch (this process is proprietary, likely non-standard across geometry kernels and potentially composed of parametric or variational approaches [30]). These rules may lead to multiple non-unique solutions to the sketch-solve as well. In such cases, the geometry kernel may infer an “intended”<sup>2</sup> solution or utilize implicit constraints not imposed explicitly by the user (see [80], [43], [51] and [23] for discussions on solving geometry constraint problems for sketches). Secondly, the sketch solution becomes an input to the surface parameterizations, namely the  $(u, v)$  definition, which depend on the sketch geometry and constraints in a non-transparent manner to the user.

---

<sup>2</sup>The “intended” design motion for sketch entities can be inferred by the sketch solve algorithm. However, the resulting design motion may *not* actually reflect the designer’s desired design motion for the sketch. This can occur, for example, in under-constrained sketches.

## 3.2 Geometry Gradients of Sketches for Sketch-Driven Surfaces

Since sketch *primitives* are the starting point for model geometry in a CAD system, understanding how parameterized sketches are solved leads to a formulation for sketch sensitivities. These tools are necessary to determine surface sensitivities on extrude, revolve and sweep *features* as well. The geometry sensitivities of B-spline curves and surfaces (i.e., lofts and blends) are discussed in Chapter 5.

### 3.2.1 Solving a Parameterized Sketch

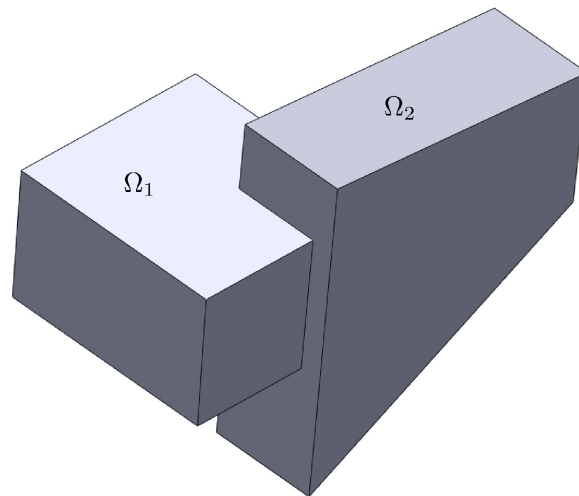
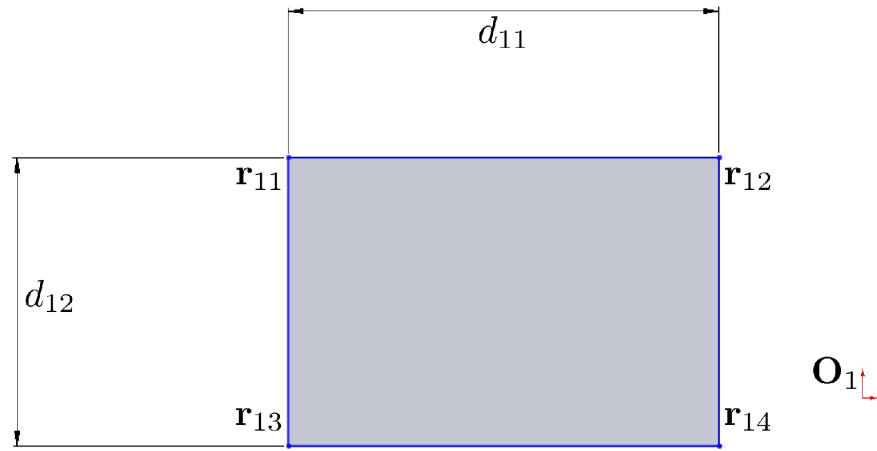
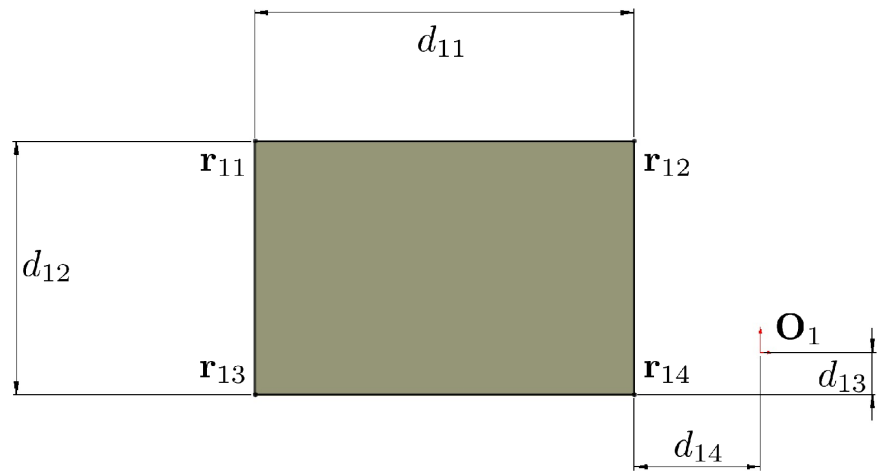


Figure 3-2: An example model containing two extrusion *features*,  $\Omega_1$  and  $\Omega_2$ , consisting of simple and complex sketch-driving geometry.

A few examples help illustrate the challenges in automatically associating driving sketch dimensions to model surfaces. First, many CAD systems permit users to under-define or fully-define sketches. To see the impact of these two sketch designations, the model in Figure 3-2 (generated using the SolidWorks CAD system) is analyzed by considering the extrusion *feature*  $\Omega_1$  and its driving sketch. Figure 3-3(a) depicts an under-defined sketch and Figure 3-3(b) shows the same sketch entities augmented with additional dimensions to make it fully-defined. In each case the vertices  $\mathbf{r}_{11} \dots \mathbf{r}_{14}$  consist of 2D coordinates  $(x_{11}, y_{11}) \dots (x_{14}, y_{14})$ , respectively; the local sketch origin represents  $\mathbf{O}_1 = (x_{10}, y_{10})$  (which is known) in both sketches as well. The simple geometry constraints for the sketch in 3-3(a)



(a)



(b)

Figure 3-3: (a) An under-defined sketch of two line segments, constrained vertically, that are coincident to two line segments, constrained horizontally, with corresponding dimensions. (b) A fully-defined version of the sketch in (a) due to the addition of two additional dimensions ( $d_{13}$  and  $d_{14}$ ).



are summarized as:

$$\begin{aligned}
 x_{11} - x_{12} - d_{11} &= 0 \\
 x_{13} - x_{14} - d_{11} &= 0 \\
 y_{11} - y_{13} - d_{12} &= 0 \\
 y_{12} - y_{14} - d_{12} &= 0
 \end{aligned}
 \tag{3.1}$$

A system of linear equations is easily created to solve for  $(x_{11}, y_{11}) \dots (x_{14}, y_{14})$  by writing

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{11} & y_{11} \\ x_{12} & y_{12} \\ x_{13} & y_{13} \\ x_{14} & y_{14} \end{bmatrix} = \begin{bmatrix} d_{11} & 0 \\ d_{11} & 0 \\ 0 & d_{12} \\ 0 & d_{12} \end{bmatrix}.
 \tag{3.2}$$

The problem, though, is that the under-defined sketch implies a matrix on the left-hand side in 3.2 that is singular. In the CAD system the sketch itself is solved in the GUI due to *implicit* constraints added to (3.2) by the geometry kernel in order for the left-hand side coefficient matrix to be non-singular. These hidden internal constraints are not known a priori and thus make the situation ambiguous for an automated geometry management system. In particular, only an ambiguous answer is possible to questions such as: should  $\mathbf{r}_{11}$ ,  $\mathbf{r}_{12}$  or both exhibit design motion when  $d_{11}$  is changed in Figure 3-3(a)? The answer to this will yield very different geometry gradients with respect to  $d_{11}$  for the resulting *feature* surfaces.

Conversely, adding the dimensions  $d_{13}$  and  $d_{14}$  in Figure 3-3(b) fully-defines the sketch by creating a reduction in the degrees of freedom by their defining equations

$$\begin{aligned}
 x_{14} - x_{10} - d_{14} &= 0 \\
 y_{14} - y_{10} - d_{13} &= 0.
 \end{aligned}$$

Hence a new, non-singular system can be formed to solve for all vertex coordinates in Figure

3-3(b):

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{11} & y_{11} \\ x_{12} & y_{12} \\ x_{13} & y_{13} \end{bmatrix} = \begin{bmatrix} d_{11} & 0 \\ d_{11} + x_{10} + d_{14} & y_{10} + d_{13} \\ 0 & d_{12} \end{bmatrix}. \quad (3.3)$$

The fully-defined system results in a single solution to the sketch (this was verified using the SolidWorks CAD system). In general, it is possible to confront both under- and fully-defined sketch types. Typically the geometry kernel will not permit over-constrained sketches (where no solution is possible) and request that the user remove redundant or conflicting dimensions/constraints. In under-constrained sketches (i.e., under-defined), the geometry kernel may suggest a solution based on its own internal implicit constraints and permit leaving the sketch seemingly under-defined to the user.

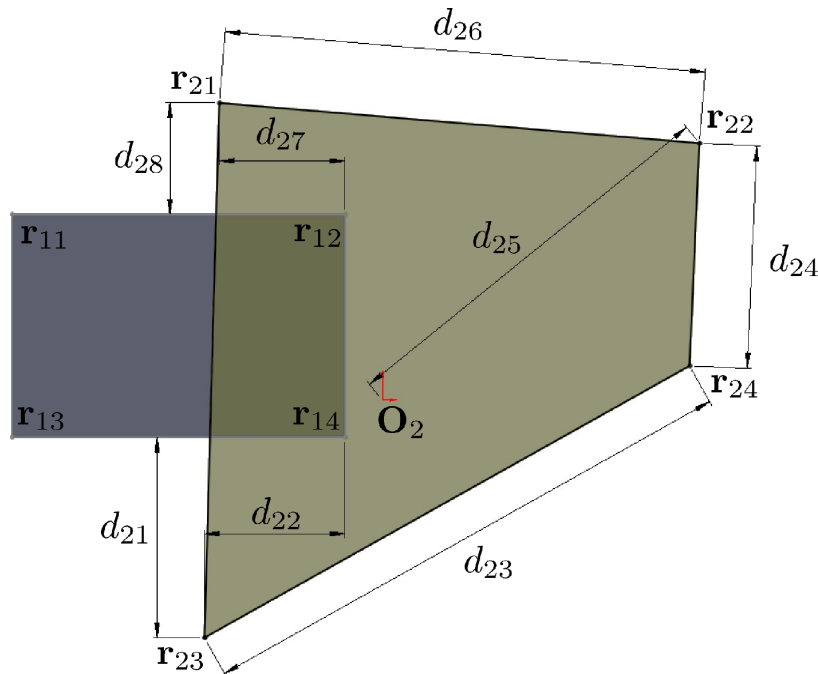


Figure 3-4: The fully-defined driving sketch for *feature*  $\Omega_2$  in Figure 3-2.

Complexity increases in the driving sketch of *feature*  $\Omega_2$  in Figure 3-2. This sketch is shown in Figure 3-4 and consists solely of line segments and dimensions (i.e., no geometry constraints are present). The vertices  $\mathbf{r}_{21}$  and  $\mathbf{r}_{23}$  are dimensioned relative to vertices  $\mathbf{r}_{12}$  and  $\mathbf{r}_{14}$ , respectively, from the sketch generating  $\Omega_1$ . The set of geometry constraint equations

thus become (again, with respect to the SolidWorks system):

$$\begin{aligned}
 (x_{21} - x_{22})^2 + (y_{21} - y_{22})^2 - d_{26}^2 &= 0 \\
 (x_{22} - x_{24})^2 + (y_{22} - y_{24})^2 - d_{24}^2 &= 0 \\
 (x_{23} - x_{24})^2 + (y_{23} - y_{24})^2 - d_{23}^2 &= 0 \\
 (x_{22} - x_{20})^2 + (y_{22} - y_{20})^2 - d_{25}^2 &= 0 \\
 -x_{21} + x_{12} - d_{27} &= 0 \\
 y_{21} - y_{12} - d_{28} &= 0 \\
 -x_{23} + x_{14} - d_{22} &= 0 \\
 -y_{23} + y_{14} - d_{21} &= 0,
 \end{aligned} \tag{3.4}$$

which need to be solved iteratively (e.g., using a Newton-Raphson method, or other commonly used method [30]). It is important to note that the driving dimensions in these sketches explicitly drive the *end-points* of the sketch entities. This inferred detail implies that the sketch entities themselves are explicitly parameterized by their end-points and implicitly driven by the sketch dimensions. Therefore, the geometry constraint equations for sketches will only consist of statements relating the horizontal or vertical placement of sketch end-points with respect to one another.

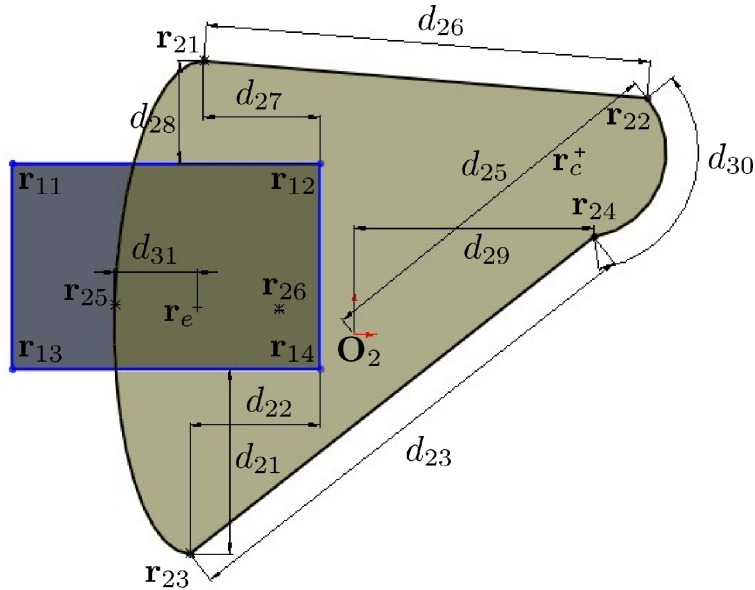


Figure 3-5: The fully-defined driving sketch containing elliptical and arc sketch entities added to the sketch in Figure 3-4.

An additional example of a sketch of added complexity is shown in Figure 3-5, where elliptical and arc sketch entities were added to the sketch chain in Figure 3-4. This scenario contains added degrees of freedom due to the arc center point, ellipse center point and floating ellipse quadrant point ( $\mathbf{r}_{26}$ ), as well as an angular dimension ( $d_{30}$ ) and added horizontal dimension ( $d_{31}$ ). The geometry constraint system for this scenario is found to be:

$$\begin{aligned}
(x_{21} - x_{22})^2 + (y_{21} - y_{22})^2 - d_{26}^2 &= 0 \\
(x_{22} - x_c)(x_{24} - x_c) + (y_{22} - y_c)(y_{24} - y_c) - \\
\frac{1}{4} (|[x_{22} - x_c, y_{22} - y_c]| + |[x_{24} - x_c, y_{24} - y_c]|)^2 \cos(d_{30}) &= 0 \\
(x_{22} - x_c)^2 + (y_{22} - y_c)^2 - (x_{24} - x_c)^2 - (y_{24} - y_c)^2 &= 0 \\
(x_{23} - x_{24})^2 + (y_{23} - y_{24})^2 - d_{23}^2 &= 0 \\
(x_{22} - x_{20})^2 + (y_{22} - y_{20})^2 - d_{25}^2 &= 0 \\
-x_{21} + x_{12} - d_{27} &= 0 \\
y_{21} - y_{12} - d_{28} &= 0 \\
-x_{23} + x_{14} - d_{22} &= 0 \\
-y_{23} + y_{14} - d_{21} &= 0, \\
x_{24} - x_{20} - d_{29} &= 0, \\
x_e - x_{25} - d_{31} &= 0, \\
(y_{25} - y_e)^2 - (y_{26} - y_e)^2 &= 0, \\
(x_{21} - x_e)^2 + (y_{21} - y_e)^2 - (x_{23} - x_e)^2 - (y_{23} - y_e)^2 &= 0, \\
(x_{25} - x_{21})^2 + (y_{25} - y_{21})^2 - (x_{25} - x_{23})^2 - (y_{25} - y_{23})^2 &= 0, \\
(x_{25} - x_e)^2 - (x_{26} - x_e)^2 &= 0, \\
(x_{21} - x_e)^2 - (x_{23} - x_e)^2 &= 0.
\end{aligned} \tag{3.5}$$

This system can also be solved using a Newton-Raphson approach for all of the sketch entity end-points, the arc center point, ellipse center point and floating ellipse quadrant point. It is possible to rewrite some of the geometry constraint equations in (3.5) in a different manner that results in an equivalent system. Determining these geometry constraint relationships for sketches in a CAD system is necessary to identify how parameters drive the model at

the fundamental level. Once they are obtained for each available sketch *primitive* in a CAD system, then differentiation of the sketch geometry with respect to its driving dimensions is possible. If the given sketch is not fully-defined, then a complete geometry constraint set cannot be clearly created unless the implicit constraint rules of the geometry kernel are discovered. Without such information, the sketch system is under-determined and an infinite number of solutions become possible.

### 3.2.2 Design Velocity at Sketch Entity End-Points

Once a fully-defined sketch system is obtained and its geometry constraints understood, the Newton system can be used to obtain geometry gradients for the sketch. The example of Figure 3-5 serves to demonstrate this point. The degrees of freedom for the sketch are organized in a vector  $\mathbf{x}$ . A residual vector  $\mathbf{f} = \mathbf{f}(\mathbf{x}, \mathcal{P})$  contains rows with each equation in (3.5) and a Jacobian  $\mathbf{J}$  matrix is then created from  $\mathbf{f}$ :

$$\mathbf{x} = [x_{21}, y_{21}, x_{22}, y_{22}, x_{23}, y_{23}, x_{24}, y_{24}, x_{25}, y_{25}, x_{26}, y_{26}, x_c, y_c, x_e, y_e]^T$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_{16}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_{16}}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{f}_{16}}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{f}_{16}}{\partial \mathbf{x}_{16}} \end{bmatrix}.$$

After linearizing  $\mathbf{f}$  with Taylor expansions, the sketch is solved via the Newton system

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{J}^{-1}\mathbf{f}, \quad (3.6)$$

where the iterate  $i$  is augmented with each  $\mathbf{x}$  update until the residuals fall below a specified tolerance. This usually requires 1 to 3 Newton iterations (even with added geometry “noise”) to converge to within  $1.0 \times 10^{-10}$  for the geometry constraints in (3.5). If the sketch is solved exactly, then (3.6) is satisfied with  $\mathbf{x}^{i+1} = \mathbf{x}^i$ , leaving

$$\mathbf{f} = \mathbf{0}.$$

If the derivative with respect to a sketch dimension  $\mathcal{P}$  is desired, then this outcome can be rewritten as follows since  $\mathbf{f} = \mathbf{f}(\mathbf{x}, \mathcal{P})$ :

$$\frac{d\mathbf{f}}{d\mathcal{P}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathcal{P}} + \frac{\partial \mathbf{f}}{\partial \mathcal{P}} = \mathbf{0}.$$

By noting that  $\partial \mathbf{f} / \partial \mathbf{x} = \mathbf{J}$ , this becomes

$$\frac{\partial \mathbf{x}}{\partial \mathcal{P}} = -\mathbf{J}^{-1} \frac{\partial \mathbf{f}}{\partial \mathcal{P}}. \quad (3.7)$$

Solving equation (3.7) results in the derivative of *all* sketch degrees of freedom with respect to *any* driving dimension found in that sketch. The Jacobian  $\mathbf{J}$  used to solve the sketch system remains exactly the same when differentiating the sketch. As long as the sketch is solved, this method will provide geometry sensitivity of the sketch end-points. If the sketch is unsolved, then geometry sensitivities are clearly unobtainable until sketch problems are remedied. Once the end-point design velocities are known, the design velocity along sketch entities can be found next. Due to the associativity of sketch entities to resulting *features*, the design velocity of *feature* surfaces can then be found as well. That is the first step in completely finding the geometry gradient of a CAD model. Subsequent steps require determining the design velocity on BRep edges and nodes based on the design velocity of their parent *feature* surfaces (this will be further explored in Chapter 4) or sketch *primitives*.

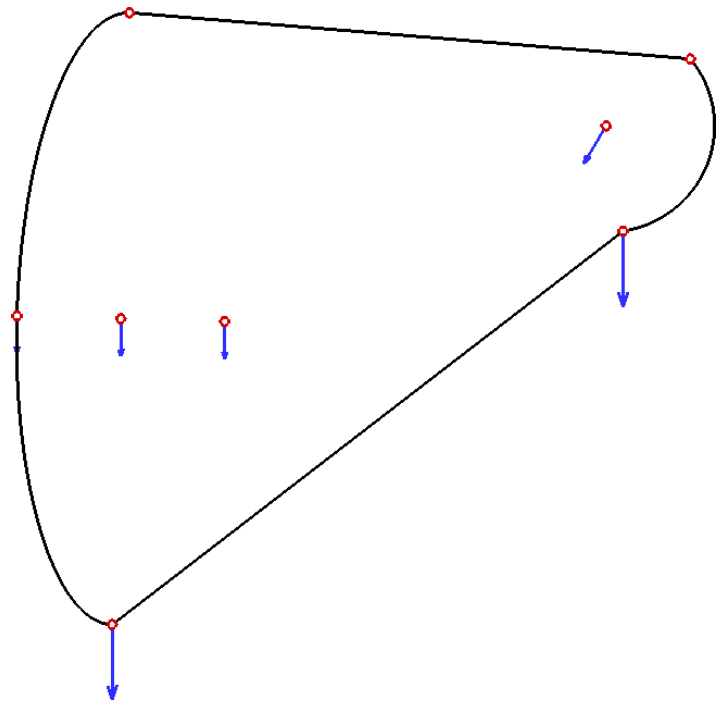
If the sketch is not solved to machine-precision due to CAD tolerances, then residual geometry “noise” remains such that  $\mathbf{x}^{i+1} - \mathbf{x}^i = \boldsymbol{\epsilon}_s$  with  $\boldsymbol{\epsilon}_s$  approximately the same order of magnitude as the Newton tolerance. This changes the sketch derivative result in (3.7) to

$$\frac{\partial \hat{\mathbf{x}}}{\partial \mathcal{P}} = -\mathbf{J}^{-1} \frac{\partial \mathbf{f}}{\partial \mathcal{P}} - \tilde{\boldsymbol{\epsilon}}_s, \quad (3.8)$$

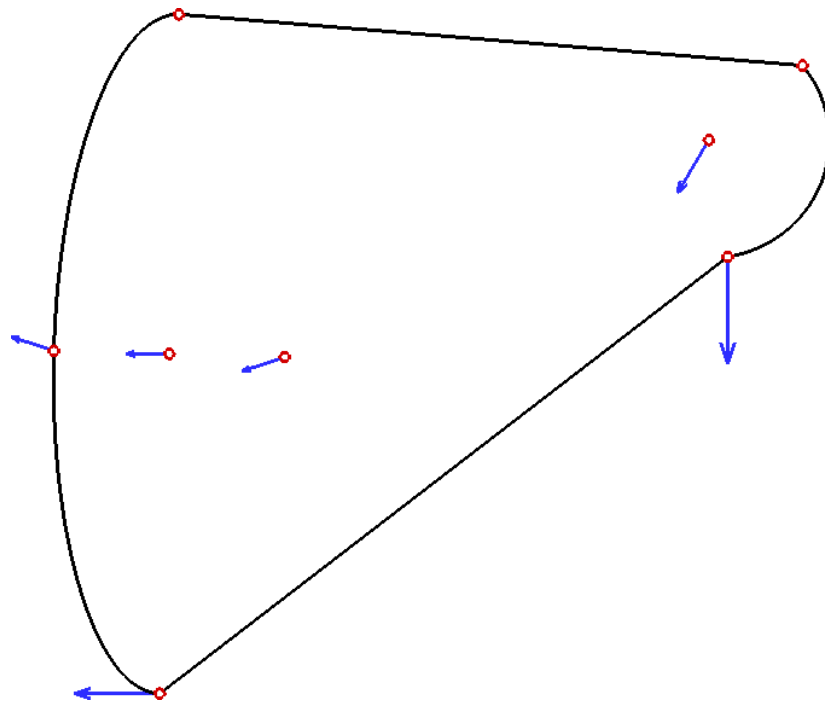
where  $\tilde{\boldsymbol{\epsilon}}_s = \partial \boldsymbol{\epsilon}_s / \partial \mathcal{P}$  is an error in the sketch derivative of unknown magnitude. This result implies  $\frac{\partial \hat{\mathbf{x}}}{\partial \mathcal{P}} \neq \frac{\partial \mathbf{x}}{\partial \mathcal{P}}$  compared to the true derivative  $\frac{\partial \mathbf{x}}{\partial \mathcal{P}}$ . Nevertheless, the sketch derivative is as accurate as the CAD system tolerance permits because as  $\boldsymbol{\epsilon}_s \rightarrow \mathbf{0}$  then  $\tilde{\boldsymbol{\epsilon}}_s \rightarrow \mathbf{0}$  and  $\frac{\partial \hat{\mathbf{x}}}{\partial \mathcal{P}} \rightarrow \frac{\partial \mathbf{x}}{\partial \mathcal{P}}$ .

For the sketch in Figure 3-5, the design velocity of the sketch entity end-points were determined with respect to each parameter  $\mathcal{P} \in \{d_{21}, d_{22}, d_{23}, d_{25}, \dots, d_{31}\}$ . The resulting design velocity for each case are shown in Figures 3-6 through 3-10, where the relevant

vector magnitudes are scaled for each individual sub-figure separately. The design velocity information shown in these Figures gives indication of which sketch entities are driven by a dimension and likewise which *feature* surfaces will depend on that parameter. As a result of this methodology for differentiating sketches, the design velocity along the sketch entities themselves can now be determined.



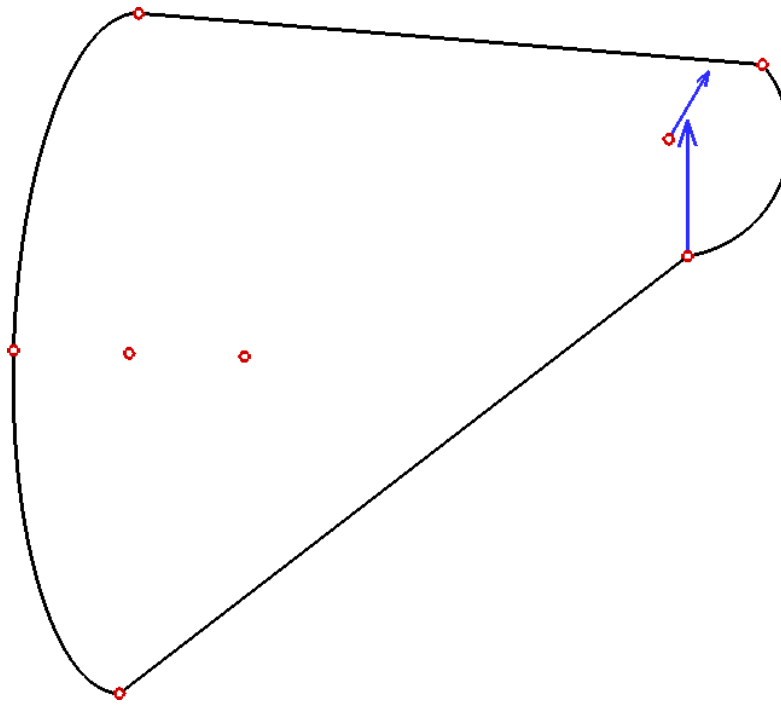
(a)  $\partial \mathbf{x} / \partial d_{21}$



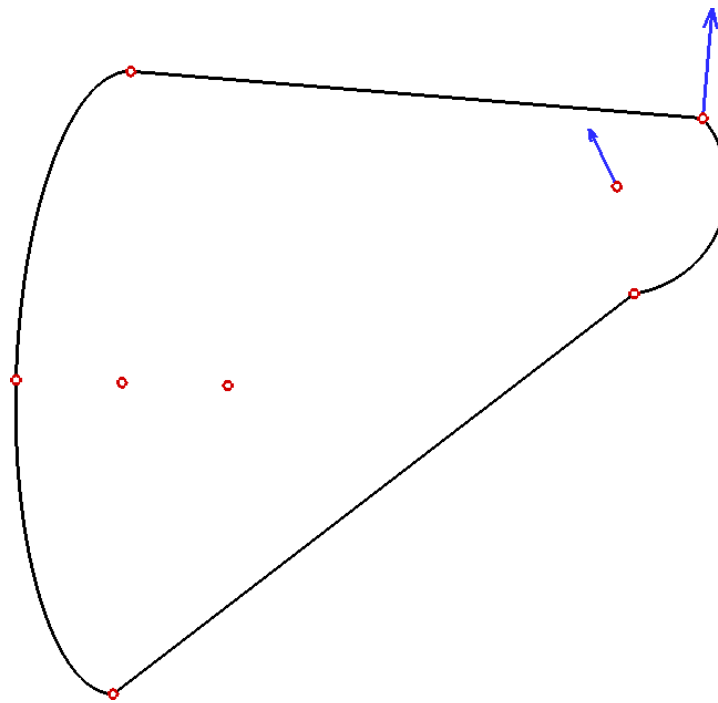
(b)  $\partial \mathbf{x} / \partial d_{22}$

Figure 3-6: The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.



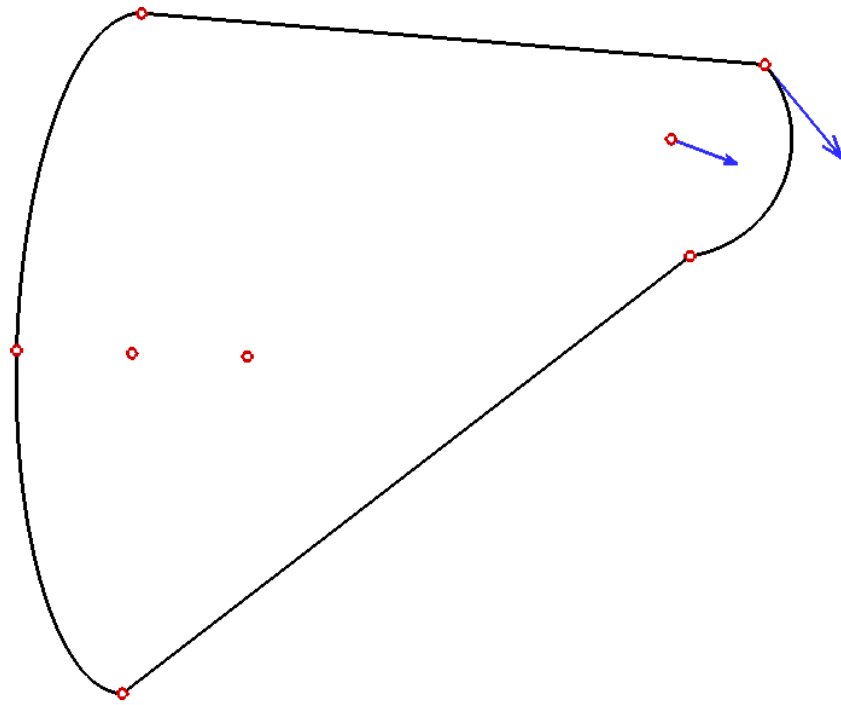


(a)  $\partial \mathbf{x} / \partial d_{23}$

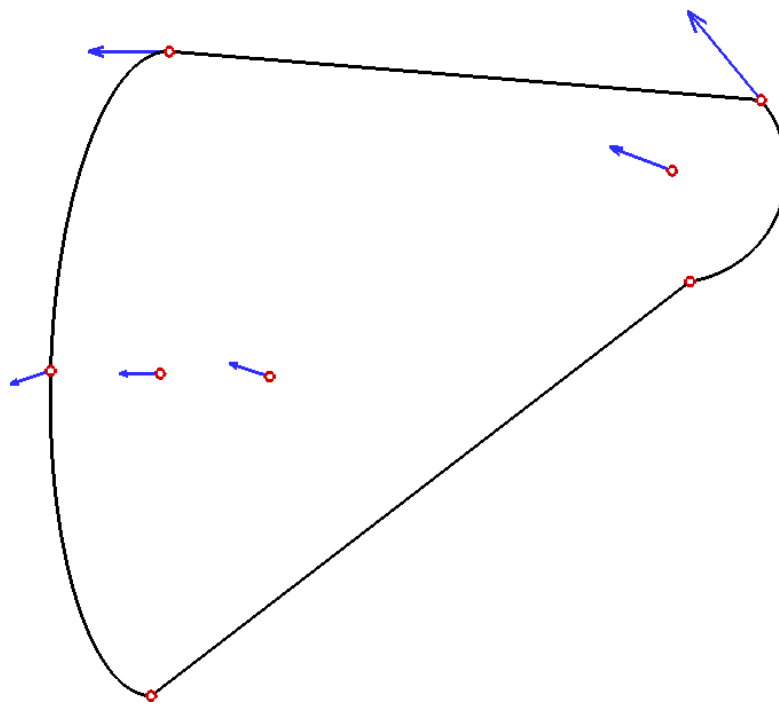


(b)  $\partial \mathbf{x} / \partial d_{25}$

Figure 3-7: The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.

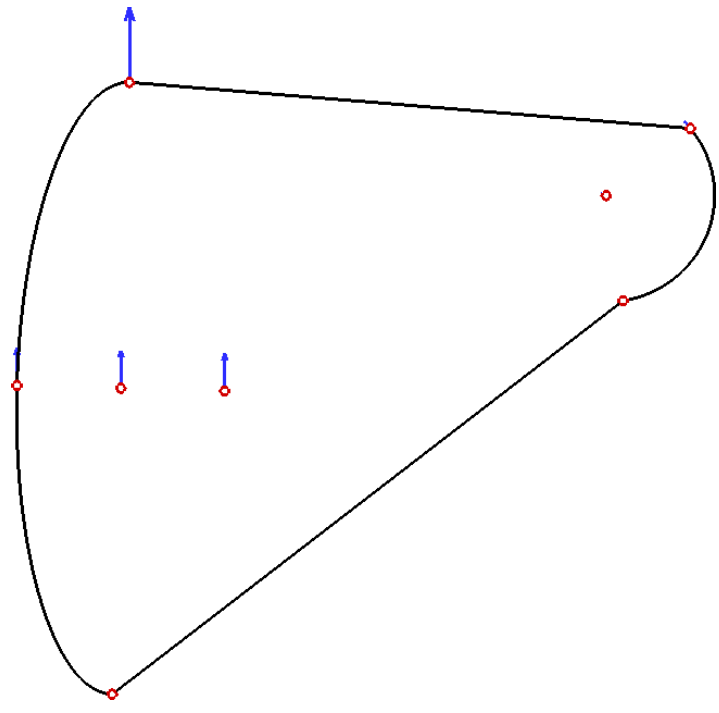


(a)  $\partial \mathbf{x} / \partial d_{26}$

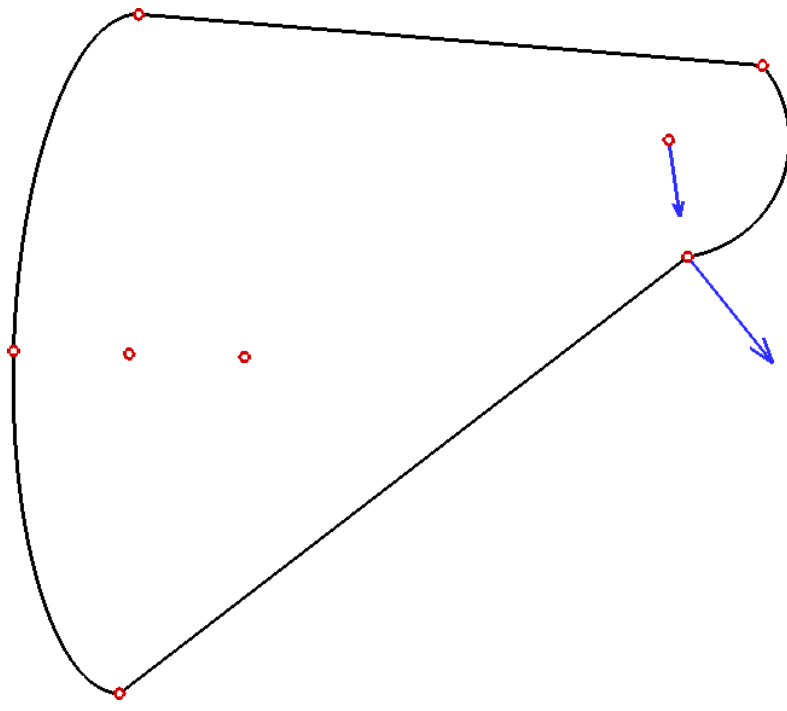


(b)  $\partial \mathbf{x} / \partial d_{27}$

Figure 3-8: The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.

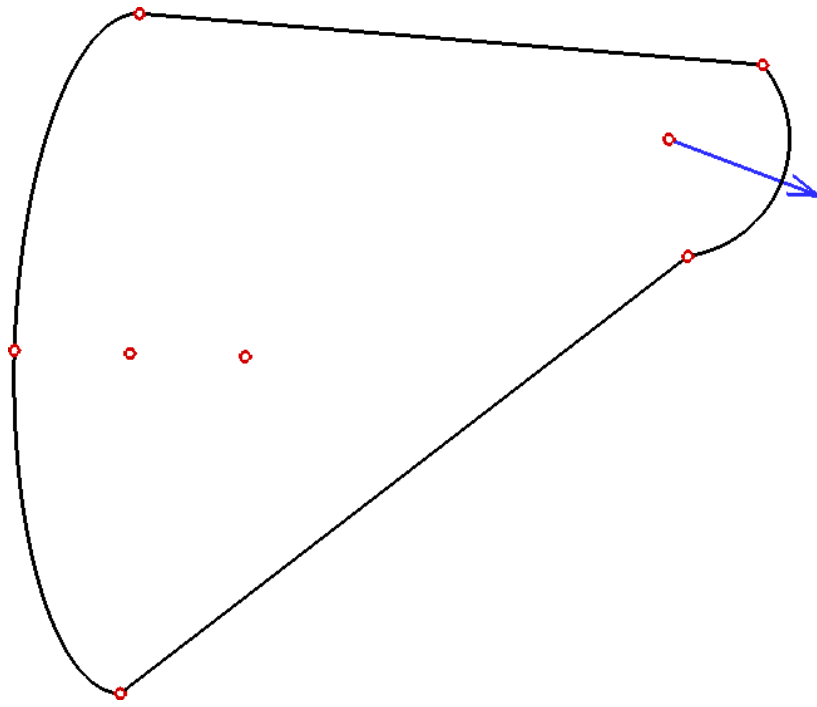


(a)  $\partial \mathbf{x} / \partial d_{28}$

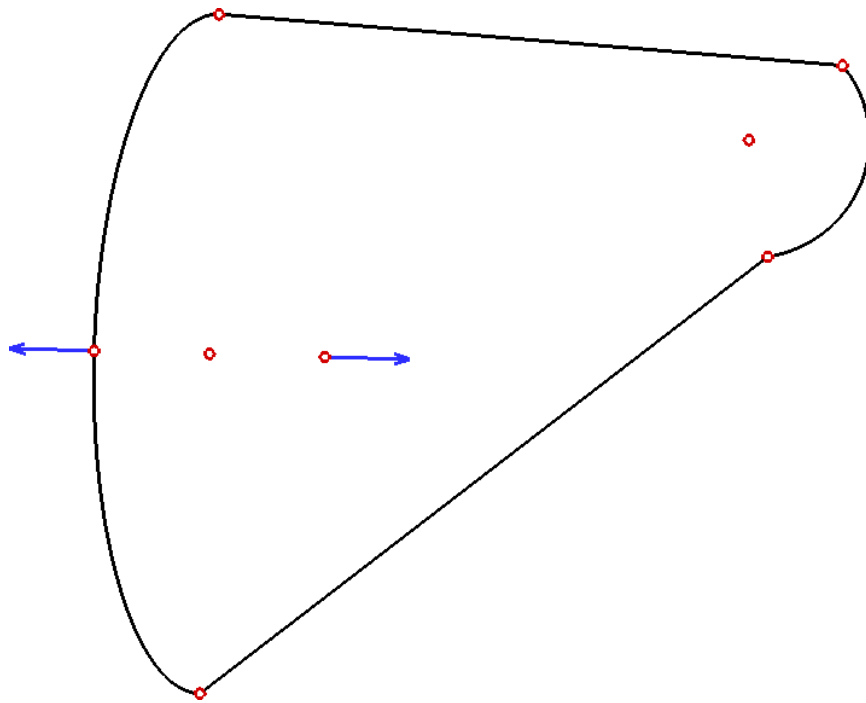


(b)  $\partial \mathbf{x} / \partial d_{29}$

Figure 3-9: The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.



(a)  $\partial \mathbf{x} / \partial d_{30}$



(b)  $\partial \mathbf{x} / \partial d_{31}$

Figure 3-10: The design velocity of sketch end-points from Figure 3-5 are shown with respect to each driving parameter in the sketch.

### 3.2.3 Design Velocity Along Sketch Entities

The parameterization of each sketch entity is required in order to find the analytic design velocity along its geometry. It is unclear if each CAD system geometry kernel utilizes a unique parameterization for the sketch entities it supports. Therefore, until other CAD systems are tested, the parameterizations shown here are only verified for 2D sketches created in the SolidWorks CAD system.

#### Line Segments

For a given  $(x_{\min}, y_{\min})$  and  $(x_{\max}, y_{\max})$  as end-points to a line segment, a running parameter  $t \in [t_{\min}, t_{\max}]$  is defined as

$$\begin{aligned} t_{\min} &= 0 \\ t_{\max} &= \sqrt{(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2}. \end{aligned} \quad (3.9)$$

Thus, for a given  $t \in [t_{\min}, t_{\max}]$ , a position  $\mathbf{r} \in \mathbb{R}^2$  along the line segment satisfies

$$\mathbf{r} = \begin{bmatrix} \left( \frac{t_{\max} - t}{t_{\max} - t_{\min}} \right) x_{\min} + \left( \frac{t - t_{\min}}{t_{\max} - t_{\min}} \right) x_{\max} \\ \left( \frac{t_{\max} - t}{t_{\max} - t_{\min}} \right) y_{\min} + \left( \frac{t - t_{\min}}{t_{\max} - t_{\min}} \right) y_{\max} \end{bmatrix} \quad (3.10)$$

The design velocity along the line segment with respect to a parameter  $\mathcal{P}$  in the sketch then becomes:

$$\frac{\partial \mathbf{r}}{\partial \mathcal{P}} = \begin{bmatrix} \left( \frac{t_{\max} - t}{t_{\max} - t_{\min}} \right) \frac{\partial x_{\min}}{\partial \mathcal{P}} + \left( \frac{t - t_{\min}}{t_{\max} - t_{\min}} \right) \frac{\partial x_{\max}}{\partial \mathcal{P}} \\ \left( \frac{t_{\max} - t}{t_{\max} - t_{\min}} \right) \frac{\partial y_{\min}}{\partial \mathcal{P}} + \left( \frac{t - t_{\min}}{t_{\max} - t_{\min}} \right) \frac{\partial y_{\max}}{\partial \mathcal{P}} \end{bmatrix}, \quad (3.11)$$

where the derivatives  $\partial x_{\min}/\partial \mathcal{P}$ ,  $\partial x_{\max}/\partial \mathcal{P}$ ,  $\partial y_{\min}/\partial \mathcal{P}$  and  $\partial y_{\max}/\partial \mathcal{P}$  are determined by the end-point geometry gradients resulting from solving (3.7).

#### Circular-Arc Segments

A circular-arc segment is determined to be parameterized between two known points  $(x_{\min}, y_{\min})$  and  $(x_{\max}, y_{\max})$  with a given center-point  $(h, k)$ . The radius for such a circular-arc is de-

defined as

$$R = \frac{|\mathbf{s}_{\min}| + |\mathbf{s}_{\max}|}{2} \quad (3.12)$$

when  $\mathbf{s}_{\min} = [x_{\min} - h, y_{\min} - k]$  and  $\mathbf{s}_{\max} = [x_{\max} - h, y_{\max} - k]$ . The parameterization for running parameter  $t \in [t_{\min}, t_{\max}]$  is found as

$$\mathbf{r} = \mathbf{A}^{-1}(\mathbf{F} + \mathbf{A}\mathbf{C}), \quad (3.13)$$

where

$$\mathbf{A} = \begin{bmatrix} (x_{\min} - h) & (y_{\min} - k) \\ (x_{\max} - h) & (y_{\max} - k) \end{bmatrix}, \mathbf{F} = \begin{bmatrix} R^2 \cos(t - t_{\min}) \\ R^2 \cos(t_{\max} - t) \end{bmatrix}, \mathbf{C} = \begin{bmatrix} h \\ k \end{bmatrix} \quad (3.14)$$

and (using  $-\mathbf{e}_1 = [-1, 0]$ )

$$t_{\min} = \cos^{-1} \left( \frac{-\mathbf{e}_1 \cdot \mathbf{s}_{\min}}{|\mathbf{s}_{\min}|} \right) \quad (3.15)$$

$$t_{\max} = t_{\min} + d_{30} \frac{\pi}{180}. \quad (3.16)$$

For this parameterization, the geometry gradient requires differentiation of each component in (3.13), yielding

$$\frac{\partial \mathbf{r}}{\partial \mathcal{P}} = \mathbf{A}^{-1} \left( \frac{\partial \mathbf{F}}{\partial \mathcal{P}} + \frac{\partial \mathbf{A}}{\partial \mathcal{P}} (\mathbf{C} - \mathbf{X}) \right) + \frac{\partial \mathbf{C}}{\partial \mathcal{P}}. \quad (3.17)$$

The terms involved in (3.17) are defined as:

$$\frac{\partial \mathbf{F}}{\partial \mathcal{P}} = \begin{bmatrix} 2R \frac{\partial R}{\partial \mathcal{P}} \cos(t - t_{\min}) - R^2 \sin(t - t_{\min}) \left( \frac{\partial t}{\partial \mathcal{P}} - \frac{\partial t_{\min}}{\partial \mathcal{P}} \right) \\ 2R \frac{\partial R}{\partial \mathcal{P}} \cos(t_{\max} - t) - R^2 \sin(t_{\max} - t) \left( \frac{\partial t_{\max}}{\partial \mathcal{P}} - \frac{\partial t}{\partial \mathcal{P}} \right) \end{bmatrix},$$

$$\frac{\partial R}{\partial \mathcal{P}} = \frac{1}{2} \left( \frac{\partial |\mathbf{s}_{\min}|}{\partial \mathcal{P}} + \frac{\partial |\mathbf{s}_{\max}|}{\partial \mathcal{P}} \right),$$

$$\frac{\partial |\mathbf{s}_{\min}|}{\partial \mathcal{P}} = \frac{1}{|\mathbf{s}_{\min}|} \left[ (x_{\min} - h) \left( \frac{\partial x_{\min}}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}} \right) + (y_{\min} - k) \left( \frac{\partial y_{\min}}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}} \right) \right],$$

$$\frac{\partial |\mathbf{s}_{\max}|}{\partial \mathcal{P}} = \frac{1}{|\mathbf{s}_{\max}|} \left[ (x_{\max} - h) \left( \frac{\partial x_{\max}}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}} \right) + (y_{\max} - k) \left( \frac{\partial y_{\max}}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}} \right) \right],$$

and

$$\frac{\partial t_{\min}}{\partial \mathcal{P}} = \frac{-1}{|\mathbf{s}_{\min}|^2} \left( \frac{\left( -\mathbf{e}_1 \cdot \frac{\partial \mathbf{s}_{\min}}{\partial \mathcal{P}} \right) |\mathbf{s}_{\min}| - (-\mathbf{e}_1 \cdot \mathbf{s}_{\min}) \frac{\partial |\mathbf{s}_{\min}|}{\partial \mathcal{P}}}{\sqrt{1 - \left( \frac{-\mathbf{e}_1 \cdot \mathbf{s}_{\min}}{|\mathbf{s}_{\min}|} \right)^2}} \right),$$

$$\frac{\partial \mathbf{s}_{\min}}{\partial \mathcal{P}} = \left[ \frac{\partial x_{\min}}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}}, \frac{\partial y_{\min}}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}} \right]^T,$$

$$\frac{\partial t_{\max}}{\partial \mathcal{P}} = \frac{\partial t_{\min}}{\partial \mathcal{P}} + \frac{\partial d_{30}}{\partial \mathcal{P}},$$

$$\frac{\partial d_{30}}{\partial \mathcal{P}} = \begin{cases} 1, & \mathcal{P} = d_{30} \\ 0, & \text{otherwise} \end{cases}$$

The term  $\partial t / \partial \mathcal{P}$  is typically considered ambiguous because it implies a change in the parameterization of a curve. When arc-length is used to parameterize a curve, the running parameter is defined as

$$t = t_{\min} + (t_{\max} - t_{\min})\eta \quad (3.18)$$

for  $\eta \in [0, 1]$ . Here  $\eta$  serves as a percentage between  $t_{\min}$  and  $t_{\max}$ , thus it is not a function of a parameter  $\mathcal{P}$ . This removes ambiguity in finding  $\partial t / \partial \mathcal{P}$  when a curve length depends on a parameter  $\mathcal{P}$ , thus giving the explicit expression

$$\frac{\partial t}{\partial \mathcal{P}} = \frac{\partial t_{\min}}{\partial \mathcal{P}} + \left( \frac{\partial t_{\max}}{\partial \mathcal{P}} - \frac{\partial t_{\min}}{\partial \mathcal{P}} \right) \eta. \quad (3.19)$$

### Semi-Ellipse Segments

The semi-ellipse segment of minor-axis  $a$  and major-axis  $b$  is found parameterized with a running parameter  $t \in [t_{\min}, t_{\max}]$  in the context of two quadrant points with coordinates  $(x_1, y_1)$  and  $(x_2, y_2)$  and center-point coordinates  $(h, k)$ . The calculation of  $\mathbf{r} \in \mathbb{R}^2$  along such a semi-ellipse is determined using

$$a = \sqrt{(x_1 - h)^2 + (y_1 - k)^2}$$

$$b = \sqrt{(x_2 - h)^2 + (y_2 - k)^2} \quad (3.20)$$

and

$$\begin{aligned}x' &= -a \cos(\pi/2 - t) \\y' &= -b \sin(\pi/2 - t).\end{aligned}\tag{3.21}$$

In order for the ellipse to be oriented properly between two points across its major-axis (in the case of Figure 3-5 this corresponds to  $\mathbf{r}_{21}$  and  $\mathbf{r}_{23}$ ), the ellipse must be rotated about the angle  $\theta$  between the sketch unit vector  $\mathbf{e}_2 = [0, 1]^T$  and  $\mathbf{r}_{2hk} = [x_2 - h, y_2 - k]^T$  to be coincident to both  $\mathbf{r}_{21}$  and  $\mathbf{r}_{23}$ :

$$\theta = -\cos^{-1}\left(\frac{[0, 1]^T \cdot \mathbf{r}_{2hk}}{|\mathbf{r}_{2hk}|}\right).$$

This results in

$$\mathbf{r} = \begin{bmatrix} h + x' \cos(\theta) - y' \sin(\theta) \\ k + x' \sin(\theta) + y' \cos(\theta) \end{bmatrix}.\tag{3.22}$$

The design velocity along the semi-ellipse involves differentiating each aspect of the parameterized segment and using the chain-rule. This approach results in

$$\frac{\partial \mathbf{r}}{\partial \mathcal{P}} = \begin{bmatrix} \frac{\partial h}{\partial \mathcal{P}} \\ \frac{\partial k}{\partial \mathcal{P}} \end{bmatrix} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \frac{\partial x'}{\partial \mathcal{P}} \\ \frac{\partial y'}{\partial \mathcal{P}} \end{bmatrix} + \frac{\partial \theta}{\partial \mathcal{P}} \begin{bmatrix} -\sin(\theta) & -\cos(\theta) \\ \cos(\theta) & -\sin(\theta) \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix},\tag{3.23}$$

where

$$\begin{aligned}\frac{\partial \theta}{\partial \mathcal{P}} &= \left( \frac{1}{\sqrt{1 - \left(\frac{[0, 1]^T \cdot \mathbf{r}_{2hk}}{|\mathbf{r}_{2hk}|}\right)^2}} \right) \left[ \frac{[0, 1]^T \cdot \left( \frac{\partial \mathbf{r}_{2hk}}{\partial \mathcal{P}} - \mathbf{r}_{2hk} \frac{\partial |\mathbf{r}_{2hk}|}{\partial \mathcal{P}} \right)}{|\mathbf{r}_{2hk}|^2} \right], \\ \frac{\partial \mathbf{r}_{2hk}}{\partial \mathcal{P}} &= \left[ \frac{\partial x_2}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}}, \frac{\partial y_2}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}} \right]^T, \\ \frac{\partial |\mathbf{r}_{2hk}|}{\partial \mathcal{P}} &= \left( \frac{x_2 - h}{|\mathbf{r}_{2hk}|} \right) \left( \frac{\partial x_2}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}} \right) + \left( \frac{y_2 - k}{|\mathbf{r}_{2hk}|} \right) \left( \frac{\partial y_2}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}} \right),\end{aligned}$$



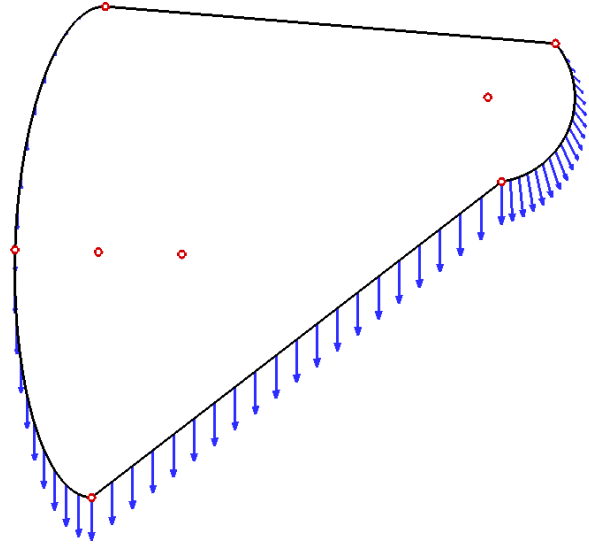
and

$$\begin{aligned}\frac{\partial x'}{\partial \mathcal{P}} &= -\frac{\partial a}{\partial \mathcal{P}} \cos(\pi/2 - t), \\ \frac{\partial y'}{\partial \mathcal{P}} &= -\frac{\partial b}{\partial \mathcal{P}} \sin(\pi/2 - t), \\ \frac{\partial a}{\partial \mathcal{P}} &= \left(\frac{x_1 - h}{a}\right) \left(\frac{\partial x_1}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}}\right) + \left(\frac{y_1 - k}{a}\right) \left(\frac{\partial y_1}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}}\right), \\ \frac{\partial b}{\partial \mathcal{P}} &= \left(\frac{x_2 - h}{b}\right) \left(\frac{\partial x_2}{\partial \mathcal{P}} - \frac{\partial h}{\partial \mathcal{P}}\right) + \left(\frac{y_2 - k}{b}\right) \left(\frac{\partial y_2}{\partial \mathcal{P}} - \frac{\partial k}{\partial \mathcal{P}}\right).\end{aligned}$$

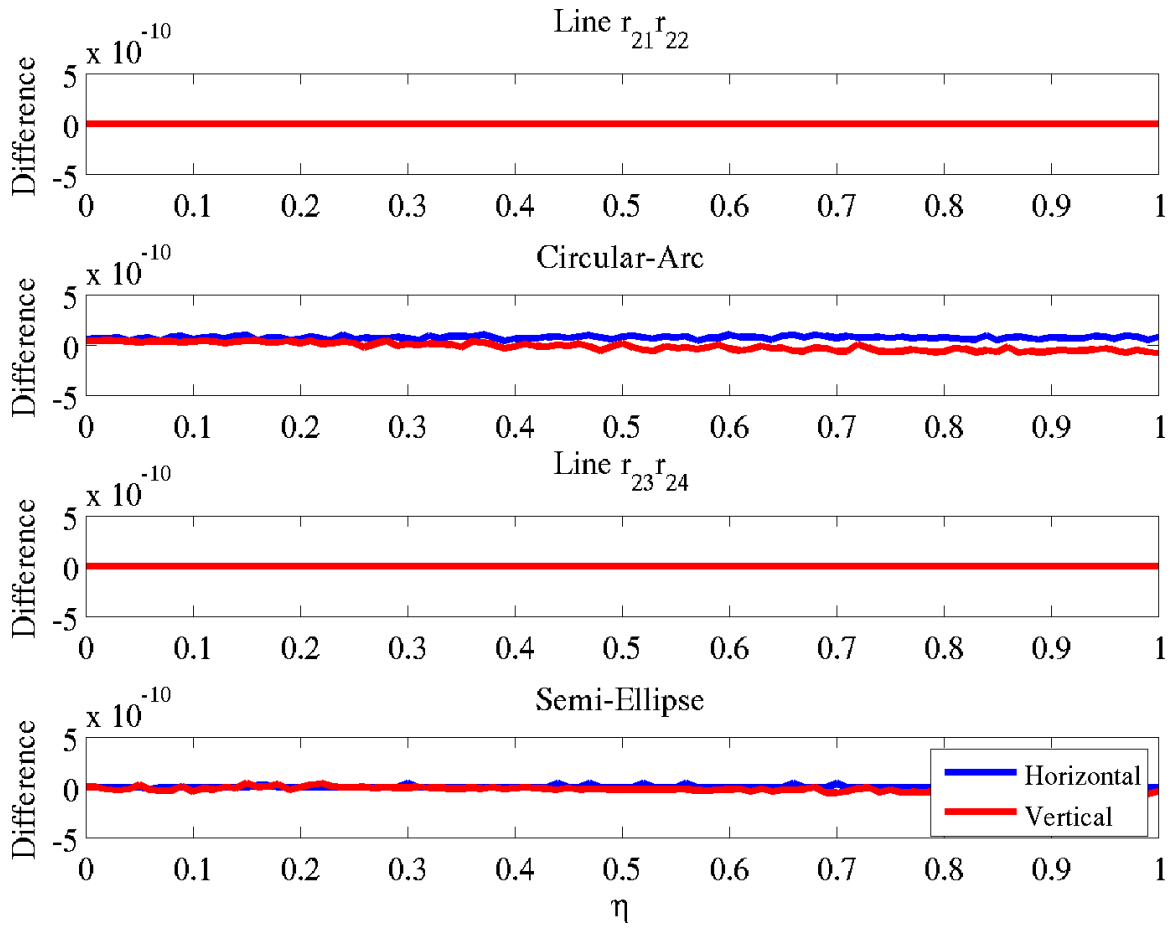
### 3.2.4 Validation Examples

With the design velocity at sketch end-points known, the design velocity *along* sketch entities is determined using the formulae obtained in the previous sections. The case of Figure 3-5 is again used for validation of the parameterization reverse-engineering discussed thus far and the validation of the sketch differentiation method that is presented.

Comparison between the design velocity obtained with the sketch differentiation method and finite-differencing of the SolidWorks CAD model is conducted for validation. Finite-differencing is done at various step-sizes ( $h \in [1.0 \times 10^{-4}, 1.0 \times 10^{-10}]$ ) by perturbing a given parameter, regenerating the model, then evaluating the coordinates  $\mathbf{r}(t)$  along each sketch segment (these segments appear as edges on the BRep of the model on the sketch plane because the model is a simple extrusion away from the sketch plane) at the *same* values of  $\eta \in [0, 1]$ , thus maintaining consistency of evaluation. The same  $\eta$  values are used with the sketch differentiation method. The relative offset between design velocity components  $\partial \mathbf{r} / \partial \mathcal{P}$  are compared in magnitude and direction for each approach to provide validation. The results for design velocities with respect to each parameter  $\mathcal{P} \in [d_{21}, d_{22}, d_{23}, d_{25}, \dots, d_{31}]$  are shown in Figures 3-11 through 3-20. In each case agreement between the two results are excellent.

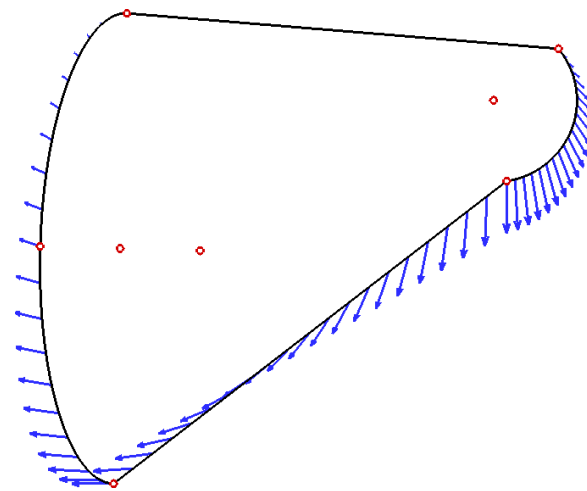


(a) Design Velocity

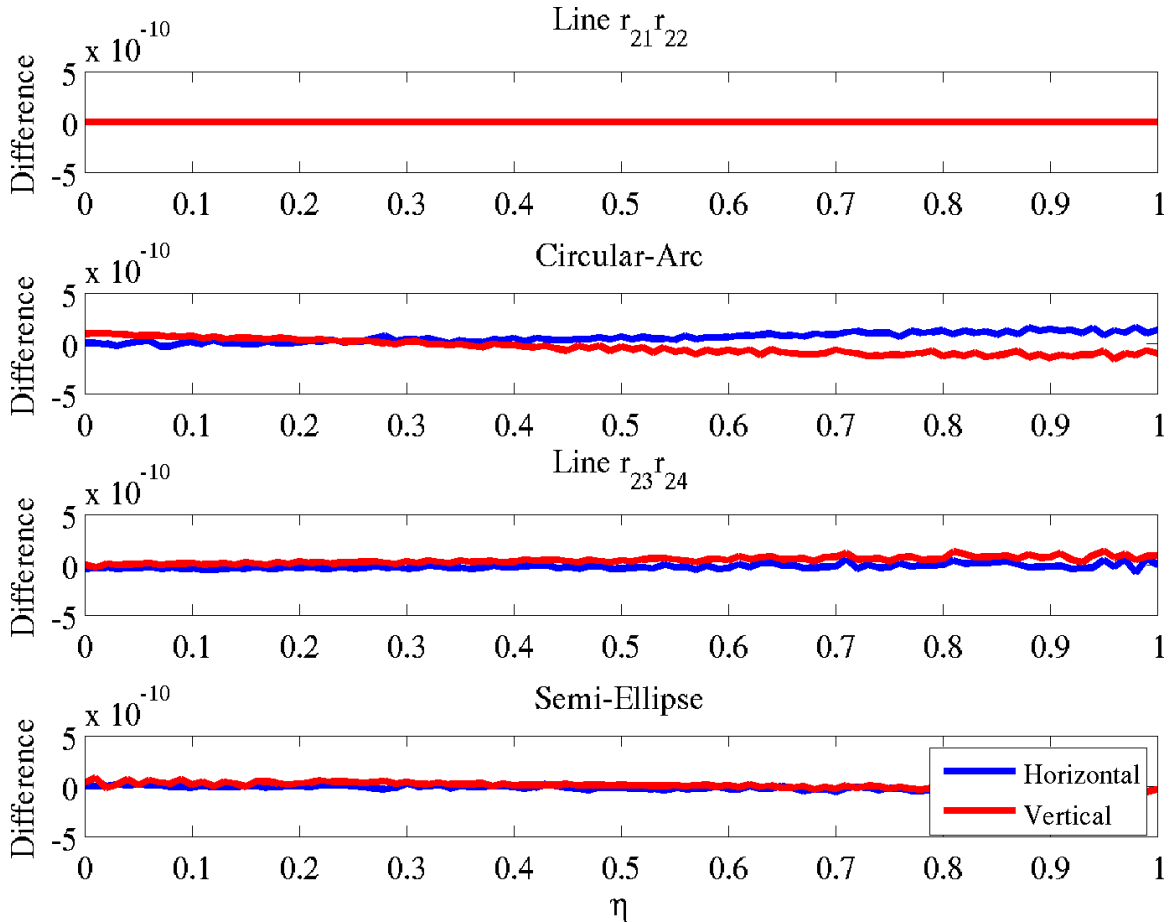


(b) Method Comparison

Figure 3-11: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{21}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.



(a) Design Velocity



(b) Method Comparison

Figure 3-12: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{22}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.

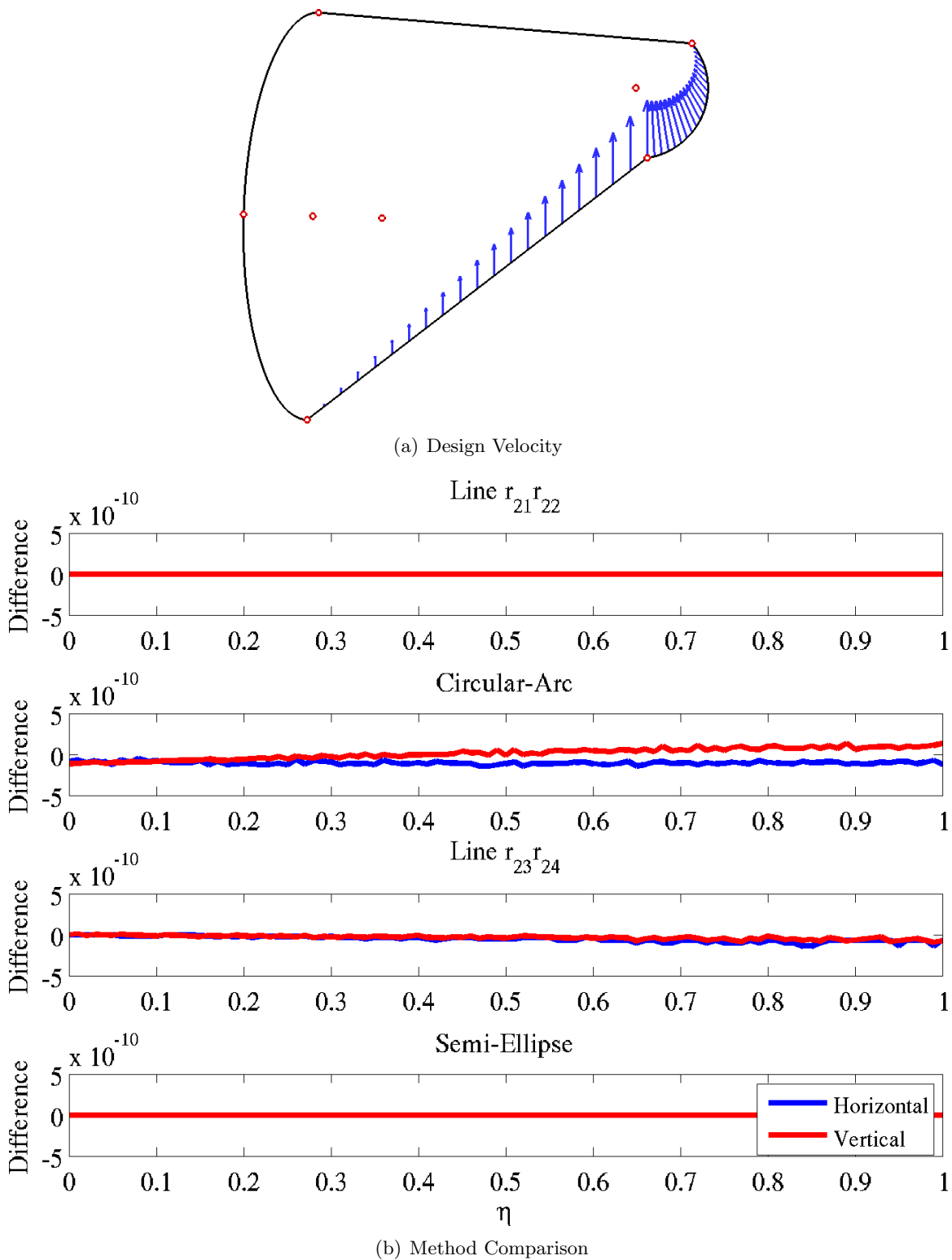


Figure 3-13: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{23}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.

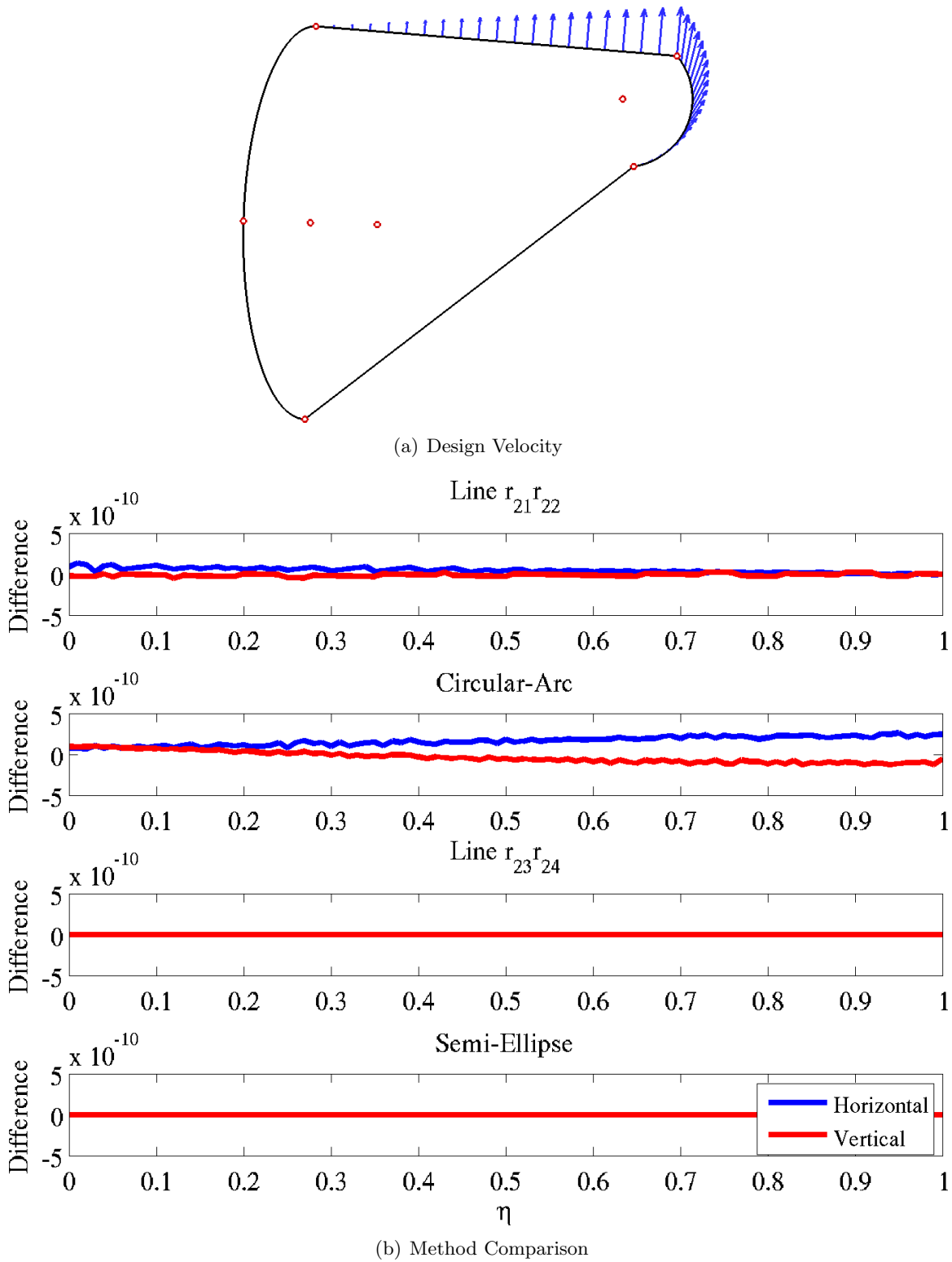


Figure 3-14: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{25}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.

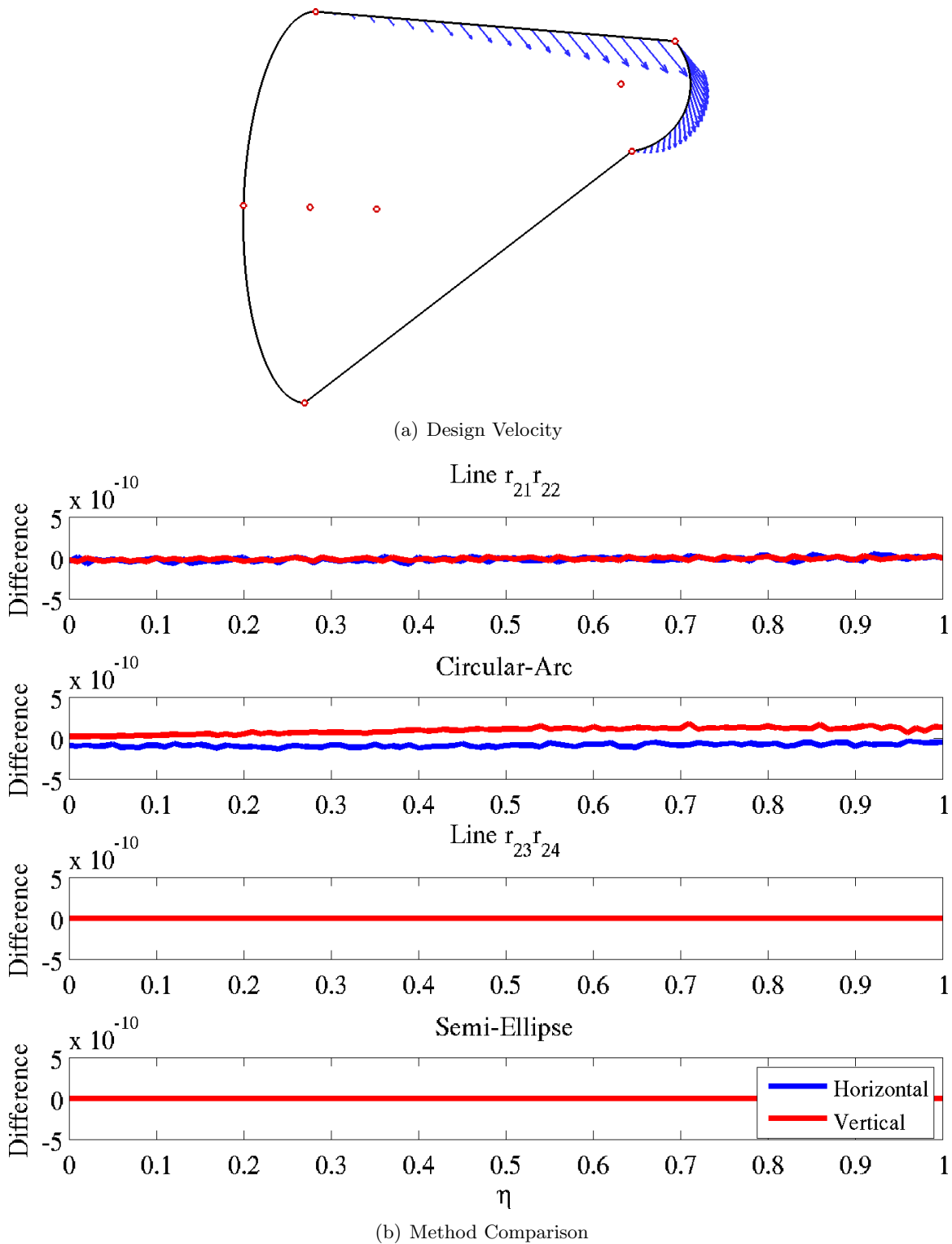
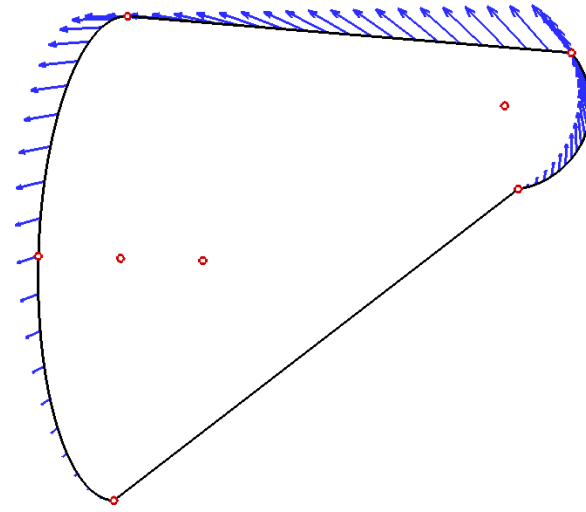
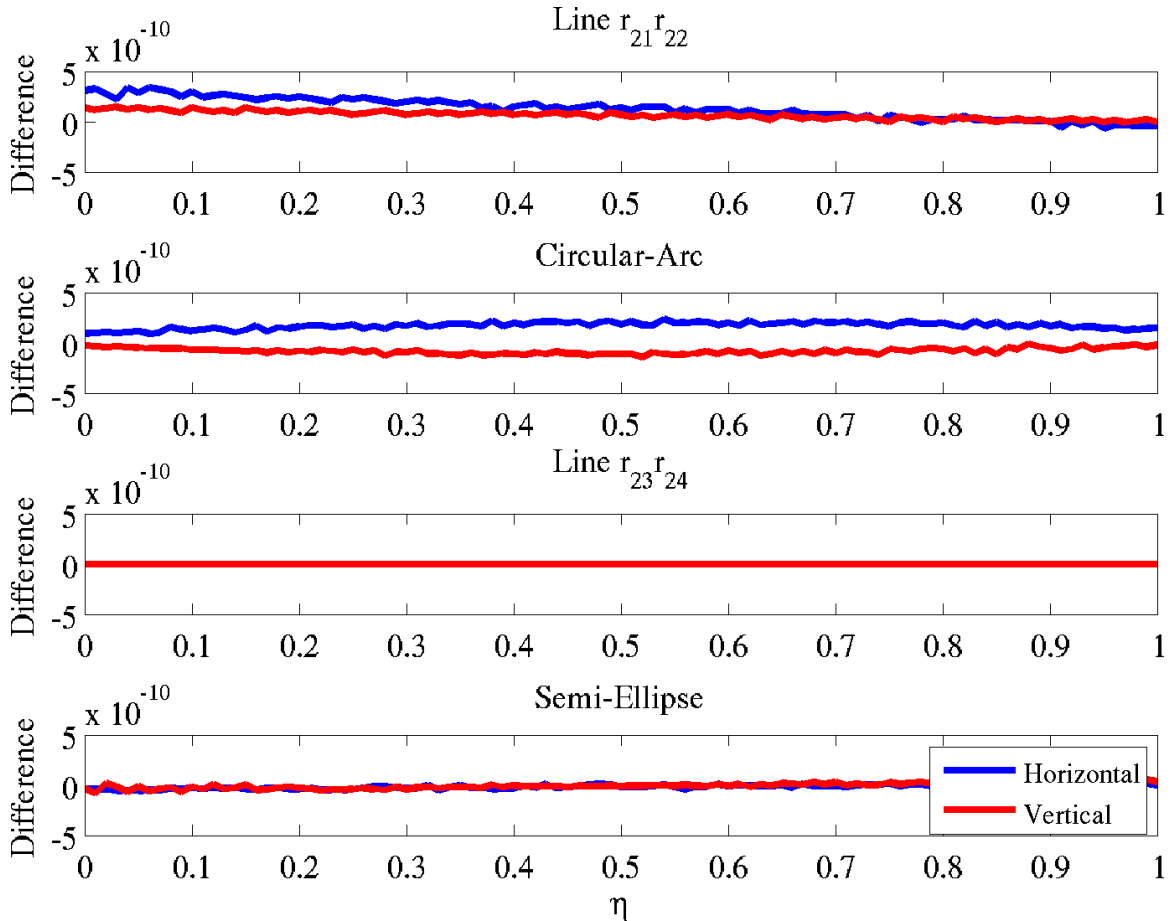


Figure 3-15: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{26}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.



(a) Design Velocity



(b) Method Comparison

Figure 3-16: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{27}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.

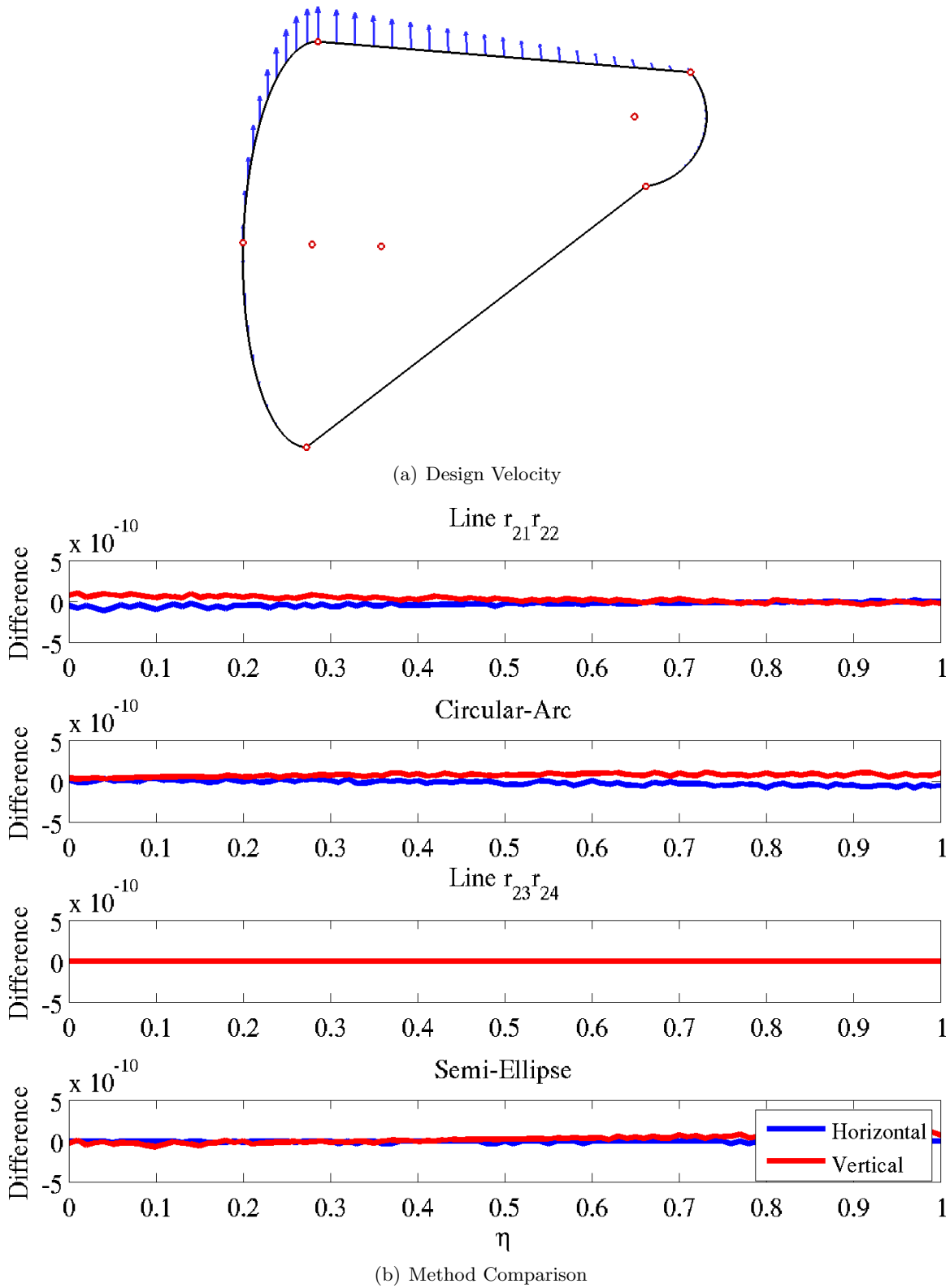
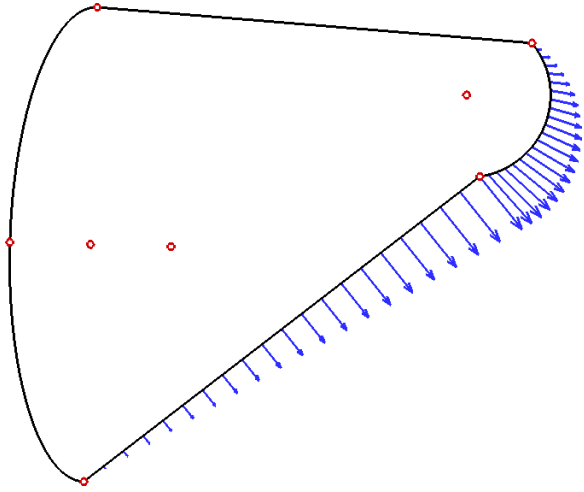
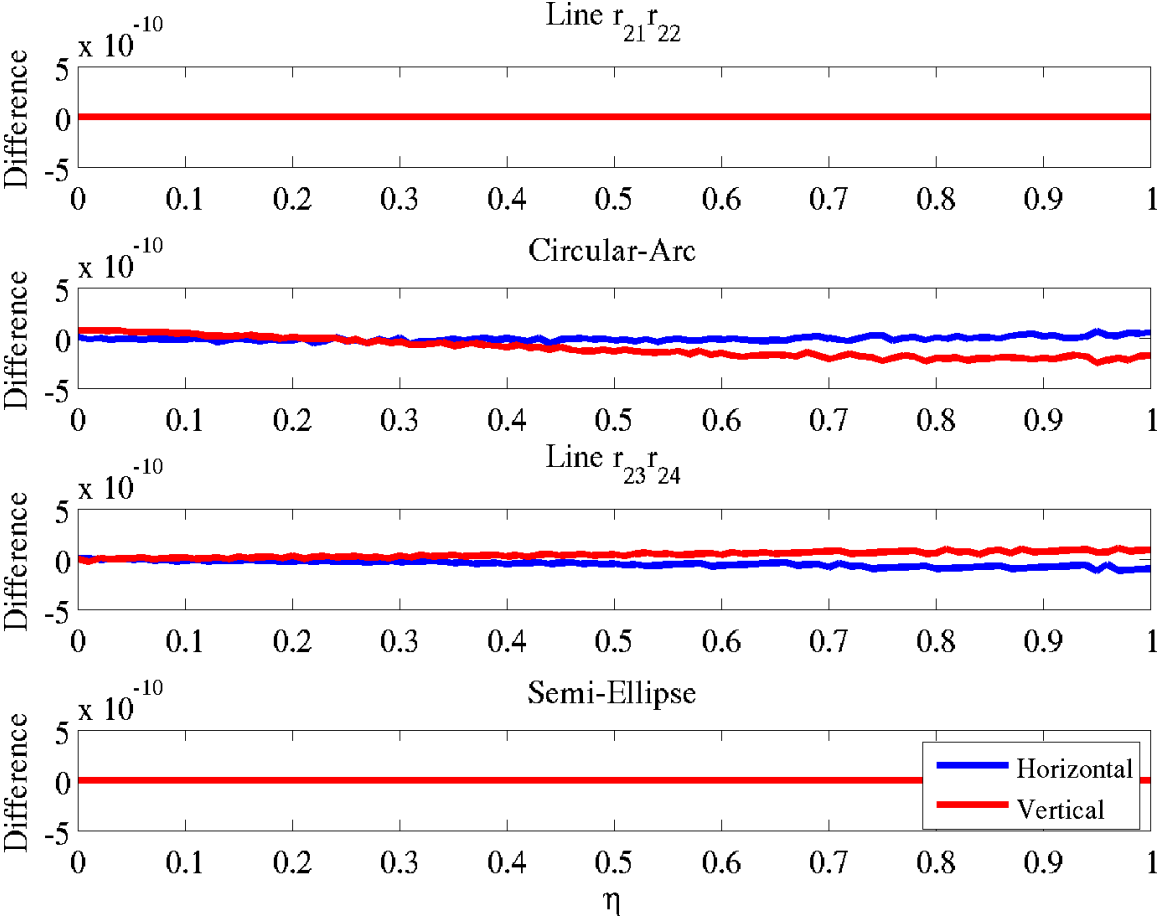


Figure 3-17: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{28}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.



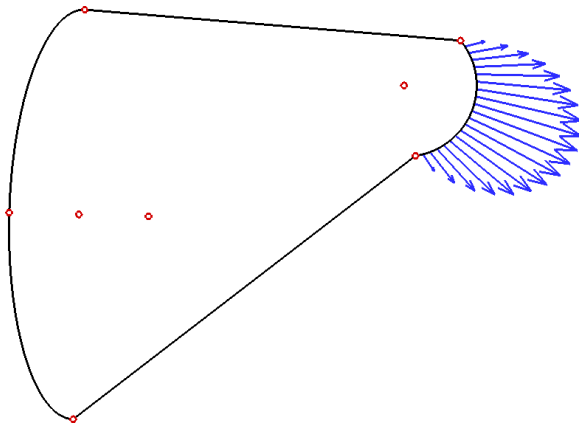


(a) Design Velocity

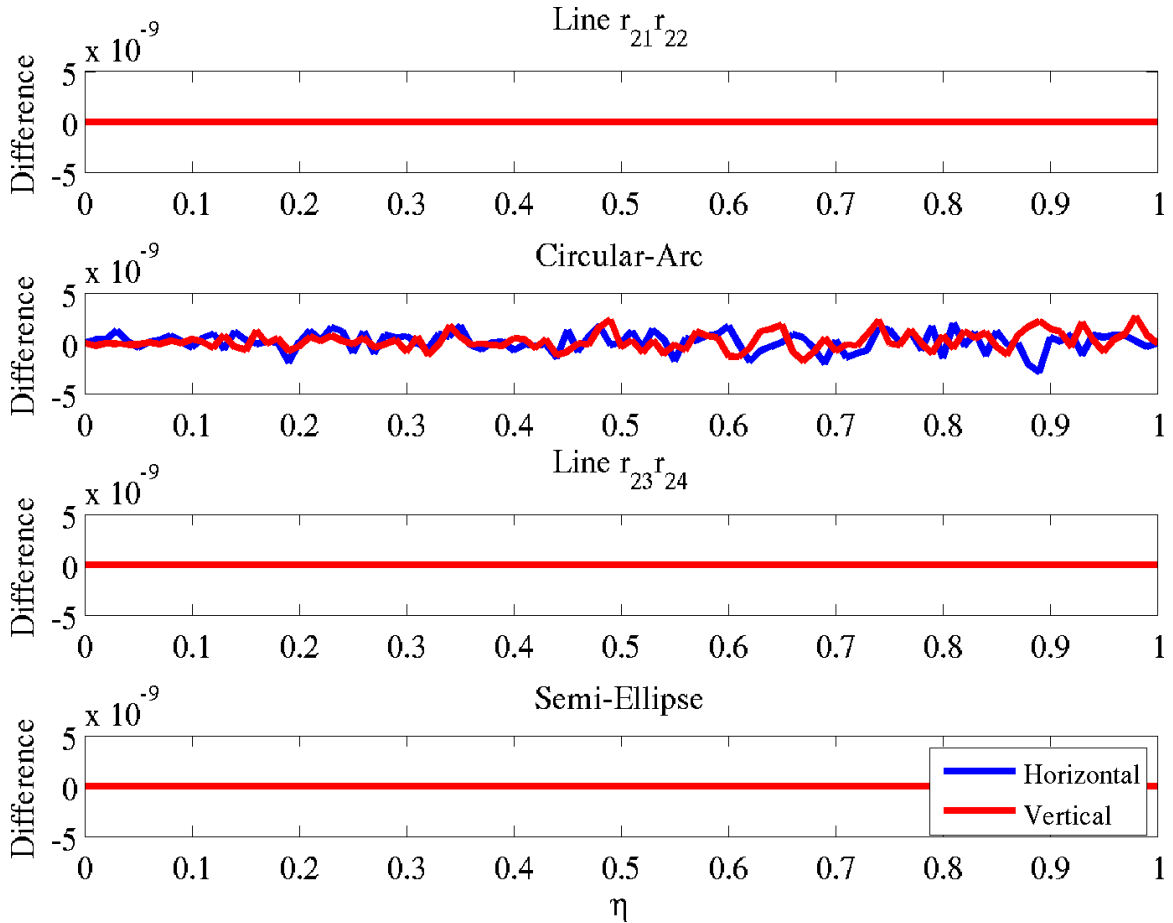


(b) Method Comparison

Figure 3-18: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{29}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.

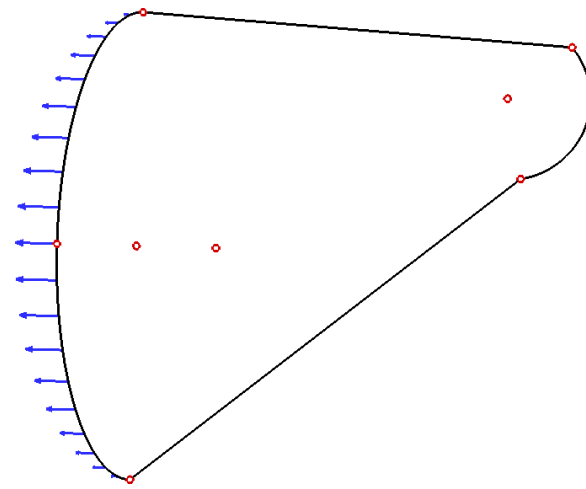


(a) Design Velocity

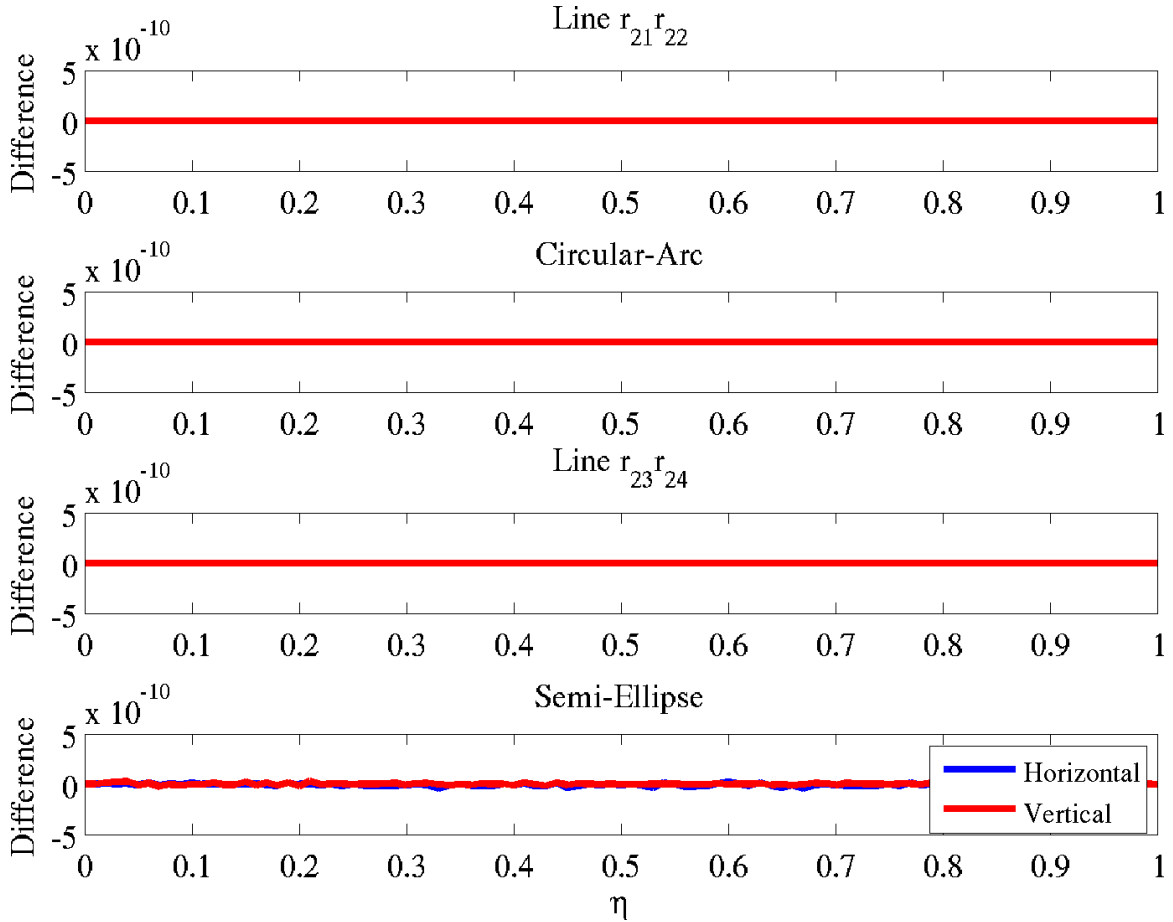


(b) Method Comparison

Figure 3-19: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{30}$  (step-size  $h = 1.0 \times 10^{-4}(\pi/180)$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.



(a) Design Velocity



(b) Method Comparison

Figure 3-20: (a) Design velocity vectors of active sketch *primitives* for  $\mathcal{P} = d_{31}$  (step-size  $h = 1.0 \times 10^{-4}$ ). (b) Offset in horizontal and vertical gradient components between finite-differencing and the sketch differentiation method.

### 3.3 Geometry Gradients of Extrude-Feature Surfaces

A simple extrusion *feature* creates a closed, solid volume of constant cross-section starting with a piecewise-continuous, closed chain of sketch entities. The resulting surfaces are parameterized by  $(u, v) \in \mathbb{R}^2$  coordinates defined within  $u \in [u_{\min}, u_{\max}]$  and  $v \in [v_{\min}, v_{\max}]$ . The  $u$ -direction is typically defined along the sketch entities, whereas the  $v$ -direction is typically defined linearly along the extrusion direction. After determining the associativity of sketch entities with sketch faces (e.g., line segments associate with planar surfaces, circular-arc segments associate with cylindrical surfaces), it is clear that the sketch endpoints correspond to  $v$ -isoparameter lines that bound resulting surfaces into faces. Furthermore, an isoparameter line in  $v$  along the extrude direction will usually maintain the *same* cross-section definition as the original sketch plane<sup>3</sup>.

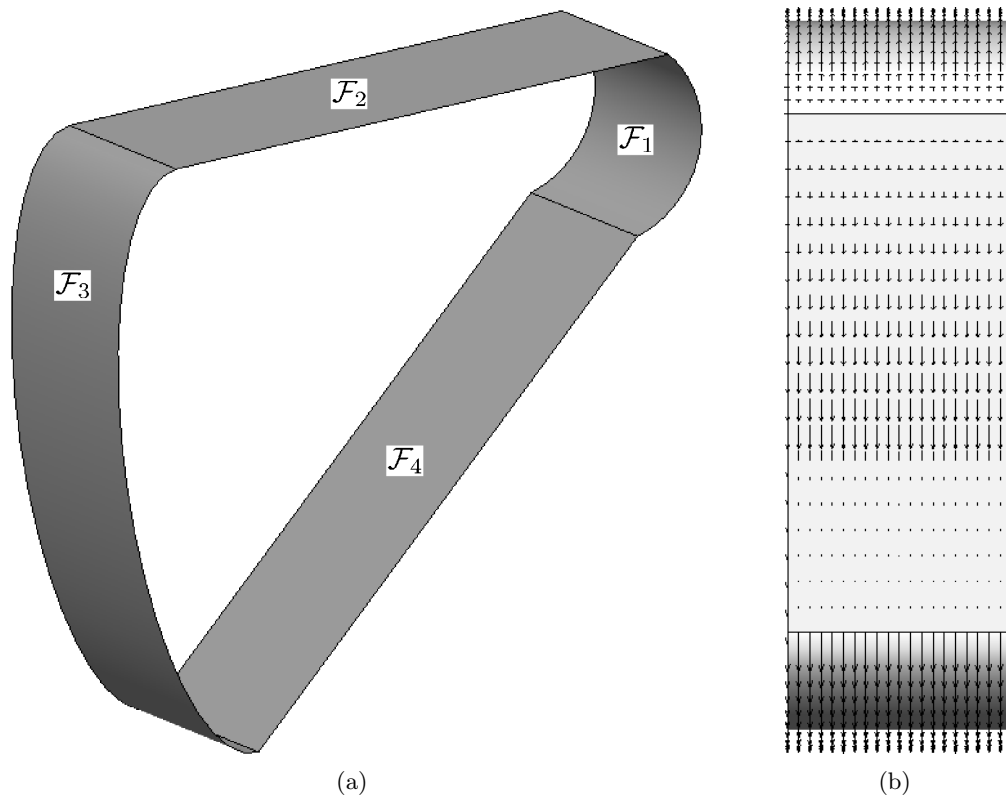


Figure 3-21: Face labels for the extruded sketch in Figure 3-5 are shown in (a) with no extrude-direction dependence of the design velocity field in (b).

<sup>3</sup>A subtlety arises when the CAD geometry kernel converts a sketch entity into a rational/non-rational B-spline curve and then extrudes, making the resulting surface a rational/non-rational B-spline surface rather than a canonical extruded surface of a given cross-section. The spline support points must be determined as sample points along the sketch entity in this case before (3.25) can be applied. The surface is then differentiated using the methods in Chapter 5.

As an example, the closed, piecewise continuous sketch in Figure 3-5 was extruded into the model geometry seen in Figure 3-21(a) (end-caps not shown). The model associativity begins with line *primitive*  $\mathbf{r}_{21}\mathbf{r}_{22}$  driving  $\mathcal{F}_2$ , circular-arc  $\mathbf{r}_{22}\mathbf{r}_{24}$  corresponding to  $\mathcal{F}_1$ , line *primitive*  $\mathbf{r}_{23}\mathbf{r}_{24}$  driving  $\mathcal{F}_4$  and semi-ellipse *primitive*  $\mathbf{r}_{21}\mathbf{r}_{23}$  associated with  $\mathcal{F}_3$ .

More formally, due to the nature of the extrude operation the sketch segment  $\mathbf{r}(t)$  associated with face  $\mathcal{F}$  on surface  $S(u, v)$  is equivalent to a  $v$ -isoparameter line of the surface  $S(u, v)$ . Thus

$$S(u, v)|_{v=\text{const}} \equiv \mathbf{r}(t). \quad (3.24)$$

This fact also implies that all parameters  $\mathcal{P}$  defined solely on sketch entity  $\mathbf{r}(t)$  *only* impact  $S(u, v)$  in the  $u$ -direction and are decoupled from the extrude-direction. Evidence of this was seen when finite-differencing the model (step-size  $1.0 \times 10^{-4}$ ) and observing design velocity fields decoupled from the extrude direction, as in Figure 3-21(b). This means that the design velocity field on  $\mathcal{F}$  is found once sketch differentiation is completed by setting

$$\left. \frac{\partial S(u, v)}{\partial \mathcal{P}} \right|_{v=\text{const}} \equiv \frac{\partial \mathbf{r}(t)}{\partial \mathcal{P}}, \quad (3.25)$$

which indicates the sketch gradient field is propagated unchanged in the extrude-direction. Only the parameter associated with the extrusion length will yield a design velocity field that is coupled to the extrude direction. For  $v = v_{\max}\eta$  and  $v_{\max} = L$ , the resulting design velocity becomes

$$\left. \frac{\partial S(u, v)}{\partial \mathcal{P}} \right|_{u=\text{const}} \equiv \eta. \quad (3.26)$$

The example extrude model from Figure 3-5 was differentiated against parameters  $\{d_{21}, d_{22}, d_{23}, d_{25}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{31}\}$  using the analytic approach and a centered-difference approach (step-size of  $h = 1.0 \times 10^{-4}$  for all parameters except  $d_{30}$ , where  $h = (1.0 \times 10^{-4})\pi/180$ ). Gradient results for both approaches are subtracted to show the difference in each gradient component, where the maximum difference is plotted in Figure 3-22; contour plots of the relative offset are shown in Appendix C with Figures C-1 through C-11, all of which demonstrate excellent agreement.

In the event that CAD tolerances vary and increase, the expected maximum difference will vary as well and agreement between analytic and finite-difference results will diminish. This is seen in Figure 3-23 as the sketch sensitivity “noise” propagates to the *feature* surfaces,

thereby increasing the maximum difference between analytic and finite-difference design velocity.

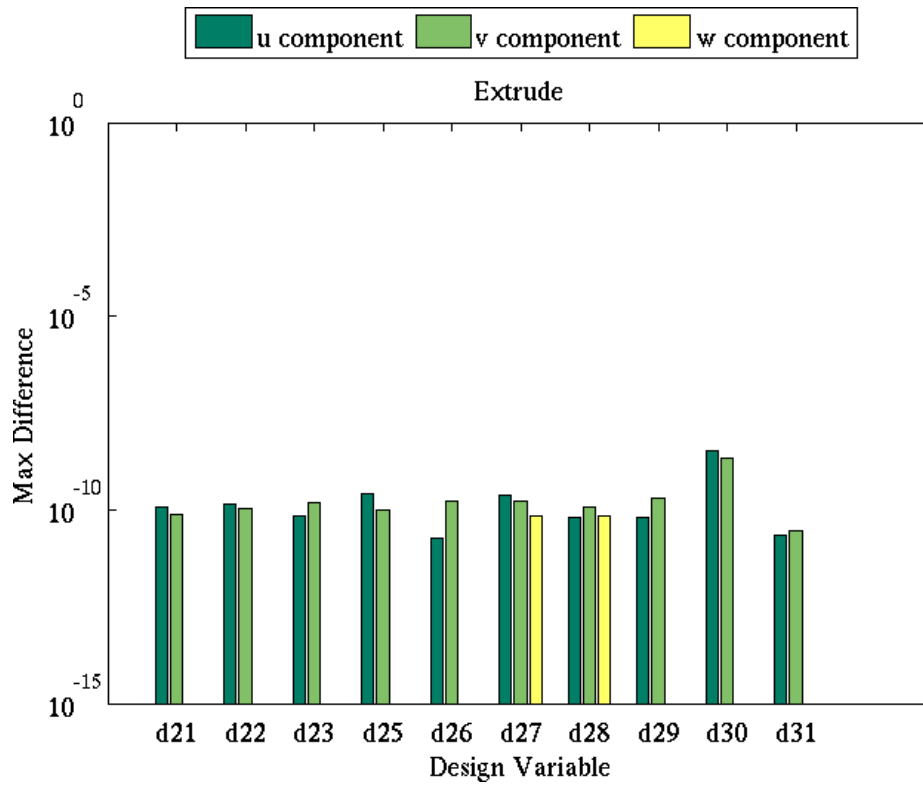
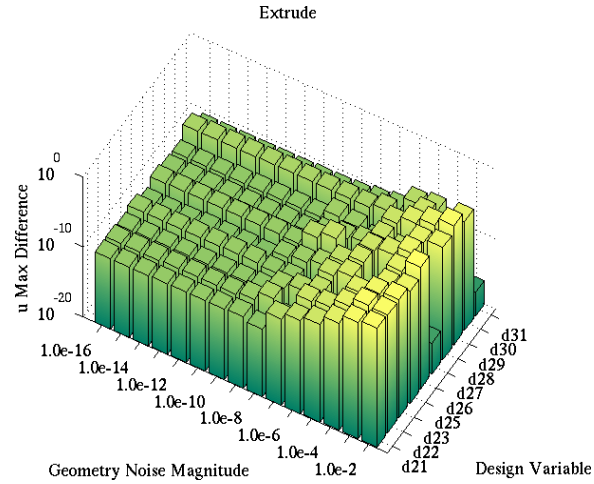
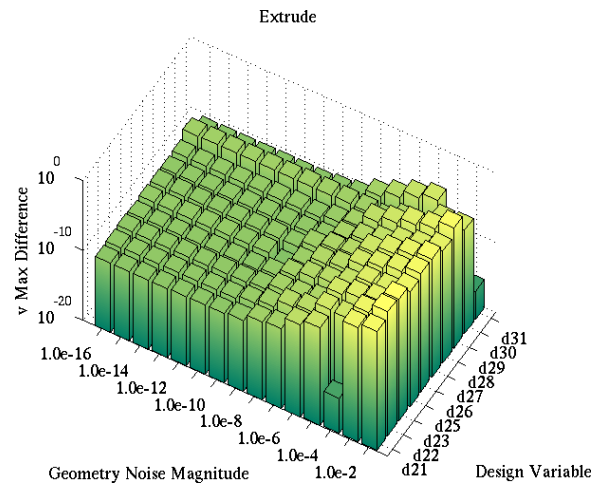


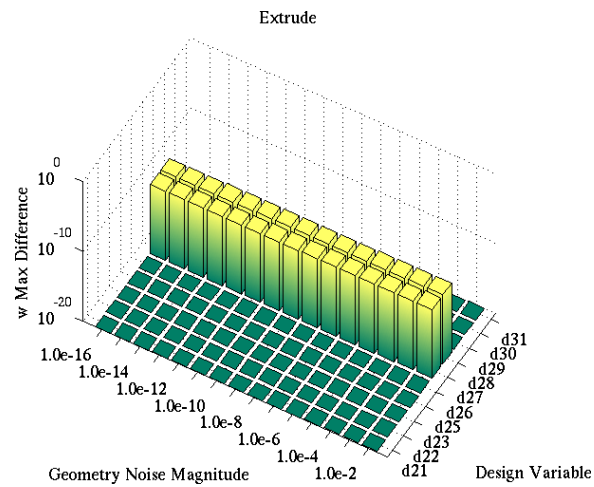
Figure 3-22: The maximum difference between the analytic and finite-difference geometry sensitivity (across all *feature* faces) is shown against the parameters in Figure 3-5.



(a)  $u$  component



(b)  $v$  component



(c)  $w$  component

Figure 3-23: The addition of geometry “noise” increases the maximum difference between analytic and finite-difference design velocity.

### 3.4 Geometry Gradients of Revolve-Feature Surfaces

The revolve *feature* is an extension of an extrude *feature* in that the piecewise-continuous, closed chain of sketch entities is revolved around a central-axis some distance  $r_d \in [r_{d,\min}, r_{d,\max}]$  from the sketch entities. Hence a revolve can be considered an extrusion wrapped around a central-axis. As in the extrude case, the sketch segments  $\mathbf{r}(t)$  associated with a face  $\mathcal{F}$  generate a surface-of-revolution  $S(u, v)$  parameterized by  $(u, v) \in \mathbb{R}^2$  coordinates defined within  $u \in [u_{\min}, u_{\max}]$  and  $v \in [v_{\min}, v_{\max}]$ . The  $v$ -direction may be defined along the sketch entities, whereas the  $u$ -direction may be defined along the revolve direction (note this is the *reverse* convention from the extrude case and all revolve surfaces might *not* follow a consistent convention). Furthermore, an isoparameter line in  $u$  along the revolved surfaces will usually maintain the *same* cross-section definition as the original sketch plane.

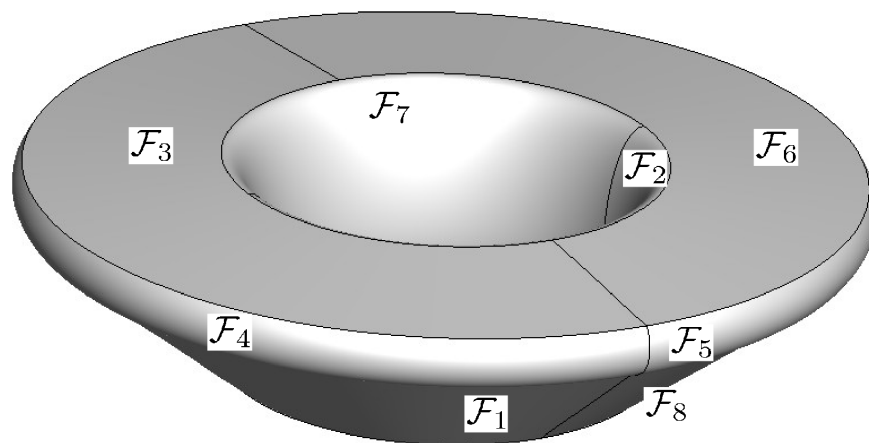


Figure 3-24: Face labels for the revolved sketch in Figure 3-5.

For example, by employing a revolve *feature* on the closed-sketch of four entities in Figure 3-5, a total of eight faces are created (each of the four periodic surfaces are split into two faces), as labeled in Figure 3-24. Line-entities  $\mathbf{r}_{21}\mathbf{r}_{22}$  and  $\mathbf{r}_{23}\mathbf{r}_{24}$  in the sketch result in faces  $\{\mathcal{F}_3, \mathcal{F}_6\}$  and  $\{\mathcal{F}_1, \mathcal{F}_8\}$ , respectively, from cone surfaces and the circular-arc  $\mathbf{r}_{22}\mathbf{r}_{24}$  results in faces  $\{\mathcal{F}_4, \mathcal{F}_5\}$  from torus surfaces. The semi-ellipse entity  $\mathbf{r}_{21}\mathbf{r}_{23}$ , though, results in faces  $\{\mathcal{F}_2, \mathcal{F}_7\}$  defined as rational B-spline surfaces<sup>4</sup>, which represent the semi-ellipse and approximate a surface-of-revolution. The cone and torus surfaces are expected outcomes from revolving the sketch in Figure 3-5. However, the rational B-spline surfaces generated

<sup>4</sup>This is known through the geometry kernel API by checking the control point weights on the surface. Weights are 1.0 when the surface is non-rational; however, this is not the case for rational B-spline surfaces, as seen in faces  $\{\mathcal{F}_2, \mathcal{F}_7\}$  of the revolve *feature*.



by the semi-ellipse entity are likely unexpected for a designer.

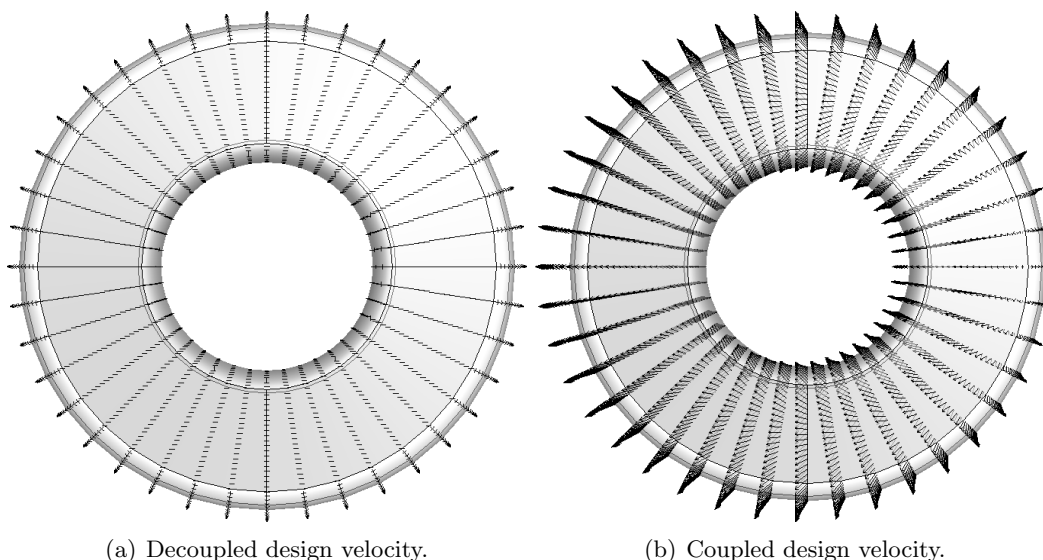


Figure 3-25: The design velocity with respect to  $d_{21}$  is decoupled to the revolve-direction in (a), whereas with respect to  $d_{22}$  the design velocity field is coupled from the revolve-direction in (b).

The sketch segment  $\mathbf{r}(t)$  associated with face  $\mathcal{F}$  on surface  $S(u, v)$  is equivalent to an  $u$ -isoparameter line of surface  $S(u, v)$ . Thus, the sketch entity associated with each face (where  $u$  is in the revolve direction) has

$$S(u, v)|_{u=\text{const}} \equiv \mathbf{r}(t). \quad (3.27)$$

In the extrude case, all parameters  $\mathcal{P}$  defined solely on sketch entity  $\mathbf{r}(t)$  *only* impact  $S(u, v)$  in the  $u$ -direction (recall that the  $u$ -direction was along the sketch entities in the extrude case) and are decoupled from the extrude-direction. For each parameter  $\{d_{21}, d_{22}, d_{23}, d_{25}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{31}\}$  in Figure 3-5 the decoupling is indeed true in the revolve case as well. Evidence of this becomes clear upon observation of the design velocity field from a finite-difference calculation (using a step-size of  $1.0 \times 10^{-4}$ ) of regenerated model geometry. This is expected due to the intended design motion designed into the model geometry. Figure 3-25 compares the design velocity fields obtained when an unexpected design motion is embedded in the model. The case in 3-25(a) shows design velocity vectors oriented along  $v$ -isoparameter lines, yet in 3-25(b) the velocity vectors show a revolve-direction dependence. The situation in 3-25(b) occurs when the axis-of-rotation is

dimensioned with respect to one of the active sketch *primitives* being differentiated. This creates a design velocity field that contains a translation and rotation effect of the model geometry (e.g., the individual surfaces can have a translation in their relative axes and origin, as was the case in 3-25(b)). This can be an unintended aspect of a model design trajectory due to poor model construction that might only be unveiled by this visual analysis of design velocities.

For the parameters  $\{d_{21}, d_{22}, d_{23}, d_{25}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{31}\}$  this means that the design velocity field on their associated faces is found once sketch differentiation is completed by setting

$$\left. \frac{\partial S(u, v)}{\partial \mathcal{P}} \right|_{u=\text{const}} \equiv \frac{\partial \mathbf{r}(t)}{\partial \mathcal{P}}, \quad (3.28)$$

which implies rotating the sketch gradient field unchanged around the revolve-axis for the *feature*. Validation of this approach is done for these parameters by comparing to a central-difference calculation of model geometry at a step-size of  $1.0 \times 10^{-4}$  (a step-size of  $(1.0 \times 10^{-4})(\pi/180)$  is used for  $d_{30}$ ). The same Cartesian surface grid is used to evaluate design velocity on both analytic and central-difference approaches. The difference between the design velocity fields from both methods are overlaid on the model geometry in Appendix C with Figures C-12 through C-21. The maximum difference seen on the entire *feature* is plotted in Figure 3-26. In each case both methods resulted in excellent agreement.

In the event that CAD tolerances vary and increase, the expected maximum difference will vary as well and agreement between analytic and finite-difference results will diminish. This is seen in Figure 3-27 as the sketch sensitivity “noise” propagates to the *feature* surfaces, thereby increasing the maximum difference between analytic and finite-difference design velocity.

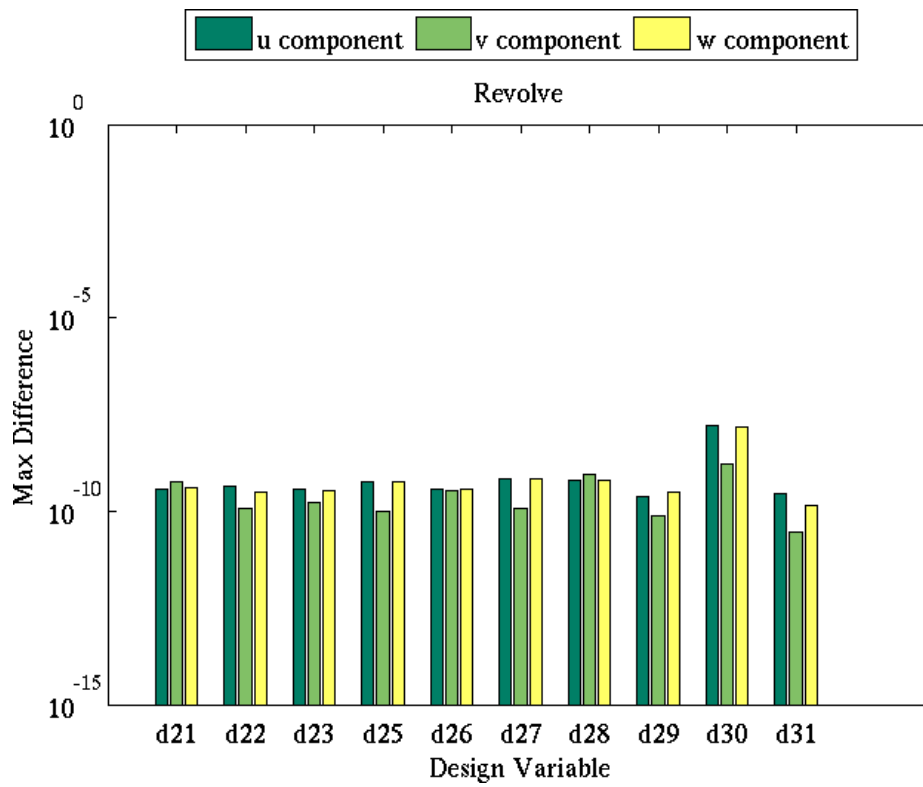
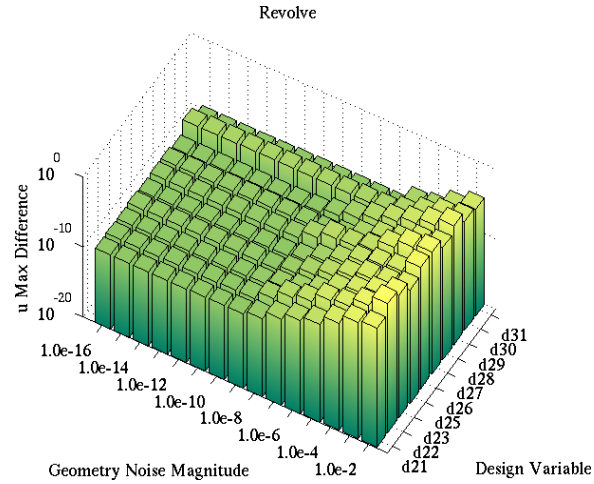
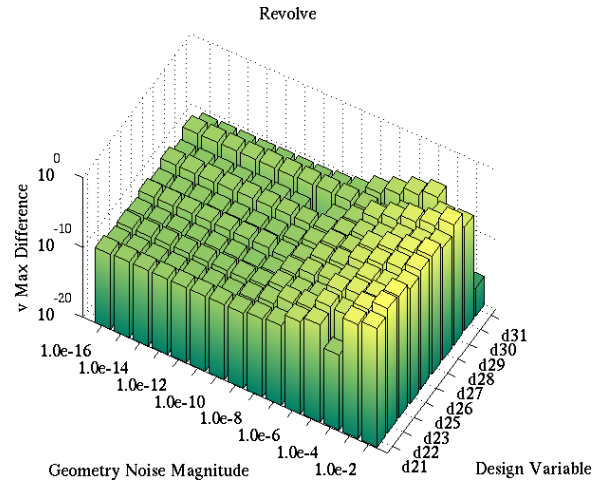


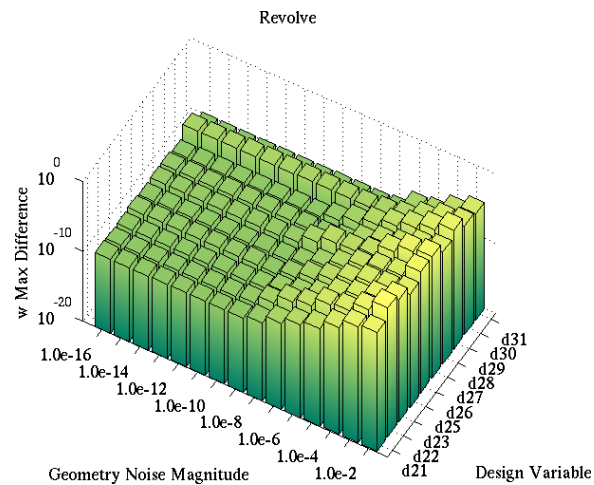
Figure 3-26: The maximum difference between the analytic and finite-difference geometry sensitivity (across all *feature* faces) is shown against the parameters in Figure 3-5.



(a)  $u$  component



(b)  $v$  component



(c)  $w$  component

Figure 3-27: The addition of geometry “noise” increases the maximum difference between analytic and finite-difference design velocity.

### 3.5 Geometry Gradients of Sweep-Feature Surfaces

A sweep *feature* is a generalization of the operation behind an extrude or revolve *feature*. For an extrude *feature*, a closed, piecewise-continuous sketch of *primitives* is extruded along a straight line in a fixed direction. In a revolve *feature*, the extrude direction constantly changes along a circular arc of fixed radius. Sweep *features* generalize the extrude direction further to any smooth curve (with the exception of curves that result in self-intersecting model geometry) passing through the sketch plane. There are at least two possible ways of extruding the sketch profile along the sweep path: (1) a normal-preserving approach and (2) a normal-following approach. The normal-preserving approach preserves the normal vector of the sketch profile to coincide with the original normal vector of the sketch plane along the entire sweep path. The normal-following approach changes the normal vector of the extruded sketch profile to match the local tangent vector direction of the sweep path. Other sweep possibilities utilize twisting of the sketch plane along a sweep path or additional guiding curves along the sweep path.

An example sweep *feature* is constructed by extruding the closed, piecewise continuous sketch of Figure 3-5 along a 2D spline curve *primitive* defined on a sketch plane orthogonal to the sketch plane of Figure 3-5. The resulting sweep *feature* shown in Figure 3-28 is normal-preserving with all faces (excluding the end-caps) generated as either B-spline surfaces or rational B-spline surfaces. As in the extrude *feature* case, the  $u$ -direction follows the sketch *primitives* and the  $v$ -direction is along the sweep path. Sketch *primitive*  $\mathbf{r}_{21}\mathbf{r}_{22}$  is a line associated with  $\mathcal{F}_5$ ; circular-arc  $\mathbf{r}_{22}\mathbf{r}_{24}$  associates with  $\mathcal{F}_6$ ; line *primitive*  $\mathbf{r}_{23}\mathbf{r}_{24}$  associates with  $\mathcal{F}_3$  and semi-ellipse  $\mathbf{r}_{21}\mathbf{r}_{23}$  corresponds to  $\mathcal{F}_4$ . The geometry kernel essentially samples points on each *primitive* for interpolation and generates the rational/non-rational B-spline surfaces in the sweep *feature*. The line *primitives* can be well approximated by maintaining collinear  $u$ -direction control points. However, there is no guarantee the circular-arc and semi-ellipse *primitives* will be represented exactly unless a rational B-spline surface is used (the algorithm choice then dictates if the rational B-spline surface properly represents the intended isoparameter *primitives*).

As a result of the bicubic rational/non-rational B-spline surface representation, an exact reverse-engineering of the sweep *feature* geometry gradient may not be possible by solely relying on the differentiated sketch *primitives*. An additional step of differentiating the

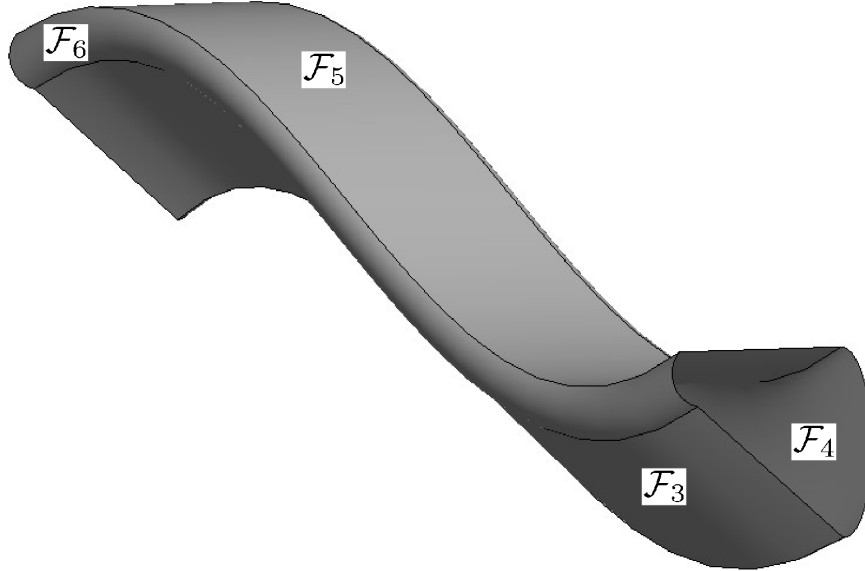


Figure 3-28: Face labels resulting from a sweep *feature* of sketch 3-5.

rational/non-rational B-spline surfaces is necessary to fully reverse-engineer the geometry gradient (see Chapter 5). Once the sketch is differentiated and the surface interpolation points are found, the sketch derivative values at those interpolation points are then passed to the B-spline surface gradient algorithm (via the term  $\partial\mathbb{B}_Q/\partial\mathcal{P}$  in (5.37)), which finally leads to the complete geometry gradient for that face.

The sketch segment  $\mathbf{r}(t)$  associated with face  $\mathcal{F}$  on surface  $S(u, v)$  is equivalent to a  $v$ -isoparameter line of surface  $S(u, v)$ . Thus, the sketch entity associated with each face has

$$S(u, v)|_{v=\text{const}} \equiv \mathbf{r}(t). \quad (3.29)$$

In the extrude case, all parameters  $\mathcal{P}$  defined solely on sketch entity  $\mathbf{r}(t)$  *only* impact  $S(u, v)$  in the  $u$ -direction and are decoupled from the extrude direction. For each parameter  $\{d_{21}, d_{22}, d_{23}, d_{25}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{31}\}$  in Figure 3-5 the decoupling is indeed true in the sweep case as well. Evidence of this becomes clear upon observation of the design velocity field from a finite-difference calculation (using a step-size of  $1.0 \times 10^{-4}$ ) of model geometry for both normal-preserving and normal-following examples in Figure 3-29. This is expected due to the intended design motion designed into the model geometry. However, another scenario of unintended design motion in a model is apparent with the design velocity field shown in 3-29(c). This apparent trajectory-dependence is a consequence of constraining the sweep-path spline *primitive* to a *primitive* on the sketch profile. The trajectory is coupled

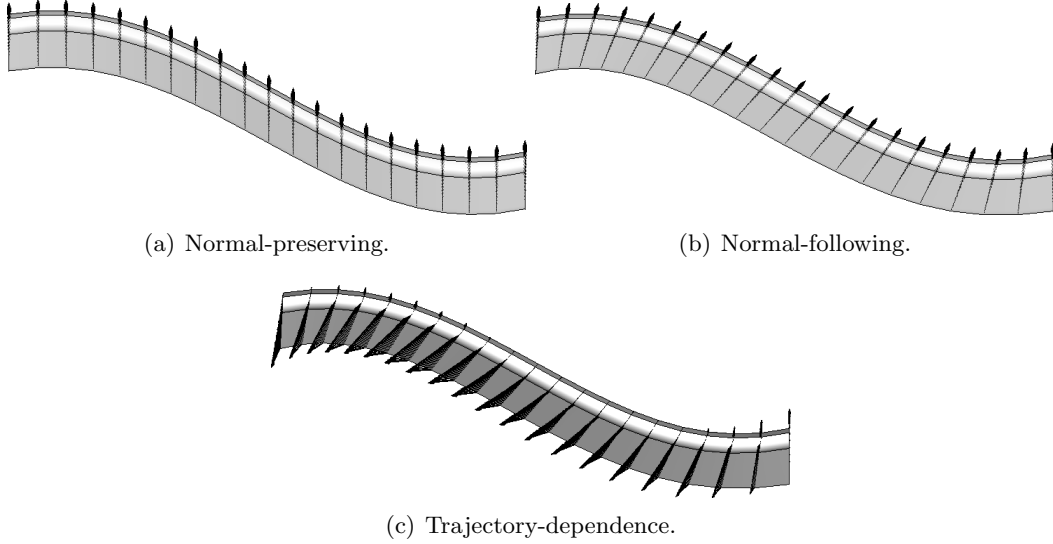


Figure 3-29: The design velocity field in (a) preserves the sketch normal-direction, whereas the field shown in (b) has a normal-following effect on the  $v$ -isoparameter line sketch gradients. Neither exhibit a dependence on the sweep path compared to the case in (c).

to the sketch and some sketch geometry gradients are transmitted as a perturbation along the spline *primitive*. The resulting design velocity field consists of the sketch gradients plus the spline *primitive* gradient along the sweep path.

For the parameters  $\{d_{21}, d_{22}, d_{23}, d_{25}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{31}\}$  this means that the design velocity field on their associated faces is *approximated* once sketch differentiation is completed by setting

$$\left. \frac{\partial S(u, v)}{\partial \mathcal{P}} \right|_{v=\text{const}} \approx \frac{\partial \mathbf{r}(t)}{\partial \mathcal{P}}, \quad (3.30)$$

or essentially propagating the sketch gradient field unchanged along the sweep path. Validation of this approach is done for these parameters by comparing to a central-difference calculation of model geometry at a step-size of  $1.0 \times 10^{-4}$  (a step-size of  $(1.0 \times 10^{-4})(\pi/180)$  was used for  $d_{30}$ ). Both methods are evaluated on a Cartesian grid with node spacing set at a fixed percentage of the total parameterization length (i.e., a percentage of  $u_{max} - u_{min}$  and  $v_{max} - v_{min}$ ). The difference between the design velocity fields from both methods are overlaid on the model geometry in Appendix C with Figures C-22 through C-31. The maximum difference is plotted in Figure 3-30 for the entire *feature* as well. In each case both methods result in good agreement only in the  $w$ -component of design velocity (meaning both methods were normal-preserving, as expected), whereas results are not in good agreement for the  $u$ - and  $v$ -component directions (also anticipated due to the model geometry

representation). This discrepancy is primarily seen on  $\mathcal{F}_4$  and  $\mathcal{F}_6$ , which correspond to the semi-ellipse and circular-arc *primitives*, respectively.

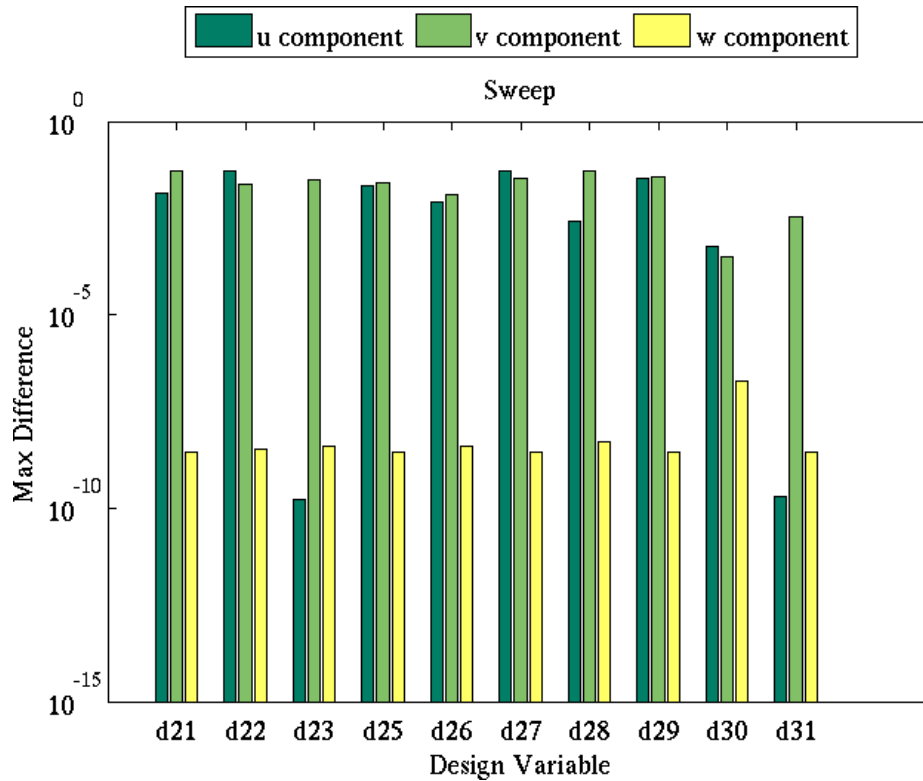


Figure 3-30: The maximum difference between the analytic and finite-difference geometry sensitivity (across all *feature* faces) is shown against the parameters in Figure 3-5.

The discrepancy arises from finite-differencing rational/non-rational B-spline surfaces in an inconsistent manner when the knot vectors and/or the support point parameterizations are not fixed between perturbed model geometry (see Section 5.7 for details on this). Appendix D tabulates the deviation in knot vector values for Faces  $\mathcal{F}_4$  and  $\mathcal{F}_6$  between perturbed model geometry and the baseline model. With the exception of  $\mathcal{P} = d_{30}$  in Table D.9, the knot vectors change for each case and the finite-difference computation becomes inconsistent. The case  $\mathcal{P} = d_{30}$  does *not* have changing knot vectors, yet it is possible that the underlying sampled support point parameterization has changed. This is possible, as seen in Figure 3-31, because the design velocity discrepancy implies a different design motion for the rational B-spline isoparameter lines compared to a pure circular-arc *primitive*. This notion is supported by comparing the validated circular-arc design velocity vectors in Figure 3-19 (obtained by analytic differentiation of the sketch) with the finite-difference results in Figure 3-31. This case implies that the rational B-spline representation stems from



a formulation that only *approximates* the circular-arc (one such formulation can be found in [33] even though other methods exist that represent conics properly [72]) or the support point parameterization has changed. In the event of an approximation by the geometry kernel, the design velocity discrepancy stems solely from a variation in design motion between two *different* geometry entities. These reasons must also contribute to the design velocity discrepancy seen for the other sketch parameters as well on  $\mathcal{F}_4$  and  $\mathcal{F}_6$ .

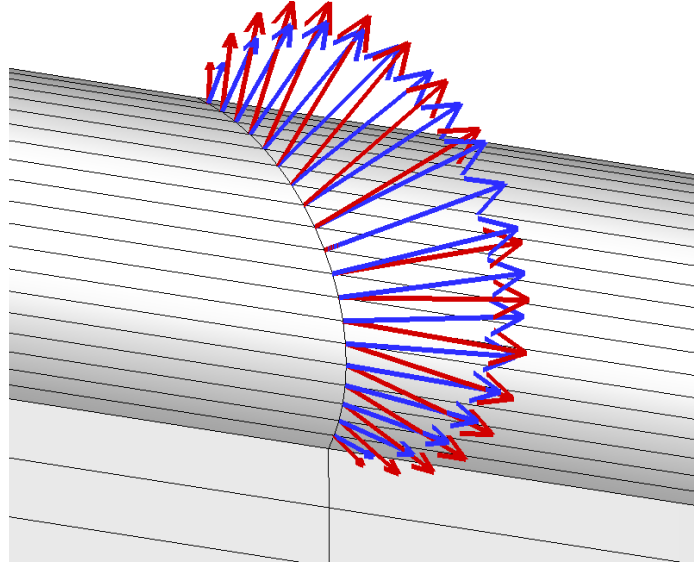


Figure 3-31: Comparison of the analytic (blue) and finite-difference (red) design velocity vectors shows a design motion discrepancy between the pure circular-arc *primitive* and the rational B-spline representation on isoparameter lines.

These design velocity discrepancies are also unavoidable if a “snapped” Cartesian grid is used, as discussed in [57], wherein the baseline grid nodes are projected onto the perturbed finite-difference model geometry. This occurs by querying an inverse evaluation of the perturbed surfaces using the baseline model grid coordinates (the geometry kernel readily outputs this type of query) so that the *nearest* points on the perturbed surface are returned and become the nodes of the perturbed Cartesian grid for differencing. This approach is a non-rigorous approximation to design velocity that leads to spurious results, as observed in [57]. Since the nearest point evaluation is a projection operation (where a baseline point is projected normal to the perturbed surface), it is unrelated to the actual design motion of a perturbed surface. Using finite-differences on “snap” grids composed of such points will result in a calculation that is unrelated to the true geometry gradient, in general<sup>5</sup>. This is

<sup>5</sup>This can work, though, when surface design motion aligns with the queried nearest-point direction, as

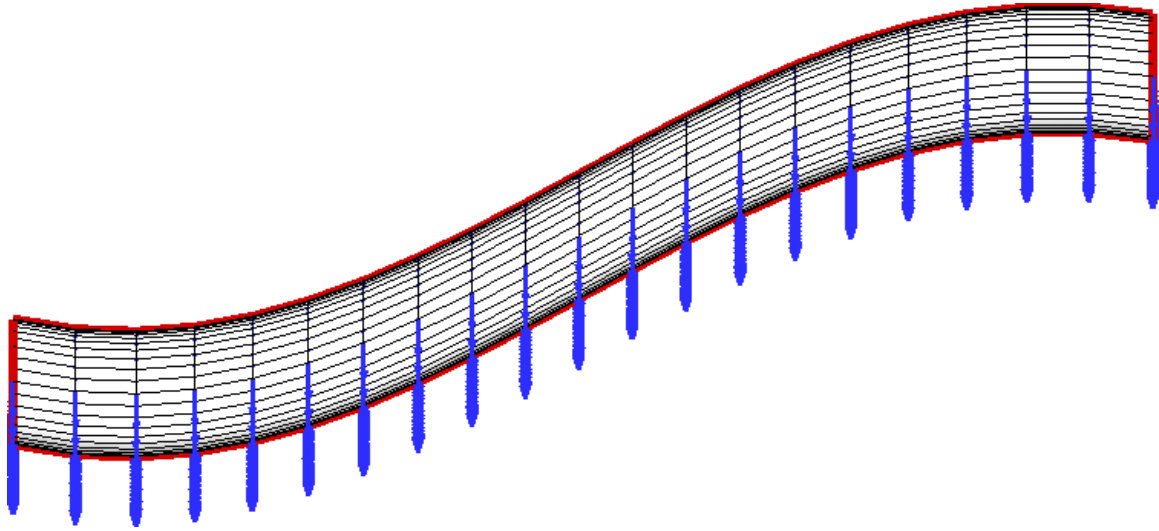
clearly seen in Figures 3-32 and 3-33 for parameter  $\mathcal{P} = d_{21}$ . Figure 3-32 displays design velocities with a spurious sweep-trajectory dependence when the “snap” grid option is used. In addition, Figure 3-33 shows that the design velocity distribution is discontinuous along the  $u$ -isoparameter lines and does not match the expected distribution validated in Figure 3-11 for the parent sketch. As expected, the fixed-spacing grid results are consistent with the normal-preserving model geometry and the design velocity distribution of Figure 3-11. Similar results are seen in Appendix E for the remaining parameters  $\mathcal{P} \in \{d_{22}, d_{23}, d_{25}, \dots, d_{31}\}$  of Figure 3-5. Since these results confirm that the nearest-point projection is generally not related to surface design motion, it is recommended that “snapped” grids be avoided as a point-tracking strategy for finite-differencing.

In summary, it may be possible to determine how the swept circular-arc and elliptical *primitives* are sampled to generate the rational B-spline surfaces. This entails “reverse-engineering” the surface by computing the sampled support points using the control points and rational B-spline basis functions. If successful, the design velocity along the primitives can be prescribed at these support points and the B-spline surface sensitivity method of Chapter 5 would yield the surface sensitivity. Otherwise, knowledge of the CAD source code appears necessary to analytically differentiate every face of sweep *features*. This could be circumvented, though, if CAD systems somehow permitted regenerating the *feature* surfaces with fixed parameterization. As further explained in Chapter 5, attempting to “correct” the finite-difference sensitivity result would require “reverse-engineering” the surface as well, thereby making the finite-difference approach unnecessary altogether. The maximum design velocity error seen when finite-differencing the sweep *feature* is expected to follow the error analysis approximations in Chapter 5 once the variation in knot vectors is known on the B-spline surfaces.

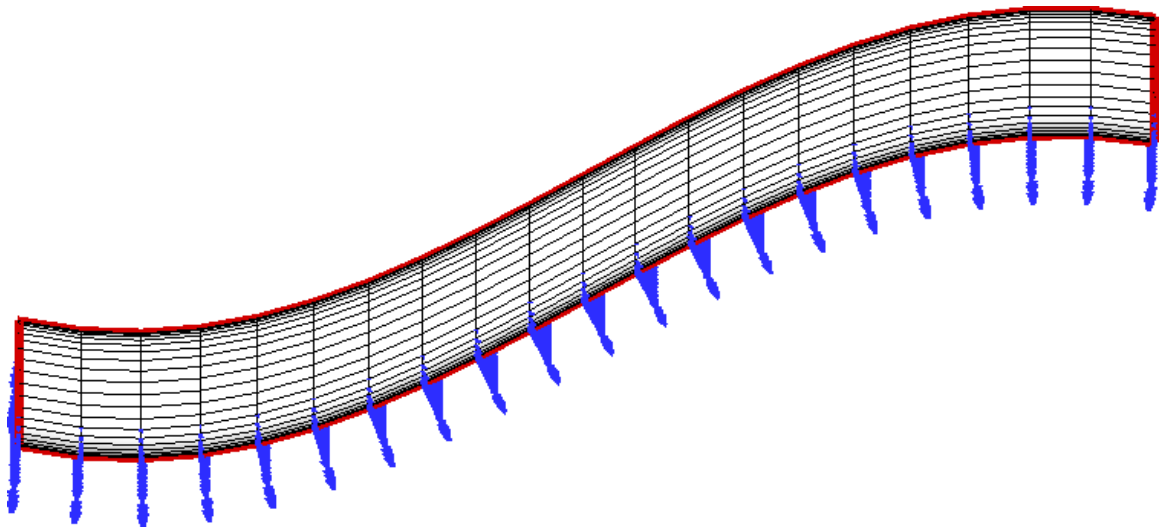
For completeness, even though the discrepancies from finite-differences are already large for the sweep *feature*, the addition of geometry “noise” does not help the situation. With increasing CAD tolerances the expected maximum difference will vary as well and agreement between analytic and finite-difference results not improve. This is seen in Figure 3-34 as the sketch sensitivity “noise” propagates to the *feature* surfaces, thereby impacting the maximum difference between analytic and finite-difference design velocity.

---

seen in translating planar surfaces or increasing the radius of circular-arcs. Spurious design velocity estimates will occur wherever this constraint is violated on a surface.

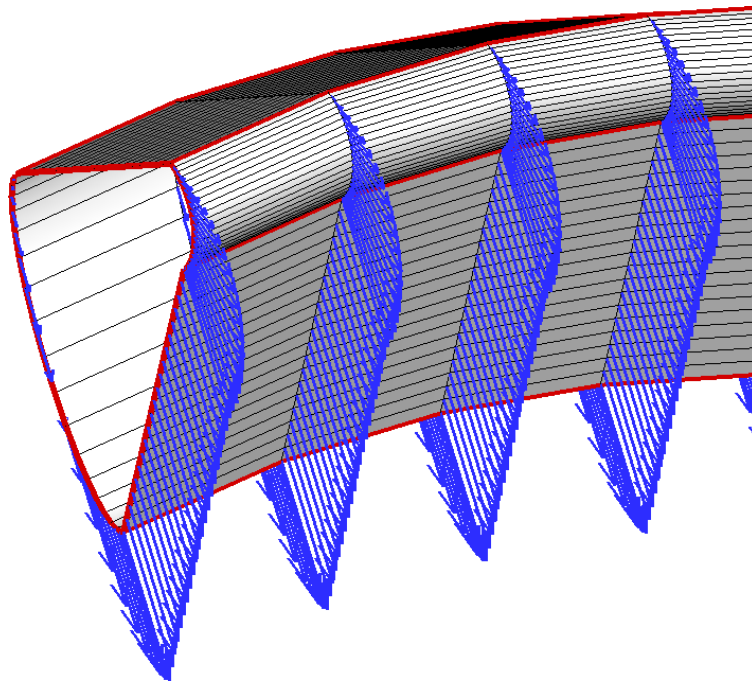


(a) Fixed-spacing Grid

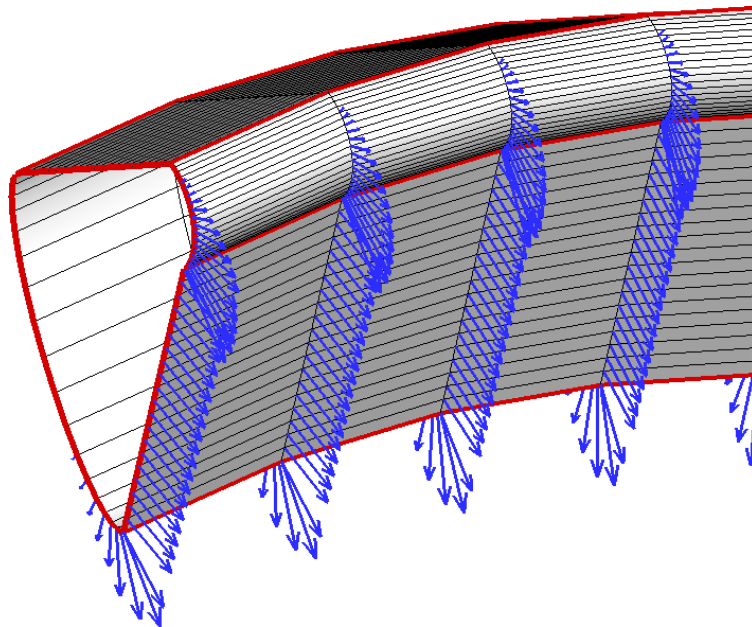


(b) "Snap" Grid

Figure 3-32: The design velocity vectors in (a) correctly follow the expected normal-preserving orientation and show no sweep-trajectory dependence. In (b), however, the "snap" grid yields non-normal-preserving design velocity results with a spurious sweep-trajectory dependence. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{21}$  and step-size  $h = 1.0 \times 10^{-4}$ .

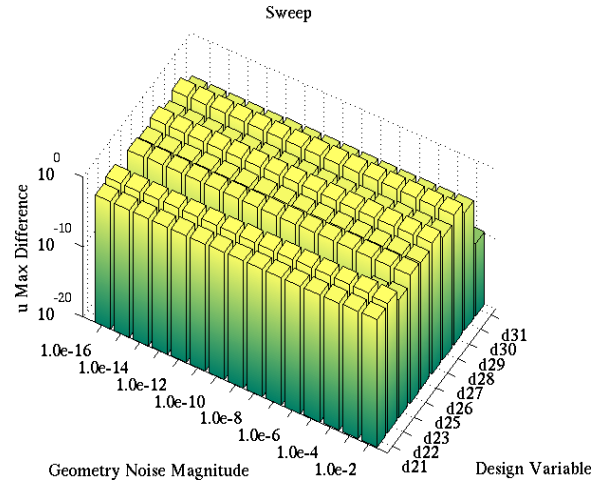


(a) Fixed-spacing Grid

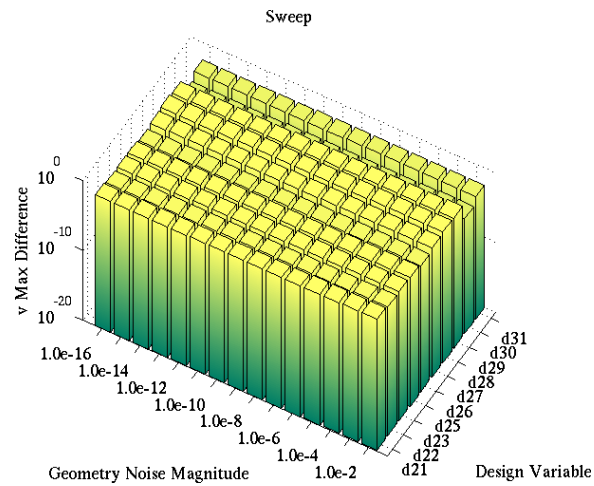


(b) "Snap" Grid

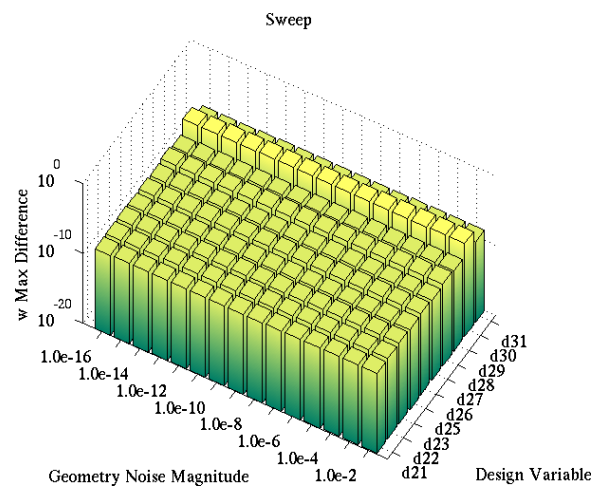
Figure 3-33: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isoparameter line (as seen in Figure 3-11). In (b), however, the "snap" grid yields incorrectly discontinuous design velocities along the  $u$ -isoparameter line. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{21}$  and step-size  $h = 1.0 \times 10^{-4}$ .



(a)  $u$  component



(b)  $v$  component



(c)  $w$  component

Figure 3-34: The addition of geometry “noise” increases the maximum difference between analytic and finite-difference design velocity.

### 3.6 Additional Commentary on Feature Geometry Gradients

The extrude, revolve and sweep *feature* examples presented in this chapter demonstrate that geometry kernels often approximate *feature* surfaces using B-spline surfaces. This is the case for  $\mathcal{F}_3$  in the extrude model (Figure 3-21),  $\mathcal{F}_2$  and  $\mathcal{F}_7$  in the revolve model (Figure 3-24) and  $\mathcal{F}_3$ ,  $\mathcal{F}_4$ ,  $\mathcal{F}_5$  and  $\mathcal{F}_6$  in the sweep model (Figure 3-28). It is important to extract BRep information from model geometry in order to identify the B-spline surfaces. As already noted, finite-difference estimates of design velocity on these surfaces may deviate from analytic results if the knot vectors on these surfaces change after perturbing the model.

Both the extrude and revolve model demonstrate excellent agreement between the analytic and finite-difference design velocity. This occurs because the underlying semi-ellipse *primitive* is represented via *rational* cubic Bezier isoparameter lines in the  $u$  direction (where conics are a special case for such curves [49]) *and* the parameterization does not change with model perturbations. The circular-arc in  $\mathcal{F}_1$  in the extrude model is modeled properly as a cylinder, whereas in the revolve model it is properly modeled as a torus section ( $\mathcal{F}_4$  and  $\mathcal{F}_5$ ). For these reasons, the finite-difference design velocity has excellent agreement with the analytic design velocity on these models. The sweep model uses rational cubic B-spline isoparameter lines (again in the  $u$  direction) to model the semi-ellipse and circular-arc because these formulations can also represent conics [49]. However it is clear that these surfaces do not have fixed parameterizations when perturbed, thereby causing discrepancies when compared to analytic sensitivities.

At this point it is clear that the geometry gradient of a CAD-generated model geometry is best accomplished by decomposing a model into its component master-model *features* and differentiating them. In this manner, topology changes can be tolerated along a design trajectory because the final set of *all* BRep faces associate with some pre-defined *feature*. An “instantaneous” design velocity for all faces on the complete model geometry is thus achieved as the superposition of design velocity fields for each *feature* in the master model.

In this chapter the three most widely used *features* in CAD-generated model geometry are reverse-engineered to yield analytic geometry gradients with respect to model parameters. Each requires attention to the individual geometry surfaces resulting from the *feature* operation in order to obtain the design velocity field for the entire *feature*. With this *primitive-to-primitive*, *surface-to-surface*, *feature-to-feature* approach, an entire CAD-generated

model can be analytically differentiated. There are additional *primitives* and *features* to reverse-engineer (B-splines and B-spline surfaces are discussed in later chapters) for a given CAD system, yet those discussed in this chapter also cover the most widely used *primitives* and dimensioning schemes. Additional geometry constraints need to be reverse-engineered as required. Once these are obtained, the process of differentiating sketches and applying the results to *feature* surfaces can ensue with the new information. With the geometry gradient on BRep faces available, the gradient along BRep edges (more specifically trim curves) and nodes also becomes possible.

THIS PAGE INTENTIONALLY LEFT BLANK



## Chapter 4

# Geometry Sensitivities for BRep Edges & Nodes

The geometry sensitivity methods discussed in Chapter 3 are augmented by considering the surface intersections of 3D CAD model geometry. A method for the design velocity on BRep edges (more specifically trim curves) and nodes is presented here. This new approach implements a minimum-velocity method that does not constrain the design velocity to a particular direction. The special case of a closed-form, three surface intersection problem is considered initially. A formulation for trim curves is then presented for two surface intersections, followed by an extension to BRep nodes with three or more surface intersections. Validation for each approach is presented, as well as comparison with other methods on example problems.

### 4.1 Components of Design Velocity

There is a distinction between edges and trim curves in the perspective of design velocity. Although trim curves are a subset of edges, edges that are not trim curves may not need an additional design velocity formulation if they represent the boundaries of adjacent surfaces defined by the full-extent of underlying *primitives*. In these cases the sketch end-point design velocity is propagated along the boundary edges as in the extrude, revolve and sweep examples discussed in Chapter 3. The same is true for nodes that are not “trim nodes” since these correspond to the sketch node design velocity as well. Trim curves, however, rely on the underlying surfaces that intersect and generally do not have a direct associativity

to design velocity on sketches. The same is true for nodes comprised of intersecting trim curves. A design velocity formulation for trim curves and trim nodes is presented in this chapter.

We consider an Euclidean space  $E \in \mathbb{R}^3$  with origin at  $\mathbf{O}_0$  and write the surface parameterization for two intersecting faces  $\mathcal{F}_1$  and  $\mathcal{F}_2$  in a model geometry BRep using  $\mathbf{r}_1, \mathbf{r}_2 \in E$  as  $\mathbf{r}_1 = \mathbf{r}_1((u, v)_1; \mathcal{P}_j)$  and  $\mathbf{r}_2 = \mathbf{r}_2((u, v)_2; \mathcal{P}_j)$ . We note that  $\mathcal{F}_1$  and  $\mathcal{F}_2$  have domains  $W_1 = U_1 \times V_1$  and  $W_2 = U_2 \times V_2$ , where  $U_1 = [u_{1,\min}, u_{1,\max}] \in \mathbb{R}$  and  $V_1 = [v_{1,\min}, v_{1,\max}] \in \mathbb{R}$  with coordinates  $(u, v)_1 \in W_1$ , whereas  $U_2 = [u_{2,\min}, u_{2,\max}] \in \mathbb{R}$  and  $V_2 = [v_{2,\min}, v_{2,\max}] \in \mathbb{R}$  with coordinates  $(u, v)_2 \in W_2$ . The set of parameters that drive both  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are  $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_j, \dots, \mathcal{P}_J\}$ .

Trim curves are often cubic B-splines, yet their parameterization is usually not provided by the surface intersection algorithms in a geometry kernel. Therefore, by using  $\mathbf{e} \in E$ , the trim curve  $\mathcal{E}$  shared by  $\mathcal{F}_1$  and  $\mathcal{F}_2$  is assumed to have the parameterization  $\mathbf{e} = \mathbf{e}(t; \mathcal{P}_j)$  on the domain  $T = [t_{\min}, t_{\max}] \in \mathbb{R}$  with coordinate  $t \in T$ . Since the trim curve is a construction from two intersecting surfaces, it shares the driving parameter set of both faces.

We first consider the geometry gradient for a face  $\mathcal{F}$  with respect to a parameter  $\mathcal{P}$ :

$$\frac{d\mathbf{r}}{d\mathcal{P}} = \frac{\partial \mathbf{r}}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}}{\partial u} \frac{\partial u}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}}{\partial v} \frac{\partial v}{\partial \mathcal{P}}. \quad (4.1)$$

By rewriting the components of design velocity in (4.1) with the following substitutions,

$$\dot{\mathbf{r}} = \frac{\partial \mathbf{r}}{\partial \mathcal{P}}, \quad \boldsymbol{\nu} = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} \partial u / \partial \mathcal{P} \\ \partial v / \partial \mathcal{P} \end{bmatrix}, \quad \nabla \equiv \begin{bmatrix} \partial / \partial u & \partial / \partial v \end{bmatrix}, \quad (4.2)$$

a more informative expression is reached:

$$\frac{d\mathbf{r}}{d\mathcal{P}} = \dot{\mathbf{r}} + \nabla \mathbf{r} \cdot \boldsymbol{\nu}. \quad (4.3)$$

The velocity  $\boldsymbol{\nu}$  in the domain space  $W$  of  $\mathcal{F}$  is projected into the space  $E$  by  $\nabla \mathbf{r}$ . In other words,  $\boldsymbol{\nu}$  is a relative design velocity term with respect to its parent faces. In an analogous manner (4.3) can be interpreted from a continuum mechanics perspective. The total design velocity consists of a term denoting “unsteadiness” of the surface ( $\dot{\mathbf{r}}$ ), or design velocity of

the entire surface, and a second “convective” term  $(\nabla \mathbf{r} \cdot \boldsymbol{\nu})$  representing a relative velocity, or design velocity relative to the surface.

A similar procedure can be followed for the description of design velocity on a trim curve. Since the actual parameterization of the trim curve is unknown, it is reasonable to presume the form  $\mathbf{e} = \mathbf{e}(t; \mathcal{P})$ . The curve design velocity is initially written as

$$\frac{d\mathbf{e}}{d\mathcal{P}} = \frac{\partial \mathbf{e}}{\partial \mathcal{P}} + \frac{\partial \mathbf{e}}{\partial t} \frac{\partial t}{\partial \mathcal{P}}. \quad (4.4)$$

The term  $\frac{\partial \mathbf{e}}{\partial t}$  in (4.4) is obtainable from the CAD geometry kernel as the local tangent to the trim curve. The remaining terms in (4.4) are unknown since the parameterization of the trim curve is not given. By making the following substitutions

$$\dot{\mathbf{e}} = \frac{\partial \mathbf{e}}{\partial \mathcal{P}}, \quad \tilde{\boldsymbol{\nu}} = [\tilde{\nu}] = [\partial t / \partial \mathcal{P}], \quad \tilde{\nabla} \equiv [\partial / \partial t],$$

we can rewrite (4.4) as

$$\frac{d\mathbf{e}}{d\mathcal{P}} = \dot{\mathbf{e}} + \tilde{\nabla} \mathbf{e} \cdot \tilde{\boldsymbol{\nu}}. \quad (4.5)$$

In this 1D case, the “unsteady” term ( $\dot{\mathbf{e}}$ ) refers to the design velocity of the entire trim curve and the second “convective” term ( $\tilde{\nabla} \mathbf{e} \cdot \tilde{\boldsymbol{\nu}}$ ) refers to the relative design velocity with respect to the original curve (here the relative velocity  $\tilde{\boldsymbol{\nu}}$  is projected into the space  $E$  by  $\tilde{\nabla} \mathbf{e}$ ). In other words, the unknown  $\tilde{\boldsymbol{\nu}}$  can be interpreted as the relative change of a point along the curve to changes in parameters because the domain coordinate  $t$  is usually expressed as a percentage of total curve length (which may reasonably depend on  $\mathcal{P}$ ).

## 4.2 Closed-Form Intersection Problem

The case of three intersecting surfaces involves a closed system of geometry constraint equations. This contrasts with the intersection trim curve problem (see Section 4.3), which consists of an under-determined system of equations, and the node intersection problem (see Section 4.4), which uses an over-determined system of equations. In this scenario three intersecting surfaces ( $\mathbf{r}_1$ ,  $\mathbf{r}_2$  and  $\mathbf{r}_3$ ) share a common intersection node  $\mathcal{N}$  with coordinates

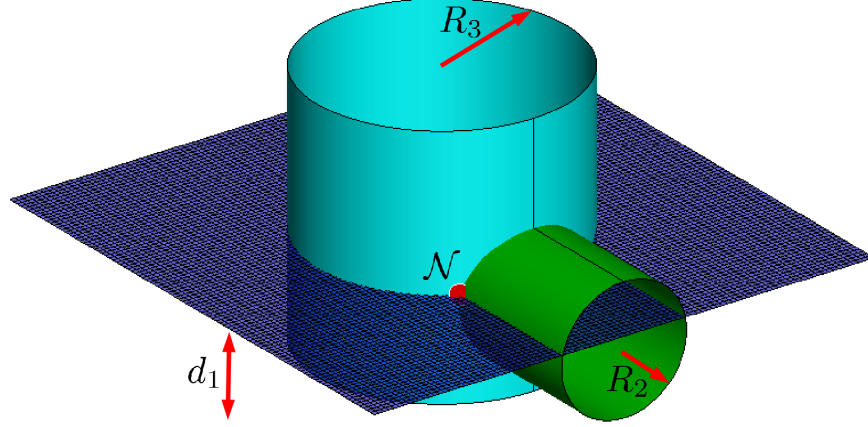


Figure 4-1: The intersection of two cylinders and a plane at a node.

$\eta \in R^3$ , as seen in Figure 4-1, where the geometry relations are

$$\begin{aligned}
 \mathbf{r}_1 - \mathbf{r}_2 &= \mathbf{0} \\
 \mathbf{r}_2 - \mathbf{r}_3 &= \mathbf{0} \\
 \mathbf{r}_3 - \mathbf{r}_1 &= \mathbf{0}.
 \end{aligned} \tag{4.6}$$

Then a first-order Taylor expansion in the neighborhood of the intersection point can be written for each surface as

$$\mathbf{r}_1((u, v)_1; \mathcal{P} + \delta\mathcal{P})' = \mathbf{r}_1((u, v)_1; \mathcal{P}) + \frac{d\mathbf{r}_1}{d\mathcal{P}} \delta\mathcal{P} \tag{4.7}$$

$$\mathbf{r}_2((u, v)_2; \mathcal{P} + \delta\mathcal{P})' = \mathbf{r}_2((u, v)_2; \mathcal{P}) + \frac{d\mathbf{r}_2}{d\mathcal{P}} \delta\mathcal{P} \tag{4.8}$$

$$\mathbf{r}_3((u, v)_3; \mathcal{P} + \delta\mathcal{P})' = \mathbf{r}_3((u, v)_3; \mathcal{P}) + \frac{d\mathbf{r}_3}{d\mathcal{P}} \delta\mathcal{P} \tag{4.9}$$

$$\tag{4.10}$$

Substituting these expansions into the geometry constraint equations then yields

$$\begin{aligned}
 \left( \frac{d\mathbf{r}_1}{d\mathcal{P}} - \frac{d\mathbf{r}_2}{d\mathcal{P}} \right) \delta\mathcal{P} &= \mathbf{0} \\
 \left( \frac{d\mathbf{r}_2}{d\mathcal{P}} - \frac{d\mathbf{r}_3}{d\mathcal{P}} \right) \delta\mathcal{P} &= \mathbf{0} \\
 \left( \frac{d\mathbf{r}_3}{d\mathcal{P}} - \frac{d\mathbf{r}_1}{d\mathcal{P}} \right) \delta\mathcal{P} &= \mathbf{0},
 \end{aligned} \tag{4.11}$$

so that for any  $\delta\mathcal{P}$  we have

$$\begin{aligned}\frac{d\mathbf{r}_1}{d\mathcal{P}} - \frac{d\mathbf{r}_2}{d\mathcal{P}} &= \mathbf{0} \\ \frac{d\mathbf{r}_2}{d\mathcal{P}} - \frac{d\mathbf{r}_3}{d\mathcal{P}} &= \mathbf{0} \\ \frac{d\mathbf{r}_3}{d\mathcal{P}} - \frac{d\mathbf{r}_1}{d\mathcal{P}} &= \mathbf{0}.\end{aligned}\tag{4.12}$$

By (4.1) it is known that each surface brings two unknowns ( $\frac{\partial u_i}{\partial\mathcal{P}}$  and  $\frac{\partial v_i}{\partial\mathcal{P}}$ ) for a total of six unknowns. Thus, substituting (4.1) for each respective surface within any two equations in (4.12) allows creation of the following closed system (here the initial two constraint equations are used):

$$\begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial v_1} & -\frac{\partial x_2}{\partial u_2} & -\frac{\partial x_2}{\partial v_2} & 0 & 0 \\ \frac{\partial y_1}{\partial u_1} & \frac{\partial y_1}{\partial v_1} & -\frac{\partial y_2}{\partial u_2} & -\frac{\partial y_2}{\partial v_2} & 0 & 0 \\ \frac{\partial z_1}{\partial u_1} & \frac{\partial z_1}{\partial v_1} & -\frac{\partial z_2}{\partial u_2} & -\frac{\partial z_2}{\partial v_2} & 0 & 0 \\ 0 & 0 & \frac{\partial x_2}{\partial u_2} & \frac{\partial x_2}{\partial v_2} & -\frac{\partial x_3}{\partial u_3} & -\frac{\partial x_3}{\partial v_3} \\ 0 & 0 & \frac{\partial y_2}{\partial u_2} & \frac{\partial y_2}{\partial v_2} & -\frac{\partial y_3}{\partial u_3} & -\frac{\partial y_3}{\partial v_3} \\ 0 & 0 & \frac{\partial z_2}{\partial u_2} & \frac{\partial z_2}{\partial v_2} & -\frac{\partial z_3}{\partial u_3} & -\frac{\partial z_3}{\partial v_3} \end{bmatrix} \begin{bmatrix} \frac{\partial u_1}{\partial\mathcal{P}} \\ \frac{\partial v_1}{\partial\mathcal{P}} \\ \frac{\partial u_2}{\partial\mathcal{P}} \\ \frac{\partial v_2}{\partial\mathcal{P}} \\ \frac{\partial u_3}{\partial\mathcal{P}} \\ \frac{\partial v_3}{\partial\mathcal{P}} \end{bmatrix} = \begin{bmatrix} \frac{\partial x_2}{\partial\mathcal{P}} - \frac{\partial x_1}{\partial\mathcal{P}} \\ \frac{\partial y_2}{\partial\mathcal{P}} - \frac{\partial y_1}{\partial\mathcal{P}} \\ \frac{\partial z_2}{\partial\mathcal{P}} - \frac{\partial z_1}{\partial\mathcal{P}} \\ \frac{\partial x_3}{\partial\mathcal{P}} - \frac{\partial x_2}{\partial\mathcal{P}} \\ \frac{\partial y_3}{\partial\mathcal{P}} - \frac{\partial y_2}{\partial\mathcal{P}} \\ \frac{\partial z_3}{\partial\mathcal{P}} - \frac{\partial z_2}{\partial\mathcal{P}} \end{bmatrix}.\tag{4.13}$$

The coefficient matrix and right-hand side is readily evaluated from CAD data. Since (4.13) is a  $6 \times 6$  linear system, it can be solved using LU decomposition with pivoting and multiple back-substitutions to yield

$$\left[ \begin{array}{cccccc} \frac{\partial u_1}{\partial\mathcal{P}} & \frac{\partial v_1}{\partial\mathcal{P}} & \frac{\partial u_2}{\partial\mathcal{P}} & \frac{\partial v_2}{\partial\mathcal{P}} & \frac{\partial u_3}{\partial\mathcal{P}} & \frac{\partial v_3}{\partial\mathcal{P}} \end{array} \right]^T.\tag{4.14}$$

The intersection design velocity is calculated by substituting the solution components into

(4.1) for each surface.

$$\begin{bmatrix} \frac{\partial \boldsymbol{\eta}}{\partial d_1} & \frac{\partial \boldsymbol{\eta}}{\partial R_2} & \frac{\partial \boldsymbol{\eta}}{\partial R_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{r}_1}{\partial u_1} & \frac{\partial \mathbf{r}_1}{\partial v_1} \end{bmatrix} \begin{bmatrix} \frac{\partial u_1}{\partial d_1} & \frac{\partial u_1}{\partial R_2} & \frac{\partial u_1}{\partial R_3} \\ \frac{\partial v_1}{\partial d_1} & \frac{\partial v_1}{\partial R_2} & \frac{\partial v_1}{\partial R_3} \end{bmatrix} + \begin{bmatrix} \frac{\partial \mathbf{r}_1}{\partial d_1} & \frac{\partial \mathbf{r}_1}{\partial R_2} & \frac{\partial \mathbf{r}_1}{\partial R_3} \end{bmatrix}. \quad (4.15)$$

An analytic solution also exists for validation comparisons. This is derived by parameterizing the surfaces as

$$\begin{aligned} \textbf{Plane: } \mathbf{r}_1 &= u_1 \hat{\boldsymbol{\xi}}_{1_1} + v_1 \hat{\boldsymbol{\xi}}_{1_2} + \mathbf{O}_1 \\ \textbf{Horizontal Cylinder: } \mathbf{r}_2 &= R_2 \left[ \cos(u_2) \hat{\boldsymbol{\xi}}_{2_1} + \sin(u_2) \hat{\boldsymbol{\xi}}_{2_2} \right] + v_2 \hat{\boldsymbol{\xi}}_{2_3} + \mathbf{O}_2 \\ \textbf{Vertical Cylinder: } \mathbf{r}_3 &= R_3 \left[ \cos(u_3) \hat{\boldsymbol{\xi}}_{3_1} + \sin(u_3) \hat{\boldsymbol{\xi}}_{3_2} \right] + v_3 \hat{\boldsymbol{\xi}}_{3_3} + \mathbf{O}_3. \end{aligned} \quad (4.16)$$

With global origin  $\mathbf{O}_0 \in \mathbb{R}^3$  and relative surface origins  $\mathbf{O}_1, \mathbf{O}_2, \mathbf{O}_3 \in \mathbb{R}^3$ , these surface parameterizations have  $\mathbf{O}_0 = [0, 0, 0]^T$ ,  $\mathbf{O}_1 = [0, 0, d_1]^T$ ,  $\mathbf{O}_2 = [0, 0, 0.5]^T$  and  $\mathbf{O}_3 = [0, 0, 0]^T$  for the plane, horizontal and vertical cylinders, respectively. For  $i = 1, 2, 3$ , the  $i$ th surface is defined in its own domain space,  $W_i \in \mathbb{R}^2$ , where  $W_i = U_i \times V_i$  with  $U_i = [u_{i,\min}, u_{i,\max}] \in \mathbb{R}$  and  $V_i = [v_{i,\min}, v_{i,\max}] \in \mathbb{R}$ . The surface coordinates are simply  $u_i \in U_i$  and  $v_i \in V_i$ .

Although various parameterization options are possible, these three surfaces are driven by  $d_1 \in \mathbb{R}$ , the distance of the plane from the  $xy$ -plane, and the cylinder radii  $R_2, R_3 \in \mathbb{R}$ . Thus the parameter set is  $\mathbf{P} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\} = \{d_1, R_2, R_3\}$ . With this information the intersection node coordinates are found using elementary trigonometry:

$$\boldsymbol{\eta} = \begin{bmatrix} \sqrt{R_3^2 - \left[ R_2^2 - (d_1 - O_{2_3})^2 \right]} \\ -\sqrt{R_2^2 - (d_1 - O_{2_3})^2} \\ d_1 \end{bmatrix}, \quad (4.17)$$

where  $O_{2_3}$  refers to the third component of  $\mathbf{O}_2$  (i.e.,  $\mathbf{O}_2 = [O_{2_1}, O_{2_2}, O_{2_3}]^T$ ). An analytic

geometry gradient for node  $\mathcal{N}$  thus follows as well for each parameter in  $\mathbf{P}$ :

$$\begin{aligned}\frac{\partial \boldsymbol{\eta}}{\partial d_1} &= \begin{bmatrix} \frac{d_1 - O_{23}}{\sqrt{R_3^2 - R_2^2 + (d_1 - O_{23})^2}} & \frac{d_1 - O_{23}}{\sqrt{R_2^2 - (d_1 - O_{23})^2}} & 1 \end{bmatrix}^T, \\ \frac{\partial \boldsymbol{\eta}}{\partial R_2} &= \begin{bmatrix} \frac{-R_2}{\sqrt{R_3^2 - R_2^2 + (d_1 - O_{23})^2}} & \frac{-R_2}{\sqrt{R_2^2 - (d_1 - O_{23})^2}} & 0 \end{bmatrix}^T, \\ \frac{\partial \boldsymbol{\eta}}{\partial R_3} &= \begin{bmatrix} \frac{R_3}{\sqrt{R_3^2 - R_2^2 + (d_1 - O_{23})^2}} & 0 & 0 \end{bmatrix}^T.\end{aligned}\quad (4.18)$$

A comparison is also made by considering a linearization at  $\boldsymbol{\eta}$  with respect to each parameter in  $\mathbf{P}$ . Perturbing any parameter in  $\mathbf{P}$  results in a new intersection node  $\boldsymbol{\eta}'(d_1 + \delta d_1, R_2 + \delta R_2, R_3 + \delta R_3)$  written as

$$\boldsymbol{\eta}'(d_1 + \delta d_1, R_2 + \delta R_2, R_3 + \delta R_3) = \begin{bmatrix} \sqrt{(R_3 + \delta R_3)^2 - \left\{ (R_2 + \delta R_2)^2 - [(d_1 + \delta d_1) - O_{23}]^2 \right\}} \\ -\sqrt{(R_2 + \delta R_2)^2 - ((d_1 + \delta d_1) - O_{23})^2} \\ d_1 + \delta d_1 \end{bmatrix}.$$

Hence, a central-difference scheme is written as

$$\frac{\partial \boldsymbol{\eta}}{\partial d_1} \approx \frac{\boldsymbol{\eta}'(+\delta d_1, 0, 0) - \boldsymbol{\eta}'(-\delta d_1, 0, 0)}{2\delta d_1} \quad (4.19)$$

$$\frac{\partial \boldsymbol{\eta}}{\partial R_2} \approx \frac{\boldsymbol{\eta}'(0, +\delta R_2, 0) - \boldsymbol{\eta}'(0, -\delta R_2, 0)}{2\delta R_2} \quad (4.20)$$

$$\frac{\partial \boldsymbol{\eta}}{\partial R_3} \approx \frac{\boldsymbol{\eta}'(0, 0, +\delta R_3) - \boldsymbol{\eta}'(0, 0, -\delta R_3)}{2\delta R_3}. \quad (4.21)$$

Furthermore, the sensitivity information in Table 4.1 is needed to populate the coefficient matrix and right-hand side in (4.13). The  $(u, v)$  values for each surface at  $\boldsymbol{\eta}$  are also required and obtained by writing the residual equations

$$\begin{aligned}\mathcal{R}_1 &= f(u_1, v_1) = \|\mathbf{r}_1 - \boldsymbol{\eta}\|_2 = 0 \\ \mathcal{R}_2 &= f(u_2, v_2) = \|\mathbf{r}_2 - \boldsymbol{\eta}\|_2 = 0 \\ \mathcal{R}_3 &= f(u_3, v_3) = \|\mathbf{r}_3 - \boldsymbol{\eta}\|_2 = 0\end{aligned}\quad (4.22)$$

	$\mathbf{r}_1$	$\mathbf{r}_2$	$\mathbf{r}_3$
$\frac{\partial(\cdot)}{\partial u}$	$\hat{\xi}_{11}$	$R_2 \left[ -\sin(u_2) \hat{\xi}_{21} + \cos(u_2) \hat{\xi}_{22} \right]$	$R_3 \left[ -\sin(u_3) \hat{\xi}_{31} + \cos(u_3) \hat{\xi}_{32} \right]$
$\frac{\partial(\cdot)}{\partial v}$	$\hat{\xi}_{12}$	$\hat{\xi}_{23}$	$\hat{\xi}_{33}$
$\frac{\partial(\cdot)}{\partial \mathcal{P}_1}$	$[0, 0, 1]^T$	0	0
$\frac{\partial(\cdot)}{\partial \mathcal{P}_2}$	0	$\cos(u_2) \hat{\xi}_{21} + \sin(u_2) \hat{\xi}_{22}$	0
$\frac{\partial(\cdot)}{\partial \mathcal{P}_3}$	0	0	$\cos(u_3) \hat{\xi}_{31} + \sin(u_3) \hat{\xi}_{32}$

Table 4.1: Summary of sensitivity information needed to determine the analytic gradient at node  $\mathcal{N}$  for the cylinder-cylinder-plane intersection problem.

as second-order expansions of the form

$$f(u + \delta u, v + \delta v) = f(u, v) + \frac{\partial f}{\partial u} \delta u + \frac{\partial f}{\partial v} \delta v + \mathcal{O}(\delta u^2, \delta v^2) + \dots = 0.$$

With this expansion Newton's method trivially finds each intersection  $(u, v)$  by solving the linear  $2 \times 2$  system

$$\begin{bmatrix} \left. \frac{\partial f}{\partial u} \right|_1 & \left. \frac{\partial f}{\partial v} \right|_1 \\ \left. \frac{\partial f}{\partial u} \right|_2 & \left. \frac{\partial f}{\partial v} \right|_2 \end{bmatrix}_i \begin{bmatrix} \delta u \\ \delta v \end{bmatrix}_i = \begin{bmatrix} -f(u, v)_1 \\ -f(u, v)_2 \end{bmatrix}_i \quad (4.23)$$

for  $[\delta u, \delta v]_i^T$  and iterating with  $u_{i+1} = u_i + \delta u_i$  and  $v_{i+1} = v_i + \delta v_i$  until  $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3 \leq \epsilon$  for some tolerance,  $\epsilon$ . Note that the subscripts 1 and 2 in (4.23) refer to the components in each vector-equation  $\mathcal{R}_1, \mathcal{R}_2$  or  $\mathcal{R}_3$ . Since the sensitivities in Table 4.1 only require information for  $(u_2, v_2)$  and  $(u_3, v_3)$ , only  $\mathcal{R}_2$  and  $\mathcal{R}_3$  appear. The initial values  $(u_0, v_0)$  are chosen to ensure the appropriate intersection node is found by this procedure (two possibilities exist in this problem formulation and a similar approach is taken by CAD systems when intersection points are queried). Once  $(u_2, v_2)$  and  $(u_3, v_3)$  are determined, the coefficient matrix and right-hand side of system (4.13) are filled to calculate the desired sensitivities.

Results from this example problem are displayed in Tables 4.2 through 4.4 for the case  $d_1 = 0.6$ ,  $R_2 = 0.25$  and  $R_3 = 0.5$ . The central-difference calculations use a step-size of



$1.0 \times 10^{-8}$  for each parameter. Excellent agreement is seen among the three methods to validate the closed-form matrix approach.

$\frac{\partial \eta}{\partial d_1}$	$x$	$y$	$z$
<b>Analytic</b>	0.225017580185205	0.436435780471985	1.0000000000000000
<b>Finite-Difference</b>	0.22501758 <u>2344833</u>	0.43643578 <u>1958622</u>	1.00000000 <u>5024759</u>
<b>Constrained Method</b>	0.225017580185205	0.436435780471985	1.0000000000000000

Table 4.2: Comparison between the closed-form method, central-difference approximation and an analytic solution for the node sensitivity to  $d_1$ . Underlined digits denote mismatches with the analytic solution.

$\frac{\partial \eta}{\partial R_2}$	$x$	$y$	$z$
<b>Analytic</b>	-0.562543950463012	-1.091089451179962	0.0000000000000000
<b>Finite-Difference</b>	-0.562543950 <u>310967</u>	-1.09108945 <u>0733218</u>	0.0000000000000000
<b>Constrained Method</b>	-0.562543950463012	-1.091089451179962	0.0000000000000000

Table 4.3: Comparison between the closed-form method, central-difference approximation and an analytic solution for the node sensitivity to  $R_2$ . Underlined digits denote mismatches with the analytic solution.

### 4.3 Geometry Gradients on BRep Edges

The intersection of two surfaces has insufficient geometry information for a closed-form formulation. The concepts of design velocity presented in Section 4.1 are applied to derive the sensitivity of BRep edges (specifically trim curves) using the Minimum Velocity Method. Different formulations are presented with example results.

#### 4.3.1 Derivation Using the Minimum Velocity Method

From an analytic geometry perspective, the set of intersection for two faces,  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , will have the exact spatial coordinates as points on their associated trim curve. This implies

$\frac{\partial \eta}{\partial R_3}$	$x$	$y$	$z$
<b>Analytic</b>	1.125087900926024	0.0000000000000000	0.0000000000000000
<b>Finite-Difference</b>	1.12508790 <u>3397493</u>	0.0000000000000000	0.0000000000000000
<b>Constrained Method</b>	1.125087900926024	0.0000000000000000	0.0000000000000000

Table 4.4: Comparison between the closed-form method, central-difference approximation and an analytic solution for the node sensitivity to  $R_3$ . Underlined digits denote mismatches with the analytic solution.

$\mathcal{E} = \mathcal{F}_1 \cap \mathcal{F}_2$  for vertices obtained after discretization of the face and trim curve. Another way of writing this for an element  $\mathbf{e} \in \mathcal{E}$  is  $\mathbf{e} \subset \mathcal{F}_1$  and  $\mathbf{e} \subset \mathcal{F}_2$ , which leads to the following equations that hold at a point on  $\mathcal{E}$ :

$$\begin{aligned}
\mathbf{r}_1 - \mathbf{r}_2 &= \mathbf{0} \\
\mathbf{r}_1 - \mathbf{e} &= \mathbf{0} \\
\mathbf{r}_2 - \mathbf{e} &= \mathbf{0}
\end{aligned} \tag{4.24}$$

Although the three equations in (4.24) appear redundant, they represent the view of a point in Euclidean space  $E$  as determined by three different parameterizations:  $\mathbf{r}_1((u, v)_1; \mathcal{P})$ ,  $\mathbf{r}_2((u, v)_2; \mathcal{P})$  and  $\mathbf{e}(t; \mathcal{P})$ .

A BRep may identify a trim curve as the intersection of two surfaces even though the trim curve itself is not part of either surface. This occurs when the trimming algorithms use Newton’s method to find intersection points and then interpolate them with a B-spline curve. This trim curve approximates the true intersection space-curve that would be found with analytic geometry. Each surface and trim curve are then “intersecting” within a proximity tolerance  $\epsilon_{tol}$  (defined internal to the geometry kernel) that is typically larger than machine precision  $\epsilon$  in order to improve the computational efficiency of trimming algorithms and maintain smoothness.

By applying this reality with computational geometry the analytic constraint equations

in (4.24) become

$$\begin{aligned}
\mathbf{r}_1 - \mathbf{r}_2 &= \boldsymbol{\epsilon}_1 \\
\mathbf{r}_1 - \mathbf{e} &= \boldsymbol{\epsilon}_2 \\
\mathbf{r}_2 - \mathbf{e} &= \boldsymbol{\epsilon}_3,
\end{aligned} \tag{4.25}$$

where  $\boldsymbol{\epsilon}_1$ ,  $\boldsymbol{\epsilon}_2$  and  $\boldsymbol{\epsilon}_3$  are the offset reached when the intersection search algorithm terminates its Newton method and  $|\boldsymbol{\epsilon}_1|, |\boldsymbol{\epsilon}_2|, |\boldsymbol{\epsilon}_3| \leq \epsilon_{tol}$ . This indicates that the set of intersection points  $\mathcal{E}$  has the property  $\mathcal{E} \neq \mathcal{F}_1 \cap \mathcal{F}_2$ . A BRep in this circumstance will contain the faces  $\mathcal{F}_1$  and  $\mathcal{F}_2$  (each with adjacent boundaries  $\mathcal{F}_{1,edge} \subset \mathcal{F}_1$  and  $\mathcal{F}_{2,edge} \subset \mathcal{F}_2$ , respectively) along with the trim curve  $\mathcal{E}$ . The nearest points between the three BRep entities  $\mathcal{F}_{1,edge}$ ,  $\mathcal{F}_{2,edge}$  and  $\mathcal{E}$  have a relative proximity that is within a ball  $\mathcal{B}$  of radius  $\epsilon_{tol}$ . In this light, the constraint equations in (4.25) are not exactly redundant and still represent the view of an approximated surface-surface intersection from the perspective of three parameterizations. This approach ensures that all of the local information from the topology is used in providing an estimate for the geometry gradient. If the trim curve modeled the true intersection space-curve well, the first relation in (4.25) would suffice because the trim curve is simply a function of the intersecting surfaces.

The inexact constraints in (4.25) require some point  $\mathbf{r}_1^*$  at a  $(u, v)_1^* \in W_1$  that minimizes  $|\mathbf{r}_1^*((u, v)_1^*; \mathcal{P}) - \mathbf{e}(t; \mathcal{P})|$  for a corresponding  $t \in T$  on the curve. Similarly, on face  $\mathcal{F}_2$  some point  $\mathbf{r}_2^*$  at a  $(u, v)_2^* \in W_2$  is needed that minimizes  $|\mathbf{r}_2^*((u, v)_2^*; \mathcal{P}) - \mathbf{e}(t; \mathcal{P})|$ . By providing the vertex coordinates of  $\mathbf{e}(t; \mathcal{P})$ , the geometry kernel can return the points  $\mathbf{r}_1^*$  and  $\mathbf{r}_2^*$  (including directional derivative information at these points) on faces  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , respectively, that are nearest to  $\mathbf{e}(t; \mathcal{P})$ .

A variational analysis is now used for the geometry gradient on trim curve  $\mathcal{E}$ . This derivative is with respect to a parameter  $\mathcal{P}$  common to  $\mathcal{F}_1$ ,  $\mathcal{F}_2$  and  $\mathcal{E}$ . It may be possible that  $\epsilon_{tol} = \epsilon_{tol}(\mathcal{P})$ , yet an expression for this could only be arbitrarily specified since access to the CAD geometry kernel source code is usually not available. Thus, we assume that  $\epsilon_{tol} \neq \epsilon_{tol}(\mathcal{P})$ , as if  $\epsilon_{tol}$  were a fixed value in the geometry kernel.

A Taylor expansion around  $\mathbf{r}_1^*((u, v)_1^*; \mathcal{P})$ ,  $\mathbf{r}_2^*((u, v)_2^*; \mathcal{P})$  and  $\mathbf{e}(t; \mathcal{P})$  is considered by

writing

$$\begin{aligned}
\mathbf{r}_1^*((u, v)_1^*; \mathcal{P} + \delta\mathcal{P})' &= \mathbf{r}_1^*((u, v)_1^*; \mathcal{P}) + \frac{d\mathbf{r}_1^*}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \dots \\
\mathbf{r}_2^*((u, v)_2^*; \mathcal{P} + \delta\mathcal{P})' &= \mathbf{r}_2^*((u, v)_2^*; \mathcal{P}) + \frac{d\mathbf{r}_2^*}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \dots \\
\mathbf{e}(t; \mathcal{P} + \delta\mathcal{P})' &= \mathbf{e}(t; \mathcal{P}) + \frac{d\mathbf{e}}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2\mathbf{e}}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \dots .
\end{aligned} \tag{4.26}$$

It is assumed that the parameter perturbation,  $\delta\mathcal{P}$ , is sufficiently small such that topology is preserved after regeneration with the new parameter value  $\mathcal{P} + \delta\mathcal{P}$ . The regenerated instance would also yield offset values  $\epsilon'_1$ ,  $\epsilon'_2$  and  $\epsilon'_3$  in (4.25). In this scenario,  $|\epsilon'_1 - \epsilon_1| < \epsilon_{tol}$ ,  $|\epsilon'_2 - \epsilon_2| < \epsilon_{tol}$  and  $|\epsilon'_3 - \epsilon_3| < \epsilon_{tol}$  are true, which from an implementation standpoint means  $\epsilon'_1 = \epsilon_1$ ,  $\epsilon'_2 = \epsilon_2$  and  $\epsilon'_3 = \epsilon_3$ . This permits rewriting the first equation in (4.25) as

$$\begin{aligned}
\epsilon'_1 &= \mathbf{r}_1^*((u, v)_1^*; \mathcal{P} + \delta\mathcal{P})' - \mathbf{r}_2^*((u, v)_2^*; \mathcal{P} + \delta\mathcal{P})' \\
&= \underbrace{\mathbf{r}_1^*((u, v)_1^*; \mathcal{P}) - \mathbf{r}_2^*((u, v)_2^*; \mathcal{P})}_{\epsilon_1} + \left(\frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}}\right)\delta\mathcal{P} + \frac{1}{2}\left(\frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2}\right)\delta\mathcal{P}^2 + \dots \\
\mathbf{0} &= \left(\frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}}\right)\delta\mathcal{P} + \frac{1}{2}\left(\frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2}\right)\delta\mathcal{P}^2 + \dots .
\end{aligned} \tag{4.27}$$

The second equation in (4.25) can then be rewritten as

$$\begin{aligned}
\epsilon'_2 &= \mathbf{r}_1^*((u, v)_1^*; \mathcal{P} + \delta\mathcal{P})' - \mathbf{e}(t; \mathcal{P} + \delta\mathcal{P})' \\
&= \underbrace{\mathbf{r}_1^*((u, v)_1^*; \mathcal{P}) - \mathbf{e}(t; \mathcal{P})}_{\epsilon_2} + \left(\frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}}\right)\delta\mathcal{P} + \frac{1}{2}\left(\frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{e}}{d\mathcal{P}^2}\right)\delta\mathcal{P}^2 + \dots \\
\mathbf{0} &= \left(\frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}}\right)\delta\mathcal{P} + \frac{1}{2}\left(\frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{e}}{d\mathcal{P}^2}\right)\delta\mathcal{P}^2 + \dots .
\end{aligned} \tag{4.28}$$

The final equation in (4.25) becomes

$$\begin{aligned}
\epsilon'_3 &= \mathbf{r}_2^*((u, v)_2^*; \mathcal{P} + \delta\mathcal{P})' - \mathbf{e}(t; \mathcal{P} + \delta\mathcal{P})' \\
&= \underbrace{\mathbf{r}_2^*((u, v)_2^*; \mathcal{P}) - \mathbf{e}(t; \mathcal{P})}_{\epsilon_3} + \left(\frac{d\mathbf{r}_2^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}}\right)\delta\mathcal{P} + \frac{1}{2}\left(\frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{e}}{d\mathcal{P}^2}\right)\delta\mathcal{P}^2 + \dots \\
\mathbf{0} &= \left(\frac{d\mathbf{r}_2^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}}\right)\delta\mathcal{P} + \frac{1}{2}\left(\frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{e}}{d\mathcal{P}^2}\right)\delta\mathcal{P}^2 + \dots .
\end{aligned} \tag{4.29}$$

In order for (4.27), (4.28) and (4.29) to hold to first-order in  $\delta\mathcal{P}$ , we must have

$$\begin{aligned}\frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}} &= \mathbf{0} \\ \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}} &= \mathbf{0} \\ \frac{d\mathbf{r}_2^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}} &= \mathbf{0},\end{aligned}\tag{4.30}$$

where each term can be expanded as

$$\begin{aligned}\frac{d\mathbf{r}_1^*}{d\mathcal{P}} &= \frac{\partial\mathbf{r}_1^*}{\partial u_1} \frac{\partial u_1}{\partial\mathcal{P}} + \frac{\partial\mathbf{r}_1^*}{\partial v_1} \frac{\partial v_1}{\partial\mathcal{P}} + \frac{\partial\mathbf{r}_1^*}{\partial\mathcal{P}} = \dot{\mathbf{r}}_1^* + \nabla\mathbf{r}_1^* \cdot \boldsymbol{\nu}_1 \\ \frac{d\mathbf{r}_2^*}{d\mathcal{P}} &= \frac{\partial\mathbf{r}_2^*}{\partial u_2} \frac{\partial u_2}{\partial\mathcal{P}} + \frac{\partial\mathbf{r}_2^*}{\partial v_2} \frac{\partial v_2}{\partial\mathcal{P}} + \frac{\partial\mathbf{r}_2^*}{\partial\mathcal{P}} = \dot{\mathbf{r}}_2^* + \nabla\mathbf{r}_2^* \cdot \boldsymbol{\nu}_2 \\ \frac{d\mathbf{e}}{d\mathcal{P}} &= \frac{\partial\mathbf{e}}{\partial t} \frac{\partial t}{\partial\mathcal{P}} + \frac{\partial\mathbf{e}}{\partial\mathcal{P}} = \dot{\mathbf{e}} + \tilde{\nabla}\mathbf{e} \cdot \tilde{\boldsymbol{\nu}}.\end{aligned}\tag{4.31}$$

Then (4.30) is written using (4.31) to form the system of equations,

$$\begin{bmatrix} \nabla\mathbf{r}_1^* & -\nabla\mathbf{r}_2^* & \mathbf{0} & \mathbf{0} \\ \nabla\mathbf{r}_1^* & \mathbf{0} & -\tilde{\nabla}\mathbf{e} & -\mathbf{I} \\ \mathbf{0} & \nabla\mathbf{r}_2^* & -\tilde{\nabla}\mathbf{e} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \\ \tilde{\boldsymbol{\nu}} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_2^* \end{bmatrix},\tag{4.32}$$

which expands to

$$\begin{bmatrix} \frac{\partial\mathbf{r}_1^*}{\partial u_1} & \frac{\partial\mathbf{r}_1^*}{\partial v_1} & -\frac{\partial\mathbf{r}_2^*}{\partial u_2} & -\frac{\partial\mathbf{r}_2^*}{\partial v_2} & \mathbf{0} & \mathbf{0} \\ \frac{\partial\mathbf{r}_1^*}{\partial u_1} & \frac{\partial\mathbf{r}_1^*}{\partial v_1} & \mathbf{0} & \mathbf{0} & -\frac{\partial\mathbf{e}}{\partial t} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \frac{\partial\mathbf{r}_2^*}{\partial u_2} & \frac{\partial\mathbf{r}_2^*}{\partial v_2} & -\frac{\partial\mathbf{e}}{\partial t} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \partial u_1 / \partial\mathcal{P} \\ \partial v_1 / \partial\mathcal{P} \\ \partial u_2 / \partial\mathcal{P} \\ \partial v_2 / \partial\mathcal{P} \\ \partial t / \partial\mathcal{P} \\ \partial\mathbf{e} / \partial\mathcal{P} \end{bmatrix} = \begin{bmatrix} \frac{\partial\mathbf{r}_2^*}{\partial\mathcal{P}} - \frac{\partial\mathbf{r}_1^*}{\partial\mathcal{P}} \\ -\frac{\partial\mathbf{r}_1^*}{\partial\mathcal{P}} \\ -\frac{\partial\mathbf{r}_2^*}{\partial\mathcal{P}} \end{bmatrix},\tag{4.33}$$

where  $\mathbf{I}$  is the identity matrix. The system in (4.33) is over-determined and can be solved in a least-squares sense as  $\mathbf{Ax} = \mathbf{b}$ . The least-squares solution,  $\mathbf{x}^*$ , is obtained by minimizing  $R_{min} = \|\mathbf{Ax}^* - \mathbf{b}\|_2$ . Since there is no guarantee that  $R_{min}$  will have order of magnitude  $\epsilon$ , substituting the components of  $\mathbf{x}^*$  back into (4.31) may result in violations of (4.30) on

the order of  $\mathcal{O}(R_{min})$ :

$$\begin{aligned}\frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}} &= \mathcal{O}(R_{min}) \neq 0 \\ \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}} &= \mathcal{O}(R_{min}) \neq 0 \\ \frac{d\mathbf{r}_2^*}{d\mathcal{P}} - \frac{d\mathbf{e}}{d\mathcal{P}} &= \mathcal{O}(R_{min}) \neq 0,\end{aligned}$$

In this situation the vertex at  $\mathbf{e}$  will need to be assigned one of three possible sensitivity values:

$$\frac{d\mathbf{r}_1^*}{d\mathcal{P}} \neq \frac{d\mathbf{r}_2^*}{d\mathcal{P}} \neq \frac{d\mathbf{e}}{d\mathcal{P}}.$$

This is remedied by augmenting the system in (4.33) with additional constraint equations in order to provide a single design velocity vector,  $\mathcal{V}$ , at any location along  $\mathcal{E}$ :

$$\mathcal{V} = \frac{\partial \mathbf{r}_1^*}{\partial u_1} \frac{\partial u_1}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}_1^*}{\partial v_1} \frac{\partial v_1}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}_1^*}{\partial \mathcal{P}} = \frac{\partial \mathbf{r}_2^*}{\partial u_2} \frac{\partial u_2}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}_2^*}{\partial v_2} \frac{\partial v_2}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}_2^*}{\partial \mathcal{P}} = \frac{\partial \mathbf{e}}{\partial t} \frac{\partial t}{\partial \mathcal{P}} + \frac{\partial \mathbf{e}}{\partial \mathcal{P}}. \quad (4.34)$$

By including these additional constraints into the over-determined system and augmenting  $\mathbf{x}$  with  $\mathcal{V}$ , we obtain the new system

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \nabla \mathbf{r}_1^* & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} \\ \mathbf{0} & \nabla \mathbf{r}_2^* & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} \\ -\nabla \mathbf{r}_1^* & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I & I \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \tilde{\nu} \\ \dot{\mathbf{e}} \\ \mathcal{V} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_2^* \\ \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_2^* \\ \mathbf{0} \end{bmatrix}. \quad (4.35)$$

We can then write (4.35) in a block structure form with

$$\underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{G} & \mathbf{A}_I \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{x} \\ \mathcal{V} \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} \mathbf{b} \\ \mathbf{b}_G \end{bmatrix}}_{\mathbf{B}}, \quad (4.36)$$

where  $\mathbf{A}$ ,  $\mathbf{x}$  and  $\mathbf{b}$  are defined as in (4.33) and

$$\mathbf{G} = \begin{bmatrix} -\frac{d\mathbf{r}_1^*}{du_1} & -\frac{d\mathbf{r}_1^*}{dv_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{d\mathbf{r}_2^*}{du_2} & -\frac{d\mathbf{r}_2^*}{dv_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\frac{d\mathbf{e}}{dt} & -\mathbf{I} \end{bmatrix}, \quad \mathbf{A}_I = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \end{bmatrix}, \quad \mathbf{b}_G = \begin{bmatrix} \frac{\partial \mathbf{r}_1^*}{\partial \mathcal{P}} \\ \frac{\partial \mathbf{r}_2^*}{\partial \mathcal{P}} \\ \mathbf{0} \end{bmatrix}.$$

The system in (4.36) remains over-determined with the form  $\mathbb{A}\mathbb{X} = \mathbb{B}$  and is solved in a least-squares sense to give  $\tilde{R}_{min} = \|\mathbb{A}\mathbb{X}^* - \mathbb{B}\|_2$  with solution  $\mathbb{X}^*$ . When  $\tilde{R}_{min} > \epsilon$ , the resulting sensitivity vector  $\mathcal{V}$  is equivalent to a weighted linear combination of the vectors  $\frac{d\mathbf{r}_1^*}{d\mathcal{P}} \neq \frac{d\mathbf{r}_2^*}{d\mathcal{P}} \neq \frac{d\mathbf{e}}{d\mathcal{P}}$ . Each is obtained by back-substituting the components  $\nu_1, \nu_2, \tilde{\nu}$  and  $\dot{\mathbf{e}}$  of  $\mathbb{X}^*$  into (4.31). In this case, we can write

$$\mathcal{V} = \lambda_1 \frac{d\mathbf{r}_1^*}{d\mathcal{P}} + \lambda_2 \frac{d\mathbf{r}_2^*}{d\mathcal{P}} + \lambda_3 \frac{d\mathbf{e}}{d\mathcal{P}}$$

and determine the weights  $\lambda_1, \lambda_2$  and  $\lambda_3$  by setting up a  $3 \times 3$  system. On the other hand, if the solution to the least-squares problem gives  $\mathcal{O}(\tilde{R}_{min}) \approx \epsilon$ , then the weights become  $\lambda_1, \lambda_2, \lambda_3 \approx \frac{1}{3}$  and

$$\mathcal{V} \approx \frac{d\mathbf{r}_1^*}{d\mathcal{P}} \approx \frac{d\mathbf{r}_2^*}{d\mathcal{P}} \approx \frac{d\mathbf{e}}{d\mathcal{P}}$$

to within machine-precision.

### Minimal Velocity Approach using Singular Value Decomposition

In order to solve the system  $\mathbb{A}\mathbb{X} = \mathbb{B}$  in a least-squares sense, it is possible to use the normal equations, QR decomposition or the singular value decomposition (SVD). The SVD is known to be the most robust approach when  $\mathbb{A}$  is rank-deficient, which can be the case when constructing  $\mathbb{A}$  with geometry information along a trim curve.

We consider  $\mathbb{A} \in \mathbb{R}^{m \times n}$  and write its truncated SVD as  $\mathbb{A} = \hat{\mathbf{U}}\tilde{\Sigma}\hat{\mathbf{V}}^T$ , where  $\hat{\mathbf{U}} \in \mathbb{R}^{m \times m}$ ,  $\hat{\mathbf{V}} \in \mathbb{R}^{m \times n}$  and  $\tilde{\Sigma} \in \mathbb{R}^{m \times n}$  can have nonzero entries  $\tilde{\sigma}_i$  (denoted singular values) only on the main diagonal. The singular values are ordered in a non-increasing manner  $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \dots \geq \tilde{\sigma}_{min} \geq TOL$ , where  $TOL \geq \epsilon \cdot \|\mathbb{A}\|_2$  and all singular values less than  $TOL$  from the full SVD of  $\mathbb{A}$  are discarded. This is useful because the solution norm  $\|\mathbb{X}\|_2$  will depend on the inverse of the smallest singular value  $\tilde{\sigma}_{min}$ , which otherwise will have magnitude  $\epsilon$  in the full SVD of a rank-deficient  $\mathbb{A}$ . We can then use the pseudoinverse to obtain a solution

as

$$\mathbb{X} = \left( \hat{\mathbf{V}} \tilde{\Sigma}^{-1} \hat{\mathbf{U}}^T \right) \mathbb{B}. \quad (4.37)$$

When  $\mathbb{A}$  has full rank, the solution  $\mathbb{X}^*$  is unique. When this condition is not satisfied the solution is non-unique because adding any vector projected in the null space of  $\mathbb{A}$  to  $\mathbb{X}$  will also satisfy  $\mathbb{A}\mathbb{X} = \mathbb{B}$ . In this case, a unique solution  $\mathbb{X}_{\min}^*$  is chosen from the null space of  $\mathbb{A}$  with minimum norm  $\|\mathbb{X}_{\min}^*\|_2$ . Although this choice is standard for the pseudoinverse, its geometry interpretation implies a conservative estimate of the true intersection design velocity. The minimum norm solution  $\mathbb{X}_{\min}^*$  has minimum relative velocity magnitudes  $\boldsymbol{\nu}_1$ ,  $\boldsymbol{\nu}_2$  and  $\tilde{\boldsymbol{\nu}}$ , which leads to the most conservative estimate of sensitivity  $\mathcal{V}$  from the null space of  $\mathbb{A}$ . Other options from that null space have larger relative velocity magnitudes. With  $\mathcal{V}$  stemming from  $\mathbb{X}_{\min}^*$  a perturbed intersection curve  $\mathbf{e}(t; \mathcal{P})' = \mathbf{e}(t; \mathcal{P}) + \mathcal{V} \delta \mathcal{P}$  is minimally displaced, with respect to the original intersecting faces and trim curve, compared to using a different  $\mathcal{V}$  from other  $\mathbb{X}$  possibilities. In addition, choosing this design velocity results in a minimum “*design energy*” trajectory in the face and trim curve domains, where design energy can be represented in the domain space of  $\mathcal{F}_1$ ,  $\mathcal{F}_2$  and  $\mathcal{E}$  as  $\frac{1}{2}\|\boldsymbol{\nu}_1\|_2^2$ ,  $\frac{1}{2}\|\boldsymbol{\nu}_2\|_2^2$  and  $\frac{1}{2}\|\tilde{\boldsymbol{\nu}}\|_2^2$ , respectively.

An additional feature to this minimum velocity approach is that the resulting design velocity vector contains a component that lies in the direction of the null space of  $\mathbb{A}$ . This component direction is actually the tangent vector direction of the trim curve  $\mathcal{E}$  at the vertex  $\mathbf{e}$ . We show this is the case by writing the tangent at  $\mathbf{e}$  in a conventional manner using the local directional derivatives of  $\mathcal{F}_1$  and  $\mathcal{F}_2$  at the intersection. First, from the perspective of  $\mathcal{F}_1$  we write

$$\begin{aligned} \frac{1}{|\partial \mathbf{e} / \partial t|_2} \frac{\partial \mathbf{e}}{\partial t} &= \frac{\mathbf{n}_1 \times \mathbf{n}_2}{|\mathbf{n}_1 \times \mathbf{n}_2|} \\ &= \frac{1}{|\mathbf{n}_1 \times \mathbf{n}_2|} \left( \frac{\partial \vec{\mathbf{r}}_1}{\partial u_1} \times \frac{\partial \vec{\mathbf{r}}_1}{\partial v_1} \right) \times \left( \frac{\partial \vec{\mathbf{r}}_2}{\partial u_2} \times \frac{\partial \vec{\mathbf{r}}_2}{\partial v_2} \right) \\ &= \frac{\partial \vec{\mathbf{r}}_1}{\partial u_1} \left( \frac{\mathbf{n}_2 \cdot \partial \vec{\mathbf{r}}_1 / \partial v_1}{|\mathbf{n}_1 \times \mathbf{n}_2|} \right) - \frac{\partial \vec{\mathbf{r}}_1}{\partial v_1} \left( \frac{\mathbf{n}_2 \cdot \partial \vec{\mathbf{r}}_1 / \partial u_1}{|\mathbf{n}_1 \times \mathbf{n}_2|} \right) \\ &= \frac{\partial \vec{\mathbf{r}}_1}{\partial u_1} \omega_1 + \frac{\partial \vec{\mathbf{r}}_1}{\partial v_1} \omega_2 \end{aligned} \quad (4.38)$$

where  $\mathbf{n}_1$  is the normal vector at  $\vec{\mathbf{r}}_1$  on  $\mathcal{F}_1$  and  $\mathbf{n}_2$  is the normal vector at  $\vec{\mathbf{r}}_2$  on  $\mathcal{F}_2$ . Here



we also have

$$\omega_1 = \frac{\mathbf{n}_2 \cdot \partial \vec{\mathbf{r}}_1 / \partial v_1}{|\mathbf{n}_1 \times \mathbf{n}_2|} \quad , \quad \omega_2 = -\frac{\mathbf{n}_2 \cdot \partial \vec{\mathbf{r}}_1 / \partial u_1}{|\mathbf{n}_1 \times \mathbf{n}_2|}.$$

We then write the tangent from the perspective of  $\mathcal{F}_2$  as

$$\begin{aligned} \frac{1}{|\partial \mathbf{e} / \partial t|_2} \frac{\partial \mathbf{e}}{\partial t} &= -\frac{\mathbf{n}_2 \times \mathbf{n}_1}{|\mathbf{n}_2 \times \mathbf{n}_1|_2} \\ &= -\frac{1}{|\mathbf{n}_2 \times \mathbf{n}_1|_2} \left( \frac{\partial \vec{\mathbf{r}}_2}{\partial u_2} \times \frac{\partial \vec{\mathbf{r}}_2}{\partial v_2} \right) \times \left( \frac{\partial \vec{\mathbf{r}}_1}{\partial u_1} \times \frac{\partial \vec{\mathbf{r}}_1}{\partial v_1} \right) \\ &= -\frac{\partial \vec{\mathbf{r}}_2}{\partial u_2} \left( \frac{\mathbf{n}_1 \cdot \partial \vec{\mathbf{r}}_2 / \partial v_2}{|\mathbf{n}_2 \times \mathbf{n}_1|_2} \right) + \frac{\partial \vec{\mathbf{r}}_2}{\partial v_2} \left( \frac{\mathbf{n}_1 \cdot \partial \vec{\mathbf{r}}_2 / \partial u_2}{|\mathbf{n}_2 \times \mathbf{n}_1|_2} \right) \\ &= \frac{\partial \vec{\mathbf{r}}_2}{\partial u_2} \omega_3 + \frac{\partial \vec{\mathbf{r}}_2}{\partial v_2} \omega_4 \end{aligned} \tag{4.39}$$

where

$$\omega_3 = -\frac{\mathbf{n}_1 \cdot \partial \vec{\mathbf{r}}_2 / \partial v_2}{|\mathbf{n}_2 \times \mathbf{n}_1|_2} \quad , \quad \omega_4 = \frac{\mathbf{n}_1 \cdot \partial \vec{\mathbf{r}}_2 / \partial u_2}{|\mathbf{n}_2 \times \mathbf{n}_1|_2}.$$

Both (4.38) and (4.39) have a form similar to (4.31), except with the terms  $\dot{\vec{\mathbf{r}}}_1 = 0$  and  $\dot{\vec{\mathbf{r}}}_2 = 0$ . Therefore, if we substitute (4.38) and (4.39) into (4.30) we obtain the system

$$\begin{bmatrix} \frac{\partial \vec{\mathbf{r}}_1}{\partial u_1} & \frac{\partial \vec{\mathbf{r}}_1}{\partial v_1} & -\frac{\partial \vec{\mathbf{r}}_2}{\partial u_2} & -\frac{\partial \vec{\mathbf{r}}_2}{\partial v_2} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \vec{\mathbf{r}}_1}{\partial u_1} & \frac{\partial \vec{\mathbf{r}}_1}{\partial v_1} & \mathbf{0} & \mathbf{0} & -\frac{\partial \mathbf{e}}{\partial t} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \vec{\mathbf{r}}_2}{\partial u_2} & \frac{\partial \vec{\mathbf{r}}_2}{\partial v_2} & -\frac{\partial \mathbf{e}}{\partial t} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \tilde{\mathbf{v}} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.40}$$

Since  $\omega_1, \omega_2, \omega_3, \omega_4 \neq 0$  are already known from the geometry, we can solve for  $\tilde{\mathbf{v}}$  and  $\dot{\mathbf{e}}$  to determine the direction of the null space of  $\mathbf{A}$  as

$$\begin{aligned} \mathbf{x}_0 &= \left[ \omega_1 \quad \omega_2 \quad \omega_3 \quad \omega_4 \quad \tilde{\mathbf{v}} \quad \dot{\mathbf{e}} \right]^T \\ &= \left[ \frac{\partial u_1}{\partial \mathcal{P}} \quad \frac{\partial v_1}{\partial \mathcal{P}} \quad \frac{\partial u_2}{\partial \mathcal{P}} \quad \frac{\partial v_2}{\partial \mathcal{P}} \quad \tilde{\mathbf{v}} \quad \dot{\mathbf{e}} \right]^T \end{aligned}$$

which proves that the null space is in the direction of the trim curve tangent. From (4.36) it

is clear that this result applies to  $\mathbb{A}$  as well because its null space contains the null space of  $\mathbf{A}$ . This is a consequence of using the Minimal Velocity method. The design velocity vectors it computes will always have a component aligned with the local trim curve tangent vector. This component of the sensitivity estimate may be undesirable if the true sensitivity does not have it. However, small perturbations along this sensitivity direction will approximate the trim curve design motion well away from the curve end-points, as seen in Section 4.3.3. The estimate may not predict design motion well near a node since information about other edges at the node are not included in this formulation.

### 4.3.2 Additional System Augmentation Options

The geometry constraint equations for intersection curves can be setup in multiple ways. Information about each intersecting face and the trim curve can be combined or omitted. In general, a unique solution does not exist due to lacking geometry information, thus from a geometry standpoint there is no preferred arrangement and different formulations will yield results that share the *same* null space of  $\mathbb{A}$ .

An argument can be made to use one formulation over another for a specific implementation. For example, using trim curve information (when available) ensures that the most relevant geometry is utilized in a design velocity calculation<sup>1</sup>. Another option is to augment the geometry constraint system with constraints that enforce the same design velocity on each intersecting entity. Without this augmentation the design velocity must be constructed using  $\mathbb{X}_{\min}^*$  in (4.3) and/or (4.5). A different augmentation also allows the design velocity to vary among intersecting entities, thus a problem in the setup can be detected if the velocities do not match. Combinations of these options are presented here as seven different Methods. Each results in a different design velocity field along a trim curve, yet similar design motion for small perturbations. The advantage of including less geometry constraints is seen in a simpler formulation for implementation. However, a tradeoff with lower quality design velocity occurs in these cases that is not seen when more geometry constraints are included.

---

<sup>1</sup>The trim curve is typically a B-spline curve that interpolates surface intersection points found by the geometry kernel trim algorithms. Thus, since the trim curve only *approximates* the true intersection curve, the design velocity along the trim curve will be driven by the intersecting surfaces and B-spline curve geometry.

Method	Edge Info. Included	Constrained Design Velocity	Solution Check
1	<i>No</i>	<i>Yes</i>	<i>No</i>
2	<i>Yes</i>	<i>Yes*</i>	<i>No</i>
3	<i>Yes</i>	<i>Yes</i>	<i>No</i>
4	<i>No</i>	<i>No</i>	<i>No</i>
5	<i>No</i>	<i>Yes</i>	<i>Yes</i>
6	<i>Yes</i>	<i>No</i>	<i>No</i>
7	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>

\* Constrained to a fixed perturbation direction.

Table 4.5: Advantages and disadvantages of each trim curve sensitivity method.

### Method 1: Augmented System with Single $\nu$ , No Trim Curve Information

This approach is similar to that of (4.35), yet no trim curve information is utilized. A single design velocity constraint is enforced for each face.

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} \\ -\nabla \mathbf{r}_1^* & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\nabla \mathbf{r}_2^* & \mathbf{I} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_2^* \end{bmatrix}. \quad (4.41)$$

### Method 2: Constrained Method

This approach constrains the design velocity to a prescribed perturbation direction that is normal to the local tangent on the curve. A different prescribed direction is also possible, yet in either case the chosen direction may have no correlation to the true design motion. The design velocity is computed in a second-step by substituting the computed  $\nu_1$  and  $\nu_2$  values into (4.3).

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* \\ \left[ \frac{\partial \mathbf{r}_1^*}{\partial u} \cdot \mathbf{s} \quad \frac{\partial \mathbf{u}_1^*}{\partial v} \cdot \mathbf{s} \right] & \mathbf{0} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_1^* \cdot \mathbf{s} \end{bmatrix}. \quad (4.42)$$

### Method 3: Augmented System with Single $\nu$ , Includes Trim Curve Information

This method is discussed in Section 4.3. A single design velocity constraint is used for each face and the trim curve. If geometry “noise” is present, then this added constraint averages the differing design velocity from each component.

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \nabla \mathbf{r}_1^* & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} \\ \mathbf{0} & \nabla \mathbf{r}_2^* & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} \\ -\nabla \mathbf{r}_1^* & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I & I \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \tilde{\nu} \\ \dot{\mathbf{e}} \\ \nu \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_2^* \\ \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_2^* \\ \mathbf{0} \end{bmatrix}. \quad (4.43)$$

### Method 4: No Augmentation or Trim Curve Information

This is the simplest formulation to implement because only information from the two intersecting faces is used. The resulting coefficient matrix is under-determined. The design velocity is computed in a second-step by substituting the computed  $\nu_1$  and  $\nu_2$  values into (4.3).

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \end{bmatrix}. \quad (4.44)$$

### Method 5: Augmented System with Multiple $\nu$ , No Trim Curve Information

This approach is similar to that of Method 1, except that each face is not constrained to output the same design velocity along the intersection. Different results may occur if geometry “noise” (due to intersection tolerances) is greater than the tolerance level used in the pseudoinverse computation. In numerical tests where this was not an issue, the resulting design velocities are  $\nu_1 = \nu_2$ . If  $\nu_1 \neq \nu_2$ , then this indicates a problem with the geometry constraint system possibly related to geometry “noise.”

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} \\ -\nabla \mathbf{r}_1^* & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & -\nabla \mathbf{r}_2^* & \mathbf{0} & I \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_2^* \end{bmatrix}. \quad (4.45)$$

### Method 6: No Augmentation, Includes Trim Curve Information

This approach pertains to equation (4.32) in the derivation of Section 4.3, prior to augmentation of the system. The design velocity is computed in a second-step by substituting the computed  $\nu_1$ ,  $\nu_2$  and  $\tilde{\nu}$  values into (4.31).

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} \\ \nabla \mathbf{r}_1^* & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I \\ \mathbf{0} & \nabla \mathbf{r}_2^* & -\tilde{\nabla} \mathbf{e} & -I \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \tilde{\nu} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_2^* \end{bmatrix}, \quad (4.46)$$

### Method 7: Augmented System with Multiple $\nu$ , Includes Trim Curve Information

Similar to Method 3, this approach also employs the possibility of capturing the effects of geometry “noise” by computing  $\nu_1$ ,  $\nu_2$  and  $\nu_3$  separately. Again, numerical tests showed  $\nu_1 = \nu_2 = \nu_3$  when geometry “noise” was not an issue along the intersection curve. This is not the case, though, when geometry “noise” exceeds the tolerance level in the pseudoinverse calculation.

$$\begin{bmatrix} \nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \nabla \mathbf{r}_1^* & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \nabla \mathbf{r}_2^* & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\nabla \mathbf{r}_1^* & \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\nabla \mathbf{r}_2^* & \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\tilde{\nabla} \mathbf{e} & -I & \mathbf{0} & \mathbf{0} & I \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \tilde{\nu} \\ \dot{\mathbf{e}} \\ \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_2^* - \dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_1^* \\ -\dot{\mathbf{r}}_2^* \\ \dot{\mathbf{r}}_1^* \\ \dot{\mathbf{r}}_2^* \\ \mathbf{0} \end{bmatrix}. \quad (4.47)$$

### 4.3.3 Validation and Comparison of Methods

The same cone-plane intersection problem used in Section 3.1 is used for validation and comparison of the seven methods explained in Section 4.3. Although the design velocities do not agree with the analytic case due to the sensitivity component aligned with the edge tangent, they are each used to determine the design motion of the intersection curve by linearly perturbing the cutting-plane parameter  $d$  by  $\delta d = 0.0004$ . Figure 4-2 illustrates

the offset norm in design motion between the analytic and other method results<sup>2</sup>. There is better agreement in the design motion sense among these results. The “Regenerated” result is obtained by evaluating the coordinates of the new curve at the  $t$  value of the original curve after regenerating the model with  $d + \delta d$  for  $\delta d = 0.0004$ .

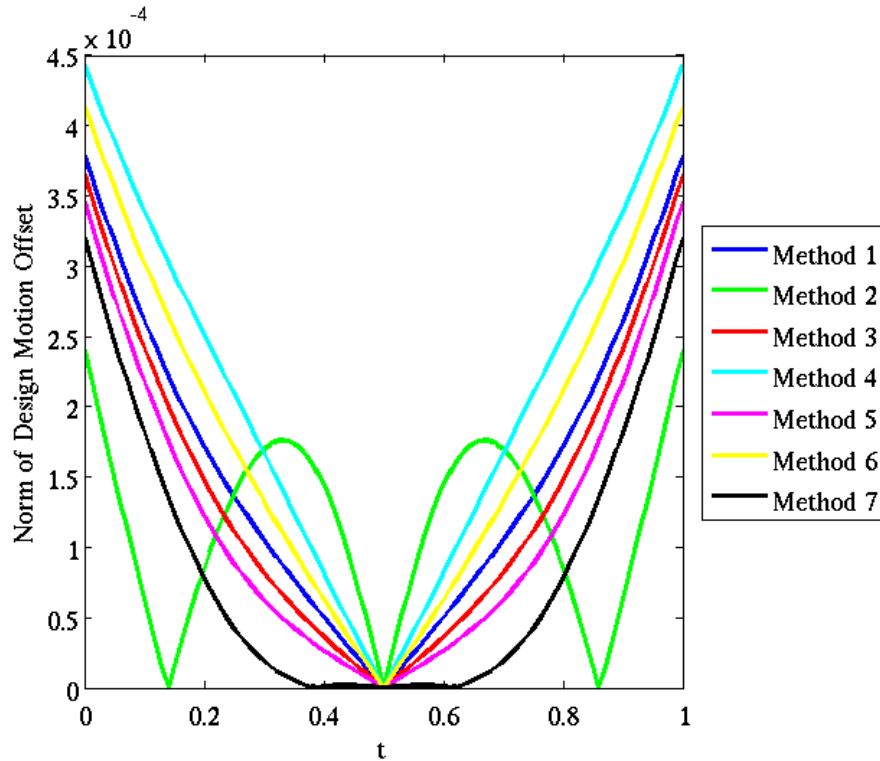


Figure 4-2: The norm of design motion offsets are plotted across  $t = [0, 1]$  for seven methods with respect to an analytic solution.

Ambiguity concerning which result is “best” is attributed to the value of  $\tilde{\nu} = \partial t / \partial \mathcal{P}$ . The magnitude of this term impacts the relative design velocity along the curve that occurs when a parameter is perturbed. The resulting  $\tilde{\nu}$  value for methods 3, 6 and 7 are plotted in Figure 4-4 to see how the added trim curve information correlates with relative design velocity. It is obvious that methods 1, 2, 4 and 5 do not calculate  $\tilde{\nu}$  and are thus not plotted. The amount of “stretching” in  $t$  occurs by the same magnitude, yet opposite direction, on either side of the hyperbola midpoint ( $t = 0.5$ ), where  $\tilde{\nu} = 0$ , for methods 3, 6 and 7.

The discrepancy in design motion seen in Figure 4-2 occurs because each method has its own inherent distribution of  $\partial t / \partial \mathcal{P}$ , as shown by the data. For this example problem,

<sup>2</sup>A design motion comparison is done between methods by checking the coordinates of the new trim curve they generate at the same  $t$  of the original curve.

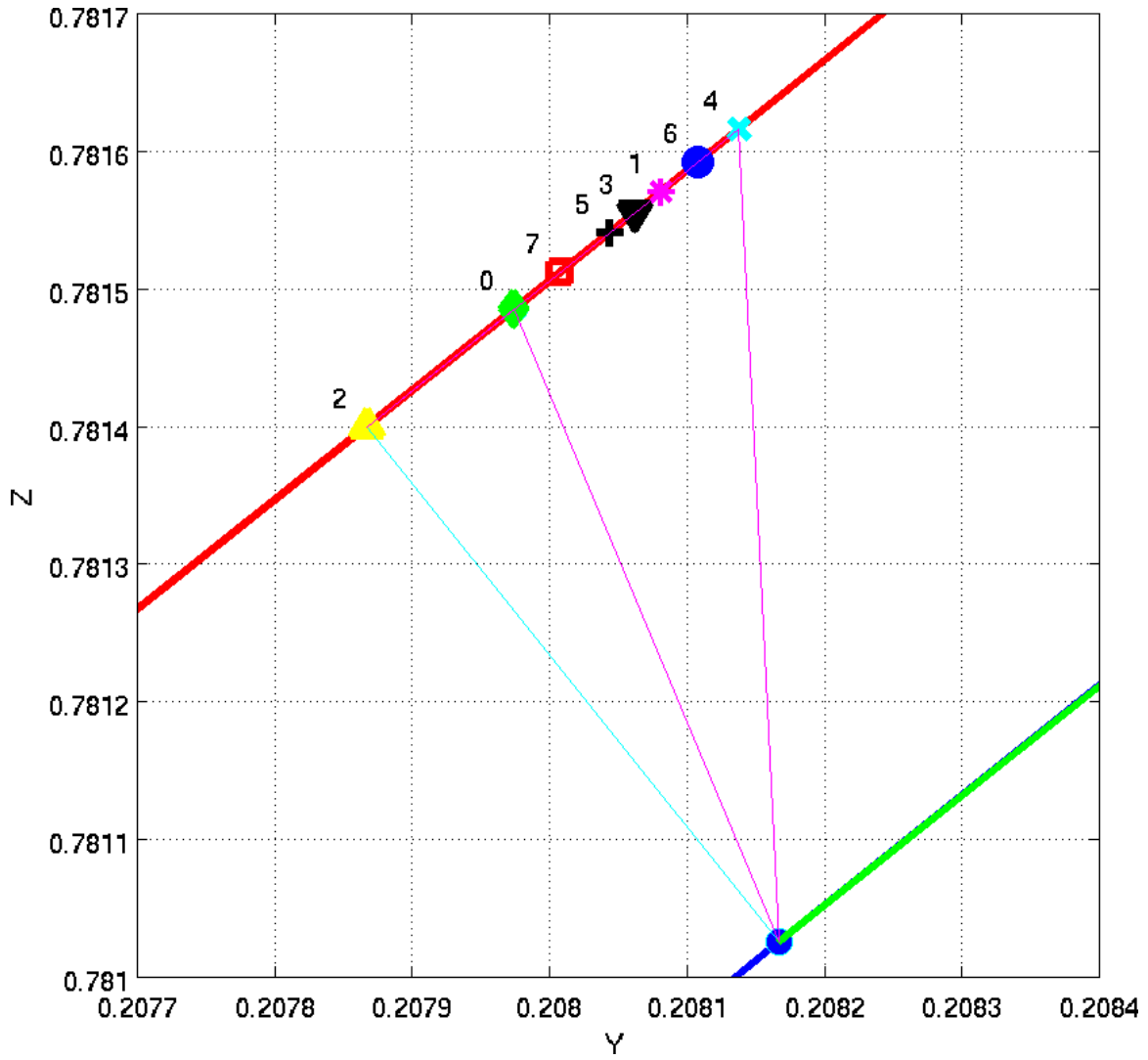


Figure 4-3: The design velocities of seven Methods (labeled 1 through 7) were used to determine design motion by a linear perturbation of the cone-plane parameter  $\delta d = 0.0004$ . The analytic result is labeled 0; the initial lower curve (blue) overlaps with a tangent vector (green) and the upper curve (red) is the perturbed curve.

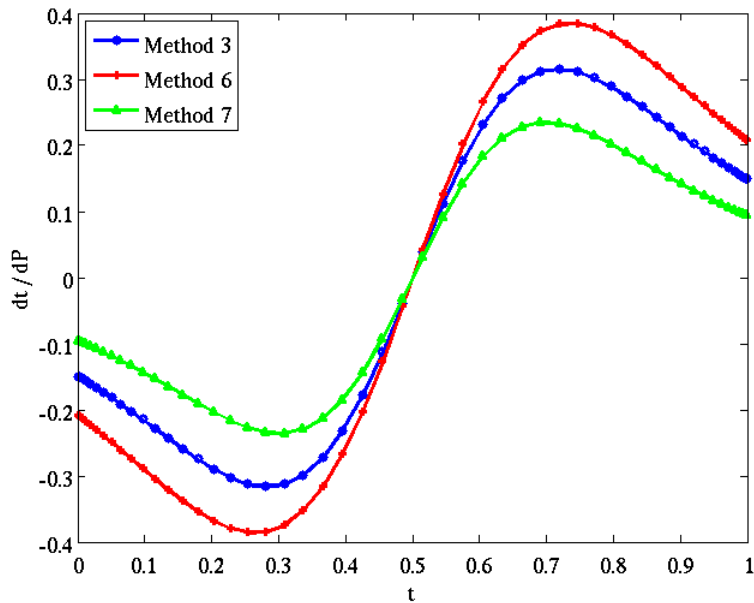
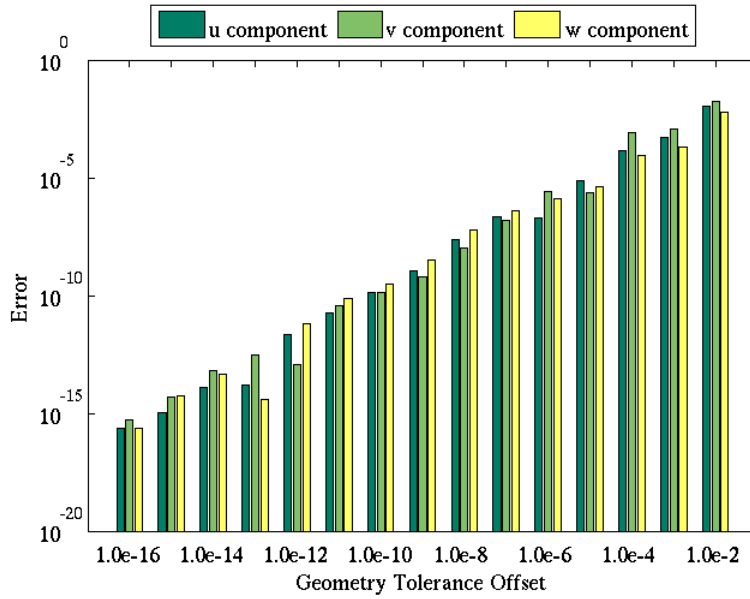


Figure 4-4: The value of  $\tilde{\nu}$ , or relative design velocity on the curve, as determined by methods 3, 6 and 7.

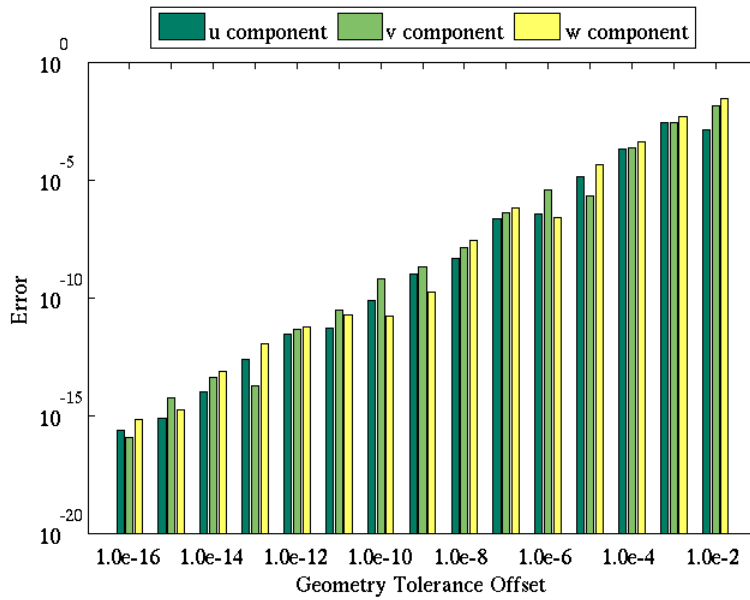
the location  $t = 0.5$  exhibits design motion normal to the curve tangent, thus each method collapses to the same design velocity and design motion. This result is encouraging since each method preserves the symmetry of the problem. Beyond  $t = 0.5$ , though, the inherent relative design velocity in each method indicates that a “perturbation direction” is preferred in each case, similar to having the constrained Method #2 prefer a perturbation that is normal to the curve tangent.

Methods #5 and #7 also benefit by computing the design velocity from the perspective of each surface and trim curve (i.e.,  $\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3$ ) at an intersection. As a check these should be equivalent to within machine-precision, yet it is possible this will not occur if each BRep entity has its own associated geometry tolerance (a form of “noise” that contributes to the right-hand side of each Method). Since it is unclear which design velocity result is “more correct” in this scenario, an average design velocity will simplify implementation. In the cone-plane example, Figure 4-5 demonstrates that this implementation gives design velocity results that deviate from a “noise-less” scenario by approximately the order-of-magnitude of the geometry tolerance offset itself. Since such offsets are expected to be small in geometry kernels, if they exist, the design velocity deviation is expected to be small compared to the “noise-less” scenario.





(a) Method 7



(b) Method 5

Figure 4-5: The error between design velocity results when geometry tolerances do not match (geometry “noise”) are shown for both Method 7 and Method 5.

## 4.4 Geometry Gradients at BRep Nodes

The sensitivity formulation on trim curves is a special case of the more-general derivation shown here for a BRep Node (more specifically, trim nodes). Compared to the closed-form problem for three surfaces, the general node intersection problem has more geometry information than is needed to determine a design velocity. Therefore, using the same tools as in Section 4.3, a derivation of design velocity at BRep nodes is presented using the Minimum Velocity Method as well. Validation examples are shown for various topology scenarios.

### 4.4.1 Extending the Trim Curve Sensitivity Derivation

A node may be considered as a point on a trim curve, however more than two intersecting trim curves (and hence faces) contribute to its position<sup>3</sup>. To find the geometry gradient at a node  $\mathcal{N}$  using the Minimal Velocity Method, we begin by extending (4.25) for all combinations of intersecting faces  $\{\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_n\}$  with  $n > 2$ . Coordinates for the node are written as

$$\boldsymbol{\eta} = \boldsymbol{\eta}(\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_n; \mathcal{P}) \in E,$$

where the parameter  $\mathcal{P}$  drives some or all of the faces intersecting at  $\mathcal{N}$ . We also consider that the geometry kernel intersects trim curves within a tolerance ball of radius  $\epsilon_{tol}$  at the node. The intersection of faces at the node also occurs with a tolerance that may or may not be  $\epsilon_{tol}$  (the geometry kernel source is required to verify this). For this derivation we assume the same tolerance  $\epsilon_{tol}$  is used for each trim curve and face intersecting at a node. In addition, the point on face  $\mathcal{F}_j$  that minimizes  $|\mathbf{r}_j((u, v)_j; \mathcal{P}) - \boldsymbol{\eta}|$  is denoted  $\mathbf{r}_j^*((u, v)_j^*; \mathcal{P})$ , thus the face-face constraint equation in (4.25) becomes

---

<sup>3</sup>We emphasize that nodes resulting from the trimming of trim curves may be considered a type of “trim nodes.” These differ from nodes that correspond to end-points of sketch *primitives*, which simply define the extent of *primitives* without involvement of trimming algorithms.

$$\left\{ \begin{array}{l} \mathbf{r}_1^* - \mathbf{r}_2^* = \boldsymbol{\epsilon}_{1,2} \\ \vdots \\ \mathbf{r}_1^* - \mathbf{r}_n^* = \boldsymbol{\epsilon}_{1,n} \end{array} \right\}, \quad \left\{ \begin{array}{l} \mathbf{r}_2^* - \mathbf{r}_3^* = \boldsymbol{\epsilon}_{2,3} \\ \vdots \\ \mathbf{r}_2^* - \mathbf{r}_n^* = \boldsymbol{\epsilon}_{2,n} \end{array} \right\}, \quad \dots, \quad (4.48)$$

$$\left\{ \begin{array}{l} \mathbf{r}_{n-2}^* - \mathbf{r}_{n-1}^* = \boldsymbol{\epsilon}_{n-2,n-1} \\ \mathbf{r}_{n-2}^* - \mathbf{r}_n^* = \boldsymbol{\epsilon}_{n-2,n} \end{array} \right\}, \quad \left\{ \mathbf{r}_{n-1}^* - \mathbf{r}_n^* = \boldsymbol{\epsilon}_{n-1,n} \right\},$$

where each equation is from the perspective of a face relative to all other intersecting faces. We also note the possibility for an offset between the intersecting faces and the node itself, where

$$\begin{aligned} |\boldsymbol{\epsilon}_{1,2}| \leq \epsilon_{tol} \quad \dots \quad |\boldsymbol{\epsilon}_{1,n}| \leq \epsilon_{tol} \\ |\boldsymbol{\epsilon}_{2,3}| \leq \epsilon_{tol} \quad \dots \quad |\boldsymbol{\epsilon}_{2,n}| \leq \epsilon_{tol} \\ |\boldsymbol{\epsilon}_{3,4}| \leq \epsilon_{tol} \quad \dots \quad |\boldsymbol{\epsilon}_{3,n}| \leq \epsilon_{tol} \\ \vdots \\ |\boldsymbol{\epsilon}_{n-2,n-1}| \leq \epsilon_{tol} \quad \dots \quad |\boldsymbol{\epsilon}_{n-2,n}| \leq \epsilon_{tol} \\ |\boldsymbol{\epsilon}_{n-1,n}| \leq \epsilon_{tol}. \end{aligned}$$

We can also add the perspective of each face to the node as

$$\left\{ \begin{array}{l} \mathbf{r}_1^* - \boldsymbol{\eta} = \boldsymbol{\epsilon}_{\mathcal{N},1} \\ \vdots \\ \mathbf{r}_n^* - \boldsymbol{\eta} = \boldsymbol{\epsilon}_{\mathcal{N},n} \end{array} \right\}, \quad (4.49)$$

with similar offsets written as

$$|\boldsymbol{\epsilon}_{\mathcal{N},1}| \leq \epsilon_{tol}, \quad |\boldsymbol{\epsilon}_{\mathcal{N},2}| \leq \epsilon_{tol}, \quad \dots, \quad |\boldsymbol{\epsilon}_{\mathcal{N},n}| \leq \epsilon_{tol}.$$

At this point we recognize that the geometry kernel provides no parameterization for  $\boldsymbol{\eta}$  and continue with a variational analysis. We first write a Taylor expansion around  $\boldsymbol{\eta}$  and each

$\mathbf{r}_j^*$  for  $j = 1, \dots, n$  as

$$\boldsymbol{\eta}' = \boldsymbol{\eta} + \frac{d\boldsymbol{\eta}}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2\boldsymbol{\eta}}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \dots \quad (4.50)$$

$$\mathbf{r}_j^*((u, v)_j^*; \mathcal{P} + \delta\mathcal{P})' = \mathbf{r}_j^*((u, v)_j^*; \mathcal{P}) + \frac{d\mathbf{r}_j^*}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2\mathbf{r}_j^*}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \dots \quad (4.51)$$

We assume that the parameter perturbation,  $\delta\mathcal{P}$ , is sufficiently small to preserve topology after regenerating with the new parameter  $\mathcal{P} + \delta\mathcal{P}$ . The new model instance also yields the offset vectors  $\boldsymbol{\epsilon}'_{1,2}$ ,  $\boldsymbol{\epsilon}'_{\mathcal{N},1}$ , etc., such that  $|\boldsymbol{\epsilon}'_{1,2} - \boldsymbol{\epsilon}_{1,2}| \leq \epsilon_{tol}$ ,  $|\boldsymbol{\epsilon}'_{\mathcal{N},1} - \boldsymbol{\epsilon}_{\mathcal{N},1}| \leq \epsilon_{tol}$ , etc., as well. From an implementation perspective, we simplify by setting  $\boldsymbol{\epsilon}'_{1,2} = \boldsymbol{\epsilon}_{1,2}$ ,  $\boldsymbol{\epsilon}'_{\mathcal{N},1} = \boldsymbol{\epsilon}_{\mathcal{N},1}$ , etc.

The first equation in (4.48) is used here to represent how all face-to-face node constraint equations are written:

$$\begin{aligned} \boldsymbol{\epsilon}'_{1,2} &= \mathbf{r}_1^*((u, v)_1^*; \mathcal{P} + \delta\mathcal{P})' - \mathbf{r}_2^*((u, v)_2^*; \mathcal{P} + \delta\mathcal{P})'_2 \\ &= \underbrace{\mathbf{r}_1^*((u, v)_1^*; \mathcal{P}) - \mathbf{r}_2^*((u, v)_2^*; \mathcal{P})}_{\boldsymbol{\epsilon}_{1,2}} + \left( \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}} \right) \delta\mathcal{P} + \frac{1}{2} \left( \frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2} \right) \delta\mathcal{P}^2 + \dots \\ \mathbf{0} &= \left( \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}} \right) \delta\mathcal{P} + \frac{1}{2} \left( \frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\mathbf{r}_2^*}{d\mathcal{P}^2} \right) \delta\mathcal{P}^2 + \dots \end{aligned} \quad (4.52)$$

The first equation in (4.49) is also taken as representative of how to rewrite the face-to-node constraint equations:

$$\begin{aligned} \boldsymbol{\epsilon}'_{\mathcal{N},1} &= \mathbf{r}_1^*((u, v)_1^*; \mathcal{P} + \delta\mathcal{P})' - \boldsymbol{\eta}' \\ &= \underbrace{\mathbf{r}_1^*((u, v)_1^*; \mathcal{P}) - \boldsymbol{\eta}}_{\boldsymbol{\epsilon}_{\mathcal{N},1}} + \left( \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\boldsymbol{\eta}}{d\mathcal{P}} \right) \delta\mathcal{P} + \frac{1}{2} \left( \frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\boldsymbol{\eta}}{d\mathcal{P}^2} \right) \delta\mathcal{P}^2 + \dots \\ \mathbf{0} &= \left( \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\boldsymbol{\eta}}{d\mathcal{P}} \right) \delta\mathcal{P} + \frac{1}{2} \left( \frac{d^2\mathbf{r}_1^*}{d\mathcal{P}^2} - \frac{d^2\boldsymbol{\eta}}{d\mathcal{P}^2} \right) \delta\mathcal{P}^2 + \dots \end{aligned} \quad (4.53)$$

In order for (4.52) and (4.53) to hold to first-order in  $\delta\mathcal{P}$ , the first term in parenthesis must

equal  $\mathbf{0}$ , which gives the following for each node constraint equation:

$$\left\{ \begin{array}{l} \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_2^*}{d\mathcal{P}} = \mathbf{0} \\ \vdots \\ \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\mathbf{r}_n^*}{d\mathcal{P}} = \mathbf{0} \end{array} \right\}, \quad \left\{ \begin{array}{l} \frac{d\mathbf{r}_2^*}{d\mathcal{P}} - \frac{d\mathbf{r}_3^*}{d\mathcal{P}} = \mathbf{0} \\ \vdots \\ \frac{d\mathbf{r}_2^*}{d\mathcal{P}} - \frac{d\mathbf{r}_n^*}{d\mathcal{P}} = \mathbf{0} \end{array} \right\}, \quad \dots \quad (4.54)$$

$$\left\{ \begin{array}{l} \frac{d\mathbf{r}_{n-2}^*}{d\mathcal{P}} - \frac{d\mathbf{r}_{n-1}^*}{d\mathcal{P}} = \mathbf{0} \\ \frac{d\mathbf{r}_{n-2}^*}{d\mathcal{P}} - \frac{d\mathbf{r}_n^*}{d\mathcal{P}} = \mathbf{0} \end{array} \right\}, \quad \left\{ \frac{d\mathbf{r}_{n-1}^*}{d\mathcal{P}} - \frac{d\mathbf{r}_n^*}{d\mathcal{P}} = \mathbf{0} \right\}$$

and

$$\left\{ \begin{array}{l} \frac{d\mathbf{r}_1^*}{d\mathcal{P}} - \frac{d\boldsymbol{\eta}}{d\mathcal{P}} = \mathbf{0} \\ \vdots \\ \frac{d\mathbf{r}_n^*}{d\mathcal{P}} - \frac{d\boldsymbol{\eta}}{d\mathcal{P}} = \mathbf{0} \end{array} \right\}. \quad (4.55)$$

Each  $d\mathbf{r}_j^*/d\mathcal{P}$  (where  $j = 1, \dots, n$ ) is also expanded to

$$\frac{d\mathbf{r}_j^*}{d\mathcal{P}} = \frac{\partial \mathbf{r}_j^*}{\partial u} \frac{\partial u}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}_j^*}{\partial v} \frac{\partial v}{\partial \mathcal{P}} + \frac{\partial \mathbf{r}_j^*}{\partial \mathcal{P}} = \dot{\mathbf{r}}_j^* + \nabla_{\mathbf{r}_j^*} \cdot \boldsymbol{\nu}_j.$$

Without needing to augment the equation set further (as in the trim curve case) to determine  $d\mathbf{r}_j^*/d\mathcal{P}$ , a linear system is written as

$$\begin{array}{c}
\left[ \begin{array}{cccccc}
\nabla \mathbf{r}_1^* & -\nabla \mathbf{r}_2^* & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\nabla \mathbf{r}_1^* & \mathbf{0} & -\nabla \mathbf{r}_3^* & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\nabla \mathbf{r}_1^* & \mathbf{0} & \cdots & \cdots & \nabla \mathbf{r}_n^* & \mathbf{0}
\end{array} \right] \\
\hline
\left[ \begin{array}{cccccc}
\mathbf{0} & \nabla \mathbf{r}_2^* & -\nabla \mathbf{r}_3^* & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \nabla \mathbf{r}_2^* & \mathbf{0} & -\nabla \mathbf{r}_4^* & \cdots & \mathbf{0} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\mathbf{0} & \nabla \mathbf{r}_2^* & \mathbf{0} & \cdots & \nabla \mathbf{r}_n^* & \mathbf{0}
\end{array} \right] \\
\hline
\left[ \begin{array}{cccccc}
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots
\end{array} \right] \\
\hline
\left[ \begin{array}{cccccc}
\mathbf{0} & \cdots & \mathbf{0} & \nabla \mathbf{r}_{n-1}^* & -\nabla \mathbf{r}_n^* & \mathbf{0}
\end{array} \right] \\
\hline
\left[ \begin{array}{cccccc}
-\nabla \mathbf{r}_1^* & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{I} \\
\mathbf{0} & -\nabla \mathbf{r}_2^* & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\mathbf{0} & \cdots & \cdots & \mathbf{0} & -\nabla \mathbf{r}_n^* & \mathbf{I}
\end{array} \right] \\
\hline
\mathbb{A}_{\mathcal{N}}
\end{array}
\begin{array}{c}
\left[ \begin{array}{c}
\nu_1 \\
\nu_2 \\
\nu_3 \\
\vdots \\
\nu_n
\end{array} \right] \\
\underbrace{\hspace{1.5cm}}_{\mathbb{X}_{\mathcal{N}}} \\
= \\
\left[ \begin{array}{c}
\mathbf{r}_2^* - \mathbf{r}_1^* \\
\mathbf{r}_3^* - \mathbf{r}_1^* \\
\vdots \\
\mathbf{r}_n^* - \mathbf{r}_1^* \\
\hline
\mathbf{r}_3^* - \mathbf{r}_2^* \\
\mathbf{r}_4^* - \mathbf{r}_2^* \\
\vdots \\
\mathbf{r}_n^* - \mathbf{r}_2^* \\
\hline
\vdots \\
\hline
\mathbf{r}_n^* - \mathbf{r}_{n-1}^* \\
\hline
\mathbf{r}_1^* \\
\mathbf{r}_2^* \\
\vdots \\
\mathbf{r}_n^*
\end{array} \right] \\
\mathbb{B}_{\mathcal{N}}
\end{array}
\tag{4.56}$$

The over-determined system  $\mathbb{A}_{\mathcal{N}}\mathbb{X}_{\mathcal{N}} = \mathbb{B}_{\mathcal{N}}$  is also solved in a least-squares sense using the truncated SVD approach. It is likely that  $\mathbb{A}_{\mathcal{N}}$  may have linearly dependent rows (as seen in some redundant geometry cases). In these situations the only unique solution we can use is  $\mathbb{X}_{\mathcal{N}}^*$ , which minimizes  $\|\mathbb{A}_{\mathcal{N}}\mathbb{X}_{\mathcal{N}}^* - \mathbb{B}_{\mathcal{N}}\|_2$  while having a minimum value for  $\|\mathbb{X}_{\mathcal{N}}^*\|_2$ . As in the trim curve case, the geometry perspective for this solution implies a minimum design velocity embodied in  $\nu_j$  for the design velocity of the node relative to the domain space of  $\mathcal{F}_j$ , whereas the design velocity for the node in Euclidean space is found with  $d\eta/d\mathcal{P}$ .

#### 4.4.2 Considerations for Redundant Geometry

There are various reasons why particular topology scenarios are redundant. For example, Figure 4-6(a) depicts a canopy-fuselage intersection modeled as a four-surface case with the

canopy distance,  $d$ , as a parameter of interest. The canopy consists of two ellipsoid swaths and the fuselage nose consists of two revolved non-linear functions. Although Figure 4-6(a) depicts seam  $\mathcal{E}_1$  separating the two-halves of the nose and  $\mathcal{E}_4$  separating the canopy, this situation has redundant geometry information compared to the intersection node seen in Section 4.2. This four-surface intersection node is actually a two-surface intersection node. Seams  $\mathcal{E}_1$  and  $\mathcal{E}_4$  are an *isoparameter line* on the fuselage and canopy geometry. Unlike the case of  $\mathcal{E}_2$  and  $\mathcal{E}_3$ , they do not represent a trim curve between two unique surfaces. The BRep connects this node to four faces with only two underlying surfaces at the intersection.

The example in Figure 4-6(b) has radius  $R$  as the parameter of interest with four faces and seams  $\mathcal{E}_1 \dots \mathcal{E}_4$  intersecting at node  $\mathcal{N}$ . All four faces share the same underlying cylinder surface. Seams  $\mathcal{E}_1 \dots \mathcal{E}_4$  are in the BRep to split the periodic surface, thus the coefficient matrix again contains linearly-dependent rows since all edges stem from the same surface. In the example of Figure 4-6(c) wing chord  $c$  is the parameter of interest and redundant geometry exists at node  $\mathcal{N}$  because a subset of surfaces have *aligned*  $\partial \mathbf{r} / \partial u$  or  $\partial \mathbf{r} / \partial v$  components. This can lead to linearly-dependent rows in the coefficient matrix as well.

Figure 4-7 shows other simple scenarios where redundant geometry may be found. In the case of 4-7(a) the parameter of interest is the radius  $R_1$  and the edge  $\mathcal{E}$  is an isoparameter line of both surfaces. The components of  $\partial \mathbf{r} / \partial u$  and  $\partial \mathbf{r} / \partial v$  are also aligned on the edge. Figure 4-7(b) shows a case where  $\partial \mathbf{r}_i / \partial \mathcal{P} = \mathbf{0}$  on each surface for the parameter  $\alpha$ . The curve  $\mathcal{E}$  also has its tangent aligned with  $\partial \mathbf{r}_i / \partial v$ . In these circumstances Method 2 results in a right-hand side that is zero and a singular coefficient matrix, yet the Minimum Velocity Method can provide a solution.

Situations of redundant geometry can be handled using the same edge or node algorithms presented here. No BRep processing is needed to uncover redundant geometry because the Minimum Velocity Method can provide a solution when the coefficient matrix is rank-deficient. In general, these solutions stem from the null space of the coefficient matrix even if redundant geometry information were removed, unless the three-surface intersection case is recovered by doing so. In the validation cases that follow the Minimum Velocity Method was used without the added complexity of identifying redundant geometry intersections.

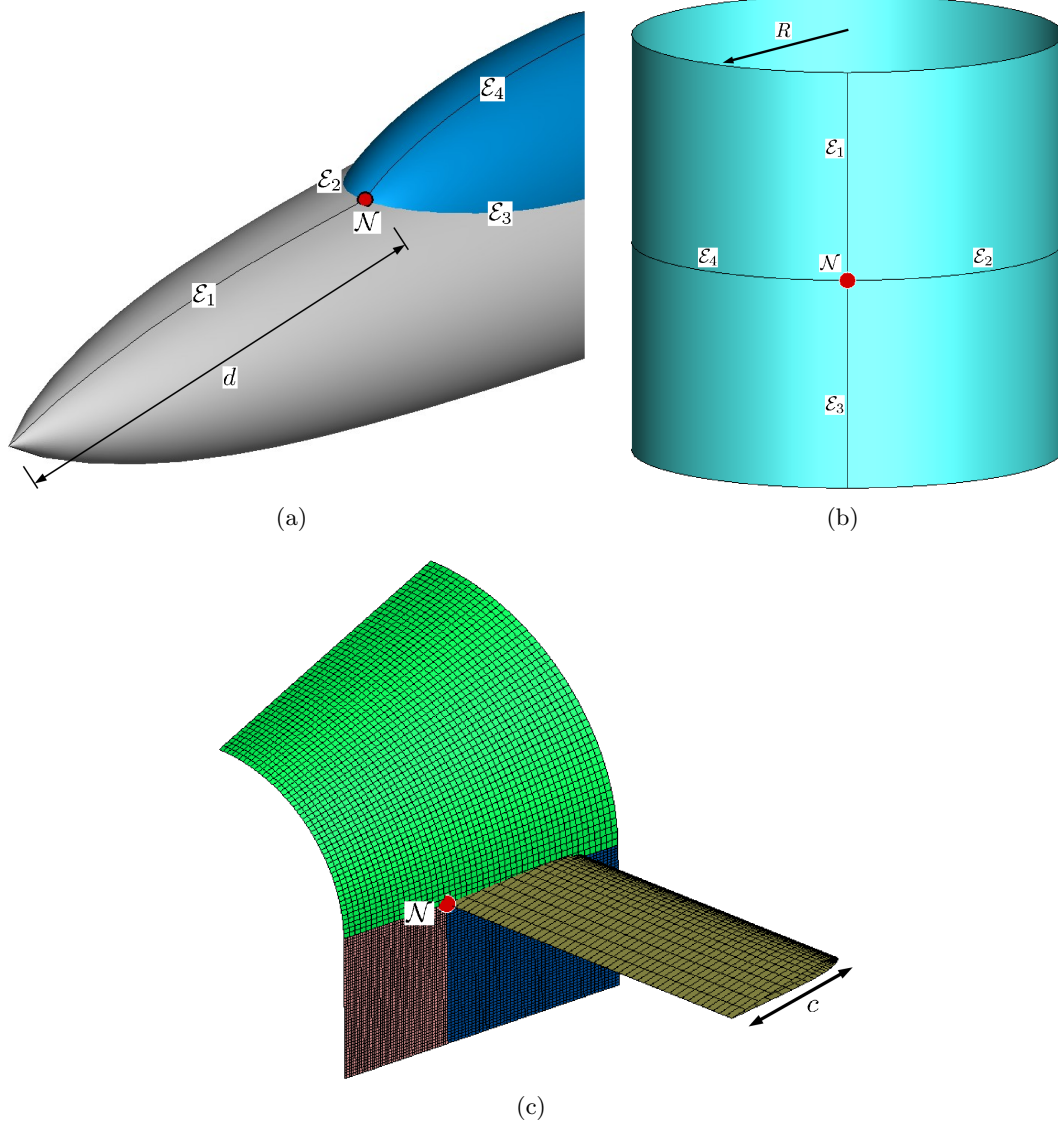


Figure 4-6: Node validation cases tested on (a) a two-surface, four-face node intersection, (b) a one-surface, four-face intersection, and (c) a five-surface intersection.



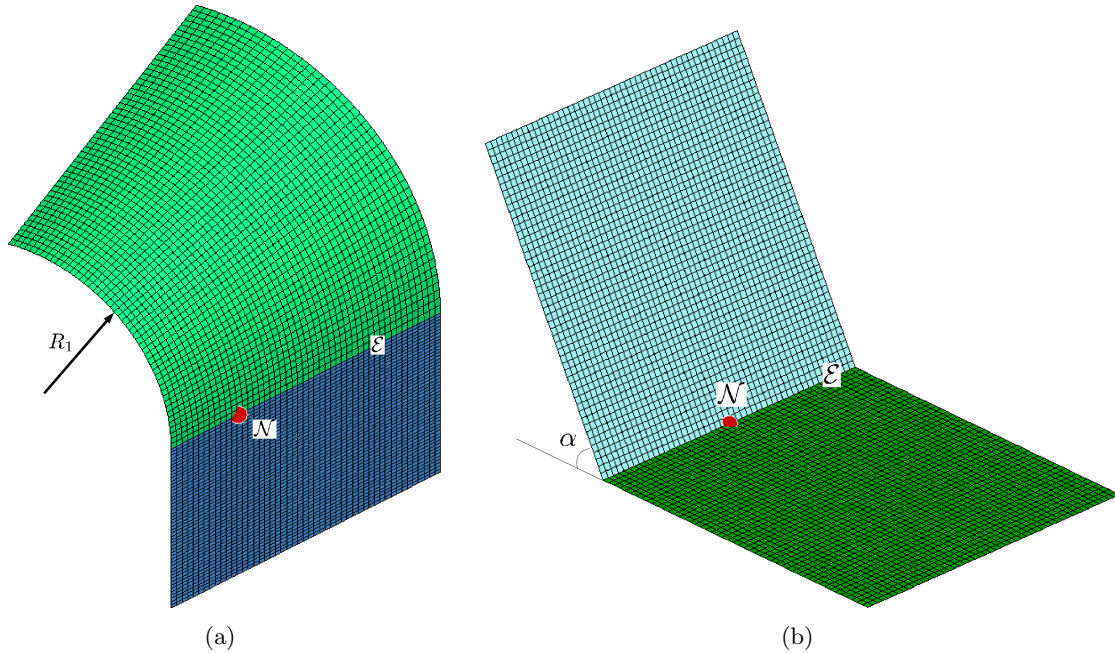


Figure 4-7: Application of the node algorithm (equivalent to Method #1 for edges) on (a) a trim curve where  $\partial\mathbf{r}/\partial u$  and/or  $\partial\mathbf{r}/\partial v$  are aligned for both surfaces, and (b) an edge where both surfaces have  $\partial\mathbf{r}/\partial\mathcal{P} = \mathbf{0}$ .

### 4.4.3 Validation for BRep Nodes

The sensitivity method for BRep nodes is validated using the analytic geometry cases seen in Figures 4-6 and 4-7. For the cases in Figure 4-7 the node algorithm is used (equivalent to using Method #1 for edges) to test the possibility of using a general algorithm for both edges and nodes. Results for each case are summarized in Table 4.6. Agreement between the analytic results and the new methods are excellent.

Numerical experiments also validate the sensitivity methodology when geometry “noise” exists for each redundant and non-redundant geometry case discussed thus far. Figures 4-8 through 4-13 indicate that in some cases the error between “noise-less” and “noisy” scenarios approximately follows the order-of-magnitude of the tolerance offset. Cases similar to Figures 4-9, 4-10 and 4-12 indicate that beyond an offset of  $10^{-9}$  the pseudoinverse tolerance is surpassed and more redundant geometry information is used to solve for the design velocity (making the problem more ill-posed). From a BRep topology perspective, this threshold also corresponds to the likelihood that the local node topology may change if tolerances are too coarse. This generates a large design velocity error, as expected, because the BRep may no longer be closed.

Case	Sensitivity		
	$x$	$y$	$z$
<b>Figure 4-6(a)</b>	$\partial \mathbf{r}_N / \partial d$		
<i>Analytic</i>	1.0000000000000000	0.0000000000000000	0.060412414523193
<i>Node Method</i>	1.0000000000000000	-0.0000000000000000	0.060412414523193
<b>Figure 4-6(b)</b>	$\partial \mathbf{r}_N / \partial R$		
<i>Analytic</i>	0.433012701892219	-0.2500000000000000	0.0000000000000000
<i>Node Method</i>	0.433012701892219	-0.2500000000000000	0.0000000000000000
<b>Figure 4-6(c)</b>	$\partial \mathbf{r}_N / \partial c$		
<i>Analytic</i>	1.0000000000000000	-0.2500000000000000	0.0000000000000000
<i>Node Method</i>	1.0000000000000000	-0.2500000000000000	0.0000000000000000
<b>Figure 4-7(a)</b>	$\partial \mathbf{r}_N / \partial R_1$		
<i>Analytic</i>	0.0000000000000000	0.7500000000000000	0.0000000000000000
<i>Node Method</i>	0.0000000000000000	0.7500000000000000	0.0000000000000000
<b>Figure 4-7(b)</b>	$\partial \mathbf{r}_N / \partial \alpha$		
<i>Analytic</i>	0.0000000000000000	0.0000000000000000	0.0000000000000000
<i>Node Method</i>	0.0000000000000000	0.0000000000000000	0.0000000000000000

Table 4.6: Validation results for the node sensitivity algorithm on the geometry cases in Figures 4-6 and 4-7.

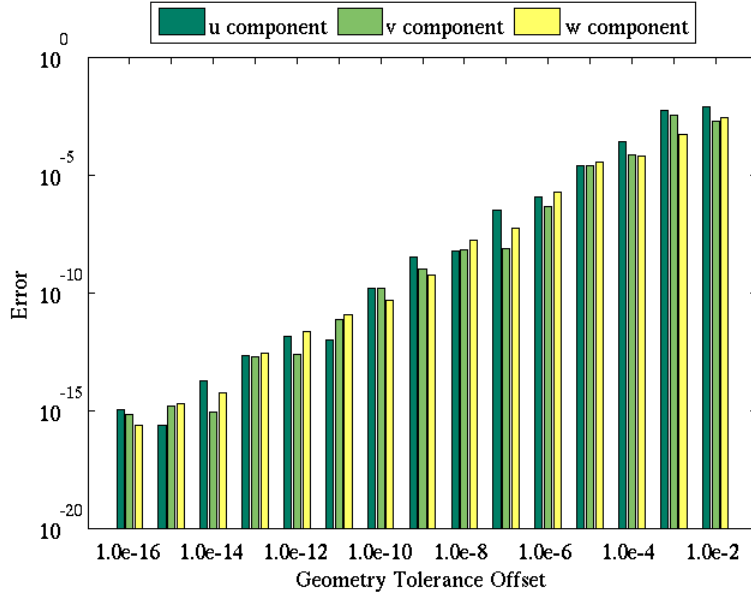


Figure 4-8: Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the three surface, closed-form intersection formulation in Figure 4-1.

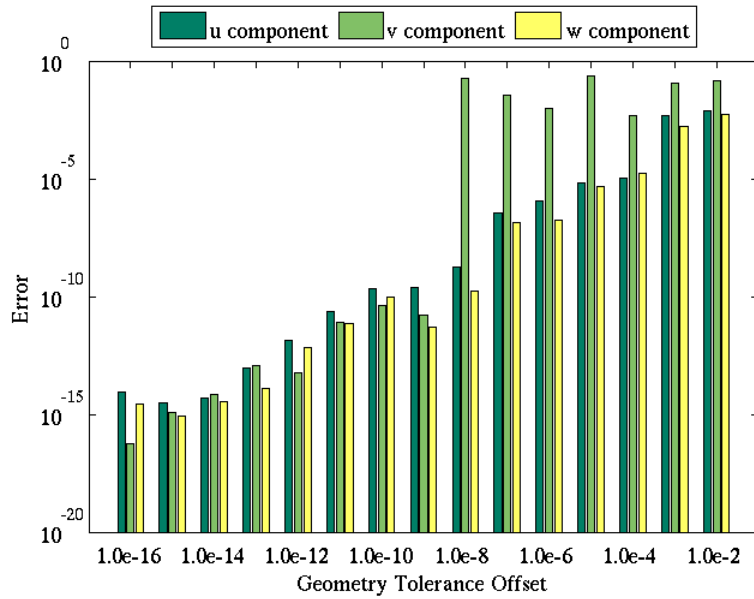


Figure 4-9: Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-6(a).

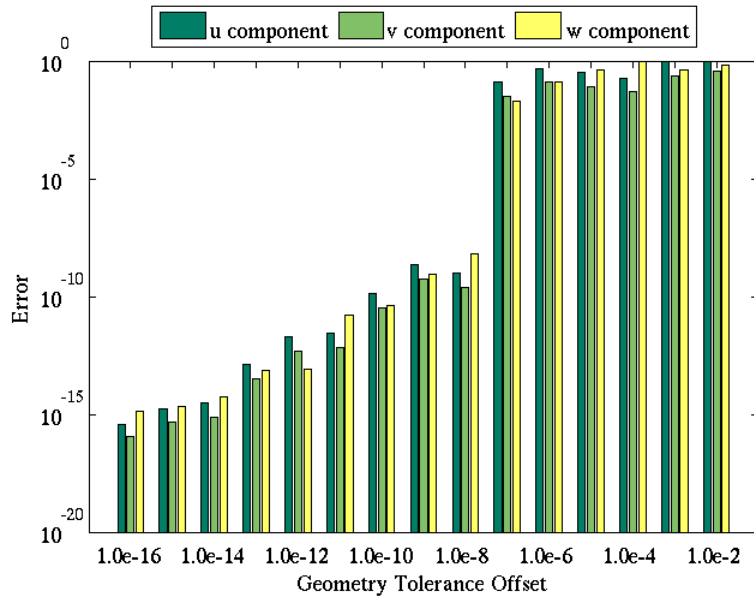


Figure 4-10: Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-7(a).

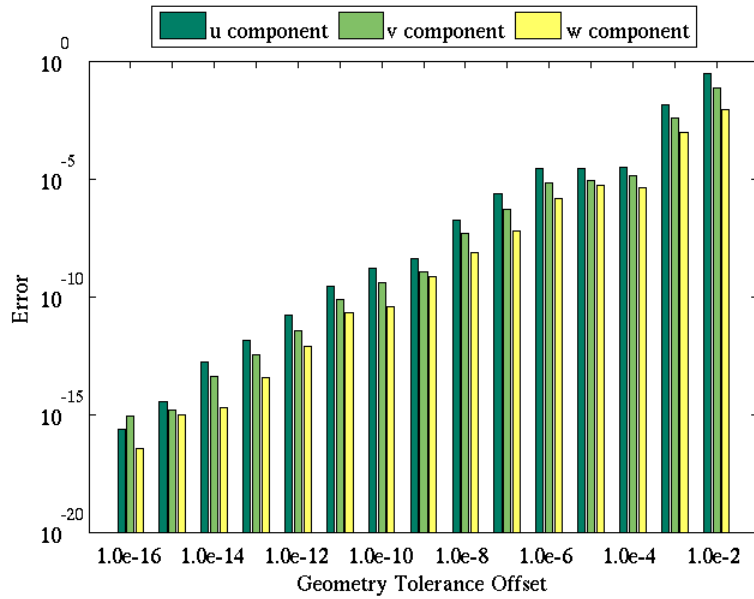


Figure 4-11: Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-6(c).

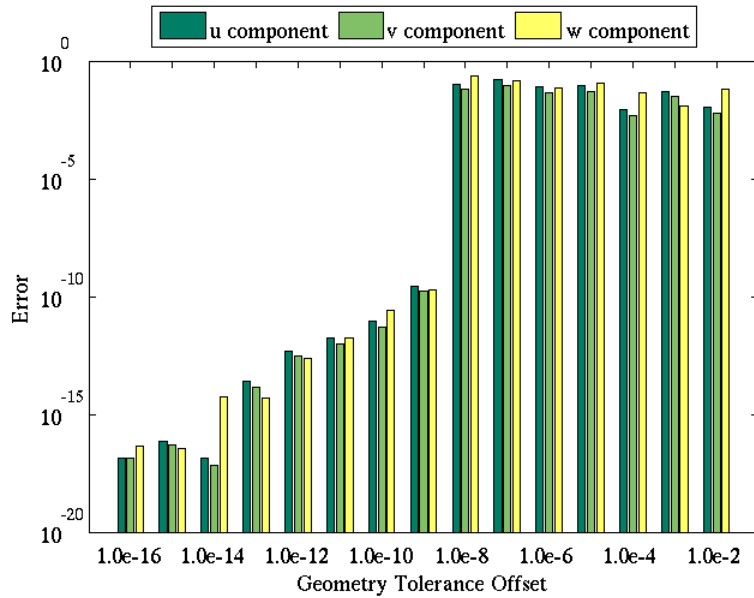


Figure 4-12: Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-6(b).

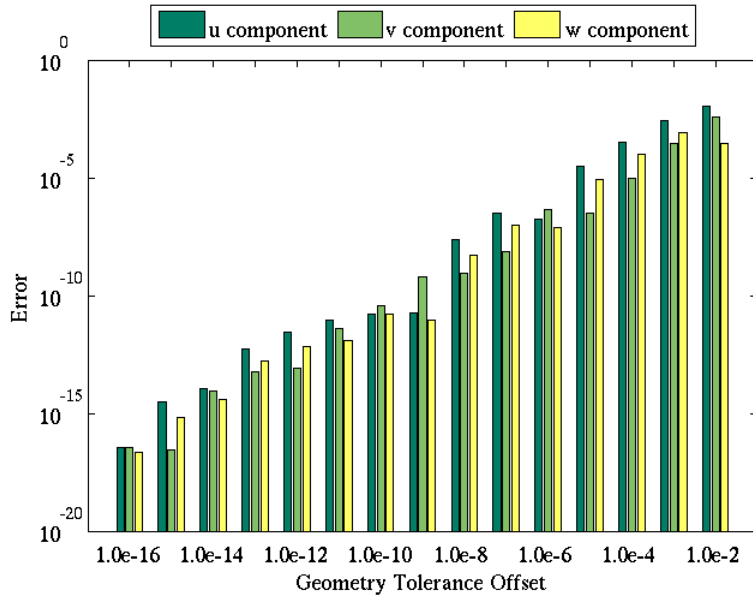


Figure 4-13: Error between “noise-less” and “noisy” design velocity results stemming from variations in geometry tolerance among BRep entities. This case corresponds to the scenario in Figure 4-7(b).

## 4.5 BRep Intersection Sensitivity Summary

Of the various BRep edge sensitivity methods presented, Method #7 is recommended as the preferred choice in order to incorporate all available BRep information. From a design motion perspective, the trim curve information is an *approximation* of the true intersection space-curve and thereby contributes to design velocity along itself via its own geometry characteristics. Since the tolerance used in generating an intersection trim curve is unknown, using all descriptors of the trim curve provides the most likely scenario in representing its design velocity accurately. If the trim curve represented the *exact* intersection trim curve, then its inclusion is redundant to the surface information already used because the trim curve is directly computed from the surfaces. In such a case Method #5 is recommended. Results for the cone-plane example in Figures 4-2 and 4-3 indicate Method #7 has the smallest offset from analytic results with Method #5 following, hence these recommendations are consistent with observed numerical experiment. If the added implementation effort is undesired, application of Method #5 may suffice with only a minor penalty in accuracy.

The geometry constraint system for nodes in (4.56) is similar to an expanded version of Method #1. If trim curve geometry is neglected, then it becomes possible to use the

same algorithm for *both* BRep edge and node sensitivities. This combined algorithm would essentially create  $\mathbb{A}_{\mathcal{N}}$ ,  $\mathbb{X}_{\mathcal{N}}$  and  $\mathbb{B}_{\mathcal{N}}$  based on the number of intersecting surfaces at a point (either on a trim curve or at a node). Although method #1 did not compare as well as the test results for Methods #5 or #7, this approach simplifies implementation to a single sensitivity code.

## Chapter 5

# Geometry Sensitivities for B-Spline Curves & Surfaces

Special consideration of B-spline curves and surfaces is given here for geometry management of CAD models in aerospace design applications. The development of analytic sensitivities for B-spline curves and surfaces is presented. First, the derived construction of B-spline curves and surfaces is discussed from an analytic perspective. This is followed by their geometry gradient formulations and a discussion of issues stemming from finite-differencing such geometry with a CAD system. Much of the notation in describing the analytic B-spline curve and surface construction follows the convention found in the Piegl and Tillers text The NURBS Book [64].

### 5.1 B-Spline Curve Construction

A multiple step procedure is usually followed when interpolating a B-spline through support points. By first understanding the construction of interpolating B-spline curves, the B-spline design velocity field is subsequently understood. The following example is only one approach to defining a B-spline curve (i.e., interpolation rather than “fitting”), thus the geometry gradient method will need to be adapted to other B-spline curve definitions accordingly. In general, though, an interpolating B-spline curve consists of piecewise polynomial segments joined at “knots” with continuity  $\mathcal{C}^{p-1}$ , where  $p$  is the polynomial degree. These serve as the parameter domain bounds for each piecewise polynomial that make up the final B-spline curve. Each polynomial segment is generated by a weighted summation of B-spline basis

functions, where the weights are called “control points.”

For a given set of support points  $\mathbf{B}_Q(\mathcal{P}) = \{\mathbf{Q}_k(\mathcal{P})\}$  (possibly driven by some parameter  $\mathcal{P}$ ), where  $\mathbf{Q}_k = (Q_{k,x}, Q_{k,y}, Q_{k,z}) \in \mathbb{R}^3$  for  $k = 0, \dots, n$ , an interpolating B-spline curve through the  $\mathbf{B}_Q$  is parameterized by a running parameter  $t \in [0, 1]$ . Within the domain of  $t$  the support points  $\mathbf{Q}_k$  are mapped to the parameters  $\bar{t}_k$ ,  $\mathbf{Q}_k \rightarrow \bar{t}_k$ . Various approaches are possible to determine the  $\bar{t}_k$ , yet one of the most common is called a “chord method,” where the normalized Euclidean distance between support points (i.e., the chord) is used as a coarse approximation of the B-spline curve arc-length. The chord method provides

$$\bar{t}_k = \bar{t}_{k-1} + \frac{|\mathbf{Q}_k - \mathbf{Q}_{k-1}|}{\mathcal{D}} \quad (5.1)$$

$$\mathcal{D} = \sum_{s=1}^n |\mathbf{Q}_s - \mathbf{Q}_{s-1}|. \quad (5.2)$$

With the  $\bar{t}_k$  available a knot vector  $U = \{u_i\}$ , for  $i = 0, \dots, m$ , can be established. This consists of knots  $u_i$  that mark the extent of the  $t$  domain where each of  $n + 1$  B-spline basis functions are non-zero. Various approaches also exist for defining the knots  $u_i$ , including setting them to the  $\bar{t}_k$  values. A common method is used by averaging the  $\bar{t}_k$  to obtain a distribution of  $u_i$ :

$$u_i = \begin{cases} 0, & 0 \leq i \leq p \\ \frac{1}{p} \sum_{j=i-p}^{i-1} \bar{t}_j, & 1 + p \leq i \leq m - p - 1 \\ 1, & m - p \leq i \leq m \end{cases} \quad (5.3)$$

The number of knots in  $U$  is  $m = n + p + 1$  for a fundamental<sup>1</sup> interpolating B-spline curve. Once a knot vector is known (or chosen, rather) the B-spline basis functions are calculated with

$$N_{i,p}(t) = \begin{cases} 0, & p = 0 \text{ and } t \notin [u_i, u_{i+1}) \\ 1, & p = 0 \text{ and } t \in [u_i, u_{i+1}) \\ \frac{t - u_i}{u_{i+p} - u_i} N_{i,p-1}(t) + \frac{u_{i+p+1} - t}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(t), & \text{otherwise} \end{cases} \quad (5.4)$$

The Euclidean coordinates of a support point are then expressed as the sum of weighted

---

<sup>1</sup>Here “fundamental” indicates that only interpolating support points are used, compared to more complex situations where tangent and curvature also parameterize the curve.



B-spline basis functions:

$$\mathbf{Q}_k(\mathcal{P}) = \sum_{i=0}^n N_{i,p}(\bar{t}_k) \mathbf{P}_i, \quad \text{for } k = 0, \dots, n, \quad (5.5)$$

where the set  $\mathbf{X}_P = \{\mathbf{P}_i\}$  contains  $\mathbf{P}_i = (P_{i,x}, P_{i,y}, P_{i,z}) \in \mathbb{R}^3$  control points (for  $i = 0, \dots, n$ ). Having  $n + 1$  support points requires  $n + 1$  basis functions and control points. The recursive basis functions are evaluated at  $\bar{t}_k$  using (5.4). However, all  $n + 1$  basis functions need not be evaluated for each  $\bar{t}_k$  because only a subset of the basis functions are non-zero in the knot span  $\bar{t}_k \in [u_i, u_{i+p+1})$ , namely  $N_{i-p,p}(\bar{t}_k), \dots, N_{i,p}(\bar{t}_k)$  (this is a property of B-spline curves). By utilizing this property, the control points are obtained using a closed, invertible linear system as follows:

$$\mathbf{A}_{\bar{N}} \mathbf{X}_P = \mathbf{B}_Q \quad (5.6)$$

where

$$\mathbf{X}_P = \begin{bmatrix} \vdots \\ P_{k,x} & P_{k,y} & P_{k,z} \\ \vdots \end{bmatrix}, \quad \mathbf{B}_Q = \begin{bmatrix} \vdots \\ Q_{k,x} & Q_{k,y} & Q_{k,z} \\ \vdots \end{bmatrix},$$

and<sup>2</sup>

$$\mathbf{A}_{\bar{N}} = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ N_{0,p}(\bar{t}_1) & \dots & N_{i,p}(\bar{t}_1) & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & N_{i-p,p}(\bar{t}_k) & \dots & N_{i,p}(\bar{t}_k) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & N_{n-p,p}(\bar{t}_{n-1}) & \dots & N_{n,p}(\bar{t}_{n-1}) \\ 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 & 1 \end{bmatrix}.$$

The unknown columns of  $\mathbf{X}_P$  are obtained using standard LU-decomposition and Gaussian Elimination. Since  $\mathbf{B}_Q = \mathbf{B}_Q(\mathcal{P})$ , then  $\mathbf{X}_P = \mathbf{X}_P(\mathcal{P})$  as well. These control points make up a *control polygon* for the B-spline curve which serves as the convex hull of the curve. It is evident that the B-spline end-points correspond to the control polygon end-points as

<sup>2</sup>Here we denote that  $\mathbf{A}_{\bar{N}} = \mathbf{A}_{\bar{N}}(\bar{t}, U)$ , yet for simplified notation the parameters  $\bar{t}$  and  $U$  are omitted since they are implied in context. The same applies later to  $\mathbf{A}_N = \mathbf{A}_N(t, U)$ .

well (i.e., this is a *clamped* B-spline curve), as denoted by the first and last rows of  $\mathbf{A}_{\bar{N}}$ . At this point there is sufficient information to construct the B-spline curve itself for all  $t \in [0, 1]$ . If  $t$  is discretized into the set  $\{t_a\} \in [0, 1]$  along the B-spline curve domain, then the discretized B-spline curve is written as

$$\mathbf{C}(t_a; \mathcal{P}) = \sum_{i=0}^n N_{i,p}(t_a) \mathbf{P}_i(\mathcal{P}) \quad \text{for } a = 0, \dots, n_C. \quad (5.7)$$

The right hand side of (5.7) is known and an analogous system to (5.6) can be written to calculate any point along the B-spline curve:

$$\mathbf{Y}(\mathcal{P}) = \mathbf{A}_N \mathbf{X}_P(\mathcal{P}), \quad (5.8)$$

where  $\mathbf{X}_P$  is as before and

$$\mathbf{Y} = \begin{bmatrix} \vdots \\ C_{a,x} & C_{a,y} & C_{a,z} \\ \vdots \end{bmatrix},$$

$$\mathbf{A}_N = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ N_{0,p}(t_1) & \dots & N_{i,p}(t_1) & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & N_{i-p,p}(t_a) & \dots & N_{i,p}(t_a) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & N_{n-p,p}(t_{n_C-1}) & \dots & N_{n,p}(t_{n_C-1}) \\ 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 & 1 \end{bmatrix}.$$

## 5.2 B-Spline Curve Geometry Gradient

With the B-spline curve construction method in place, a design velocity field across  $\mathbf{C}(t_a; \mathcal{P})$  can be derived with respect to a parameter  $\mathcal{P}$ . This derivation is succinctly described as the differentiation of the B-spline construction procedure because the knots, control points and basis functions are coupled through the support points. Although B-spline geometry gradients are known with respect to control points, differentiating with respect to support points seems to be overlooked by the research community. In many design instances a B-spline curve is used to interpolate airfoil coordinates driven by an external parameterization.

By differentiating the B-spline curve against its support points and using the chain-rule, the sensitivity of the B-spline to the external airfoil parameterization is then possible.

A B-spline curve from (5.8) has a design velocity with respect to a parameter  $\mathcal{P}$  as

$$\frac{d\mathbf{Y}}{d\mathcal{P}} = \frac{d}{d\mathcal{P}} (A_{\bar{N}} \mathbf{X}_{\mathcal{P}}). \quad (5.9)$$

It is clear that the gradient of *control points* to support points will be required. These are found by differentiating (5.5) as

$$\begin{aligned} \frac{d\mathbf{Q}_k}{d\mathcal{P}} &= \frac{d}{d\mathcal{P}} \left( \sum_{i=0}^n N_{i,p}(\bar{t}_k) \mathbf{P}_i \right) \\ &= \sum_{i=0}^n \left( \frac{dN_{i,p}(\bar{t}_k)}{d\mathcal{P}} \mathbf{P}_i + N_{i,p}(\bar{t}_k) \frac{d\mathbf{P}_i}{d\mathcal{P}} \right). \end{aligned} \quad (5.10)$$

The term  $\frac{d\mathbf{P}_i}{d\mathcal{P}}$  is unknown. The left-hand side will be a dense vector in general since  $\mathbf{Q}_k = \mathbf{Q}_k(\mathcal{P})$ . If  $\mathcal{P} = Q_{k,y}$ , for example, then  $\frac{d\mathbf{Q}_k}{d\mathcal{P}} = [0, 1, 0]^T$  and only a single entry is non-zero.

The term  $\frac{dN_{i,p}(\bar{t}_k)}{d\mathcal{P}}$  is expanded by first setting  $f_{i,p} = (\bar{t}_k - u_i)/(u_{i+p} - u_i)$  and  $g_{i,p} = (u_{i+p+1} - \bar{t}_k)/(u_{i+p+1} - u_{i+1})$  to get the recursive relation

$$\begin{aligned} \frac{dN_{i,p}(\bar{t}_k)}{d\mathcal{P}} &= \frac{d}{d\mathcal{P}} [fN_{i,p-1}(\bar{t}_k) + gN_{i+1,p-1}(\bar{t}_k)] \\ &= \frac{df_{i,p}}{d\mathcal{P}} N_{i,p-1}(\bar{t}_k) + f_{i,p} \frac{dN_{i,p-1}(\bar{t}_k)}{d\mathcal{P}} + 0 + 0, \end{aligned} \quad (5.11)$$

where the last two terms go to zero because the basis functions  $N_{i+1,p-1} = 0$  within the knot span of interest. Expanding  $\frac{df_{i,p}}{d\mathcal{P}}$  as

$$\frac{df_{i,p}}{d\mathcal{P}} = \frac{1}{(u_{i+p} - u_i)^2} \left[ (u_{i+p} - u_i) \frac{d\bar{t}_k}{d\mathcal{P}} - (\bar{t}_k - u_i) \frac{du_{i+p}}{d\mathcal{P}} + (\bar{t}_k - u_{i+p}) \frac{du_i}{d\mathcal{P}} \right] \quad (5.12)$$

requires the gradient of  $\bar{t}_k$ , the knots  $u_i$  and  $u_{i+p}$  with respect to  $\mathcal{P}$ . First, the term  $\frac{d\bar{t}_k}{d\mathcal{P}}$  is found by differentiating (5.1) after setting  $\mathbf{V}_k = \mathbf{Q}_k - \mathbf{Q}_{k-1}$  to obtain

$$\frac{d\bar{t}_k}{d\mathcal{P}} = \frac{\partial \bar{t}_{k-1}}{\partial \mathcal{P}} + \frac{1}{D} \left\{ \frac{\mathbf{V}_k}{|\mathbf{V}_k|} \cdot \left[ \frac{\partial Q_{k,x}}{\partial \mathcal{P}}, \frac{\partial Q_{k,y}}{\partial \mathcal{P}} \right] - \frac{\mathbf{V}_k}{|\mathbf{V}_k|} \cdot \left[ \frac{\partial Q_{k-1,x}}{\partial \mathcal{P}}, \frac{\partial Q_{k-1,y}}{\partial \mathcal{P}} \right] \right\} - \frac{|\mathbf{V}_k|}{D^2} \frac{\partial D}{\partial \mathcal{P}}, \quad (5.13)$$

where

$$\frac{\partial \mathcal{D}}{\partial \mathcal{P}} = \begin{cases} \frac{\mathbf{V}_k}{|\mathbf{V}_k|} \left[ \frac{\partial Q_{k,x}}{\partial \mathcal{P}}, \frac{\partial Q_{k,y}}{\partial \mathcal{P}}, \frac{\partial Q_{k,z}}{\partial \mathcal{P}} \right] - \frac{\mathbf{V}_{k+1}}{|\mathbf{V}_{k+1}|} \left[ \frac{\partial Q_{k-1,x}}{\partial \mathcal{P}}, \frac{\partial Q_{k-1,y}}{\partial \mathcal{P}}, \frac{\partial Q_{k-1,z}}{\partial \mathcal{P}} \right], & k \in (0, n) \\ -\frac{\mathbf{V}_1}{|\mathbf{V}_1|} \left[ \frac{\partial Q_{0,x}}{\partial \mathcal{P}}, \frac{\partial Q_{0,y}}{\partial \mathcal{P}}, \frac{\partial Q_{0,z}}{\partial \mathcal{P}} \right], & k = 0 \\ \frac{\mathbf{V}_n}{|\mathbf{V}_n|} \left[ \frac{\partial Q_{n,x}}{\partial \mathcal{P}}, \frac{\partial Q_{n,y}}{\partial \mathcal{P}}, \frac{\partial Q_{n,z}}{\partial \mathcal{P}} \right], & k = n. \end{cases} \quad (5.14)$$

Second, the term  $\frac{du_i}{d\mathcal{P}}$  is obtained by differentiating (5.3) to obtain

$$\frac{du_i}{d\mathcal{P}} = \begin{cases} 0, & 0 \leq i \leq p \\ \frac{1}{p} \sum_{j=i-p}^{i-1} \frac{d\bar{t}_j}{d\mathcal{P}}, & 1+p \leq i \leq m-p-1 \\ 0, & m-p \leq i \leq m \end{cases} . \quad (5.15)$$

Third, a very similar expression for  $\frac{du_{i+p}}{d\mathcal{P}}$  is written by changing the limits of summation

$$\frac{du_{i+p}}{d\mathcal{P}} = \begin{cases} 0, & 0 \leq i \leq p \\ \frac{1}{p} \sum_{j=i}^{i+p-1} \frac{d\bar{t}_j}{d\mathcal{P}}, & 1+p \leq i \leq m-p-1 \\ 0, & m-p \leq i \leq m \end{cases} . \quad (5.16)$$

Each term in (5.11) is now defined and a linear system can be written for the unknowns  $\frac{d\mathbf{P}_i}{d\mathcal{P}}$  by using (5.9). The following notation will be useful:

$$\partial_{\mathcal{P}} \mathbf{A}_{\bar{N}} = \frac{d}{d\mathcal{P}} (\mathbf{A}_{\bar{N}}), \quad \partial_{\mathcal{P}} \mathbf{B}_Q = \frac{d}{d\mathcal{P}} (\mathbf{B}_Q), \quad \partial_{\mathcal{P}} \mathbf{X} = \frac{d}{d\mathcal{P}} (\mathbf{X}_P).$$

Since the control points in  $\mathbf{A}_{\bar{N}} \mathbf{X}_P = \mathbf{B}_Q$  are found with  $\mathbf{X}_P = \mathbf{A}_{\bar{N}}^{-1} \mathbf{B}_Q$ , differentiating

against  $\mathcal{P}$  will yield:

$$\begin{aligned}
\frac{d}{d\mathcal{P}}(\mathbf{X}_P) &= \frac{d}{d\mathcal{P}}(\mathbf{A}_{\bar{N}}^{-1}\mathbf{B}_Q) \\
&= \frac{d}{d\mathcal{P}}(\mathbf{A}_{\bar{N}}^{-1})\mathbf{B}_Q + \mathbf{A}_{\bar{N}}^{-1}\partial_{\mathcal{P}}\mathbf{B}_Q \\
&= -\mathbf{A}_{\bar{N}}^{-1}\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}}\mathbf{A}_{\bar{N}}^{-1}\mathbf{B}_Q + \mathbf{A}_{\bar{N}}^{-1}\partial_{\mathcal{P}}\mathbf{B}_Q \\
&= \mathbf{A}_{\bar{N}}^{-1}(\partial_{\mathcal{P}}\mathbf{B}_Q - (\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})\mathbf{X}_P). \tag{5.17}
\end{aligned}$$

Solving (5.17) for  $\partial_{\mathcal{P}}\mathbf{X}_P$  returns the gradient of the control points  $\mathbf{P}_i$ . If  $\mathcal{P}$  is a component direction for some  $\mathbf{Q}_k$ , then  $\partial_{\mathcal{P}}\mathbf{X}_P$  is the control point gradient with respect to support point  $\mathbf{Q}_k$ .

The gradient of the B-spline curve  $\mathbf{C}(t)$  becomes possible now at any location  $t \in [0, 1]$  in its domain. This derivation holds for the continuous B-spline curve and a discretized B-spline curve. For example, if  $t$  is discretized into a set of distinct  $\{t_a\}$  (with  $a = 0, \dots, n_C$ ), the gradient derivation continues by differentiating (5.7) with respect to  $\mathcal{P}$  at each  $t_a$ :

$$\frac{d\mathbf{C}(t_a)}{d\mathcal{P}} = \sum_{i=0}^n \left[ \frac{dN_{i,p}(t_a)}{d\mathcal{P}}\mathbf{P}_i + N_{i,p}(t_a)\frac{d\mathbf{P}_i}{d\mathcal{P}} \right], \tag{5.18}$$

where  $\frac{dN_{i,p}(t_a)}{d\mathcal{P}}$  is expanded slightly differently than in (5.11) because there is no dependence of  $t_a$  on  $\mathcal{P}$  (by construction). Therefore

$$\begin{aligned}
\frac{dN_{i,p}(t_a)}{d\mathcal{P}} &= \frac{d}{d\mathcal{P}} [fN_{i,p-1}(t_a) + gN_{i+1,p-1}(t_a)] \\
&= \frac{df_{i,p}}{d\mathcal{P}}N_{i,p-1}(t_a) + f_{i,p}\frac{dN_{i,p-1}(t_a)}{d\mathcal{P}} + 0 + 0 \tag{5.19}
\end{aligned}$$

and

$$\frac{df_{i,p}}{d\mathcal{P}} = \frac{1}{(u_{i+p} - u_i)^2} \left[ (u_i - t_a)\frac{du_{i+p}}{d\mathcal{P}} + (t_a - u_{i+p})\frac{du_i}{d\mathcal{P}} \right]. \tag{5.20}$$

The knot sensitivities  $\frac{du_i}{d\mathcal{P}}$  and  $\frac{du_{i+p}}{d\mathcal{P}}$  are found in the same manner as (5.15) and (5.16), respectively. Since  $\frac{d\mathbf{P}_i}{d\mathcal{P}}$  is known from (5.17), the only unknowns in (5.18) are the  $\frac{d\mathbf{C}(t_a)}{d\mathcal{P}}$ . Thus the B-spline curve  $\mathbf{Y} = \mathbf{A}_N\mathbf{X}_P$  is differentiated as

$$\partial_{\mathcal{P}}\mathbf{Y} = (\partial_{\mathcal{P}}\mathbf{A}_N)\mathbf{X}_P + \mathbf{A}_N\partial_{\mathcal{P}}\mathbf{X}, \tag{5.21}$$

where  $\mathbf{A}_N$ ,  $\mathbf{X}_P$  and  $\partial_{\mathcal{P}}\mathbf{X}$  are as before, and

$$\partial_{\mathcal{P}}\mathbf{Y} = \frac{d}{d\mathcal{P}}(\mathbf{Y}), \quad \partial_{\mathcal{P}}\mathbf{A}_N = \frac{d}{d\mathcal{P}}(\mathbf{A}_N).$$

This is rewritten in a final form by substituting (5.17) and rearranging to obtain

$$\partial_{\mathcal{P}}\mathbf{Y} = \left( \partial_{\mathcal{P}}\mathbf{A}_N - \mathbf{A}_N\mathbf{A}_N^{-1}(\partial_{\mathcal{P}}\mathbf{A}_N) \right) \mathbf{X}_P + \mathbf{A}_N\mathbf{A}_N^{-1}\partial_{\mathcal{P}}\mathbf{B}_Q. \quad (5.22)$$

At this stage the gradient of a B-spline curve with respect to a parameter  $\mathcal{P}$  is defined. If  $\mathcal{P}$  is the component direction for some support point  $\mathbf{Q}_k$ , then the sensitivity of the B-spline curve to an interpolating support point is now known.

### 5.3 B-Spline Surface Construction

The analytic construction of B-spline surfaces is an extension of the formulation for B-spline curves (only non-rational B-spline surfaces are treated here). A B-spline surface,  $\mathbf{S}(u, v) \in \mathbb{R}^3$ , is parameterized by two coordinates  $(u, v) \in \mathbb{R}^2$  which typically range from  $(u, v) \in [0, 1] \times [0, 1]$ . These surfaces are often generated by interpolating<sup>3</sup> a field of support points  $\mathbf{Q}_{k,l}(\mathcal{P}) \in \mathbb{R}^3$  ( $k = 0, \dots, n$  and  $l = 0, \dots, m$ ) with weighted-sums of B-spline basis functions of degree  $p$  and  $q$  in the  $u$  and  $v$  directions, respectively. The B-spline surface shape is also driven by a selection of knot vectors  $U$  and  $V$  containing individual knots  $\bar{u}_k$  and  $\bar{v}_l$ . As in the case of B-spline curves, various approaches exist to characterize the knot vectors in some manner related to the  $\mathbf{Q}_{k,l}$  (e.g., the chord method utilized in constructing B-spline curves). A unique B-spline surface shape is obtained for each given knot vector  $U$  and  $V$ .

The B-spline surface interpolation is written as a tensor product surface:

$$\mathbf{Q}_{k,l}(\mathcal{P}) = \mathbf{S}(\bar{u}_k, \bar{v}_l; \mathcal{P}) = \sum_{i=0}^n \sum_{j=0}^m N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j}, \quad (5.23)$$

where  $N_{i,p}$  and  $N_{j,q}$  are B-spline basis functions of degree  $p$  and  $q$  in respective  $u$  and  $v$  directions (the general case of  $\mathbf{Q}_{k,l} = \mathbf{Q}_{k,l}(\mathcal{P})$  is still considered here). Also,  $\mathbf{P}_{i,j} \in \mathbb{R}^3$  represent basis function “weights” that are also called control points. Once a field of control

---

<sup>3</sup>A “fit” approximation to support points is also possible but not treated here.

points are calculated from the given  $\mathbf{Q}_{k,l}$  and knot vectors, any point on the B-spline surface within the domain of  $(u, v)$  is found with

$$\mathbf{S}(u, v; \mathcal{P}) = \sum_{i=0}^n \sum_{j=0}^m N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j}. \quad (5.24)$$

The control points in (5.23) can be found using a matrix formulation and splitting the problem into two parts. The given  $\mathbf{Q}_{k,l}$  represent coordinates along a cross-section of the B-spline surface at locations of constant- $v$ . Since (5.23) is a tensor product surface, it can be written as

$$\begin{aligned} \mathbf{Q}_{k,l}(\mathcal{P}) &= \sum_{i=0}^n \sum_{j=0}^m N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \\ &= \sum_{i=0}^n N_{i,p}(\bar{u}_k) \left( \sum_{j=0}^m N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \right) \\ &= \sum_{i=0}^n N_{i,p}(\bar{u}_k) \mathbf{R}_{i,l}, \end{aligned} \quad (5.25)$$

where

$$\mathbf{R}_{i,l} = \sum_{j=0}^m N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j}. \quad (5.26)$$

In this form the control points are calculated using sequential curve interpolations, first in  $u$  and afterwards in  $v$ . It is important to note that, unlike the B-spline curve interpolation addressed earlier, all of the curve interpolations are done using the *same*  $U$  and  $V$  vectors instead of cross-section-specific knot vectors. Writing these equations in matrix form yields

$$\begin{aligned} \mathbb{B}_Q(\mathcal{P}) &= \mathbf{A}_{\bar{N}}(\bar{u}) \mathcal{R} \\ \mathcal{R}^T &= \mathbf{A}_{\bar{N}}(\bar{v}) \mathbb{P}^T, \end{aligned}$$

where

$$\mathbb{B}_Q(\mathcal{P}) \equiv \begin{bmatrix} \mathbf{Q}_{0,0} & \mathbf{Q}_{0,1} & \cdots & \mathbf{Q}_{0,m} \\ \vdots & \vdots & \mathbf{Q}_{k,l} & \vdots \\ \mathbf{Q}_{n,0} & \mathbf{Q}_{n,1} & \cdots & \mathbf{Q}_{n,m} \end{bmatrix}, \quad \mathcal{R} \equiv \begin{bmatrix} \mathbf{R}_{0,0} & \mathbf{R}_{0,1} & \cdots & \mathbf{R}_{0,m} \\ \vdots & \vdots & \mathbf{R}_{k,l} & \vdots \\ \mathbf{R}_{n,0} & \mathbf{R}_{n,1} & \cdots & \mathbf{R}_{n,m} \end{bmatrix},$$

$$\mathbb{P} \equiv \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \cdots & \mathbf{P}_{0,m} \\ \vdots & \vdots & \mathbf{P}_{k,l} & \vdots \\ \mathbf{P}_{n,0} & \mathbf{P}_{n,1} & \cdots & \mathbf{P}_{n,m} \end{bmatrix},$$

and  $\mathbf{A}_{\bar{N}}$  is defined the same as for B-spline curves. The arguments  $\bar{u}$  or  $\bar{v}$  imply that the elements of  $\mathbf{A}_{\bar{N}}$  are either  $N_{i,p}(\bar{u}_k)$  or  $N_{j,q}(\bar{v}_l)$ , respectively, for  $i = 0, \dots, n$  and  $j = 0, \dots, m$ . The columns of  $\mathbb{B}_Q$  and  $\mathcal{R}$  represent the B-spline curve interpolation problem at constant- $v$  with inputs  $\mathbf{Q}_{k,l=\text{const}}$ . The rows of  $\mathcal{R}$  and  $\mathbb{P}$  represent B-spline curve interpolation at constant- $u$  with the *control points* of the  $u$ -direction B-splines as inputs. These two linear matrix problems are combined to form a single B-spline surface tensor-product:

$$\mathbb{B}_Q(\mathcal{P}) = \mathbf{A}_{\bar{N}}(\bar{u})\mathbb{P}\mathbf{A}_{\bar{N}}^T(\bar{v}). \quad (5.27)$$

Once the surface control points  $\mathbb{P}$  are obtained, it is clear that  $\mathbb{P} = \mathbb{P}(\mathcal{P})$ . The Euclidean coordinates on the B-spline surface are found for any set of discretized  $\{(u_a, v_a)\}$  ( $a = 0, \dots, n_C$ ) using

$$\mathbf{S}(u_a, v_a; \mathcal{P}) = \mathbf{A}_N(u_a)\mathbb{P}(\mathcal{P})\mathbf{A}_N^T(v_a). \quad (5.28)$$

## 5.4 B-Spline Surface Geometry Gradient

The geometry gradients of B-spline curves serve as the starting point to develop the B-spline surface sensitivity. It is clear that differentiating (5.28) will require the gradient of surface control points to support points:

$$\frac{d\mathbf{S}(u_a, v_a; \mathcal{P})}{d\mathcal{P}} = \frac{d}{d\mathcal{P}} [\mathbf{A}_N(u_a)\mathbb{P}\mathbf{A}_N^T(v_a)]. \quad (5.29)$$

---

<sup>4</sup>As in the B-spline curve case, we note that  $\mathbf{A}_{\bar{N}}(\bar{u}) = \mathbf{A}_{\bar{N}}(\bar{u}, U)$ ,  $\mathbf{A}_{\bar{N}}(\bar{v}) = \mathbf{A}_{\bar{N}}(\bar{v}, V)$ ,  $\mathbf{A}_{\bar{N}}(u) = \mathbf{A}_{\bar{N}}(u, U)$  and  $\mathbf{A}_{\bar{N}}(v) = \mathbf{A}_{\bar{N}}(v, V)$ . For simpler notation the knot vectors are omitted since they are implied in context.



First the calculation of

$$\frac{d\mathbb{B}_Q}{d\mathcal{P}} = \frac{d}{d\mathcal{P}} [\mathbf{A}_{\bar{N}}(\bar{u})\mathbb{P}\mathbf{A}_{\bar{N}}^T(\bar{v})] \quad (5.30)$$

is considered by splitting the tensor product into two parts. If the coupling between support points and knot vectors  $U$  and  $V$  is known, then the notation

$$\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}} = \frac{d}{d\mathcal{P}}(\mathbf{A}_{\bar{N}})$$

is helpful in writing

$$\frac{d\mathbb{B}_Q}{d\mathcal{P}} = \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathcal{R} + \mathbf{A}_{\bar{N}}(\bar{u})\frac{d\mathcal{R}}{d\mathcal{P}} \quad (5.31)$$

$$\frac{d\mathcal{R}^T}{d\mathcal{P}} = \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathbb{P}^T + \mathbf{A}_{\bar{N}}(\bar{v})\frac{d\mathbb{P}^T}{d\mathcal{P}}. \quad (5.32)$$

This expanded formulation can also be combined into a single statement,

$$\begin{aligned} \frac{d\mathbb{B}_Q}{d\mathcal{P}} &= \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathcal{R} + \mathbf{A}_{\bar{N}}(\bar{u}) \left[ \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathbb{P}^T + \mathbf{A}_{\bar{N}}(\bar{v})\frac{d\mathbb{P}^T}{d\mathcal{P}} \right]^T \\ &= \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathcal{R} + \mathbf{A}_{\bar{N}}(\bar{u})\mathbb{P}(\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})^T(\bar{v}) + \mathbf{A}_{\bar{N}}(\bar{u})\frac{d\mathbb{P}}{d\mathcal{P}}\mathbf{A}_{\bar{N}}^T(\bar{v}) \\ &= \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathbb{P}\mathbf{A}_{\bar{N}}^T(\bar{v}) + \mathbf{A}_{\bar{N}}(\bar{u})\mathbb{P}(\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})^T(\bar{v}) + \mathbf{A}_{\bar{N}}(\bar{u})\frac{d\mathbb{P}}{d\mathcal{P}}\mathbf{A}_{\bar{N}}^T(\bar{v}), \end{aligned}$$

and a solution for the unknown  $\frac{d\mathbb{P}}{d\mathcal{P}}$  is obtained via

$$\frac{d\mathbb{P}}{d\mathcal{P}} = \mathbf{A}_{\bar{N}}^{-1}(\bar{u}) \left[ \frac{d\mathbb{B}_Q}{d\mathcal{P}} - \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathbb{P}\mathbf{A}_{\bar{N}}^T(\bar{v}) - \mathbf{A}_{\bar{N}}(\bar{u})\mathbb{P}(\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})^T(\bar{v}) \right] \mathbf{A}_{\bar{N}}^{-T}(\bar{v}). \quad (5.33)$$

The geometry gradient anywhere on the B-spline surface then becomes

$$\begin{aligned} \frac{d\mathbf{S}(u_a, v_a; \mathcal{P})}{d\mathcal{P}} &= \frac{d}{d\mathcal{P}} [\mathbf{A}_N(u_a)\mathbb{P}\mathbf{A}_N^T(v_a)] \\ &= \partial_{\mathcal{P}}(\mathbf{A}_N(u_a))\mathbb{P}\mathbf{A}_N^T(v_a) + \mathbf{A}_N(u_a)\mathbb{P}\partial_{\mathcal{P}}(\mathbf{A}_N^T(v_a)) + \mathbf{A}_N(u_a)\frac{d\mathbb{P}}{d\mathcal{P}}\mathbf{A}_N^T(v_a). \end{aligned} \quad (5.34)$$

## 5.5 B-spline Curves & Surfaces in CAD Models

It is unclear how the knot vector  $U$  is generated for a general B-spline curve within a geometry kernel. This is made evident by considering how  $U$  varies when support points

are perturbed. Initially,  $U$  knots for a simple B-spline curve may be determined via the construction method presented in Section 5.1. However, it has been observed that small changes in the support points yield unintuitive changes in  $U$  as calculated by a geometry kernel. In some cases it is clear that  $U$  is obtained by knot insertion or removal to expand/contract a fundamental knot vector that is not given to the designer. The exact algorithm and logic used by the geometry kernel for this is unknown. It is also found that an end-point tangency is specified by the geometry kernel even if the user does not parameterize the B-spline curve with tangency information<sup>5</sup>. The given tangent direction may or may not suit the desired design intent for the curve and, as often declared in the CAD literature [24], may only provide an aesthetically “pleasant” curve shape with no inherent engineering design consideration. Similar situations arise when considering the  $U$  and  $V$  knot vectors for B-spline surfaces<sup>6</sup>. These cases may yield even more radical knot vectors despite interpolating across simple cross-section B-spline curves with few support points.

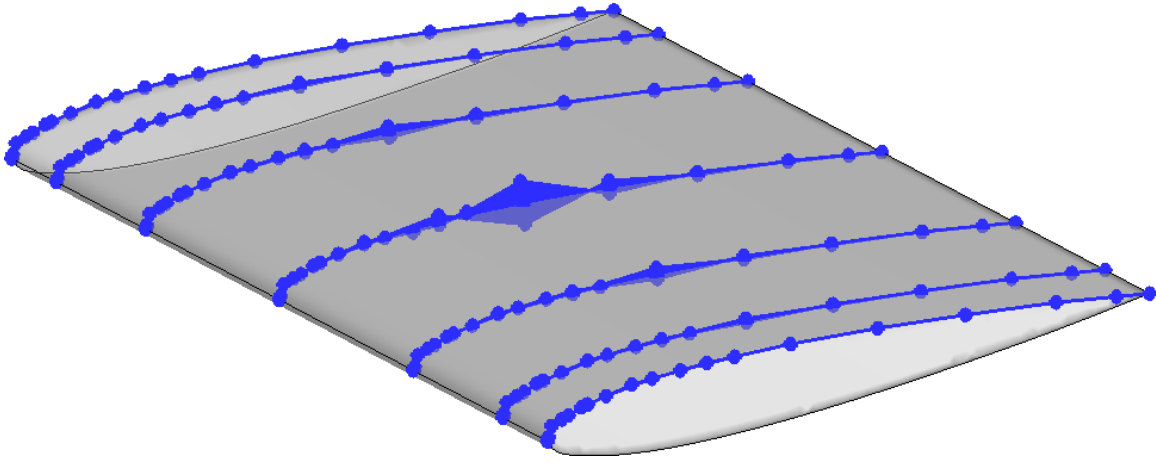
A simple example of this scenario for a B-spline surface is shown in Figure 5-1, where the design motion of surface control points are depicted by perturbing a single support point ( $y$ -coordinate) between the range  $[-0.034693, -0.023129]$ . The subsequent change in the  $U$  knot vector is shown in Figure 5-2(a)–(b). As illustrated in these results, the support point variation enabled the geometry kernel to allow knot insertion and variations in knot values of approximately  $10^{-5}$  in magnitude. This reality implies that the simple, static  $U$  construction in Section 5.1 insufficiently captures the design motion for B-spline curves and surfaces in a general geometry kernel.

This scenario exhibits two interesting consequences. First, the gradient methodology in Sections 5.2 and 5.4 can be greatly simplified since the function that maps the  $\mathbf{Q}_k$  support points to  $U$ ,  $\mathbf{Q}_k \rightarrow U$ , is unknown. Therefore, it is proposed that assuming a specific mapping function is as equally arbitrary as assuming no dependence of  $U$  on  $\mathbf{Q}_k$ . This assumption follows the generation procedure for a B-spline interpolant because the curve designer *chooses* the parameter span for each piecewise polynomial segment of a spline *and* specifies their end-point continuity (both defined through the knot vector and curve degree). Due to insufficient geometry information to dictate otherwise, any knot vector generation

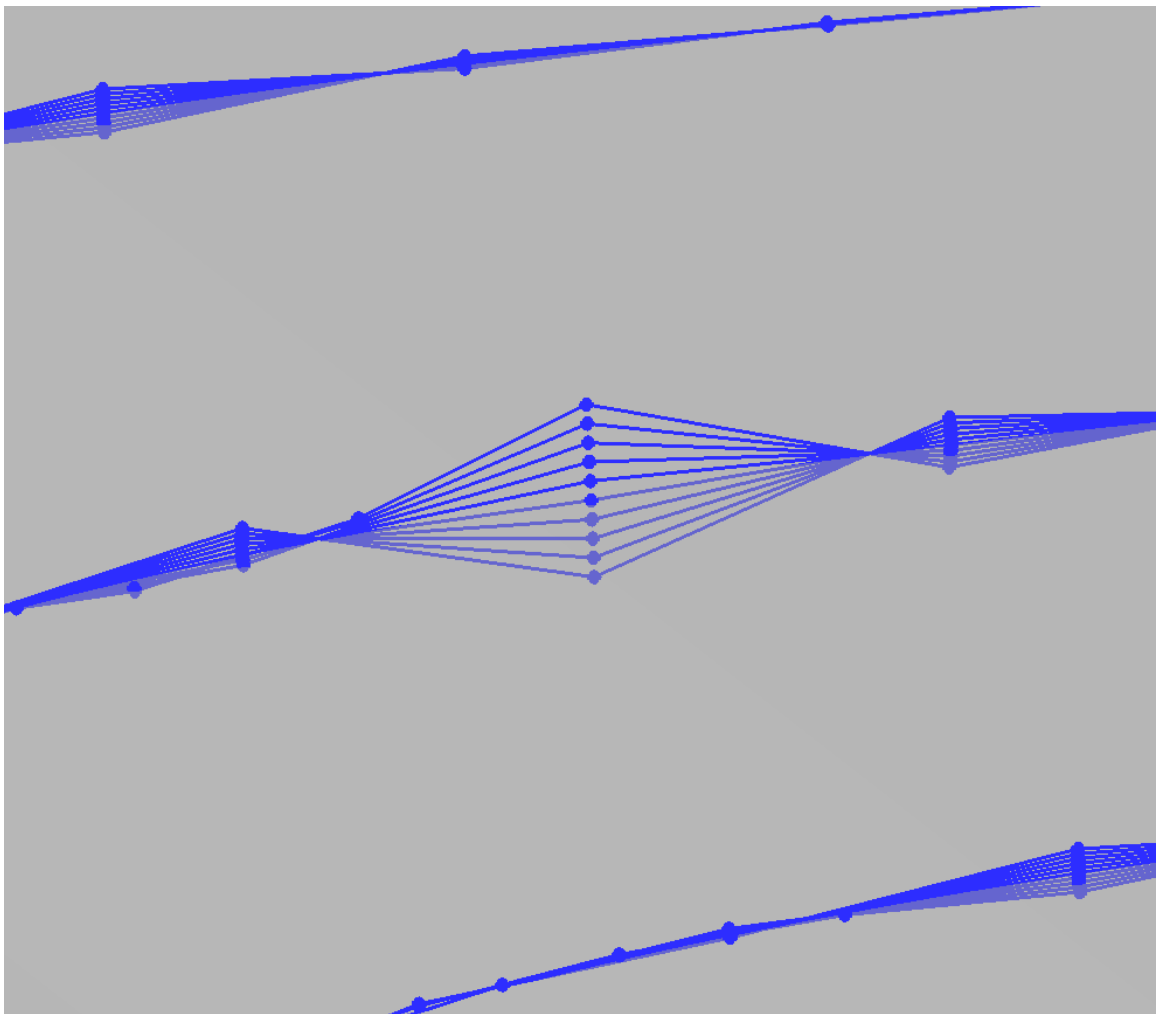
---

<sup>5</sup>This can vary between CAD system versions as well

<sup>6</sup>This has been seen for both  $u$  and  $v$  directions. In such cases  $\mathbf{A}_{\bar{N}}$  is augmented with a new second row as  $[-1, 1, 0, \dots, 0]$  and a new penultimate row as  $[0, \dots, 0, -1, 1]$  to handle these tangents.  $\mathbf{B}_Q$  is also augmented in like fashion with entries  $\mathbf{T}$  that reflect the tangency values at the second and penultimate rows.

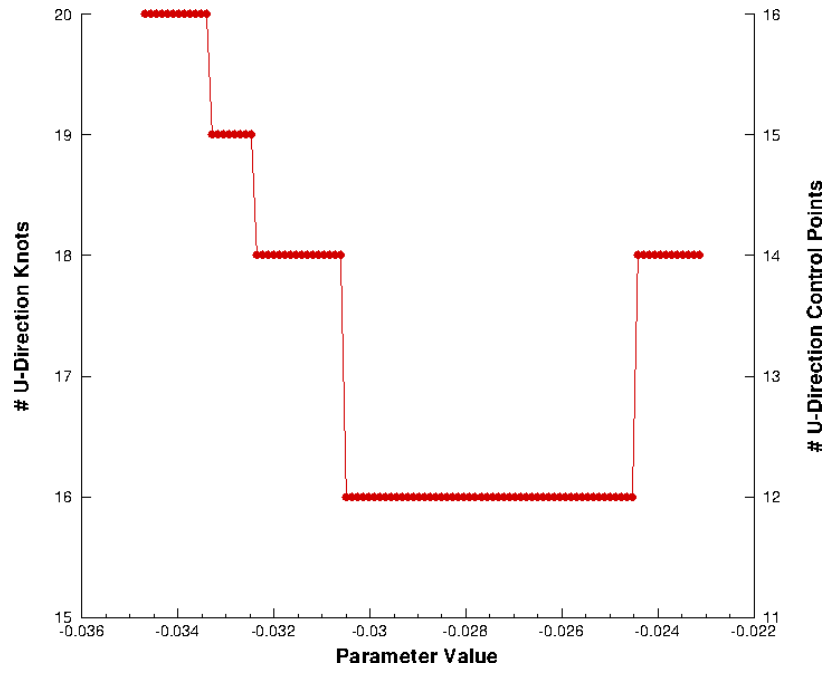


(a)

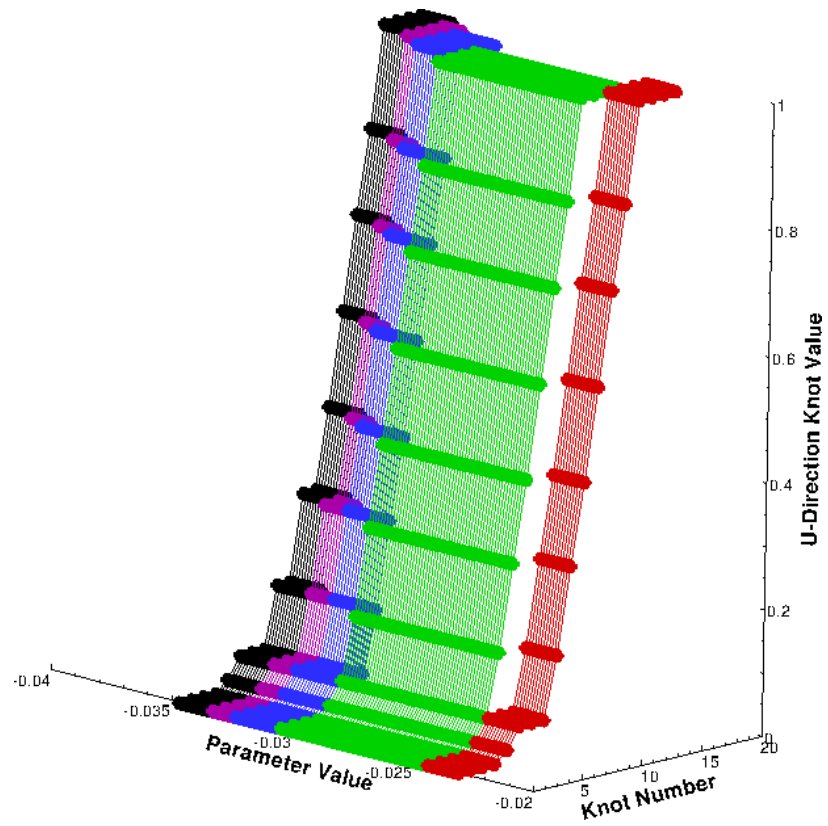


(b)

Figure 5-1: (a) Overall design motion of surface control points for a simple loft after perturbing a single support point ( $y$ -coordinate). (b) Zoomed view of the control point design trajectories.



(a)



(b)

Figure 5-2: (a) The variation in number of  $U$  knots during design motion and (b) the design motion for the  $U$  knots.

procedure can be declared plausible if the resulting B-spline satisfies the objective merits for a needed curve design. By this perspective, the assumption that  $U$  is not a function of  $\mathbf{Q}_k$  allows for  $\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}} = 0$ . This occurs since  $\bar{u}$  and  $U$ , the sole parameters in the basis functions  $N_{i,p}(u)$ , are assumed to have no dependence on  $\mathcal{P}$  if they are decoupled from  $\mathbf{Q}_k$ . Secondly, it is clear that using finite-differences between perturbed B-spline surfaces with different knots (which is possible even for small perturbations in certain parts of the design trajectory) will be an inconsistent operation (discussed further in Section 5.7).

These assumptions on  $U$  bring a simplification of the B-spline curve and surface geometry gradients. For B-spline curves the control point gradient simplifies from

$$\frac{d}{d\mathcal{P}}(\mathbf{X}_P) = \mathbf{A}_{\bar{N}}^{-1}(\partial_{\mathcal{P}}\mathbf{B}_Q - (\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})\mathbf{X}_P)$$

to

$$\frac{d}{d\mathcal{P}}(\mathbf{X}_P) = \mathbf{A}_{\bar{N}}^{-1}\partial_{\mathcal{P}}\mathbf{B}_Q. \quad (5.35)$$

The subsequent geometry gradient along the B-spline curve then simplifies from

$$\partial_{\mathcal{P}}\mathbf{Y} = \left(\partial_{\mathcal{P}}\mathbf{A}_N - \mathbf{A}_N\mathbf{A}_{\bar{N}}^{-1}(\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})\right)\mathbf{X}_P + \mathbf{A}_N\mathbf{A}_{\bar{N}}^{-1}\partial_{\mathcal{P}}\mathbf{B}_Q$$

to

$$\partial_{\mathcal{P}}\mathbf{Y} = \mathbf{A}_N\mathbf{A}_{\bar{N}}^{-1}\partial_{\mathcal{P}}\mathbf{B}_Q. \quad (5.36)$$

For B-spline surfaces, the control point gradient simplifies from

$$\frac{d\mathbb{P}}{d\mathcal{P}} = \mathbf{A}_{\bar{N}}^{-1}(\bar{u}) \left[ \frac{d\mathbb{B}_Q}{d\mathcal{P}} - \partial_{\mathcal{P}}(\mathbf{A}_{\bar{N}}(\bar{u}))\mathbb{P}\mathbf{A}_{\bar{N}}^T(\bar{v}) - \mathbf{A}_{\bar{N}}(\bar{u})\mathbb{P}(\partial_{\mathcal{P}}\mathbf{A}_{\bar{N}})^T(\bar{v}) \right] \mathbf{A}_{\bar{N}}^{-T}(\bar{v})$$

to

$$\frac{d\mathbb{P}}{d\mathcal{P}} = \mathbf{A}_{\bar{N}}^{-1}(\bar{u}) \frac{d\mathbb{B}_Q}{d\mathcal{P}} \mathbf{A}_{\bar{N}}^{-T}(\bar{v}). \quad (5.37)$$

The subsequent geometry gradient across the B-spline surface then simplifies from

$$\frac{d\mathbf{S}(u_a, v_a; \mathcal{P})}{d\mathcal{P}} = \partial_{\mathcal{P}}(\mathbf{A}_N(u_a))\mathbb{P}\mathbf{A}_N^T(v_a) + \mathbf{A}_N(u_a)\mathbb{P}\partial_{\mathcal{P}}(\mathbf{A}_N^T(v_a)) + \mathbf{A}_N(u_a) \frac{d\mathbb{P}}{d\mathcal{P}} \mathbf{A}_N^T(v_a)$$

to

$$\frac{d\mathbf{S}(u_a, v_a; \mathcal{P})}{d\mathcal{P}} = \mathbf{A}_N(u_a) \frac{d\mathbb{P}}{d\mathcal{P}} \mathbf{A}_N^T(v_a). \quad (5.38)$$

The final versions of the new sensitivity equation have the form

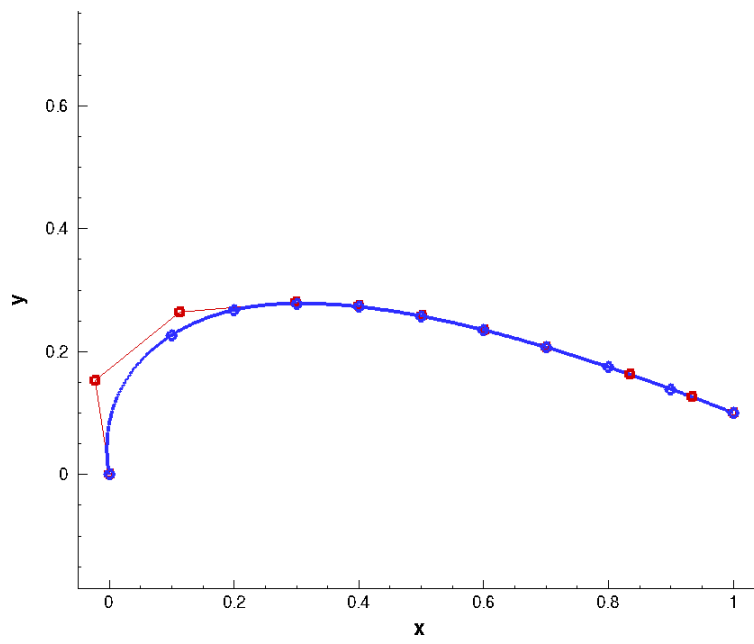
$$\begin{array}{ll} \mathbf{B}\text{-Spline Curve} & \mathbf{B}\text{-Spline Surface} \\ \mathbf{Y}' = \mathbf{A}'\mathbf{X}'_P & \mathbf{S}' = \mathbf{A}'(u)\mathbb{P}'\mathbf{A}'^T(v) \end{array}$$

These equations have the appearance of B-spline curve and B-spline surface definitions, which means that the geometry gradient of B-spline curves and B-spline surfaces are *another* B-spline curve or surface. The new “sensitivity B-spline curve” or “sensitivity B-spline surface” geometry, however, consists of interpolation through support points that are *gradients* and returning control polygons that are also *gradients*. Mathematically the geometry is represented in the same way, yet in this case interpolation is done through gradient information instead of Euclidean coordinates. Since the knot vectors for a B-spline curve or surface are preserved in their geometry gradient form, each exists in a consistent spline space (described in Section 5.7).

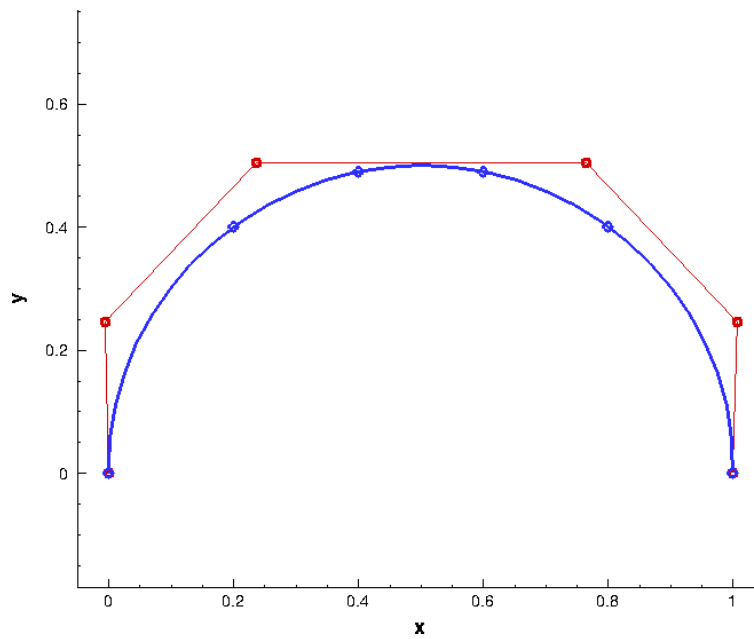
## 5.6 Examples of B-Spline Curve Geometry Gradients

Figure 5-3 illustrates two generic examples where support points are interpolated by B-spline curves within corresponding control polygons. Each is generated using the methodology in Section 5.1. The design velocity method from Section 5.5 is validated against finite-differencing for these examples using different parameters.

The finite-difference gradient is found by perturbing a baseline support point with a user-defined step-size ( $h = \pm 10^{-7}$ ) and interpolating new cubic B-spline curves through both perturbed sets of support points. Differencing is done at the same  $t$  values across each B-spline. Comparisons are made between both methods with the assumption that the knots  $U = \{u_i\}$  and parameters  $\bar{t}_k$  did not depend on the support points (i.e., the knots and parameters are fixed in the finite-difference computation). Results for these comparisons are given in Figures 5-4 through 5-8, where the absolute value of offset between both results are shown for non-zero design velocity components. In all example cases there is excellent agreement between the proposed analytic method and finite-differencing results.



(a)



(b)

Figure 5-3: Example B-spline curves interpolated through support points (blue circles) with a control polygon (red circles) determined by the method in Section 5.1.

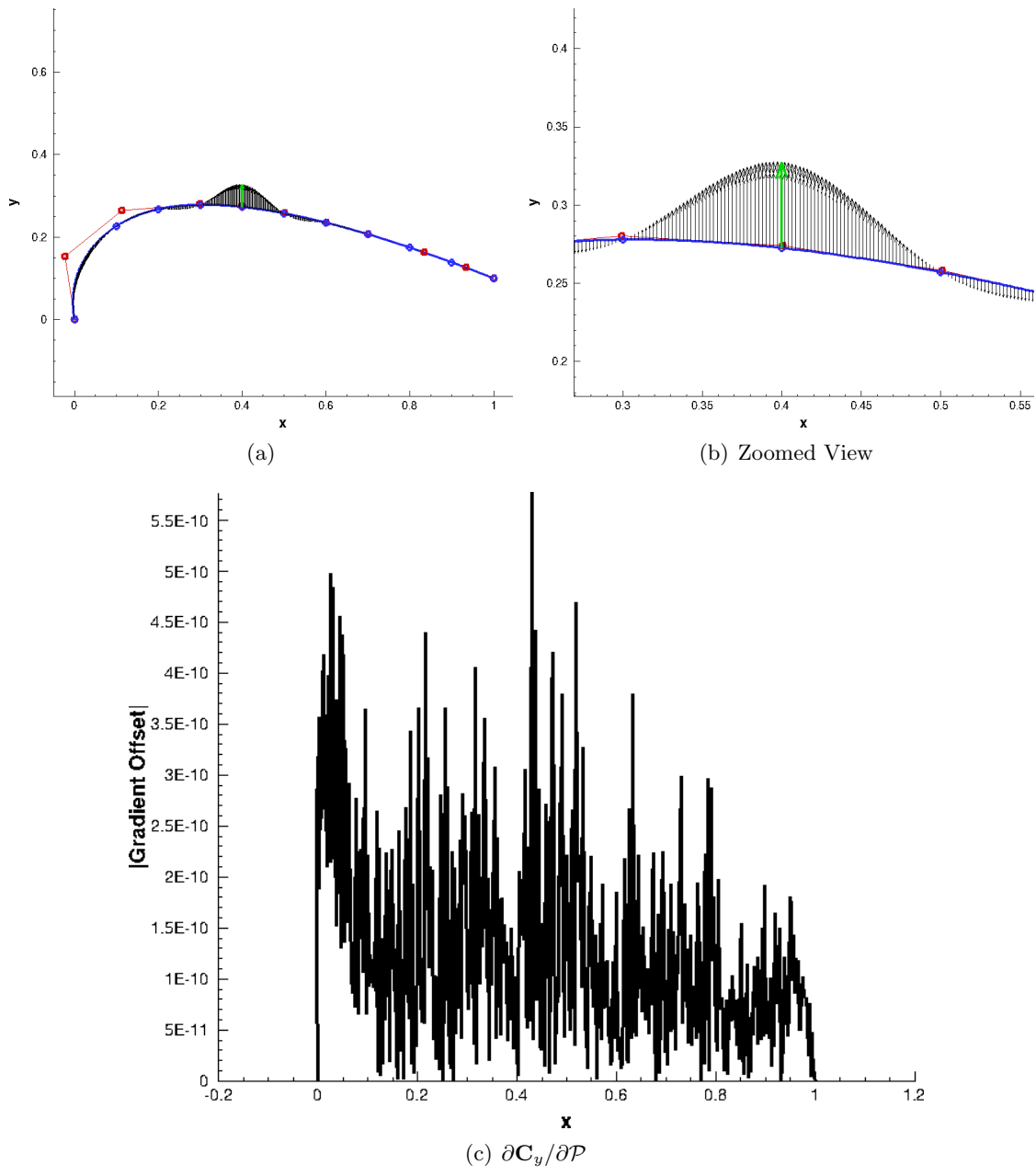


Figure 5-4: (a) Design velocity field with respect to  $\mathbf{Q}_{k,y}$  for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero).



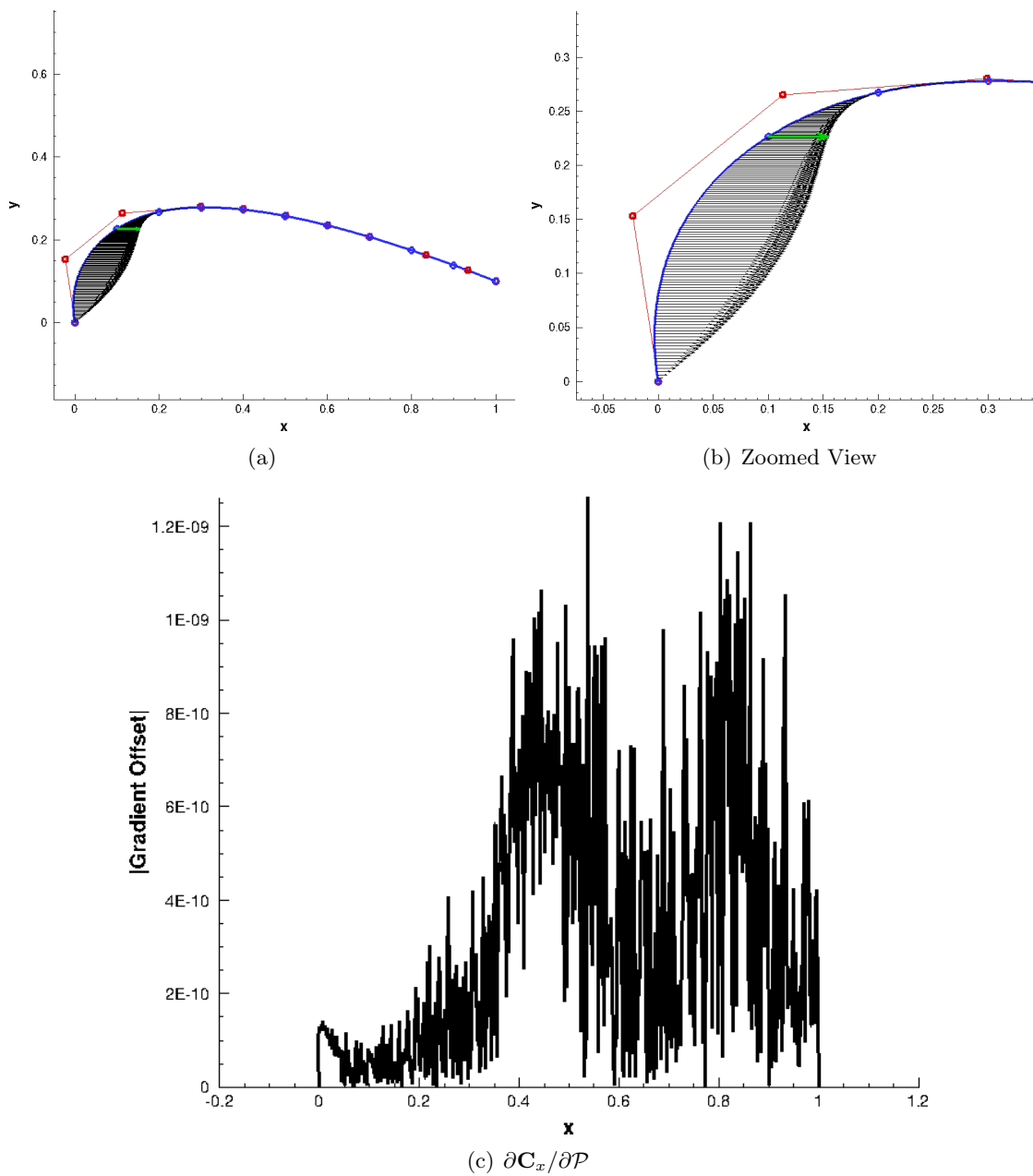


Figure 5-5: (a) Design velocity field with respect to  $\mathbf{Q}_{k,x}$  for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero).

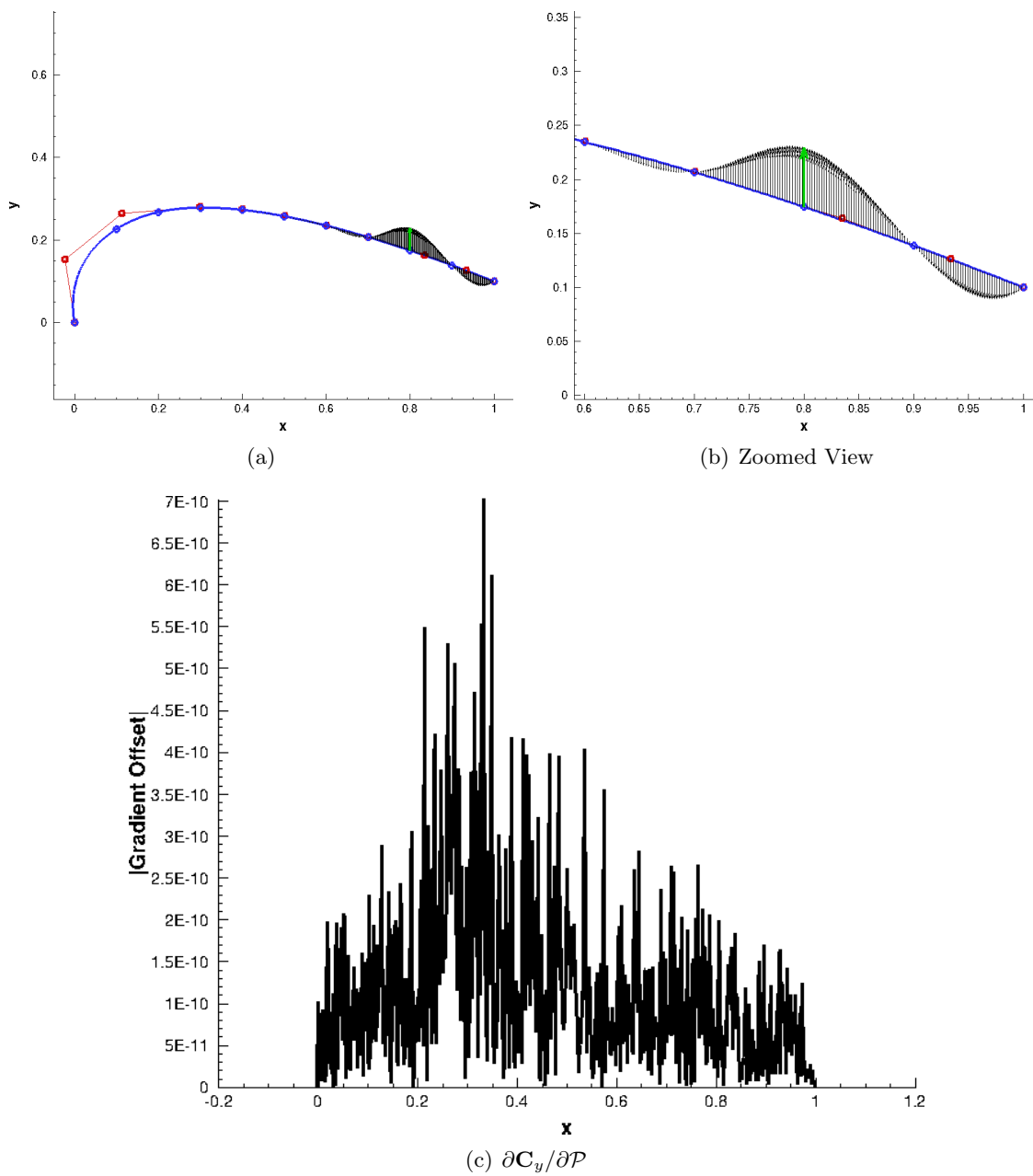
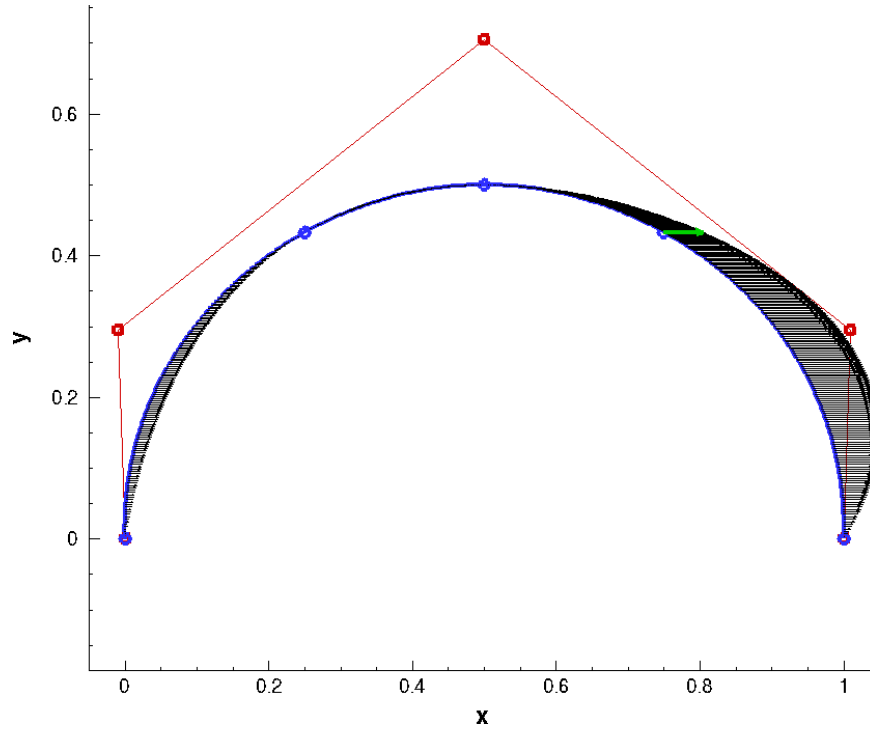
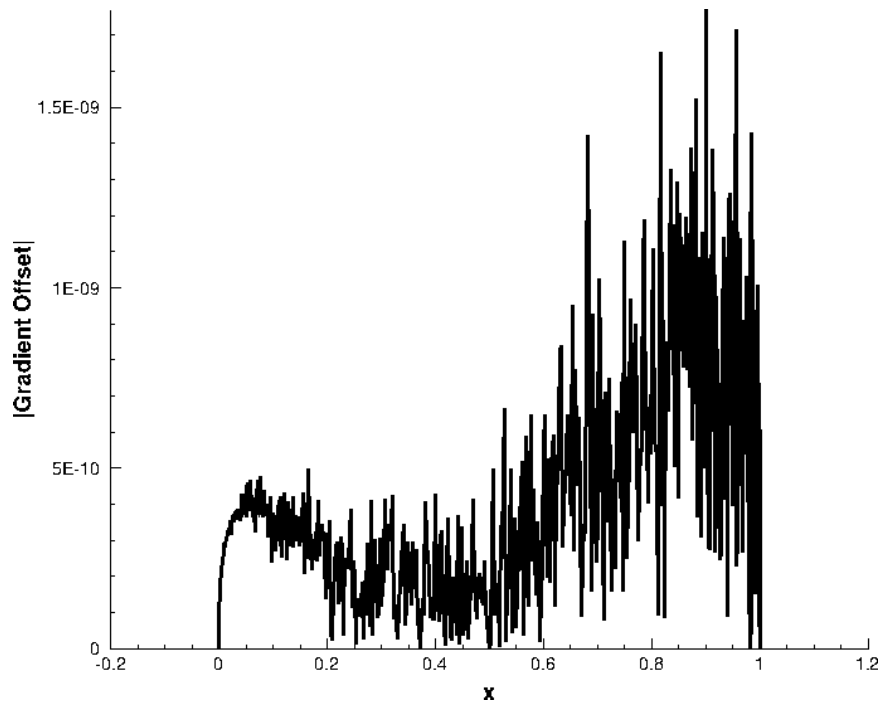


Figure 5-6: (a) Design velocity field with respect to  $\mathbf{Q}_{k,y}$  for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero).



(a)



(b)  $\partial C_x / \partial \mathcal{P}$

Figure 5-7: (a) Design velocity field with respect to  $\mathbf{Q}_{k,x}$  for an example cubic B-spline curve. The green vector denotes the perturbation direction (unit magnitude). (b) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero).

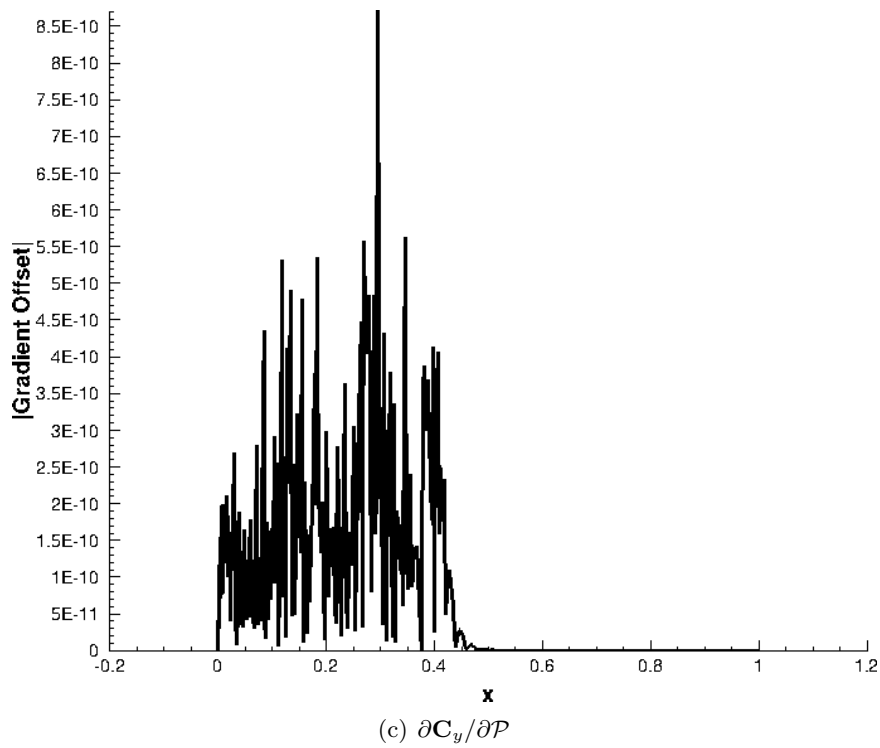
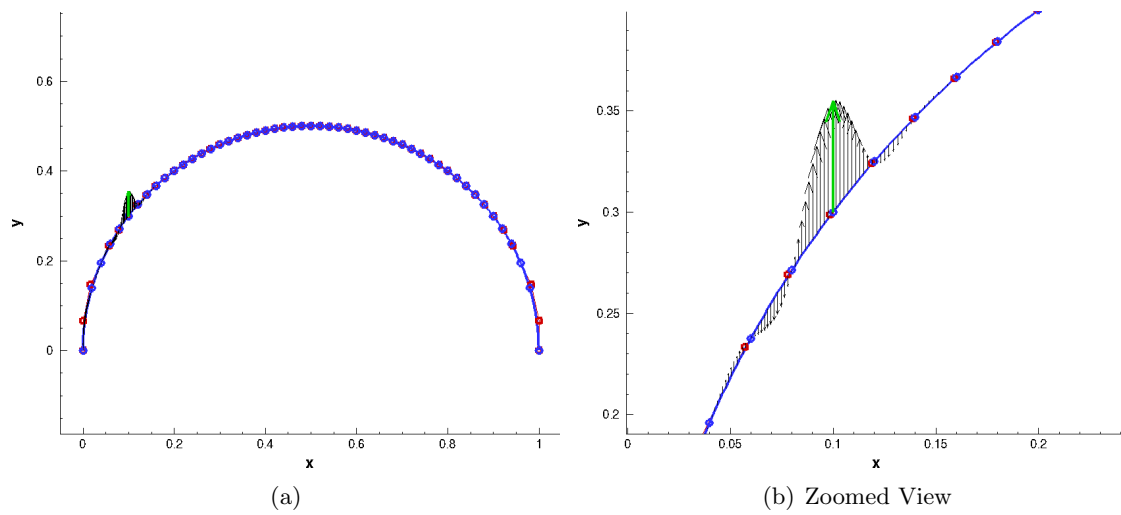


Figure 5-8: (a) Design velocity field with respect to  $\mathbf{Q}_{k,y}$  for an example cubic B-spline curve, with (b) a detailed-view around the perturbed parameter location. The green vector denotes the perturbation direction (unit magnitude). (c) Offset between finite-difference (knots fixed) and the analytic methodology gradient results (other gradient components are zero).

## 5.7 Examples of B-Spline Surface Geometry Gradients

A solid loft *feature* (i.e., a B-spline surface) created in the SolidWorks CAD system is seen in Figure 5-9(a). This loft has five cross-sections containing cubic B-splines and cubic interpolation span-wise. The gradient magnitude with respect to the  $z$ -coordinate of a single support is shown as a contour plot in Figure 5-9(b). Maximum design velocity is clearly seen where the driving “perturbation” exists. As expected, a patchwork plot of peaks and valleys is evident in the design velocity because there is no design velocity along isoparameter lines corresponding to the knot vector values  $(\bar{u}_k, \bar{v}_l)$  in  $U$  and  $V$ . Only at the intersection of isoparameter lines corresponding to the differentiated support point is a non-zero design velocity found. Figure 5-9(c) includes an overlay of design velocity vectors as well. Due to the local support property of B-spline surfaces, the design velocity magnitude decreases to zero at the edge of the influence region (i.e.,  $p + 1$  knot spans) for the cubic non-zero basis functions. In this case the entire B-spline surface encompasses that region.

A second solid loft *feature* shown in Figure 5-10(a) was created with SolidWorks. This loft contains nine linearly-scaled cross-section B-splines curves added between each of the five original cross-sections in the loft of Figure 5-9(a). A contour plot of the resulting design velocity magnitude field is shown in Figure 5-10(b) for the variation in  $z$ -coordinate direction at a support point. The design velocity vectors are also overlaid in Figure 5-10(c). Since the extent of non-zero design velocity is limited (by the local support property) to the knot span where local basis functions are non-zero, the region of influence for a support point variation is much less expansive here compared to the case in Figure 5-9.

A third example of a solid loft *feature* is shown in Figure 5-11(a). This loft was created with SolidWorks and contains B-spline curves oriented to approximate circular cross-sections. A contour plot of design velocity corresponding to a variation in  $y$ -coordinate for a single support point is shown in Figure 5-11(b), whereas the design velocity vectors are included in Figure 5-11(c). The design velocity influence is broad due to the large spacing between some cross-sections in the knot spans surrounding the varied support point.

As an aside, the design velocity field can also be used to forecast the design motion for a B-spline curve or surface (or other features as well). Unintended design motion is easily identified if the non-zero sensitivity region is larger, smaller, or in a different direction than expected for a given parameter. This information allows a designer to adapt the model

construction in order to limit or remove the potential for unintended design motion. The case of Figure 5-11(a) is illustrative if a designer does not want design motion aft of the nose when modifying the nose-section. Additional cross-section profiles will be needed in that region to mitigate its sensitivity to a support point near the nose.

A trade-off may quickly emerge when planning a model design trajectory, as in the case of Figures 5-9(a) and 5-10(a). Limiting the span-wise influence of a support point requires additional cross-sections, yet this increases the design space of the model. This may or may not be a hindrance for the optimization problems conducted with these models. On the other hand, fewer support points can limit the extent of “wavy” surfaces and provide a smaller design space. If transonic aerodynamics analysis is conducted on these models, there is an added concern in having large sensitivity regions with fewer support points, or “wavy” surfaces with too many support points. Both impact the design space differently by potentially introducing unexpected local minima. A designer will need to consider these issues when defining their model design space and construction approach. This is in accordance with the model construction methodology presented in Chapter 2.

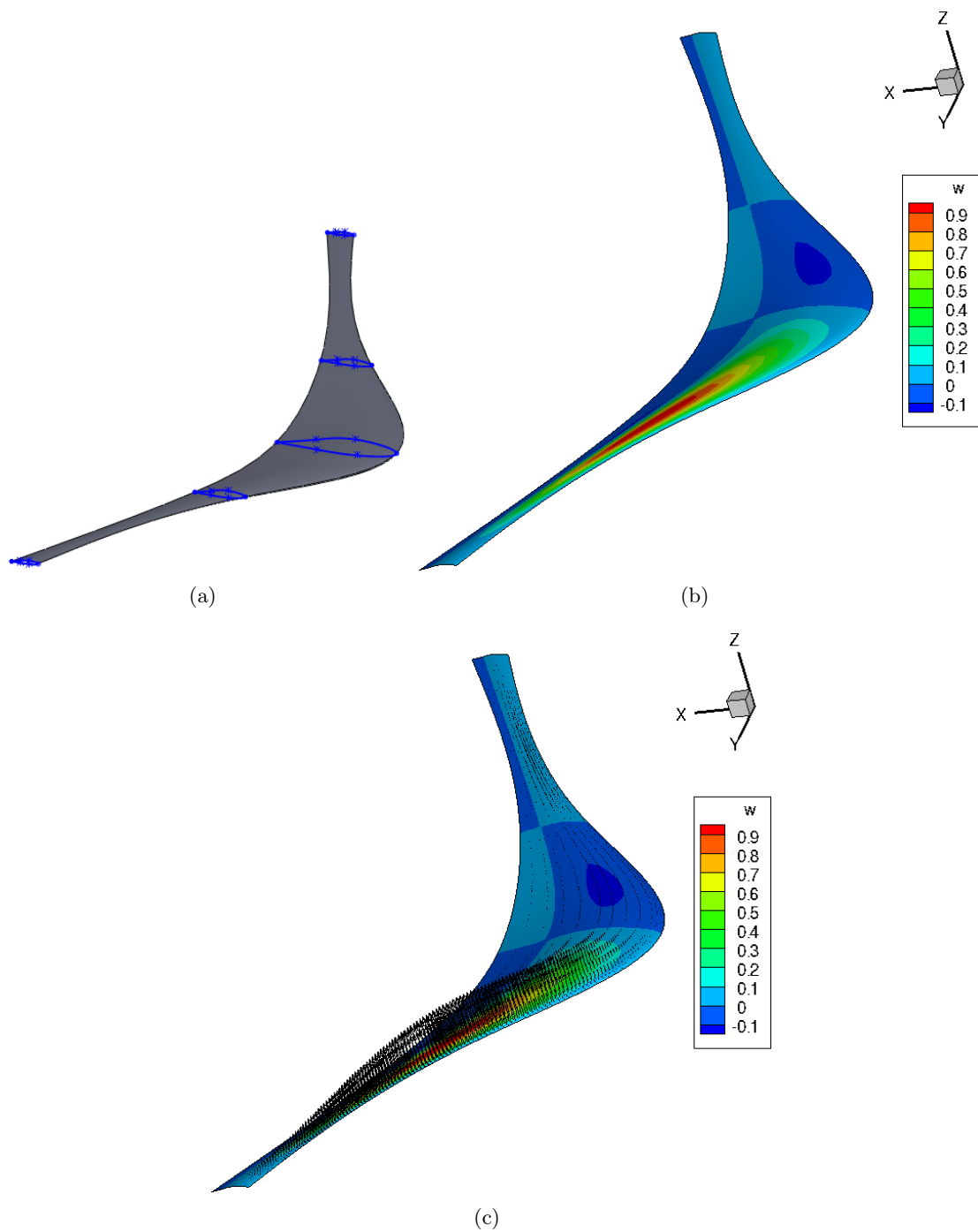


Figure 5-9: (a) A SolidWorks model of a generic, solid loft *feature* with five cross-section B-spline curves, (b) contour regions denote the gradient magnitude in the  $z$ -coordinate direction with respect to a variation in  $z$  at a single support point, and (c) an overlay of design velocity vectors for the same case.

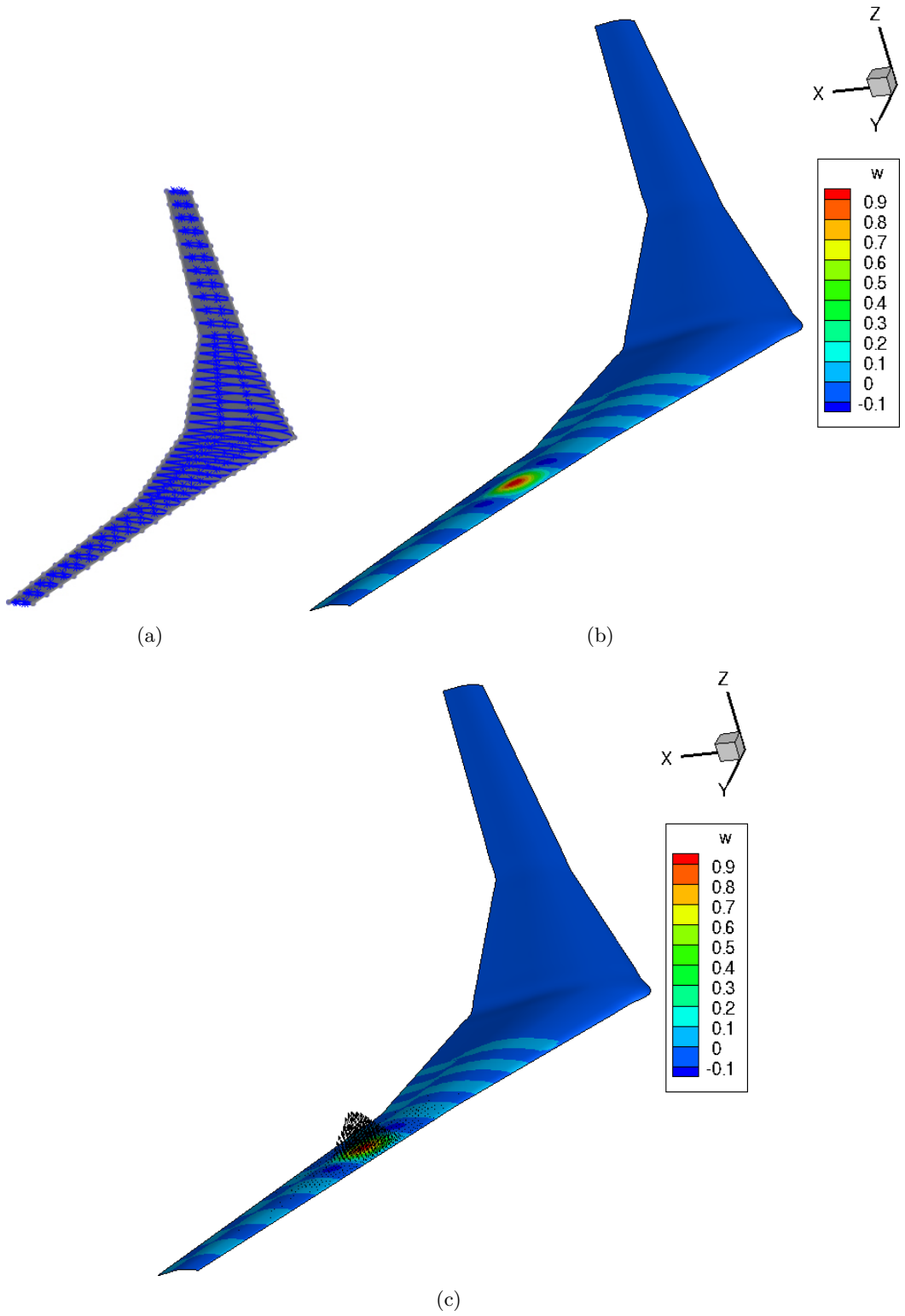
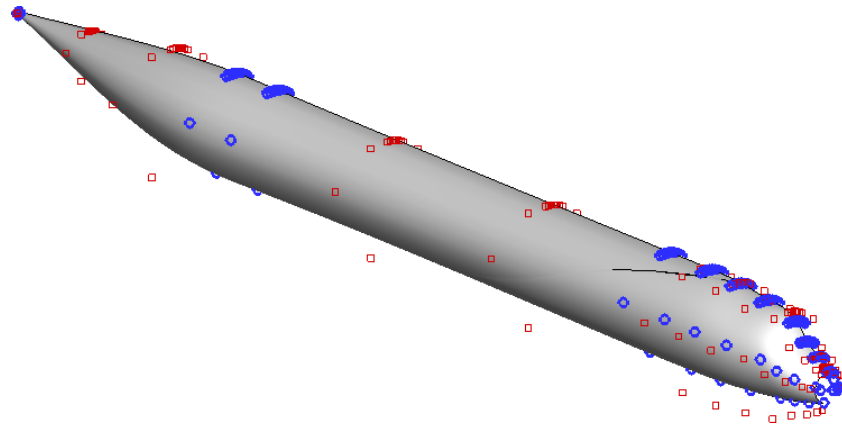
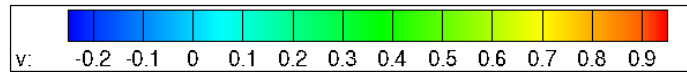
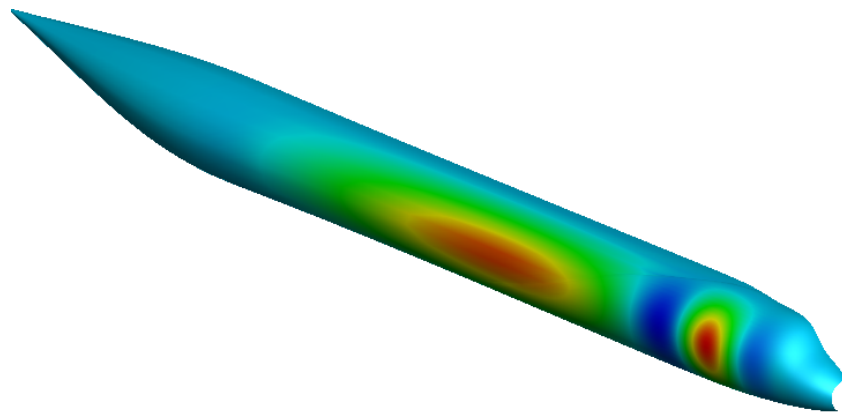


Figure 5-10: (a) A SolidWorks model of a generic, solid loft *feature* constructed with additional linearly-scaled cross-section B-spline curves, (b) contour regions denote the gradient magnitude in the  $z$ -coordinate direction with respect to a variation in  $z$  at a single support point, and (c) an overlay of design velocity vectors for the same case.

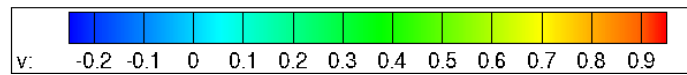
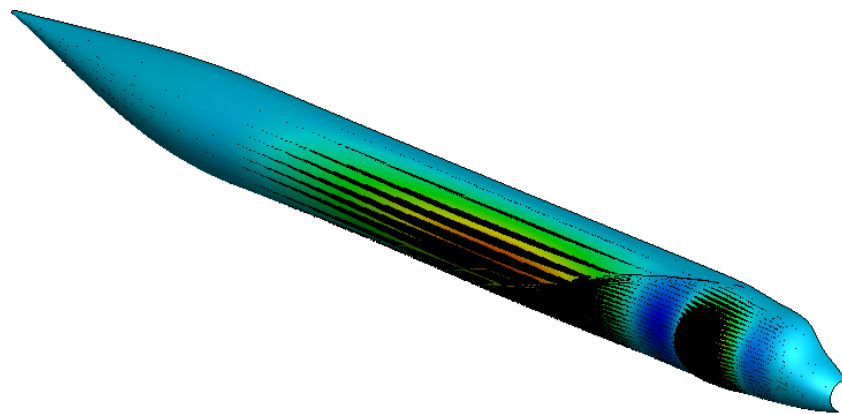




(a)



(b)



(c)

Figure 5-11: (a) A SolidWorks model of a generic, solid loft *feature* with various cross-section B-spline curves, (b) contour regions denote the gradient magnitude in the  $y$ -coordinate direction with respect to a variation in  $y$  at a single support point, and (c) an overlay of design velocity vectors for the same case.

## 5.8 Scenarios Creating Inconsistent Finite-Difference Results

An analytic B-spline surface is generated to compare geometry gradients obtained via finite-differencing and the analytic approach in Section 5.5. The geometry is a cubic B-spline surface interpolation of five cross-sectional cubic B-spline curves that crudely approximate a NACA 0012 profile<sup>7</sup>. The design velocity field from the analytic method is shown in Figure 5-12 with respect to a support point  $y$ -component.

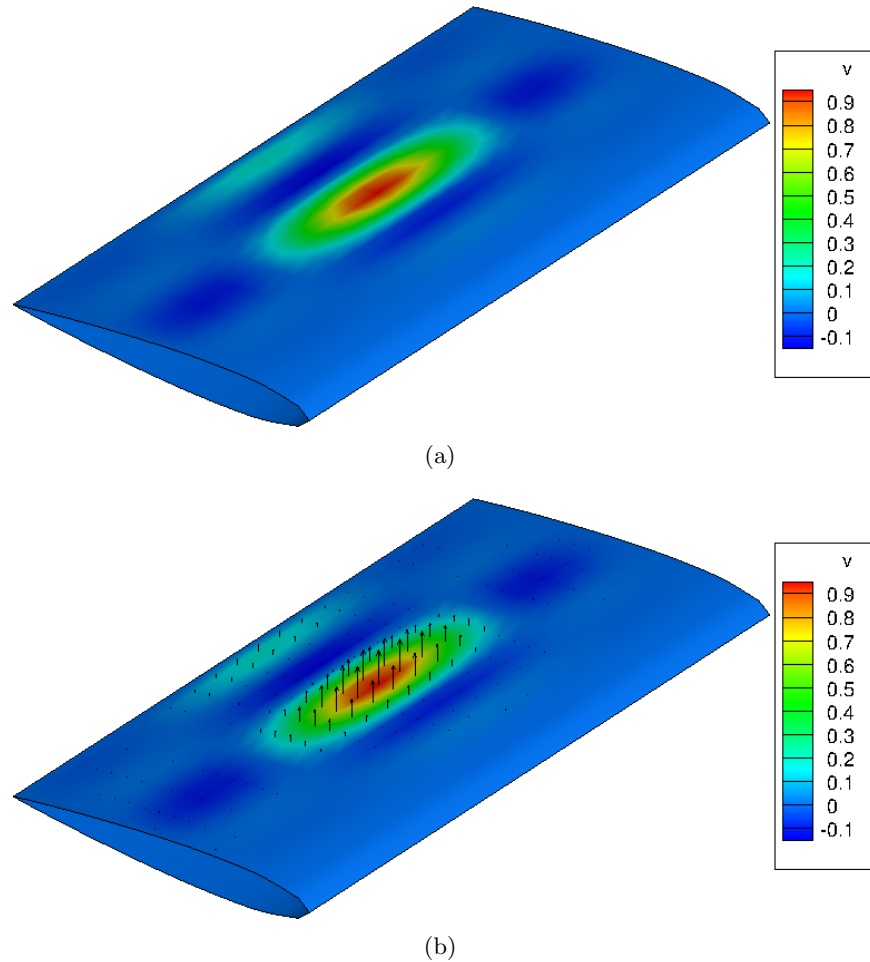


Figure 5-12: (a) The magnitude of design velocity with respect to a support point  $y$ -component with (b) associated design velocity vectors.

Using a finite-difference step-size of  $h = 10^{-5}$  (chord value of 0.5), a knot *preserving* and *non-preserving* comparison is made with the analytic method. The preserving method fixes the baseline B-spline surface knots (including cross-sectional B-spline curve knots) when

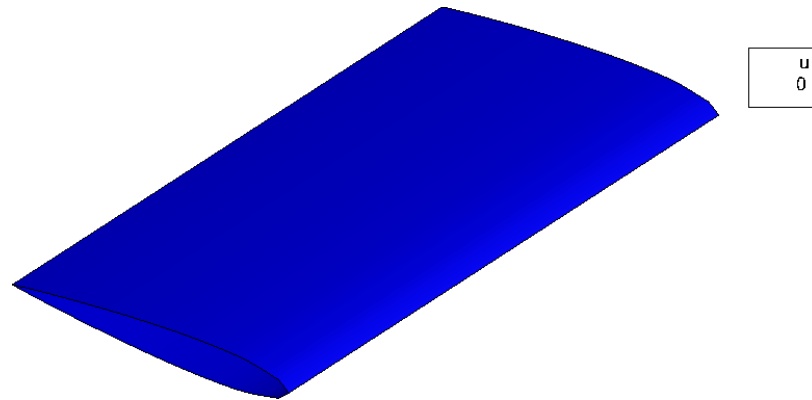
---

<sup>7</sup>Few support points are used here in order to highlight discrepancies between the design velocity fields from both methods instead of resolving the airfoil shape properly.

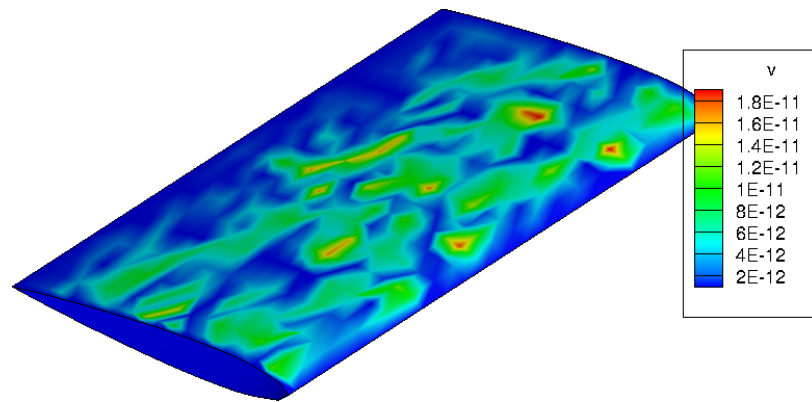
generating the  $+h$  and  $-h$  geometry perturbations. Conversely, the non-preserving method *recalculates* the knots along B-spline curves and the B-spline surface after perturbing with  $\pm h$ . Figure 5-13 depicts the absolute value offset between design velocity components for the knot preserving finite-difference method and the analytic approach. In this case agreement between the two methods is excellent. Figure 5-14 shows the absolute value offset between the knot non-preserving finite-difference method and the analytic approach. These results show a discrepancy between the two approaches. A profile view of the design velocity vectors in Figure 5-15 indicates that the results are similar, yet the finite-difference result contains non-zero  $u$ -components in design velocity.

The reason for the discrepancy in Figure 5-14 is found in the computation of the knot non-preserving finite-difference operation. It is important to understand that interpolation with B-spline curves is accomplished as piecewise polynomials that meet at knots with continuity based on polynomial degree (see Section 5.1). This is very different from interpolation using a single polynomial. An example helps illustrate how this difference in definition leads to the discrepancy above. A single polynomial,  $P_1$ , of degree  $p$  defined over a domain  $t_1 \in [0, 1]$  can be compared in a consistent manner with another polynomial,  $P_2$ , of degree  $p$  defined over the same domain. In other words, both  $P_1$  and  $P_2$  are defined in the same polynomial space and share the same domain. This implies that the operation  $P_1 + P_2 = P_3$  and  $P_3$  belongs in the same polynomial space as  $P_1$  and  $P_2$  in a consistent manner. However, if  $P_2$  were defined over the domain  $t_2 \in [-2, 2]$ , then it would exist in a different domain space *unless* it were reparameterized such that  $t_2 \rightarrow t_1$ . Without reparameterizing  $P_2$ , then a consistent comparison between  $P_1$  and  $P_2$  could not be made. This implies that the operation  $P_1 + P_2 = P_4$  would not be consistent and  $P_4$  would not be in the same domain space as  $P_3$ .

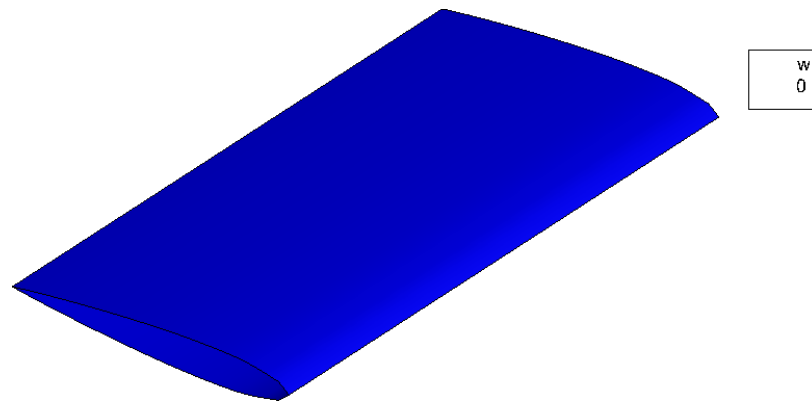
In like manner, an interpolating B-spline curve can be considered a *spline space* [5] which is also a vector space. Piecewise polynomials are segments of a B-spline curve that originate from a weighted summation of B-spline basis functions defined in a domain space (typically  $t \in [0, 1]$ ). These segments span half-intervals (i.e.,  $[u_i, u_{i+1})$ ) between the knots in a knot vector  $U$  in that domain. The inclusion of  $U$  distinguishes the spline space from the polynomial space (a polynomial space is a subspace of a spline space). Therefore, a B-spline curve  $S_1$  defined in  $t_1 \in [0, 1]$  with  $U_1$  resides in the same spline space as  $S_2$  if  $S_2$  is defined over the same domain and knot vector. This allows  $S_1 + S_2 = S_3$  to be consistent and



(a)

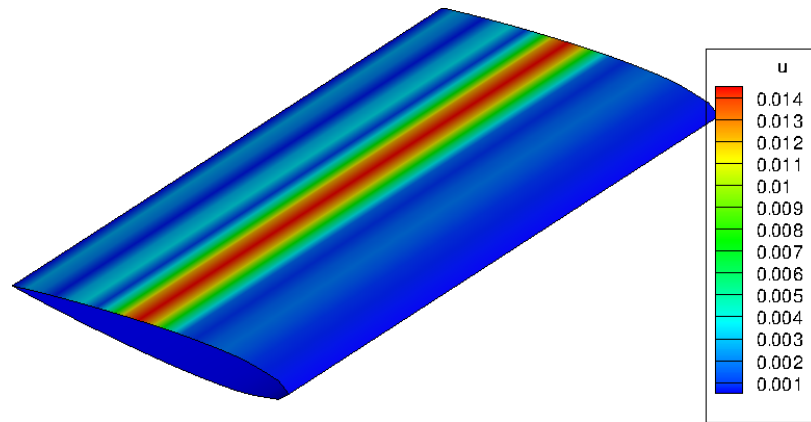


(b)

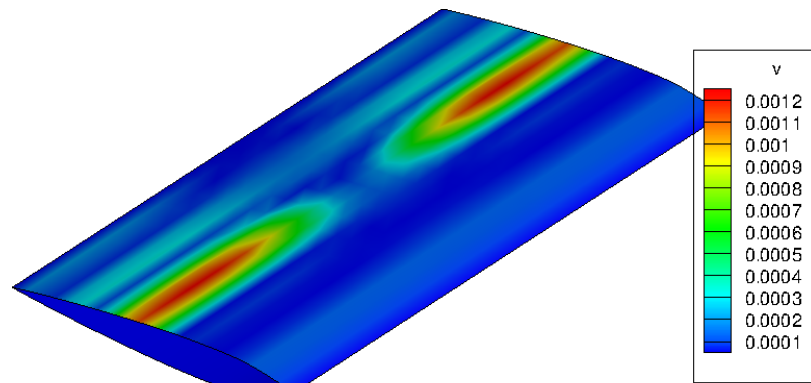


(c)

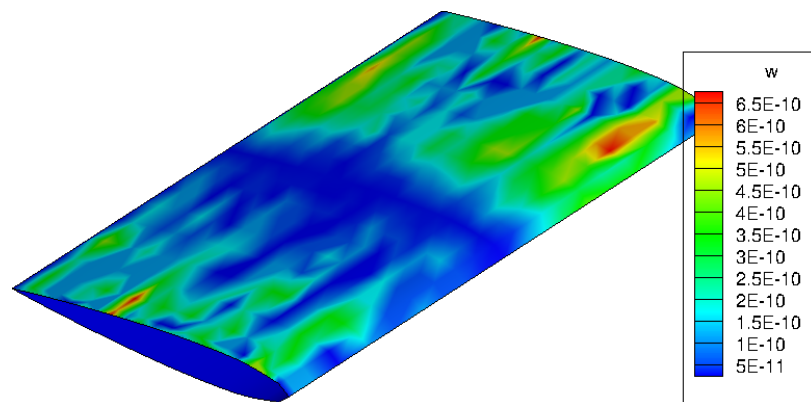
Figure 5-13: (a)–(c) Comparison of geometry gradient offset from finite-difference (knot-preserving) and Section 5.5 approaches. The  $u$ - and  $w$ -component offset are exactly zero in this case.



(a)



(b)



(c)

Figure 5-14: (a)–(c) Comparison of geometry gradient offset from finite-difference (non-preserving knots) and Section 5.5 approaches.

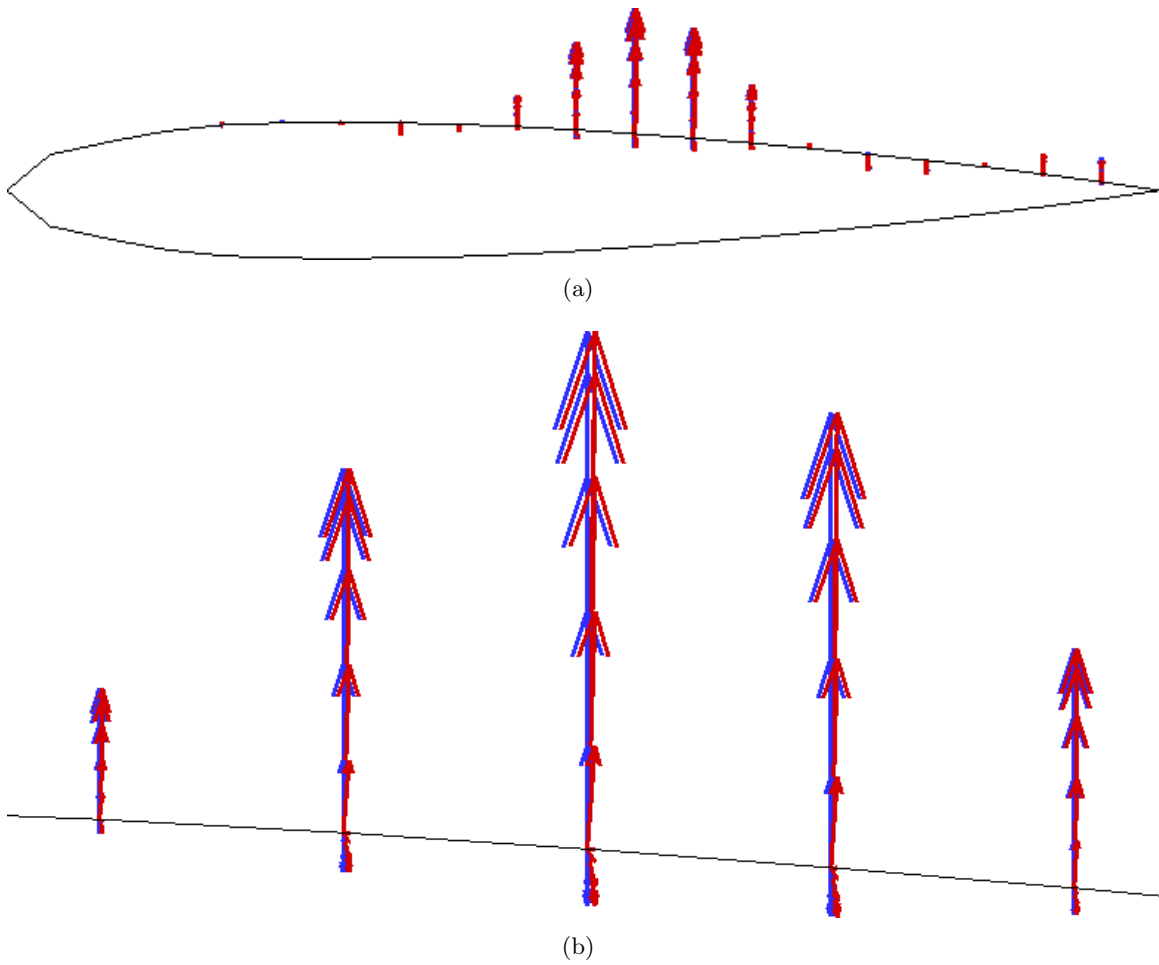


Figure 5-15: (a) A profile view of the design velocity vectors from the knot non-preserving central-difference and analytic method results of Figure 5-14. (b) A detailed view of the design velocity discrepancy. Analytic gradient results are in blue and finite-difference results (knot non-preserving) are in red.

$S_3$  belongs to the same spline space as  $S_1$  and  $S_2$ . However, if  $S_2$  is defined over the same domain as  $S_1$  but has its own knot vector  $U_2$ , then  $S_2$  belongs to its own spline space *unless* it is reparameterized with a new knot vector. The only way for  $S_1$  and  $S_2$  to exist in the same spline space if initially  $U_1 \neq U_2$  is for a new knot vector  $U_3 = U_1 \cup U_2$  to be defined for both  $S_1$  and  $S_2$ . If this is not done, then  $S_1 + S_2 = S_4$  is not consistent and  $S_4$  does not exist in the same spline space as  $S_1$  or  $S_2$ . Inconsistency stems from the fact that the B-spline curve is made of piecewise polynomials defined across each knot span. If the knot vectors do not match between two B-spline curves, then the piecewise polynomials will not share the same knot span domain. Furthermore, if there are repeated knots in one knot vector that are not found in the other (multiplicity is greater than 1), then the piecewise polynomials will not share the same continuity (meaning the differentiability of the polynomials at that location will not match) at those knot values and a similar inconsistency will occur when comparing such B-spline curves without reparameterization [5].

This realization explains the discrepancy seen in Figure 5-14.<sup>8</sup> The finite-difference operation introduces a spurious design velocity in the  $u$ -direction because knots in that direction are recomputed. Even relatively “small” differences between the reparameterized knots leads to the spurious result. This outcome is seen in Tables 5.1 and 5.2, where the offset between B-spline surface and curve  $U$  knot vectors are shown for the case in Figures 5-14. In contrast, no spurious design velocity results are seen (Figure 5-13) when knots are preserved in the finite-difference operation.

---

<sup>8</sup>Although this discussion pertains to continuous B-spline curves, it extends to continuous B-spline surfaces. This discussion also applies to discretized B-spline curves and surfaces, where the piecewise polynomial segments of a B-spline curve are compared (or patches for B-spline surfaces) at discrete locations rather than the entire continuous geometry.

Baseline $U$	$U_{+h}$	$U_{-h}$	$(U_{+h})-(U_{-h})$
0.0000000000000000	0.0000000000000000	0.0000000000000000	0.000000e+00
0.0000000000000000	0.0000000000000000	0.0000000000000000	0.000000e+00
0.0000000000000000	0.0000000000000000	0.0000000000000000	0.000000e+00
0.0000000000000000	0.0000000000000000	0.0000000000000000	0.000000e+00
0.072929342779053	0.072929342578264	0.072929342979796	-4.015316e-10
0.162220912504359	0.162220912057733	0.162220912950883	-8.931496e-10
0.282837207075283	0.282837206296577	0.282837207853811	-1.557234e-09
0.419686459338646	0.419686456322491	0.419686462354641	-6.032150e-09
0.560361415142910	0.560361412657183	0.560361417628601	-4.971419e-09
0.702491753786752	0.702491751827443	0.702491755746144	-3.918701e-09
1.0000000000000000	1.0000000000000000	1.0000000000000000	0.000000e+00
1.0000000000000000	1.0000000000000000	1.0000000000000000	0.000000e+00
1.0000000000000000	1.0000000000000000	1.0000000000000000	0.000000e+00
1.0000000000000000	1.0000000000000000	1.0000000000000000	0.000000e+00

Table 5.1: A comparison between the  $U$ -direction knot vector for the B-spline surfaces generated in the central-difference problem of Figure 5-14. Underlined digits denote a variation from the baseline value. The step-size used in this case was  $h = 10^{-5}$ .

B-Spline Curve Cross-Sections				
# 1	# 2	# 3	# 4	# 5
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-2.007658e-09	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-4.465748e-09	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-7.786171e-09	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-3.016075e-08	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-2.485709e-08	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-1.959350e-08	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

Table 5.2: The offset between the  $i$ th B-spline curve knot vectors  $U_{+h}^i - U_{-h}^i$  ( $i = 1, \dots, 5$ ) obtained in central-differencing for Figure 5-14 are shown for the non-preserving case. Non-zero offsets exist where the geometry gradient was computed ( $h = 10^{-5}$ ).



Since the actual algorithm for generating  $U$  and  $V$  knot vectors is not known for a given geometry kernel, it is expected that finite-differencing B-spline surfaces within CAD systems may yield as large, or larger, spurious design velocity results and greater knot offsets than those seen with the analytic example above. The analytic loft example is recreated in the SolidWorks CAD system (see Figure 6-10(a)) to conduct finite-differencing against the same support point (with step-size  $h = 1.0 \times 10^{-5}$  as well) to validate this. Results are displayed in Figure 5-16, which shows larger offsets in design velocity and cross-section knot values than that seen in the analytic case. The reason for this is seen in Tables 5.3 through 5.5. Even though the surface knot vector  $U$  does not change (there is a small change in the  $V$  knots which impacts the  $w$ -component of design velocity, as seen in Table 5.4), the cross-section B-spline curve *does* show an offset between perturbed states in Table 5.5. In this case the initial surface  $U$  knots match the cross-section knots. Perturbing the cross-section spline creates new knot vectors that imply new  $\bar{u}$  values (i.e., support point parameters), which are set to the interior knot values by the geometry kernel in this case. Thus, inconsistent finite-differencing occurs when the perturbed surface support points are reparameterized with different  $\bar{u}$  values that are not at the end of individual knot spans in  $U$ . This explanation is strengthened by considering the finite-difference derivation for B-spline curves and surfaces that follows.

### 5.8.1 Analysis for Linearized B-Spline Curves

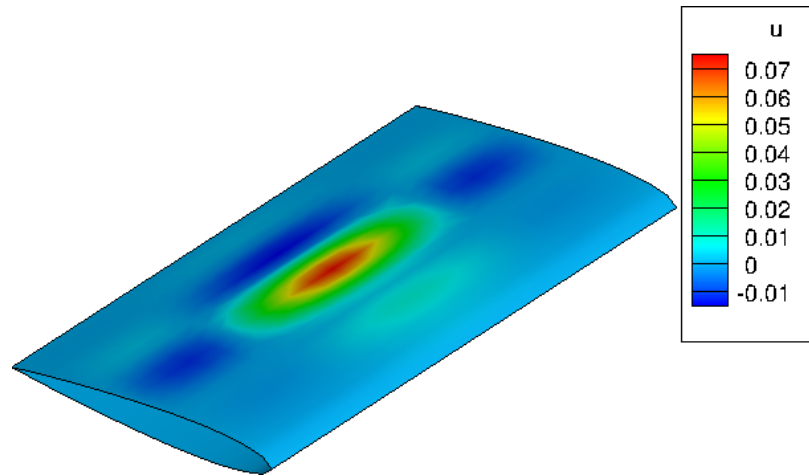
For B-spline curves we define

$$g(\mathbf{B}_Q; \mathcal{P}) = \mathbf{B}_Q = \mathbf{A}_{\bar{N}}(\bar{t}, U) \mathbf{X}_P \quad (5.39)$$

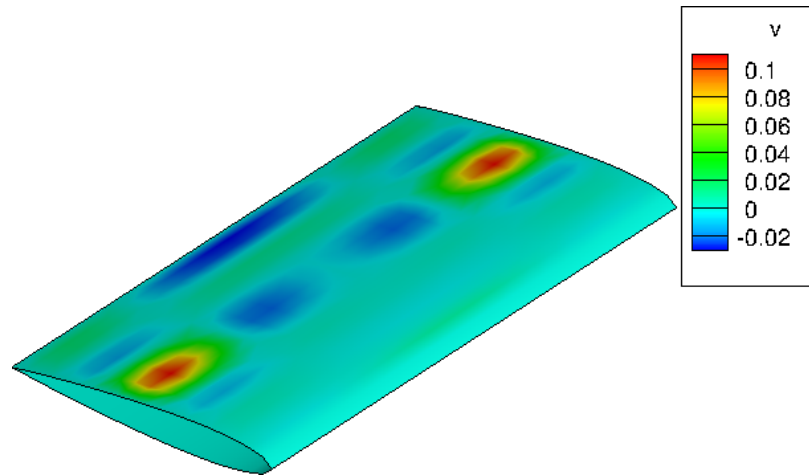
$$\begin{aligned} g_+(\mathbf{B}_Q; \mathcal{P} + \delta\mathcal{P}) &= \mathbf{B}_Q + \delta\mathbf{B}_Q = \mathbf{A}_{\bar{N}_+}(\bar{t}_+, U_+) (\mathbf{X}_{P_+} + \delta\mathbf{X}_{P_+}) \\ &= \mathbf{A}_{\bar{N}_+}(\bar{t}_+, U_+) \mathbf{X}'_{P_+} \end{aligned} \quad (5.40)$$

$$\begin{aligned} g_-(\mathbf{B}_Q; \mathcal{P} - \delta\mathcal{P}) &= \mathbf{B}_Q - \delta\mathbf{B}_Q = \mathbf{A}_{\bar{N}_-}(\bar{t}_-, U_-) (\mathbf{X}_{P_-} + \delta\mathbf{X}_{P_-}) \\ &= \mathbf{A}_{\bar{N}_-}(\bar{t}_-, U_-) \mathbf{X}'_{P_-} \end{aligned} \quad (5.41)$$

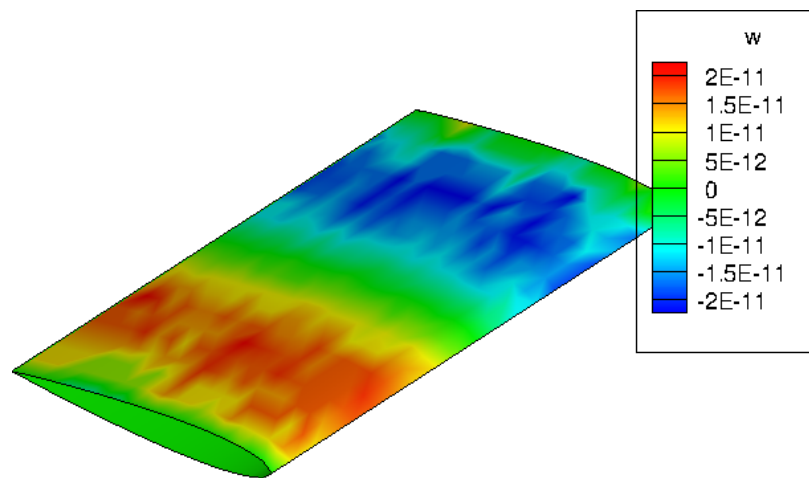
with  $\mathbf{X}'_{P_+} = \mathbf{X}_{P_+} + \delta\mathbf{X}_{P_+}$  and  $\mathbf{X}'_{P_-} = \mathbf{X}_{P_-} + \delta\mathbf{X}_{P_-}$  for some control point perturbations  $\delta\mathbf{X}_{P_+}$  and  $\delta\mathbf{X}_{P_-}$  that result from a parameter perturbation  $\delta\mathcal{P}/\mathcal{P} \ll 1$ . Also  $\bar{t}_+$  and  $\bar{t}_-$  are the new support point parameterizations and  $U_+$  and  $U_-$  stem from non-preserved knot vectors. By assuming a Taylor expansion representation of the B-spline curve at all  $t$



(a)



(b)



(c)

Figure 5-16: (a)–(c) Comparison of geometry gradient offset between finite-difference results and the Section 5.5 approach on a CAD model geometry.

<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$	$(U_{+h})-(U_{-h})$
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.017932446381	0.017932446381	0.017932446381	0.000000e+00
0.051549762576	0.051549762576	0.051549762576	0.000000e+00
0.149305819381	0.149305819381	0.149305819381	0.000000e+00
0.285807155557	0.285807155557	0.285807155557	0.000000e+00
0.413398646288	0.413398646288	0.413398646288	0.000000e+00
0.559853576171	0.559853576171	0.559853576171	0.000000e+00
0.707832022969	0.707832022969	0.707832022969	0.000000e+00
0.83978966222	0.83978966222	0.83978966222	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00

Table 5.3: A comparison between the  $U$  knot vector for the B-spline surfaces generated by finite-differencing the CAD model from Figure 5-16. The step-size used in this case is  $h = 10^{-5}$ , which resulted in no change of the knot vectors.

<b>Baseline <math>V</math></b>	$V_{+h}$	$V_{-h}$	$(V_{+h})-(V_{-h})$
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.0000000000	0.0000000000	0.0000000000	0.000000e+00
0.2500000000	0.249999999988	0.249999999988	3.885781e-16
0.5000000000	0.5000000000	0.5000000000	0.000000e+00
0.7500000000	0.750000000012	0.750000000012	-3.330669e-16
1.0000000000	1.0000000000	1.0000000000	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00
1.0000000000	1.0000000000	1.0000000000	0.000000e+00

Table 5.4: A comparison between the  $V$  knot vector for the B-spline surfaces generated by finite-differencing the CAD model from Figure 5-16. The step-size used in this case is  $h = 10^{-5}$ , which resulted in a minor change of the knot vectors.

<b>B-Spline Curve Cross-Sections</b>				
# 1	# 2	# 3	# 4	# 5
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	2.006753e-08	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	5.768741e-08	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	1.670826e-07	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	3.198361e-07	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	4.626190e-07	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	2.895703e-06	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-3.269543e-07	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	-1.792854e-07	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

Table 5.5: The offset between the  $i$ th B-spline curve knot vectors  $U_{+h}^i - U_{-h}^i$  ( $i = 1, \dots, 5$ ) obtained in finite-differencing the CAD model in Figure 5-16 are shown. Non-zero offsets exist where the geometry gradient was computed ( $h = 10^{-5}$ ).

coordinates in its domain, we write

$$\begin{aligned}
g_+(\mathbf{B}_Q; \mathcal{P} + \delta\mathcal{P}) &= g(\mathbf{B}_Q; \mathcal{P}) + \frac{dg}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2g}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \frac{1}{6}\frac{d^3g}{d\mathcal{P}^3}\delta\mathcal{P}^3 + \dots \\
g_-(\mathbf{B}_Q; \mathcal{P} - \delta\mathcal{P}) &= g(\mathbf{B}_Q; \mathcal{P}) - \frac{dg}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2g}{d\mathcal{P}^2}\delta\mathcal{P}^2 - \frac{1}{6}\frac{d^3g}{d\mathcal{P}^3}\delta\mathcal{P}^3 + \dots
\end{aligned}$$

and subtract to obtain

$$\frac{dg}{d\mathcal{P}} \approx \frac{g_+ - g_-}{2\delta\mathcal{P}} - \mathcal{O}(\delta\mathcal{P}^2), \tag{5.42}$$

which is rewritten as

$$\frac{d\mathbf{B}_Q}{d\mathcal{P}} \approx \left( \frac{1}{2\delta\mathcal{P}} \right) \left[ \mathbf{A}_{\bar{N}_+}(\bar{t}_+, U_+) \mathbf{X}'_{P_+} - \mathbf{A}_{\bar{N}_-}(\bar{t}_-, U_-) \mathbf{X}'_{P_-} \right] - \mathcal{O}(\delta\mathcal{P}^2). \tag{5.43}$$

When knots and support point parameterizations are preserved, the finite-difference derivative simplifies because

$$\mathbf{A}_{\bar{N}_+}(\bar{t}_+, U_+) = \mathbf{A}_{\bar{N}_-}(\bar{t}_-, U_-) = \mathbf{A}_{\bar{N}}(\bar{t}, U),$$

with  $\bar{t}_+ = \bar{t}_- = \bar{t}$  and  $U_+ = U_- = U$ . Also, the perturbed control points become  $\mathbf{X}'_{P_+} = \mathbf{X}_{P_+}$  and  $\mathbf{X}'_{P_-} = \mathbf{X}_{P_-}$ , resulting in

$$\frac{d\mathbf{B}_Q}{d\mathcal{P}} \approx \left( \frac{1}{2\delta\mathcal{P}} \right) \mathbf{A}_{\bar{N}}(\bar{t}, U) (\mathbf{X}_{P_+} - \mathbf{X}_{P_-}) - \mathcal{O}(\delta\mathcal{P}^2), \quad (5.44)$$

which takes a familiar form in the limit of  $\delta\mathcal{P} \rightarrow 0$  (as found in the analytic case of Section 5.5):

$$\begin{aligned} \frac{d\mathbf{B}_Q}{d\mathcal{P}} &= \mathbf{A}_{\bar{N}}(\bar{t}, U) \frac{d\mathbf{X}_P}{d\mathcal{P}} \\ \Rightarrow \frac{d\mathbf{X}_P}{d\mathcal{P}} &= \mathbf{A}_{\bar{N}}^{-1}(\bar{t}, U) \frac{d\mathbf{B}_Q}{d\mathcal{P}}. \end{aligned} \quad (5.45)$$

This ideal case is possible when knots and support point parameterizations are preserved. When this is not the case the finite-difference approximation remains in the form of (5.43), which results in spurious design velocity components because the control point gradient is made inconsistent. The issue arises either by differencing B-spline curves that do not reside in the same spline space (i.e., different  $U$  knot vectors) or by using different support point parameterizations (i.e., different  $\bar{t}$ ). To see this, we return to (5.43) and consider rewriting  $\mathbf{A}_{\bar{N}_+}(\bar{t}_+, U_+)$  and  $\mathbf{A}_{\bar{N}_-}(\bar{t}_-, U_-)$  as

$$\begin{aligned} \mathbf{A}_{\bar{N}_+}(\bar{t}_+, U_+) &= \mathbf{A}_{\bar{N}}(\bar{t}, U) + \boldsymbol{\varepsilon}_+ \\ \mathbf{A}_{\bar{N}_-}(\bar{t}_-, U_-) &= \mathbf{A}_{\bar{N}}(\bar{t}, U) + \boldsymbol{\varepsilon}_-, \end{aligned}$$

for some deviations  $\boldsymbol{\varepsilon}_+$  and  $\boldsymbol{\varepsilon}_-$  in the basis function matrix. These are used in (5.43) with  $\delta\mathcal{P} \rightarrow 0$  and  $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_+ + \boldsymbol{\varepsilon}_-$  to yield:

$$\begin{aligned} \frac{d\mathbf{B}_Q}{d\mathcal{P}} &\approx \left( \frac{1}{2\delta\mathcal{P}} \right) \left[ (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \boldsymbol{\varepsilon}_+) \mathbf{X}'_{P_+} - (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \boldsymbol{\varepsilon}_-) \mathbf{X}'_{P_-} \right] - \mathcal{O}(\delta\mathcal{P}^2) \\ &\approx \frac{\mathbf{A}_{\bar{N}}(\bar{t}, U) (\mathbf{X}'_{P_+} - \mathbf{X}'_{P_-})}{2\delta\mathcal{P}} + \frac{(\boldsymbol{\varepsilon}_+ + \boldsymbol{\varepsilon}_-) (\mathbf{X}'_{P_+} - \mathbf{X}'_{P_-})}{2\delta\mathcal{P}} - \mathcal{O}(\delta\mathcal{P}^2) \\ &= (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \boldsymbol{\varepsilon}) \frac{d\hat{\mathbf{X}}_P}{d\mathcal{P}}. \end{aligned} \quad (5.46)$$

In (5.46) it is clear that the ideal case is not obtained since solving for the control point sensitivities will yield  $\frac{d\hat{\mathbf{X}}_P}{d\mathcal{P}} \neq \frac{d\mathbf{X}_P}{d\mathcal{P}}$  when the true value is  $\frac{d\mathbf{X}_P}{d\mathcal{P}}$ . The deviation from ideal is estimated by using  $\frac{d\hat{\mathbf{X}}_P}{d\mathcal{P}} = \frac{d\mathbf{X}_P}{d\mathcal{P}} + \boldsymbol{\delta} \left( \frac{d\mathbf{X}_P}{d\mathcal{P}} \right)$  and expanding (5.46). By adopting the following

notation,

$$\partial_{\mathcal{P}}(\hat{\mathbf{X}}_P) = \frac{d\hat{\mathbf{X}}_P}{d\mathcal{P}}, \quad \partial_{\mathcal{P}}(\mathbf{X}_P) = \frac{d\mathbf{X}_P}{d\mathcal{P}}, \quad \delta\partial_{\mathcal{P}}(\mathbf{X}_P) = \delta\left(\frac{d\mathbf{X}_P}{d\mathcal{P}}\right)$$

we continue with

$$\begin{aligned} \frac{d\mathbf{B}_Q}{d\mathcal{P}} &= (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \varepsilon) (\partial_{\mathcal{P}}(\mathbf{X}_P) + \delta\partial_{\mathcal{P}}(\mathbf{X}_P)) \\ &= \mathbf{A}_{\bar{N}}(\bar{t}, U)\partial_{\mathcal{P}}(\mathbf{X}_P) + \mathbf{A}_{\bar{N}}(\bar{t}, U)\delta\partial_{\mathcal{P}}(\mathbf{X}_P) + \varepsilon\partial_{\mathcal{P}}(\mathbf{X}_P) + \varepsilon\delta\partial_{\mathcal{P}}(\mathbf{X}_P) \\ (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \varepsilon)\delta\partial_{\mathcal{P}}(\mathbf{X}_P) &= -\varepsilon\partial_{\mathcal{P}}(\mathbf{X}_P). \end{aligned} \quad (5.47)$$

By taking consistent norms and applying the triangle inequality, the deviation magnitude becomes

$$\begin{aligned} \|\mathbf{A}_{\bar{N}}(\bar{t}, U) + \varepsilon\| \cdot \|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\| &\leq \|\varepsilon\| \cdot \|\partial_{\mathcal{P}}(\mathbf{X}_P)\| \\ \Rightarrow \frac{\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\|}{\|\partial_{\mathcal{P}}(\mathbf{X}_P)\|} &\leq \frac{\|\varepsilon\|}{\|\mathbf{A}_{\bar{N}}(\bar{t}, U)\| + \|\varepsilon\|}. \end{aligned} \quad (5.48)$$

This result implies that the deviation from ideal is at most the same order of magnitude as the offset caused by reparameterization of the knot vector or support points. Numerical experiments with a cross-section B-spline curve from the geometry in Figure 5-14 confirmed this outcome, where  $\|\varepsilon\| = 9.88 \times 10^{-8}$  created a control point sensitivity deviation of  $\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\| = 4.85 \times 10^{-9}$  after perturbing a support point by  $+\mathcal{P} = 1.0 \times 10^{-5}$  (with  $\|\mathbf{A}_{\bar{N}}(\bar{t}, U)\| = 1.13$  and  $\|\partial_{\mathcal{P}}(\mathbf{X}_P)\| = 0.055$  from finite-differences). As expected, when  $\|\varepsilon\| \rightarrow 0$  the deviation becomes  $\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\| \rightarrow 0$ .

Now that the deviation in control points is known due to the reparameterization error, this deviation will create an error in each component of design velocity. This is seen by considering the linearization of a B-spline curve, where the notation  $(\ )_+$  corresponds to the regenerated curve after perturbing a support point by step-size  $+\delta\mathcal{P}$  and  $(\ )_-$  corresponds to the curve obtained after a perturbation by  $-\delta\mathcal{P}$ :

$$\left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{FD}} = \frac{\mathbf{Y}_+ - \mathbf{Y}_-}{2\delta\mathcal{P}}. \quad (5.49)$$

Expanding  $\mathbf{Y}_+$  will suffice since the case for  $\mathbf{Y}_-$  is analogous, thus giving

$$\begin{aligned}\mathbf{Y}_+ &= \mathbf{A}_N(t, U) \mathbf{X}'_{P_+} = \mathbf{A}_N(t, U) \mathbf{A}_{\bar{N}_+}^{-1}(\bar{t}_+, U_+) \mathbf{B}_{Q_+} \\ &= \mathbf{A}_N(t, U) (\mathbf{X}_{P_+} + \delta \mathbf{X}_{P_+}).\end{aligned}$$

Substituting back into (5.49) the finite-difference computation becomes

$$\begin{aligned}\left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{FD}} &= \mathbf{A}_N(t, U) \left( \frac{\mathbf{X}_{P_+} - \mathbf{X}_{P_-}}{2\delta \mathcal{P}} \right) + \mathbf{A}_N(t, U) \left( \frac{\delta \mathbf{X}_{P_+} - \delta \mathbf{X}_{P_-}}{2\delta \mathcal{P}} \right) \quad \delta \mathcal{P} \rightarrow 0 \\ &= \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{True}} + \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{Error}},\end{aligned}\tag{5.50}$$

where

$$\left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{True}} \equiv \mathbf{A}_N(t, U) \left( \frac{\mathbf{X}_{P_+} - \mathbf{X}_{P_-}}{2\delta \mathcal{P}} \right)\tag{5.51}$$

$$\left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{Error}} \equiv \mathbf{A}_N(t, U) \left( \frac{\delta \mathbf{X}_{P_+} - \delta \mathbf{X}_{P_-}}{2\delta \mathcal{P}} \right).\tag{5.52}$$

It is possible for  $(\partial \mathbf{Y} / \partial \mathcal{P})_{\text{Error}} \neq \mathbf{0}$  even in design velocity components *not* associated with a perturbation. This is seen by considering the  $(\ )_+$  perturbation scenario after perturbing a support point to make  $\mathbf{B}_{Q_+} = \mathbf{B}_Q + \delta \mathbf{B}_Q$ :

$$\mathbf{B}_{Q_+} = (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \epsilon_+) (\mathbf{X}_{P_+} + \delta \mathbf{X}_{P_+})\tag{5.53}$$

$$= (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \epsilon_+) \mathbf{X}_{P_+} + (\mathbf{A}_{\bar{N}}(\bar{t}, U) + \epsilon_+) \delta \mathbf{X}_{P_+}.\tag{5.54}$$

By recalling that  $\mathbf{B}_{Q_+} = \mathbf{A}_{\bar{N}}(\bar{t}, U) \mathbf{X}_{P_+}$ , the deviation simplifies to

$$\delta \mathbf{X}_{P_+} = -(\mathbf{A}_{\bar{N}}(\bar{t}, U) + \epsilon_+)^{-1} \epsilon_+ \mathbf{X}_{P_+}.\tag{5.55}$$

Since the right hand side of (5.55) contains only non-zero terms, then  $\delta \mathbf{X}_{P_+}$  will always be non-zero for *every* component of control point design velocity (the same is true for  $\delta \mathbf{X}_{P_-}$ ). As  $\epsilon_+ \rightarrow \mathbf{0}$  then  $\delta \mathbf{X}_{P_+} \rightarrow \mathbf{0}$  as well (likewise as  $\epsilon_- \rightarrow \mathbf{0}$  then  $\delta \mathbf{X}_{P_-} \rightarrow \mathbf{0}$ ).

These results translate to a non-zero  $(\partial \mathbf{Y} / \partial \mathcal{P})_{\text{Error}}$  for each component of the B-spline curve design velocity. Numerical experiments of design velocity error using the analytic cross-section geometry of Figure 5-15 match the results predicted using this error analysis

(to within  $1.0 \times 10^{-17}$ ) for  $\delta \mathbf{X}_{P_+}$ . This implies that the spurious design velocity can be removed from the finite-difference computation *if* the baseline and perturbed curves are “reverse-engineered” to calculate the reparameterization error. Although tests with B-splines of different order (or other piecewise continuous polynomials written as a sum of weighted basis functions) were not conducted, these cases represent different values in the matrices  $\mathbf{A}_{\bar{N}}(\bar{t}, U)$  and  $\mathbf{A}_N(t, U)$ . The error analysis here is general to any case where finite-differencing is done between such piecewise continuous polynomials that do not share the same parameter domain or knot vector.

The magnitude of design velocity error along  $\mathbf{Y}$  is expected to be

$$\left\| \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right\|_{\text{Error}} \approx \mathcal{O} \left( \frac{\| \delta \mathbf{X}_{P_+} - \delta \mathbf{X}_{P_-} \|}{|\delta \mathcal{P}|} \right). \quad (5.56)$$

In the case of a cross-section in Figure 5-14, the spurious control point design velocities were of  $\mathcal{O}(1.0 \times 10^{-8})$  for a support point perturbation of  $\delta \mathcal{P} = 1.0 \times 10^{-5}$ , thus making

$$\left\| \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right\|_{\text{Error}} \approx \mathcal{O} \left( \frac{1.0 \times 10^{-8}}{1.0 \times 10^{-5}} \right) = 1.0 \times 10^{-3},$$

which is consistent with the spurious design velocity magnitudes seen in Figure 5-14.

## 5.8.2 Analysis for Linearized B-Spline Surfaces

For B-spline surfaces we define

$$f(\mathbb{B}_Q; \mathcal{P}) = \mathbb{B}_Q = \mathbf{A}_{\bar{N}}(\bar{u}, U) \mathbb{P} \mathbf{A}_{\bar{N}}^T(\bar{v}, V) \quad (5.57)$$

$$\begin{aligned} f_+(\mathbb{B}_Q; \mathcal{P} + \delta \mathcal{P}) &= \mathbb{B}_Q + \delta \mathbb{B}_Q = \mathbf{A}_{\bar{N}_+}(\bar{u}_+, U_+) (\mathbb{P}_+ + \delta \mathbb{P}_+) \mathbf{A}_{\bar{N}_+}^T(\bar{v}_+, V_+) \\ &= \mathbf{A}_{\bar{N}_+}(\bar{u}_+, U_+) \mathbb{P}'_+ \mathbf{A}_{\bar{N}_+}^T(\bar{v}_+, V_+) \end{aligned} \quad (5.58)$$

$$\begin{aligned} f_-(\mathbb{B}_Q; \mathcal{P} - \delta \mathcal{P}) &= \mathbb{B}_Q - \delta \mathbb{B}_Q = \mathbf{A}_{\bar{N}_-}(\bar{u}_-, U_-) (\mathbb{P}_- + \delta \mathbb{P}_-) \mathbf{A}_{\bar{N}_-}^T(\bar{v}_-, V_-) \\ &= \mathbf{A}_{\bar{N}_-}(\bar{u}_-, U_-) \mathbb{P}'_- \mathbf{A}_{\bar{N}_-}^T(\bar{v}_-, V_-) \end{aligned} \quad (5.59)$$

with  $\mathbb{P}'_+ = \mathbb{P}_+ + \delta \mathbb{P}_+$  and  $\mathbb{P}'_- = \mathbb{P}_- + \delta \mathbb{P}_-$  for some control point perturbations  $\delta \mathbb{P}_+$  and  $\delta \mathbb{P}_-$  that result from a parameter perturbation  $\delta \mathcal{P} / \mathcal{P} \ll 1$ . Also  $\bar{u}_+$ ,  $\bar{v}_+$ ,  $\bar{u}_-$ ,  $\bar{v}_-$  are the new support point parameterizations and  $U_+$ ,  $U_-$ ,  $V_+$  and  $V_-$  stem from non-preserved knot vectors. By assuming a Taylor expansion representation of the B-spline surface at all



$(u, v)$  coordinates, we write

$$\begin{aligned} f_+(\mathbb{B}_Q; \mathcal{P} + \delta\mathcal{P}) &= f(\mathbb{B}_Q; \mathcal{P}) + \frac{df}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2f}{d\mathcal{P}^2}\delta\mathcal{P}^2 + \frac{1}{6}\frac{d^3f}{d\mathcal{P}^3}\delta\mathcal{P}^3 + \dots \\ f_-(\mathbb{B}_Q; \mathcal{P} - \delta\mathcal{P}) &= f(\mathbb{B}_Q; \mathcal{P}) - \frac{df}{d\mathcal{P}}\delta\mathcal{P} + \frac{1}{2}\frac{d^2f}{d\mathcal{P}^2}\delta\mathcal{P}^2 - \frac{1}{6}\frac{d^3f}{d\mathcal{P}^3}\delta\mathcal{P}^3 + \dots \end{aligned}$$

and subtract to obtain

$$\frac{df}{d\mathcal{P}} \approx \frac{f_+ - f_-}{2\delta\mathcal{P}} - \mathcal{O}(\delta\mathcal{P}^2), \quad (5.60)$$

which is rewritten as

$$\frac{d\mathbb{B}_Q}{d\mathcal{P}} \approx \left( \frac{1}{2\delta\mathcal{P}} \right) \left[ \mathbf{A}_{\bar{N}_+}(\bar{u}_+, U_+) \mathbb{P}'_+ \mathbf{A}_{\bar{N}_+}^T(\bar{v}_+, V_+) - \mathbf{A}_{\bar{N}_-}(\bar{u}_-, U_-) \mathbb{P}'_- \mathbf{A}_{\bar{N}_-}^T(\bar{v}_-, V_-) \right] - \mathcal{O}(\delta\mathcal{P}^2). \quad (5.61)$$

When knots and support point parameterizations are preserved, the finite-difference derivative simplifies because

$$\begin{aligned} \mathbf{A}_{\bar{N}_+}(\bar{u}_+, U_+) &= \mathbf{A}_{\bar{N}_-}(\bar{u}_-, U_-) = \mathbf{A}_{\bar{N}}(\bar{u}, U) \\ \mathbf{A}_{\bar{N}_+}(\bar{v}_+, V_+) &= \mathbf{A}_{\bar{N}_-}(\bar{v}_-, V_-) = \mathbf{A}_{\bar{N}}(\bar{v}, V), \end{aligned}$$

with  $\bar{u}_+ = \bar{u}_- = \bar{u}$ ,  $\bar{v}_+ = \bar{v}_- = \bar{v}$ ,  $U_+ = U_- = U$  and  $V_+ = V_- = V$ . Also,  $\mathbb{P}'_+ = \mathbb{P}_+$  and  $\mathbb{P}'_- = \mathbb{P}_-$ , resulting in

$$\frac{d\mathbb{B}_Q}{d\mathcal{P}} \approx \left( \frac{1}{2\delta\mathcal{P}} \right) \mathbf{A}_{\bar{N}}(\bar{u}, U) (\mathbb{P}_+ - \mathbb{P}_-) \mathbf{A}_{\bar{N}}^T(\bar{v}, V) - \mathcal{O}(\delta\mathcal{P}^2), \quad (5.62)$$

which takes a familiar form (as found in the analytic case of Section 5.5) in the limit of  $\delta\mathcal{P} \rightarrow 0$ :

$$\begin{aligned} \frac{d\mathbb{B}_Q}{d\mathcal{P}} &= \mathbf{A}_{\bar{N}}(\bar{u}, U) \frac{d\mathbb{P}}{d\mathcal{P}} \mathbf{A}_{\bar{N}}^T(\bar{v}, V) \\ \Rightarrow \frac{d\mathbb{P}}{d\mathcal{P}} &= \mathbf{A}_{\bar{N}}^{-1}(\bar{u}, U) \frac{d\mathbb{B}_Q}{d\mathcal{P}} \mathbf{A}_{\bar{N}}^{-T}(\bar{v}, V). \end{aligned} \quad (5.63)$$

This ideal case is possible when knots and support point parameterizations are preserved. When this is not the case the finite-difference approximation remains in the form of (5.61), which clearly results in spurious design velocity components because the control point gradient is made inconsistent. The issue arises either by differencing B-spline surfaces that do

not reside in the same spline space (i.e., different  $U$  and/or  $V$  knot vectors) or by using different support point parameterizations (i.e., different  $\bar{u}$  and  $\bar{v}$ ). In order to see this, a derivation similar to that for B-spline curves is pursued. In this case we consider writing

$$\begin{aligned}\mathbf{A}_{\bar{N}_+}(\bar{u}_+, U_+) &= \mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_{u_+} \\ \mathbf{A}_{\bar{N}_-}(\bar{u}_-, U_-) &= \mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_{u_-} \\ \mathbf{A}_{\bar{N}_+}(\bar{v}_+, V_+) &= \mathbf{A}_{\bar{N}}(\bar{v}, V) + \boldsymbol{\varepsilon}_{v_+} \\ \mathbf{A}_{\bar{N}_-}(\bar{v}_-, V_-) &= \mathbf{A}_{\bar{N}}(\bar{v}, V) + \boldsymbol{\varepsilon}_{v_-}\end{aligned}$$

for deviations  $\boldsymbol{\varepsilon}_{u_+}$ ,  $\boldsymbol{\varepsilon}_{u_-}$ ,  $\boldsymbol{\varepsilon}_{v_+}$  and  $\boldsymbol{\varepsilon}_{v_-}$  in the basis function matrices. By assuming that  $\boldsymbol{\varepsilon}_{u_+} \approx \boldsymbol{\varepsilon}_{u_-} \equiv \boldsymbol{\varepsilon}_u$  and  $\boldsymbol{\varepsilon}_{v_+} \approx \boldsymbol{\varepsilon}_{v_-} \equiv \boldsymbol{\varepsilon}_v$ , we substitute into (5.61) and apply  $\delta\mathcal{P} \rightarrow 0$  to get

$$\begin{aligned}\frac{d\mathbb{B}_Q}{d\mathcal{P}} &\approx \left(\frac{1}{2\delta\mathcal{P}}\right) [(\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)\mathbb{P}'_+(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T) \\ &\quad - (\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)\mathbb{P}'_-(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T)] - \mathcal{O}(\delta\mathcal{P}^2) \\ &\approx \left(\frac{1}{2\delta\mathcal{P}}\right) (\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)(\mathbb{P}'_+ - \mathbb{P}'_-)(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T) - \mathcal{O}(\delta\mathcal{P}^2) \\ &= (\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)\frac{d\hat{\mathbb{P}}}{d\mathcal{P}}(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T).\end{aligned}\tag{5.64}$$

It is obvious in (5.64) that the solution  $\frac{d\hat{\mathbb{P}}}{d\mathcal{P}} \neq \frac{d\mathbb{P}}{d\mathcal{P}}$  for a true  $\frac{d\mathbb{P}}{d\mathcal{P}}$ . Thus we consider writing  $\frac{d\hat{\mathbb{P}}}{d\mathcal{P}} = \frac{d\mathbb{P}}{d\mathcal{P}} + \boldsymbol{\delta}\left(\frac{d\mathbb{P}}{d\mathcal{P}}\right)$  and adopting the notation

$$\partial_{\mathcal{P}}(\hat{\mathbb{P}}) = \frac{d\hat{\mathbb{P}}}{d\mathcal{P}}, \quad \partial_{\mathcal{P}}(\mathbb{P}) = \frac{d\mathbb{P}}{d\mathcal{P}}, \quad \boldsymbol{\delta}\partial_{\mathcal{P}}(\mathbb{P}) = \boldsymbol{\delta}\left(\frac{d\mathbb{P}}{d\mathcal{P}}\right)$$

in order to rewrite (5.64) as:

$$\begin{aligned}\frac{d\mathbb{B}_Q}{d\mathcal{P}} &= (\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)(\partial_{\mathcal{P}}(\mathbb{P}) + \boldsymbol{\delta}\partial_{\mathcal{P}}(\mathbb{P}))(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T) \\ &= (\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)\partial_{\mathcal{P}}\mathbb{P}(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T) + \\ &\quad (\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u)\boldsymbol{\delta}\partial_{\mathcal{P}}(\mathbb{P})(\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T)\end{aligned}\tag{5.65}$$

Now consistent matrix norms are taken and the triangle inequality is applied to yield:

$$\begin{aligned}
\left\| \frac{d\mathbb{B}_Q}{d\mathcal{P}} \right\| &\leq \|\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u\| \cdot \|\partial_{\mathcal{P}}\mathbb{P}\| \cdot \|\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T\| \\
&\quad + \|\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u\| \cdot \|\delta\partial_{\mathcal{P}}\mathbb{P}\| \cdot \|\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T\| \\
\|\delta\partial_{\mathcal{P}}\mathbb{P}\| &\leq \frac{\left\| \frac{d\mathbb{B}_Q}{d\mathcal{P}} \right\| - \|\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u\| \cdot \|\partial_{\mathcal{P}}\mathbb{P}\| \cdot \|\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T\|}{\|\mathbf{A}_{\bar{N}}(\bar{u}, U) + \boldsymbol{\varepsilon}_u\| \cdot \|\mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \boldsymbol{\varepsilon}_v^T\|}. \tag{5.66}
\end{aligned}$$

By recalling that  $\left\| \frac{d\mathbb{B}_Q}{d\mathcal{P}} \right\| \leq \|\mathbf{A}_{\bar{N}}(\bar{u}, U)\| \cdot \|\partial_{\mathcal{P}}\mathbb{P}\| \cdot \|\mathbf{A}_{\bar{N}}^T(\bar{v}, V)\|$ , we can simplify further and write

$$\begin{aligned}
\frac{\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\|}{\|\partial_{\mathcal{P}}(\mathbf{X}_P)\|} &\leq \frac{\|\mathbf{A}_{\bar{N}}(\bar{u}, U)\| \cdot \|\mathbf{A}_{\bar{N}}^T(\bar{v}, V)\|}{(\|\mathbf{A}_{\bar{N}}(\bar{u}, U)\| + \|\boldsymbol{\varepsilon}_u\|)(\|\mathbf{A}_{\bar{N}}^T(\bar{v}, V)\| + \|\boldsymbol{\varepsilon}_v\|)} - 1 \\
\Rightarrow \frac{\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\|}{\|\partial_{\mathcal{P}}(\mathbf{X}_P)\|} &\leq \frac{1}{\left(1 + \frac{\|\boldsymbol{\varepsilon}_u\|}{\|\mathbf{A}_{\bar{N}}(\bar{u}, U)\|}\right) \left(1 + \frac{\|\boldsymbol{\varepsilon}_v\|}{\|\mathbf{A}_{\bar{N}}^T(\bar{v}, V)\|}\right)} - 1. \tag{5.67}
\end{aligned}$$

As in the B-spline curve case, the control point derivative error is at most the same order of magnitude as the deviation in basis functions caused by reparameterization of knots or support points. Numerical experiments with the B-spline surfaces in the geometry of Figure 5-14 confirmed this outcome, where  $\|\boldsymbol{\varepsilon}_u\| = 1.98 \times 10^{-8}$  and  $\|\boldsymbol{\varepsilon}_v\| = 1.31 \times 10^{-13}$  created a control point sensitivity deviation of  $\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\| = -1.83 \times 10^{-9}$  after perturbing a support point by  $+\mathcal{P} = 1.0 \times 10^{-5}$  (with  $\|\mathbf{A}_{\bar{N}}(\bar{u}, U)\| = 1.13$ ,  $\|\mathbf{A}_{\bar{N}}^T(\bar{v}, U)\| = 1.62$  and  $\|\partial_{\mathcal{P}}(\mathbf{X}_P)\| = 0.105$  from finite-differences). It is also seen that as  $\|\boldsymbol{\varepsilon}_u\| \rightarrow 0$  and  $\|\boldsymbol{\varepsilon}_v\| \rightarrow 0$  the deviation becomes  $\|\delta\partial_{\mathcal{P}}(\mathbf{X}_P)\| \rightarrow 0$ .

Surface design velocity errors are also introduced by the control point derivative error, as in the B-spline curve case. The analysis of a linearized B-spline surface shows this, where it is written as

$$\left. \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right|_{\text{FD}} = \frac{\mathbf{S}_+ - \mathbf{S}_-}{2\delta\mathcal{P}}. \tag{5.68}$$

By expanding  $\mathbf{S}_+$  (an analogous derivation is possible with  $\mathbf{S}_-$ ), the perturbed surface is represented as

$$\mathbf{S}_+ = \mathbf{A}_N(u, U)\mathbb{P}'_+ \mathbf{A}_N^T(v, V) = \mathbf{A}_N(u, U) (\mathbb{P}_x + \delta\mathbb{P}_+) \mathbf{A}_N^T(v, V) \tag{5.69}$$

Substituting back into (5.68) yields

$$\begin{aligned} \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \Big|_{\text{FD}} &= \mathbf{A}_N(u, U) \left( \frac{\mathbb{P}_+ - \mathbb{P}_-}{2\delta\mathcal{P}} \right) \mathbf{A}_N^T(v, V) + \\ &\quad \mathbf{A}_N(u, U) \left( \frac{\delta\mathbb{P}_+ - \delta\mathbb{P}_-}{2\delta\mathcal{P}} \right) \mathbf{A}_N^T(v, V). \quad \delta\mathcal{P} \rightarrow 0 \end{aligned}$$

The design velocity is thereby rewritten as

$$\frac{\partial \mathbf{S}}{\partial \mathcal{P}} \Big|_{\text{FD}} = \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \Big|_{\text{True}} + \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \Big|_{\text{Error}}, \quad (5.70)$$

where

$$\frac{\partial \mathbf{S}}{\partial \mathcal{P}} \Big|_{\text{True}} \equiv \mathbf{A}_N(u, U) \left( \frac{\mathbb{P}_+ - \mathbb{P}_-}{2\delta\mathcal{P}} \right) \mathbf{A}_N^T(v, V) \quad (5.71)$$

$$\frac{\partial \mathbf{S}}{\partial \mathcal{P}} \Big|_{\text{Error}} \equiv \mathbf{A}_N(u, U) \left( \frac{\delta\mathbb{P}_+ - \delta\mathbb{P}_-}{2\delta\mathcal{P}} \right) \mathbf{A}_N^T(v, V). \quad (5.72)$$

It is possible that the spurious design velocity  $(\partial \mathbf{S} / \partial \mathcal{P})_{\text{Error}} \neq \mathbf{0}$  for all design velocity components, even those not associated with a support point perturbation. This is seen by considering the  $(\ )_+$  for perturbed support points  $\mathbb{B}_{Q_+} = \mathbb{B}_Q + \delta\mathbb{B}_Q$  and writing the new surface as:

$$\begin{aligned} \mathbb{B}_{Q_+} &= ( \mathbf{A}_{\bar{N}}(\bar{u}, U) + \varepsilon_u ) (\mathbb{P}_+ + \delta\mathbb{P}_+) ( \mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \varepsilon_v^T ) \\ &= ( \mathbf{A}_{\bar{N}}(\bar{u}, U) + \varepsilon_u ) \mathbb{P}_+ ( \mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \varepsilon_v^T ) + \\ &\quad ( \mathbf{A}_{\bar{N}}(\bar{u}, U) + \varepsilon_u ) \delta\mathbb{P}_+ ( \mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \varepsilon_v^T ). \end{aligned}$$

By recalling that  $\mathbb{B}_{Q_+} = \mathbf{A}_{\bar{N}}(\bar{u}, U) \mathbb{P} \mathbf{A}_{\bar{N}}^T(\bar{v}, V)$ , the deviation simplifies to

$$\begin{aligned} \delta\mathbb{P}_+ &= - ( \mathbf{A}_{\bar{N}}(\bar{u}, U) + \varepsilon_{u_+} )^{-1} [ \varepsilon_{u_+} \mathbb{P} \varepsilon_{v_+}^T + \mathbf{A}_{\bar{N}}(\bar{u}, U) \mathbb{P} \varepsilon_{v_+}^T + \\ &\quad \varepsilon_{u_+} \mathbb{P} \mathbf{A}_{\bar{N}}^T(\bar{v}, V) ] \left( \mathbf{A}_{\bar{N}}^T(\bar{v}, V) + \varepsilon_{v_+}^T \right)^{-1} \end{aligned} \quad (5.73)$$

Since the right hand side of (5.73) contains only non-zero terms, then  $\delta\mathbb{P}_+$  will always be non-zero for *every* component of control point design velocity (the same is true for  $\delta\mathbb{P}_-$ ). As both  $\varepsilon_{u_+} \rightarrow \mathbf{0}$  and  $\varepsilon_{v_+} \rightarrow \mathbf{0}$  then  $\delta\mathbb{P}_+ \rightarrow \mathbf{0}$  as well (and likewise as  $\varepsilon_{u_-} \rightarrow \mathbf{0}$  and  $\varepsilon_{v_-} \rightarrow \mathbf{0}$  then  $\delta\mathbb{P}_- \rightarrow \mathbf{0}$ ).

This result translates to a non-zero  $(\partial \mathbf{S} / \partial \mathcal{P})_{\text{Error}}$  for each component of the B-spline

surface design velocity. Numerical experiments of gradient error using the analytic surface geometry of Figure 5-14 match the results predicted using this error analysis (to within  $1.0 \times 10^{-13}$ ) for  $\delta\mathbb{P}_+$ . This implies that the spurious design velocity can be removed from the finite-difference computation *if* the baseline and perturbed surfaces are “reverse-engineered” to calculate the reparameterization error. As in the B-spline curve case, tests with B-spline surfaces of varying order (or other piecewise continuous surfaces written as a sum of weighted basis functions) were not conducted because they represent different values in the matrices  $\mathbf{A}_{\bar{N}}(\bar{u}, U)$ ,  $\mathbf{A}_{\bar{N}}(\bar{v}, V)$ ,  $\mathbf{A}_N(u, U)$  and  $\mathbf{A}_N(v, V)$ . The error analysis here is general to any cases where finite-differencing is done between such piecewise continuous surfaces that do not share the same parameter domain or knot vector.

The magnitude of design velocity error along  $\mathbf{S}$  is expected to be

$$\left\| \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right\|_{\text{Error}} \approx \mathcal{O} \left( \frac{\|\delta\mathbb{P}_+ - \delta\mathbb{P}_-\|}{|\delta\mathcal{P}|} \right). \quad (5.74)$$

In the case of the surface in Figure 5-14, the spurious control point design velocities were of  $\mathcal{O}(1.0 \times 10^{-8})$  for a support point perturbation of  $\delta\mathcal{P} = 1.0 \times 10^{-5}$ , thus making

$$\left\| \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right\|_{\text{Error}} \approx \mathcal{O} \left( \frac{1.0 \times 10^{-8}}{1.0 \times 10^{-5}} \right) = 1.0 \times 10^{-3},$$

which is consistent with the spurious design velocity magnitudes seen in Figure 5-14.

### 5.8.3 Potential for Error Correction

The previous sections show that finite-difference sensitivities of support points are inconsistent when a geometry kernel computes perturbed surfaces with reparameterized knot vectors and/or support points. Design velocity error is quantified as well for the B-spline curve and surface cases *if* the curve or surface is analytically defined. This information permits correcting the finite-difference computation by writing

$$\begin{aligned} \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{True}} &= \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{FD}} - \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{Error}} \\ \left. \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right|_{\text{True}} &= \left. \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right|_{\text{FD}} - \left. \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right|_{\text{Error}}. \end{aligned}$$

However, since correctly calculating the sensitivity error requires “reverse-engineering” the analytic definition of the curve or surface, the finite-difference method is unnecessary because sufficient information then exists for the analytic sensitivity method. Having found  $\mathbf{A}_{\bar{N}}(\bar{t}, U)$  guarantees this for B-spline curves and having both  $\mathbf{A}_{\bar{N}}(\bar{u}, U)$  and  $\mathbf{A}_{\bar{N}}(\bar{v}, V)$  guarantees this for B-spline surfaces. The same goes if a mapping approach is considered instead of the linear-correcting approach above:

$$\begin{aligned} \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{True}} &= \mathbf{M} \left. \frac{\partial \mathbf{Y}}{\partial \mathcal{P}} \right|_{\text{FD}} \\ \left. \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right|_{\text{True}} &= \mathbb{M} \left. \frac{\partial \mathbf{S}}{\partial \mathcal{P}} \right|_{\text{FD}}, \end{aligned}$$

with both mappings  $\mathbf{M}$  and  $\mathbb{M}$  requiring knowledge about the analytic surface definition found in  $(\partial \mathbf{Y} / \partial \mathcal{P})_{\text{True}}$  and  $(\partial \mathbf{S} / \partial \mathcal{P})_{\text{True}}$ , respectively.

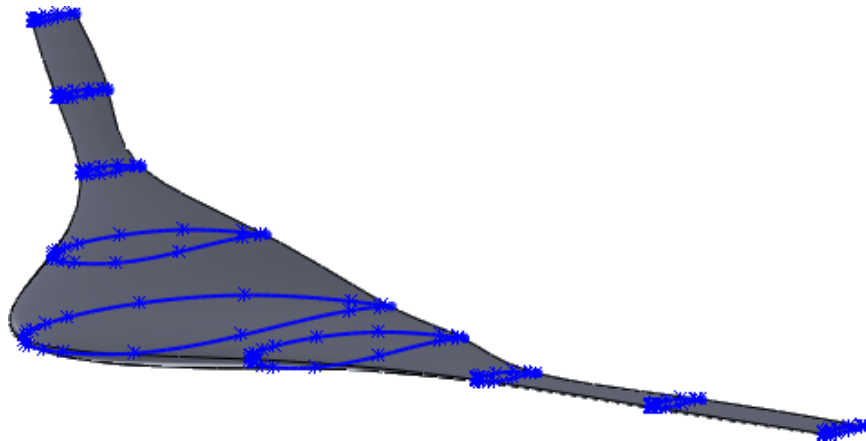


Figure 5-17: The impact of control point perturbations on knot vector reparameterization was done using this SolidWorks model geometry of a Blended-Wing-Body aircraft.

In contrast to the results seen from perturbing support points, numerical test results summarized in Tables 5.6 and 5.7 show that perturbing *control points* instead of support points does not change the knot vector  $U$  or support point parameterization  $\bar{t}$  on a cross-section spline (at least when using the SolidWorks geometry kernel). This example uses a step-size of  $1.0 \times 10^{-5}$  in the  $y$ -component of either a support or control point on a cross-section B-spline curve of the CAD model in Figure 5-17. The research literature cites various shape design examples where finite-differencing of control points is used instead of support points. Although using control points as optimization parameters is less intuitive for many designers, this approach appears to circumvent the issues discovered thus far when

finite-differencing support points. If manipulation of control points is possible outside of the CAD system<sup>9</sup>, then this is a viable alternative to the support point design variables for consistent finite-differencing.

<b>B-Spline Curve Control Point Perturbation</b>			
	<i>Baseline</i>	<i>+h Perturbation</i>	<i>-h Perturbation</i>
$\bar{t}$	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.00664434497888606	0.00664434497888606	0.00664434497888606
	0.0159796083917612	0.0159796083917612	0.0159796083917612
	0.0291008230580382	0.0291008230580382	0.0291008230580382
	0.0676990990527782	0.0676990990527782	0.0676990990527782
	0.122328812951516	0.122328812951516	0.122328812951516
	0.315346062269475	0.315346062269475	0.315346062269475
	0.599716141077589	0.599716141077589	0.599716141077589
	0.886354975172199	0.886354975172199	0.886354975172199
	0.970646767300137	0.970646767300137	0.970646767300137
	1.0000000000000000	1.0000000000000000	1.0000000000000000
	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.0000000000000000	0.0000000000000000	0.0000000000000000
0.0000000000000000	0.0000000000000000	0.0000000000000000	
0.00664434497888606	0.00664434497888606	0.00664434497888606	
0.0159796083917612	0.0159796083917612	0.0159796083917612	
0.0291008230580382	0.0291008230580382	0.0291008230580382	
0.0676990990527782	0.0676990990527782	0.0676990990527782	
0.122328812951516	0.122328812951516	0.122328812951516	
0.315346062269475	0.315346062269475	0.315346062269475	
0.599716141077589	0.599716141077589	0.599716141077589	
0.886354975172199	0.886354975172199	0.886354975172199	
0.970646767300137	0.970646767300137	0.970646767300137	
1.0000000000000000	1.0000000000000000	1.0000000000000000	
1.0000000000000000	1.0000000000000000	1.0000000000000000	
1.0000000000000000	1.0000000000000000	1.0000000000000000	
1.0000000000000000	1.0000000000000000	1.0000000000000000	

Table 5.6: Perturbation of a B-spline curve control point preserves its knot vector and support point parameterization. Perturbation size is  $h = 10^{-5}$  in the  $y$ -component of a control point in a cross-section B-spline curve of Figure 5-17.

<sup>9</sup>As of this writing the CAPRI API did not support this with SolidWorks.

<b>B-Spline Curve Support Point Perturbation</b>			
	<i>Baseline</i>	<i>+h Perturbation</i>	<i>-h Perturbation</i>
$\bar{t}$	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.00664434497888606	0.00664434489760022	0.00664434506018314
	0.0159796083917612	0.0159796081962693	0.0159796085872802
	0.0291008230580382	0.0291008227020235	0.0291008234141022
	0.0676990990527782	0.0676990982245583	0.0676990998811125
	0.122328812951516	0.122328811454965	0.122328814448274
	0.315346062269475	0.315346067654938	0.31534605688325
	0.599716141077589	0.599716145974597	0.599716136179904
	0.886354975172199	0.886354976562514	0.886354973781692
	0.970646767300137	0.97064676765924	0.970646766940985
1.0000000000000000	1.0000000000000000	1.0000000000000000	
$U$	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.0000000000000000	0.0000000000000000	0.0000000000000000
	0.00664434497888606	0.00664434489760022	0.00664434506018314
	0.0159796083917612	0.0159796081962693	0.0159796085872802
	0.0291008230580382	0.0291008227020235	0.0291008234141022
	0.0676990990527782	0.0676990982245583	0.0676990998811125
	0.122328812951516	0.122328811454965	0.122328814448274
	0.315346062269475	0.315346067654938	0.31534605688325
	0.599716141077589	0.599716145974597	0.599716136179904
	0.886354975172199	0.886354976562514	0.886354973781692
	0.970646767300137	0.97064676765924	0.970646766940985
	1.0000000000000000	1.0000000000000000	1.0000000000000000
	1.0000000000000000	1.0000000000000000	1.0000000000000000
1.0000000000000000	1.0000000000000000	1.0000000000000000	
1.0000000000000000	1.0000000000000000	1.0000000000000000	

Table 5.7: Perturbation of a B-spline curve support point leads to knot vector and support parameterization changes. Perturbation size is  $h = 10^{-5}$  in the  $y$ -component of a support point in a cross-section B-spline curve of Figure 5-17. Underlined digits reflect deviations from the baseline case.



## Chapter 6

# Geometry Management

## Demonstrations

Geometry management with CAD models places geometry at the center of a design framework and enables it to support automated multidisciplinary design optimization. This chapter highlights a few examples wherein CAD models fulfill this role using the concepts presented in prior chapters. First a discussion is given about important details behind CAD-based design frameworks. Various design examples are then presented, beginning with a 3D mechanical part design and both 2D/3D wing design. Each is conducted using gradient-based optimization. A multidisciplinary design space study with aero/structural analysis is also presented to illustrate the usefulness of CAD models in such settings.

### 6.1 Implementing CAD Model Geometry in Design Frameworks

CAD-based design frameworks differ from traditional approaches by pre-processing, constructing and mapping a central geometry representation into the various geometry models required by analysis tools. Conventional methodologies only require constructing model geometry for specific analysis tools. Each model solely exists for its specific analysis and is independent of other geometry models used with different analysis tools. Geometry processing and mapping is later conducted, though often with great difficulty, when an attempt is made to map the various models into a single, consistent representation.

A high-level overview is given here about some important details in a CAD-based design framework. Initially a review of available CAD systems and interfacing software is given. This is followed by various details pertaining to geometry processing, namely creating an inventory of geometry data, associating parameters to model topology, extracting consistent sub-model geometry, and issues with scaling design problems.

### 6.1.1 Overview of CAD Systems

Many commercial CAD systems are available and extensively used throughout multiple industries. Proprietary systems such as SolidWorks, ProENGINEER, Catia and Unigraphics have been developed over many years around proprietary geometry kernels, such as Parasolid. Open-source geometry kernels such as OpenCascade are also available. With the exception of OpenCascade, CAD systems typically require manual interaction with a GUI to generate model geometry. Each also supports an API to query, manipulate or generate the model geometry.

Numerous API routines are defined within each CAD system. These facilitate programming geometry processing and automated *primitive* and *feature* creation. For example, the SolidWorks system used throughout this dissertation contains an API that allows automatic generation of all *primitives* and *features* in the model geometry examples of Chapter 2. Information about the resulting BRep and master-model are also extracted via the API interface. For example, there exist API calls to query B-spline surfaces for  $U$  and  $V$  knot vectors, control points and surface coordinates for a given  $(u, v)$  location. Querying other surfaces, such as planes, cylinders, etc., also returns their parameterization information. The master-model parameter list is available through the API, thereby allowing direct manipulation of model geometry through its embedded driving parameters. Model regeneration is executed after varying driving parameters through the API as well.

In order to implement a CAD model within an existing design framework, an interface between the two software systems must exist. The CAD system API permits such an interface. One example of this is the *Computational Analysis PRogramming Interface* (CAPRI) developed by Haimes [27]. The CAPRI API tools provide calling routines for non-CAD software to interface with the CAD system API. These tools include geometry manipulation, regeneration, tessellation, querying and saving model instances. A design framework loads the CAD system along with “back-end” dynamic libraries for CAPRI to

communicate with the geometry kernel. This interface can occur on the same workstation or via a web services client/server setup, wherein a server communicates with the design framework and the geometry kernel remotely. Once the CAD system is loaded, the design framework can manage geometry at any time through the CAPRI API in an automated fashion. With this functionality in place, geometry management of CAD models becomes a reality for automated design frameworks.

Geometry management is evolving from a typical conventional setup seen in Figure 6-1 to the geometry-centered approach shown in Figure 6-2 and discussed in Chapter 1. The scenario in Figure 6-2 is employed throughout the demonstration frameworks used in Section 6.2, wherein a CAD model is the central source of geometry information for analysis. A conventional framework similar to Figure 6-1 has independent geometry representations for each analysis tool. The framework in Figure 6-2 manipulates, queries and extracts sub-models automatically from a central-geometry source for passage to model analysis management as needed.

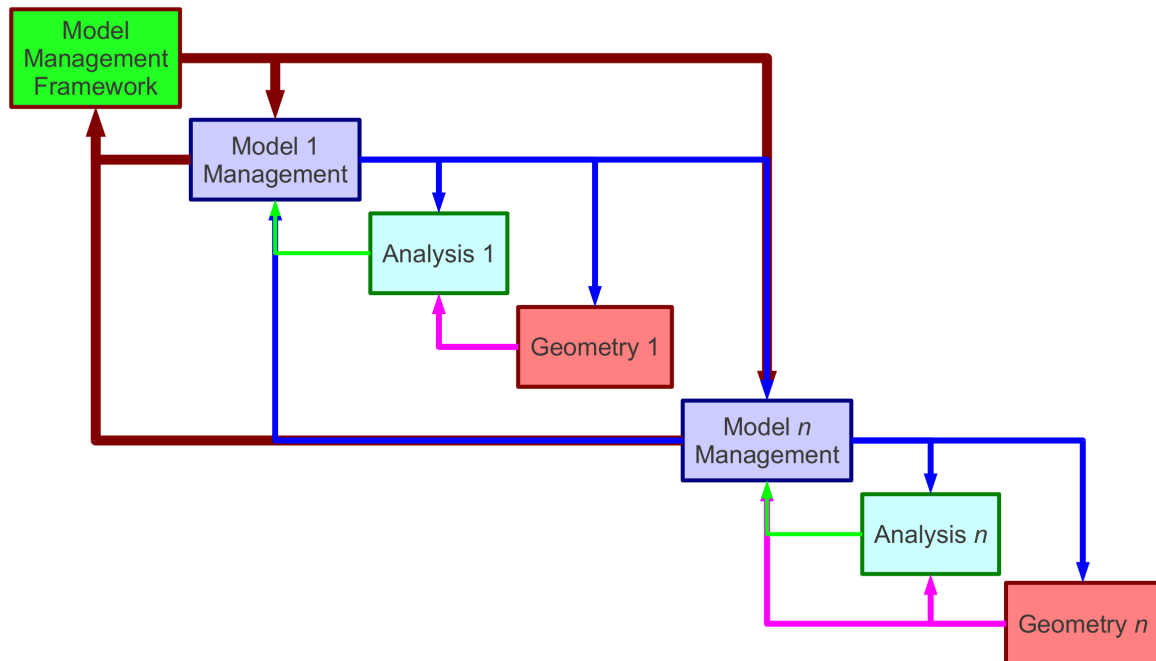


Figure 6-1: A design framework flow-chart where the geometry management paradigm has a secondary role.

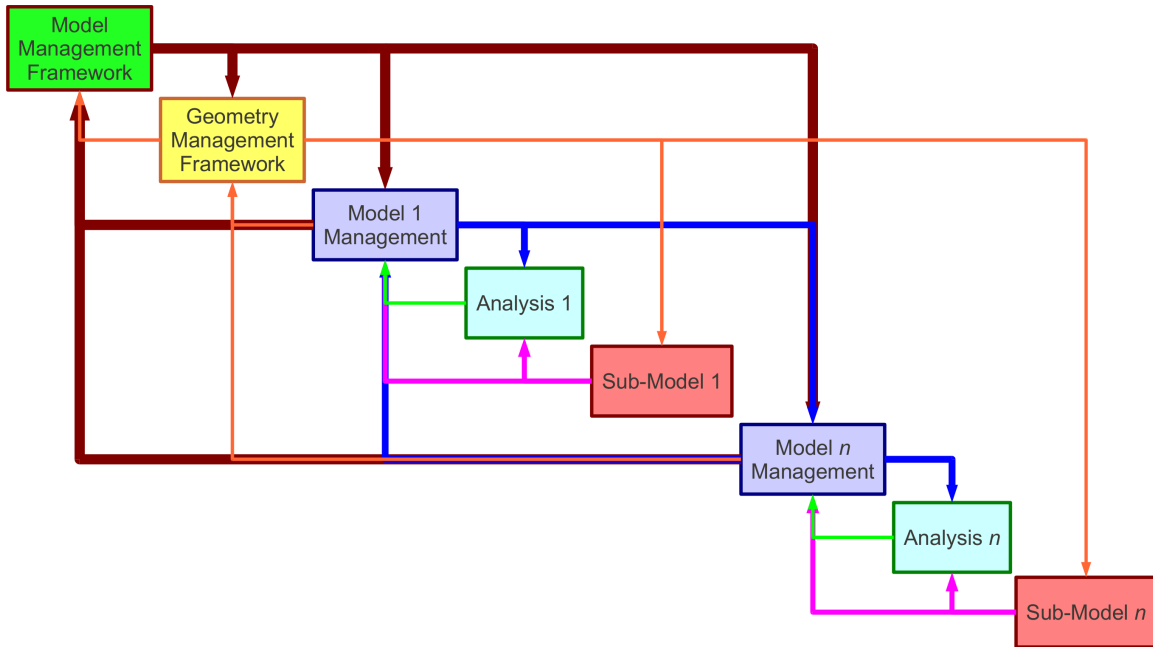


Figure 6-2: A design framework flow-chart where the geometry management paradigm has a primary role.

### 6.1.2 Inventory of Geometry Data

Using the construction principles explained in Chapter 2 (with additional detail covered in Appendix A), a 3D model geometry is generated within a CAD system to be the central geometry representation of the design framework. This model defines the high-fidelity, or fully-realizable, description of the design, where no other representation of the same design can contain *more* levels of geometry information. The central geometry representation must resolve each level of requisite geometry fidelity, either by explicit inclusion from the start or through added *features* as design matures<sup>1</sup>.

Once a 3D model is generated, the geometry- and design-specific information contained in its master-model must be catalogued. Access to the master-model simplifies this important step (in this work the CAPRI API provided this access). Without such information, it is unclear how various components are described topologically because the CAD system internally conducts the surface trimming, union/subtraction and other geometry operations in a non-transparent manner to the user. Such information is vital to create algorithms that extract lower-fidelity geometry information from the complete high-fidelity model in

<sup>1</sup>In the case of aircraft design, it is convenient to define the outer mold line first and adapt an “outside-in” approach to adding *features* pertaining to internal subsystems. Placeholder (skeletal) geometry [18] handles are required for geometry information that is not known a priori, but relevant to later analysis.

an automated fashion.

Parameter information must also be catalogued to setup design problems properly for a given CAD model. Applicable bounds can be determined through a design motion study or design of experiments. Design variables can also include the suppression status of a feature-tree branch, thus suppression information must be obtained from the parametric model. Explicitly naming model components, *features* and parameters also encourages a consistent reference system that sub-models refer to. Furthermore, the model regeneration robustness is also better understood by obtaining the parent-child relationships across the master-model. A greater number of “children” branching from any part of the feature-tree may hint to design motion limitations and a potential for lower regeneration robustness across the feasible design space.

### 6.1.3 Parameter Associativity to Model Topology

The taxonomy of parameterization levels discussed in Chapter 2 also assists in categorizing which parameters drive the resulting BRep topology for a model. Although not strictly important from the perspective of model generation, the associativity of driving parameters to the resulting model surfaces is important for conducting geometry sensitivities. Knowing which faces are driven by certain parameters simplifies the sensitivity algorithm. This avoids wasted computational time in computing the sensitivity of surfaces to unassociated parameters. Constructing this associativity table is non-trivial, however.

Three possible approaches may be taken to create the parameter associativity table. Further study is needed to identify which is the best approach, or if another is preferred altogether. First, the associativity table may be created manually as the model geometry is constructed. The addition of extra *features* in the model will expand the associativity table. By doing this, a sensitivity algorithm can be custom-tailored to the final model and be more computationally efficient. Second, if a CAD model of unknown construction is used, then it may be feasible to infer an associativity table by suppressing all *features* in the model and perturbing each parameter to see which faces show a non-zero design velocity. As each additional *feature* is un-suppressed, the parameter perturbations may permit inference of new associativity information for the new faces. Perturbation sizes will need to be less-than any critical value that induces topology changes in the model. Knowledge of such limiting perturbation sizes can be obtained via a design motion study on the model beforehand.

Third, if equations defining model geometry surfaces are known and the master-model parameters directly link to those equations, then tracing the parent/child dependencies in the model feature-tree can lead to an associativity table for models.

#### 6.1.4 Extracting Consistent Geometry Sub-models

Geometry information extracted from a CAD model becomes a geometry “sub-model.” These are inherently lower-fidelity geometry representations of the full high-fidelity model. The extraction of sub-models is a crucial step in developing a CAD-based design framework because many analysis tools only require lower-fidelity geometry information. The process for creating these sub-models must be automated, robust and able to meet all the analysis-specific geometry criteria. When analysis models are coupled, their geometry sub-models must be consistent in order to be coupled properly as well. Inconsistent sub-models are difficult to use when analysis solutions need to be passed from one discipline to another.

Consistency between sub-models can occur naturally when each originates from the same source geometry. Individually the sub-models are independent. Yet they are coupled by sharing the parent geometry *features* from which lower-fidelity geometry is extracted. For example, separate edges enclosing a face may appear in distinct sub-models, yet those edges are coupled by sharing a face in the CAD model BRep. In this manner, multi-disciplinary coupling is obtained by the connection of disciplinary sub-models to parent geometry. Discipline-specific analysis results can thus be mapped in a consistent fashion from one geometry sub-model to another by including the source geometry in the mapping procedure.

#### 6.1.5 Impact of Problem Scaling on Geometry Management

There is an expected increase in the computational cost associated with geometry processing as the number of parameters in a CAD model increases. If the larger parameter set is also dimensioned, then the geometry constraint set increases for the CAD model as well. A scaling study was conducted using the SolidWorks CAD system with a single-loft representation of a Blended-Wing-Body (see Figure 6-23 in Section 6.2.5). Execution time for automated model construction, regeneration and differentiation were quantified as the number of parameters in the model increased.

The timing results for automatic model construction are shown in Figure 6-3 for indi-

vidual models containing a loft *feature* with the indicated number of cross-section sketches. Each sketch contained the same B-spline curve airfoil that is scaled according to its span-wise position. When dimensions are needed, the CAD system “Fully Defined Sketch” option is used to automatically dimension each support point with respect to the sketch origin for a total of 46 dimensions per sketch. The added dimensions create a larger geometry constraint set to solve as more sketches are added. Overall, as the model size increases the geometry kernel appears to require greater computational resources to process the model. It is also observed that the CAD system memory management may contribute added overhead if a series of larger models are constructed consecutively. Performance degrades if models are not released before larger models follow. This can be true whether in model construction, model loading or regeneration and is likely CAD system dependent. Different versions of the same CAD system may also display variability in performance. Results shown here pertain to the SolidWorks 2011 version. A scaling study like this should be conducted with a CAD-based design framework in order to identify the tractability of design optimization problems under consideration.

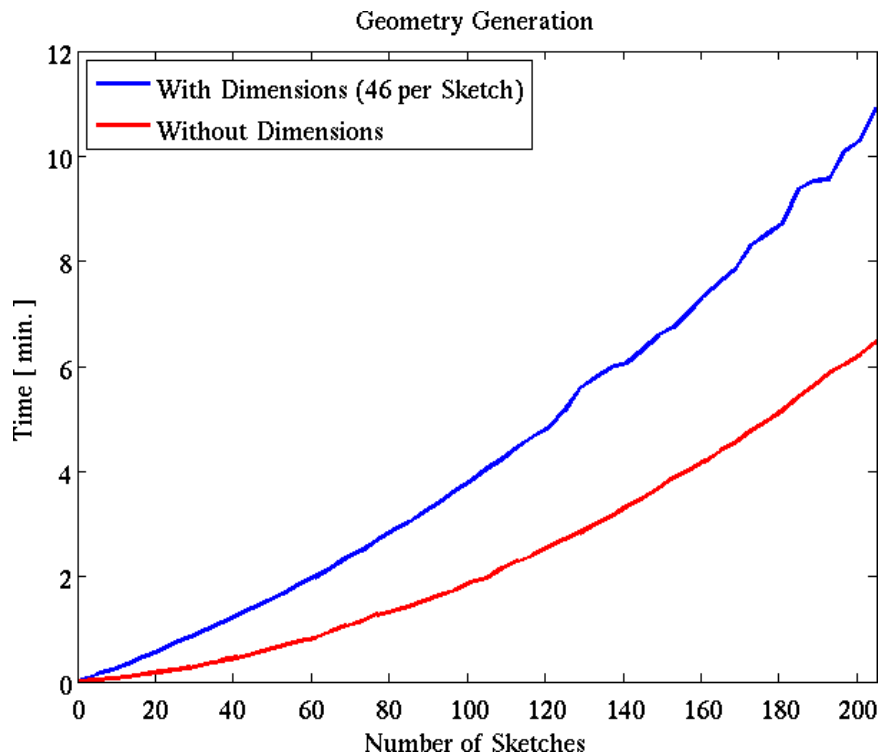
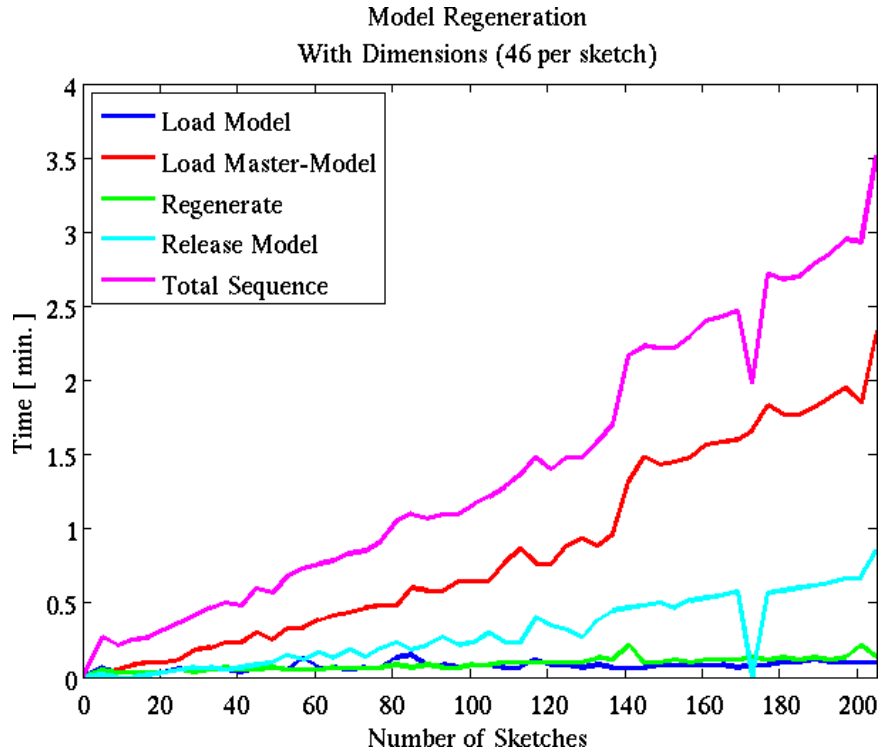


Figure 6-3: Timing tests for automated model construction of a loft *feature* with increasing number of cross-section definitions.

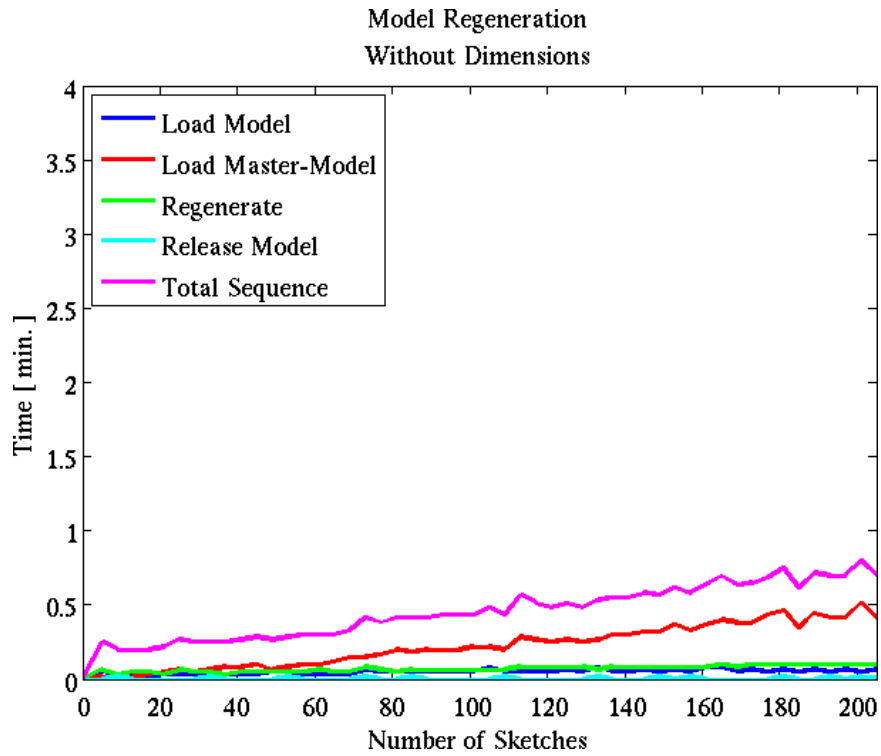
Different timing results are shown in Figure 6-4 for model regeneration. This process is decomposed into the following sequence of steps: loading the model, loading the master-model, regenerating the model after perturbing a single parameter and finally releasing the model. Whether with or without dimensions, the model regeneration time is fairly constant. However the master-model processing appears to drive the overall execution time as the number of parameters increases. This effect is exacerbated when dimensions are used in the sketches.

The combined computational expense of processing the model geometry and calculating B-spline surface sensitivities (for a tessellation of about 200000 elements) are shown in Figure 6-5. In this case the model geometry used in the prior timing tests is reused and a geometry gradient is calculated with respect to a single support point on each sketch. Clearly the analytic gradient benefits from doing only one tessellation and can compute all the necessary gradients in less time. As the model size increases, the finite-difference method suffers from potential overhead costs within the CAD system since a baseline model and two perturbed models may be loaded at a time for a central difference solution. This can hinder processing of the master-model significantly, but is less of a concern in the analytic case because only one model is loaded at a time.





(a)



(b)

Figure 6-4: A computational expense study for model regeneration as the number of model sketches increases (a) with dimensions and (b) without dimensions.

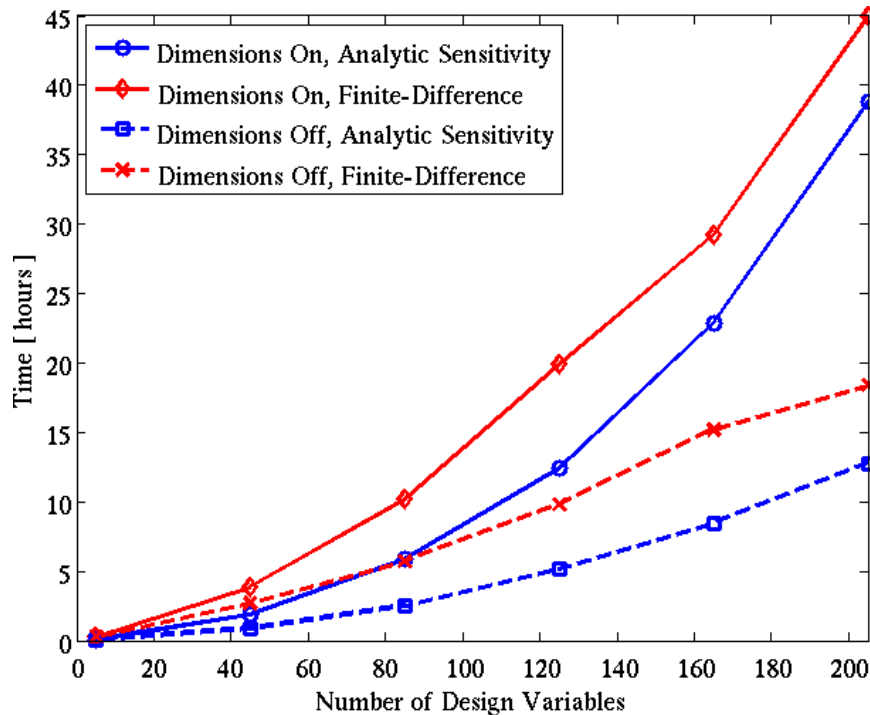


Figure 6-5: Timing tests for the geometry gradients of a loft *feature* with increasing number of cross-section definitions. Only one design variable per sketch is considered for differentiation.

## 6.2 Design Framework Demonstrations

Example design frameworks are presented here with increasing complexity to demonstrate the tools developed in this thesis and various aspects of a CAD-based design framework. First, a single-discipline design problem of a mechanical part is presented with gradient-based optimization. This is followed by 2D/3D single-discipline design problems of airfoils and wings also using gradient-based optimization. Lastly, a multidisciplinary parameter study is shown that incorporates both aerodynamic and structural analysis.

### 6.2.1 3D Single-Discipline Mechanical Design Problem

The pulley in Figure 6-6(a) is designed as a revolve *feature* using the sketch *primitives* parameterized in Figure 6-6(b). Ten linear dimensions and various geometry constraints drive the sketch end-points. The symmetric pulley is explicitly modeled by mirroring the sketch *primitives* about the horizontal axis as well. The design motion of this part, though, is limited to a bounded design space wherein sketch *primitives* do not cross (a constraint imposed by the CAD system). Otherwise, the pulley will not regenerate in those unfeasible

portions of the design space.

A constrained design problem is posed to find the dimension values that yield a reduction in the baseline pulley volume  $V_0$  by a factor  $\beta = 0.8$ :

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{J} = (\beta V_0 - V(\mathbf{r}; \mathbf{x}))^2 \\ \text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}} \end{aligned} \tag{6.1}$$

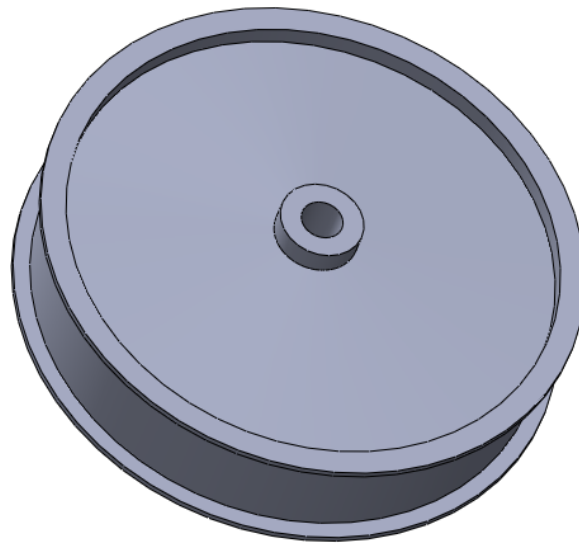
where the sketch dimensions are the design variables  $\mathbf{x} = \{\mathcal{P}_i\} = \{d_1, \dots, d_{10}\}$  constrained within side bounds  $\mathbf{x}_{\text{LB}} = 0.5\mathbf{x}_0$  and  $\mathbf{x}_{\text{UB}} = 5\mathbf{x}_0$  for baseline values  $\mathbf{x}_0$ . The  $\mathbf{r}$  refers to the sketch *primitive* parameterization.  $V(\mathbf{r}; \mathbf{x})$  is the pulley volume returned by the geometry kernel after regenerating a given design point  $\mathbf{x}$ . This volume computation was “reverse-engineered” by treating the model as a surface-of-revolution and integrating the sketch *primitives*. Validation of the analytic volume calculation was done by changing dimension values in  $\mathbf{x}$  and comparing the results with the geometry kernel answer. Excellent agreement was seen between the two results.

Having the analytic volume model also enabled an analytic computation of  $\partial V / \partial \mathbf{r}$ . After determining the geometry constraint set for the sketch, the sketch derivative is found as  $\partial \mathbf{r} / \partial \mathbf{x}$  and the objective function sensitivity becomes

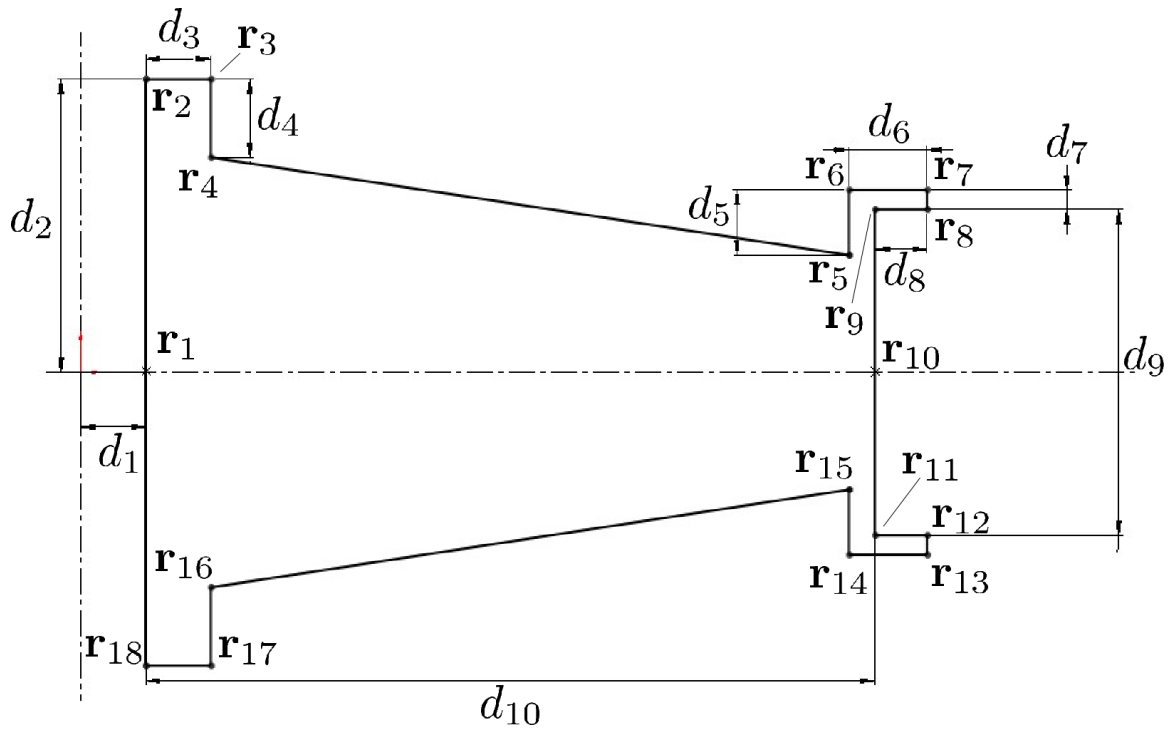
$$\frac{\partial \mathcal{J}}{\partial \mathbf{x}} = \frac{\partial V}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{x}},$$

where finite-differences validated both components of this analytic gradient.

A wrapper function provided the `fmincon` optimizer in Matlab with sensitivity information, while a separate geometry management module provided access to the pulley CAD model for regeneration and volume queries. Optimization runs were conducted using the analytic geometry gradients or finite-differencing of the CAD model (with a step-size of  $1.0 \times 10^{-2}$  for dimensions of  $\mathcal{O}(100)$ ). Figure 6-7(a) shows the objective function history comparison and Figure 6-7(b) shows both the initial and final pulley cross-section geometry. As shown in Table 6.1, both gradient methods reached the same local minimum. However, the analytic gradient provided less overall computational expense than the finite-difference gradient, as expected.

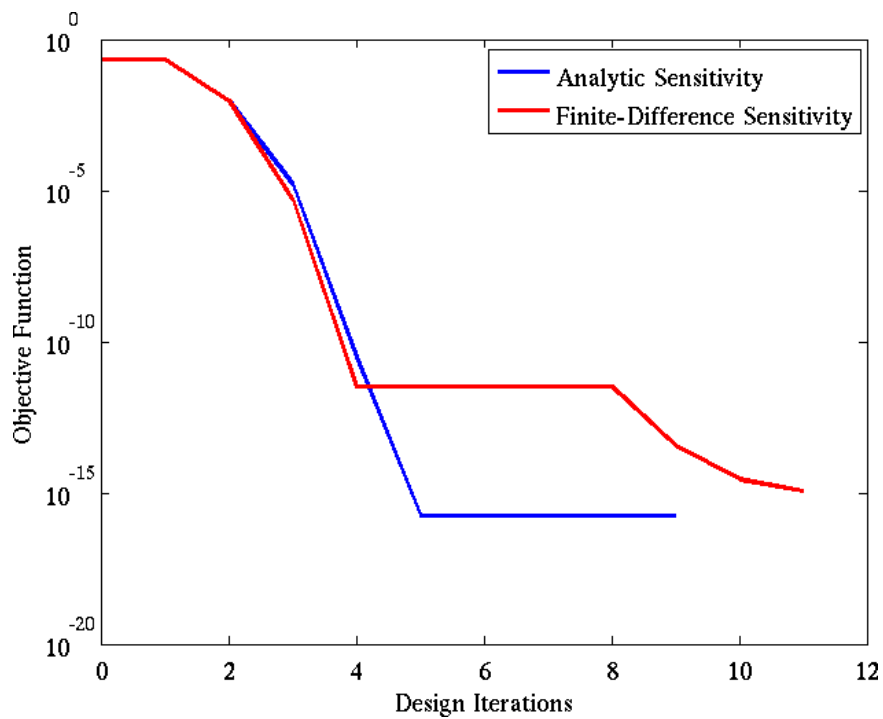


(a)

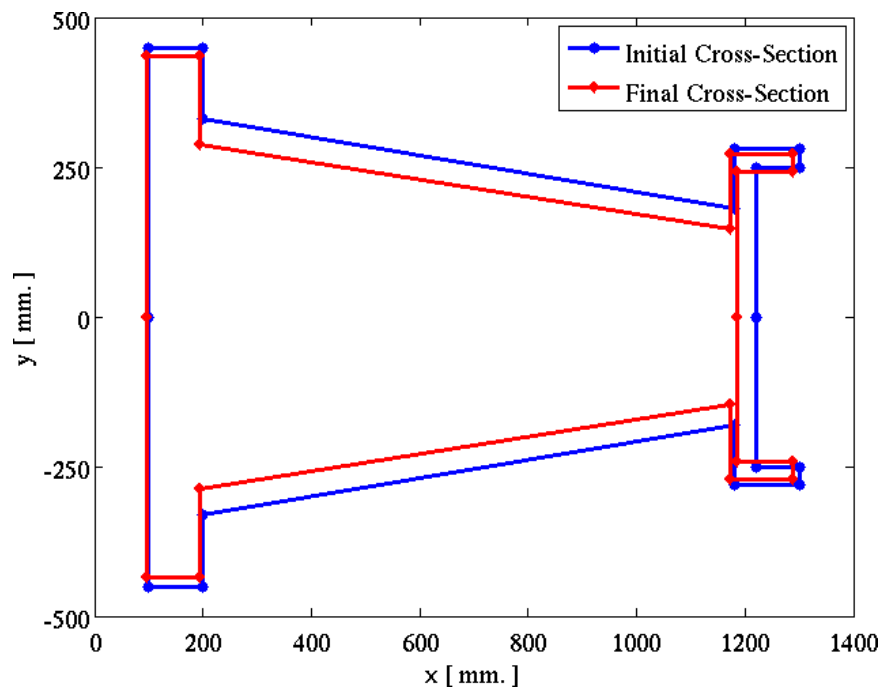


(b)

Figure 6-6: (a) A pulley generated as a revolve *feature* using the sketch *primitives* parameterized in (b).



(a)



(b)

Figure 6-7: (a) Objective function history as influenced by analytic and finite-difference geometry gradients for the pulley design problem. (b) A comparison of the initial and final cross-section geometry for the pulley after optimization.

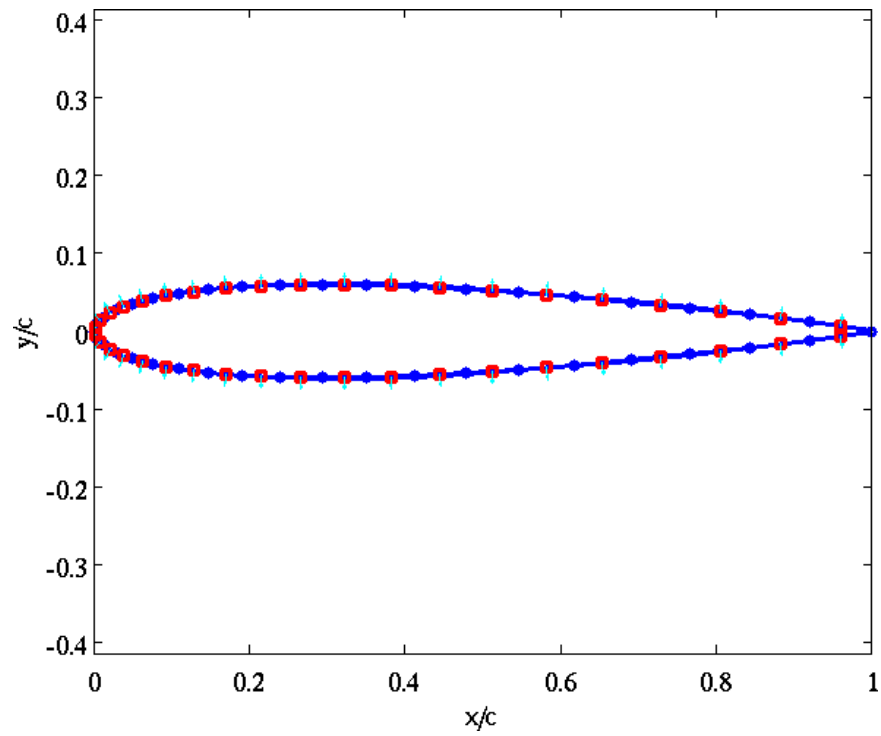
	<b>Analytic Sensitivity</b>	<b>Finite-Difference Sensitivity</b>
<i>Starting Volume [m<sup>3</sup>]</i>	2.30769	2.30769
<i>Target Volume [m<sup>3</sup>]</i>	1.8461520	1.8461520
<i>Final Volume [m<sup>3</sup>]</i>	1.84615 <u>1986</u>	1.84615 <u>1965</u>
<i>Final <math>\mathcal{J}</math> Value [m<sup>3</sup>]</i>	$1.81976 \times 10^{-16}$	$1.16494 \times 10^{-15}$
<b>Avg. Sensitivity Computation Time [min:sec]</b>	0:37	7:40

Table 6.1: A summary of results for the pulley design problem. The reported computation time is an average over all geometry gradient computations in a design run. Underlined digits correspond to deviations from the target solution.

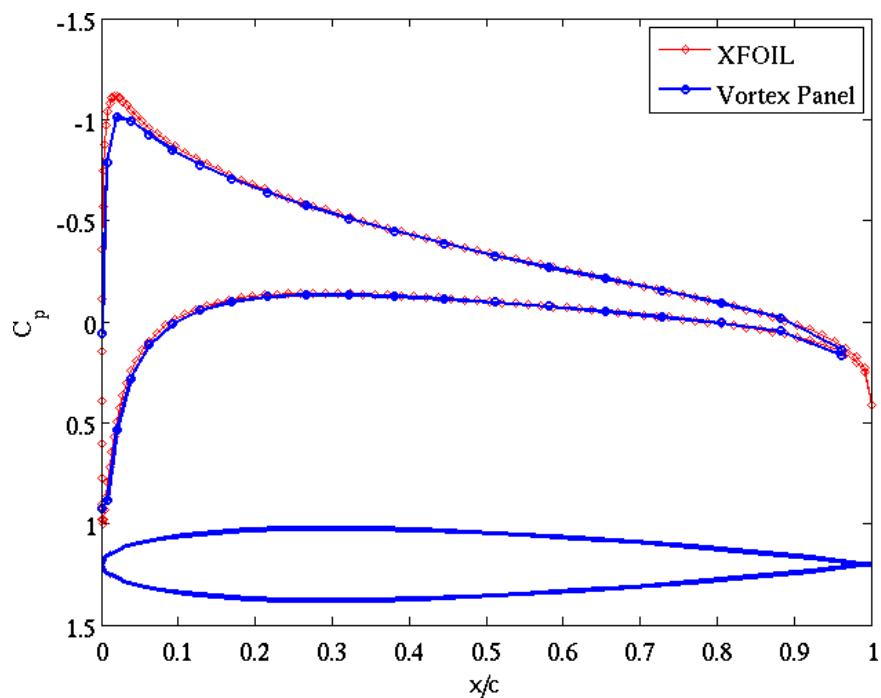
### 6.2.2 2D Single-Discipline Inverse-Design Problems

A 2D linear vortex panel method (with Karman-Tsien compressibility correction) was developed in order to have access to the source code and manually differentiate with respect to flow parameters and discretization. This allows for a complete derivative to be computed for use in a gradient-based optimization design framework. An input geometry representation consists of airfoil coordinates that serve as the panel definition. For validation, the  $C_p$  distribution results for an inviscid, incompressible 2D flow over a NACA 0012 airfoil is compared to XFOIL [20] results for various discretization choices. Two results shown in Figures 6-8 and 6-9 demonstrate excellent agreement between the two codes for an  $\alpha = 3.0$  case.

Using this newly developed 2D panel code, an inverse-design problem is posed wherein the cross-section from an existing 3D CAD-generated loft is extracted, perturbed and subsequently driven to its target baseline shape via gradient-based optimization. Two simple 3D lofts of a NACA 0012 wing and RAE2822 wing were created in the SolidWorks CAD system. The lofts consist of five defining cross-sections made of two cubic B-splines each, as shown in Figure 6-10. Both the top and bottom B-splines utilize 10 support points in approximating the NACA 0012 shape, whereas 15 support points are used for the RAE 2822 shape. Each wing has an aspect ratio  $\mathcal{R} = 1.0$ , planform area  $S = 1.0$  and chord  $c = 0.5$ . For a 3D design problem this planform shape results in large 3D-effects due to the low  $\mathcal{R}$  value, however the 2D analysis does not take this into account. When applied

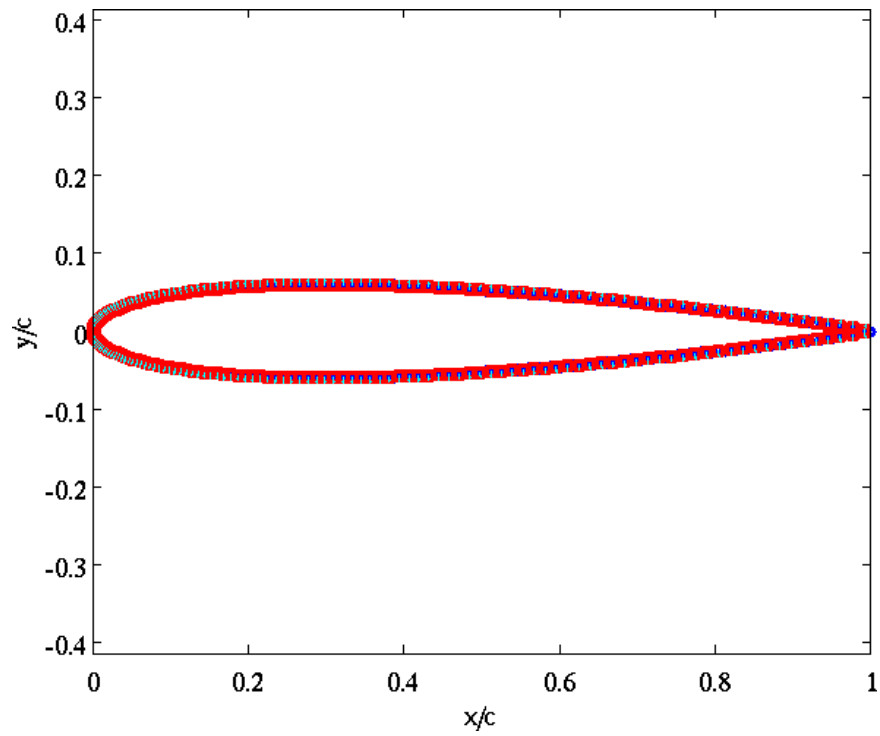


(a)

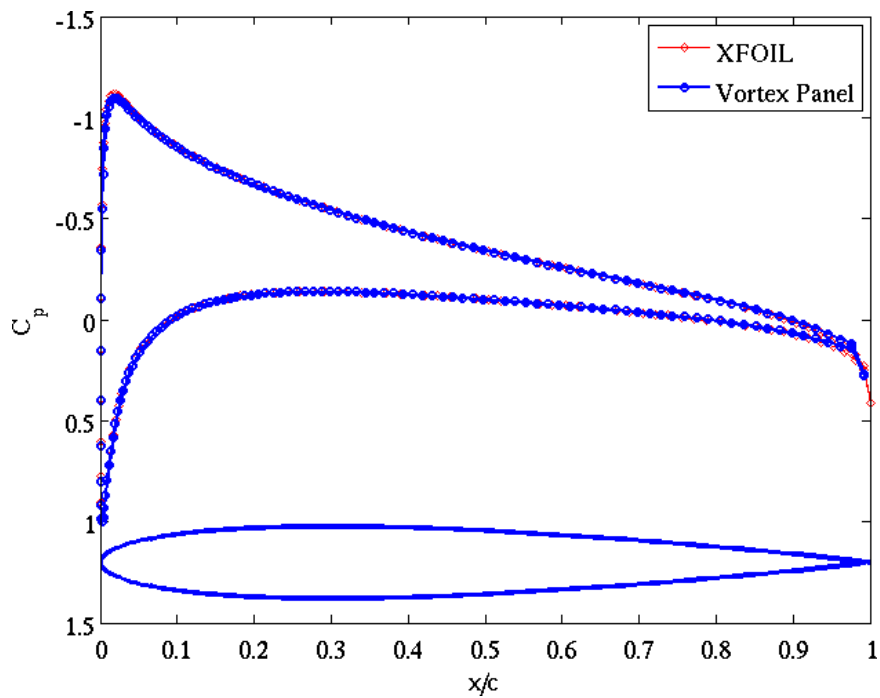


(b)

Figure 6-8: Comparison between a developed linear vortex panel code and XFOIL results with (a) 20 panels (top and bottom) on a NACA 0012 airfoil input geometry and (b)  $C_p$  results.



(a)



(b)

Figure 6-9: Comparison between a developed linear vortex panel code and XFOIL results with (a) 100 panels (top and bottom) on a NACA 0012 airfoil input geometry and (b)  $C_p$  results.



to a 3D analysis, the B-spline surface geometry gradient can be validated with these same model geometry because 3D aerodynamic effects are dominant.

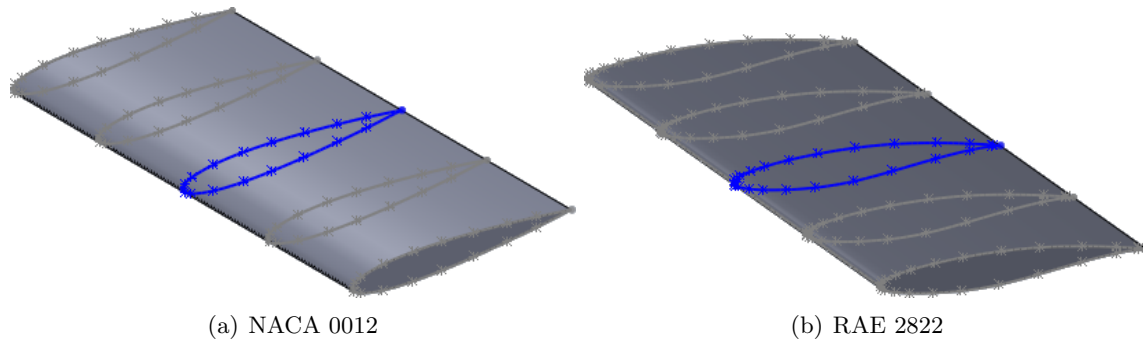


Figure 6-10: Simple 3D wing lofts were constructed in the SolidWorks CAD system to represent the (a) NACA 0012 and (b) RAE 2822 airfoil shapes.

In each case the CAD model consists of a single loft *feature* with four faces (including two trimmed planar end-caps), six edges and four nodes in its topology. Due to the simple construction approach, where the top and bottom airfoil splines have initial and final support points at the leading and trailing edge, the four loft faces consist of two end-cap planar faces and two cubic B-spline surfaces constrained at the leading/trailing edge<sup>2</sup>. The top/bottom B-spline surfaces are driven solely by the respective top/bottom B-spline curves in each cross-section profile.

Cross-section knot vectors, support points and control points are accessed through the CAPRI API, as are the B-spline surface knot vectors and control points. The middle airfoil section is selected for 2D analysis, thus the B-spline curve parameters are obtained and analytically reconstructed outside of the CAD system. This permits an arbitrary discretization of the B-spline curve since it is *not* an edge in the model topology that can be queried via the API<sup>3</sup>. Therefore, extraction of this particular B-spline curve results in a sub-model of lower-fidelity for use in the panel code analysis. A particular support point on this sub-model is chosen to be perturbed by some amount and the entire CAD model is regenerated, thereby resulting in an initial point for the design problem. As the optimizer determines new design points, new sub-models are obtained by regenerating the entire wing loft and

<sup>2</sup>This setup leads to a “sharp” leading edge. This can be remedied by clustering more support points near the leading edge to create a “rounded” shape or specifying the tangent vector at the initial support point.

<sup>3</sup>Since the model construction is sufficiently simple, it is possible to query the loft along a  $v = 0.5$  isoparameter line and obtain a discretization that would closely represent the underlying B-spline curve cross-section. Since the curve knot vector and surface knot vector are not exactly the same, the isoparameter line and cross-section curve may not be exactly the same.

again extracting the corresponding B-spline curve cross-section. This allows the final design point to be reflected in the sub-model *and* the 3D CAD model in a consistent fashion.

Using the baseline, unperturbed sub-model lift-coefficient as the target solution, an inverse-design problem is created to drive the perturbed starting point to a target pressure distribution and sub-model geometry. Constraints in the problem are set to limit a minimum thickness  $t/c = 7\%$  at  $x/c = \{15\%, 65\%\}$  and a maximum thickness anywhere at  $t/c = 25\%$ . These constraints are evaluated by querying the sub-model B-spline curves (top and bottom) at the corresponding  $x/c$  location and calculating the thickness normal to the chord-line. Finally, the design variable in the problem is the perturbed coordinate of the initially perturbed support point on the sub-model. This particular setup is selected in order to test the quality of the B-spline curve design velocity algorithm developed in Section 5.5. By driving a perturbed sub-model to a known baseline, it is fair to conclude that the design velocity prescribed to the optimizer is correct. A comparison using finite-difference design velocity is also used, where the CAD model geometry is perturbed by  $h = 1.0 \times 10^{-5}$  and the new sub-model B-spline curves are extracted for differencing in object space.

For both the NACA 0012 and RAE 2822 inverse-design problems, the optimization problem is written as

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathcal{J} = w(C_l - C_l^*)^2 \\
\text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}}, \\
& 7\% - (t/c)_1 \leq 0, \\
& 7\% - (t/c)_2 \leq 0, \\
& (t/c) - 25\% \leq 0
\end{aligned} \tag{6.2}$$

where  $C_l$  is the lift-coefficient at a given design point,  $C_l^*$  is the target lift-coefficient,  $\mathbf{x}$  is the  $y$ -coordinate of the perturbed support point and  $w$  is a weighting factor. The subscripts ‘‘LB’’ and ‘‘UB’’ denote the lower and upper side bounds on  $\mathbf{x}$ , respectively. Values for these parameters are summarized in Table 6.2. Flow conditions are set at  $M = 0.0$  (incompressible, infinite sound velocity) and  $\alpha = 3.0$ . Thickness constraint gradients are also taken from the B-spline design velocity as follows:

$$\frac{\partial(t/c)}{\partial\mathcal{P}} = \frac{1}{c} \left( \frac{\partial\mathbf{Y}(t)_{\text{upper}}}{\partial\mathcal{P}} - \frac{\partial\mathbf{Y}(t)_{\text{lower}}}{\partial\mathcal{P}} \right), \tag{6.3}$$

where the appropriate  $t$  and  $t'$  values are found on the upper/lower B-splines in order to compute the thickness normal to the chord.

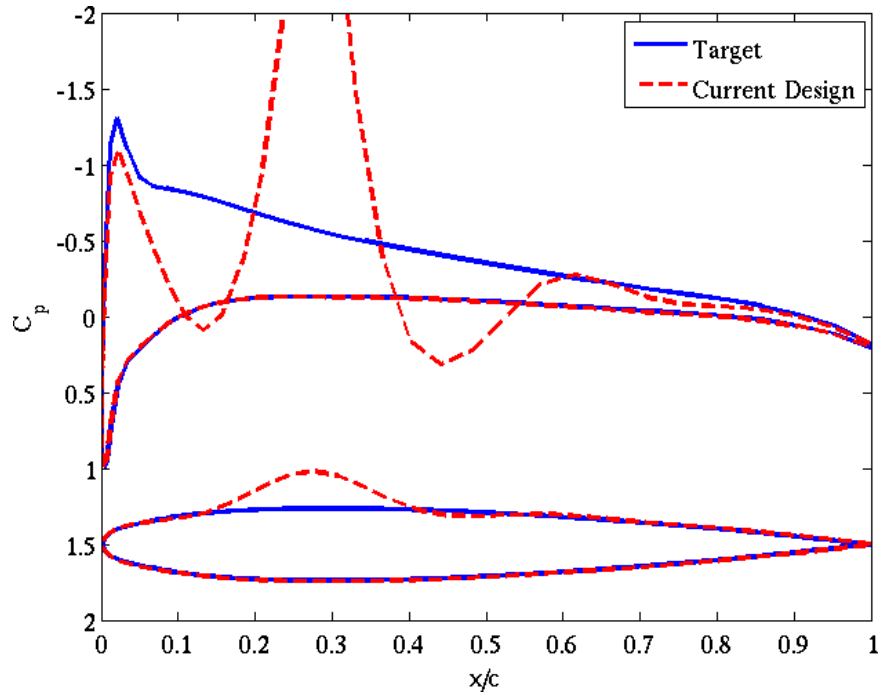
The gradient-based algorithm named `fmincon` in Matlab is used to drive the initial design point to a final design point resulting in  $\mathcal{J} \leq 1.0 \times 10^{-12}$  or a change in design variable less than  $1.0 \times 10^{-9}$ . For both wing loft cases, Figures 6-11 and 6-13 show the initial design point and the final design point obtained when the optimizer reaches a local minimum. In both examples the agreement between the final design point and the target are excellent. This serves as validation of the B-spline curve geometry gradient as well. Furthermore, this method also agrees well with the results using finite-difference design velocity, as seen in the objective history in Figure 6-12(a) and Figure 6-14(a), or the design trajectory in Figure 6-12(b) and Figure 6-14(b).

	<b>NACA 0012</b>	<b>RAE 2822</b>	<b>NACA 0012 (MSES)</b>
<i>Initial Perturbation</i>	2.0y	1.8y	1.12y
<i>Side Bounds</i>	[0.5y, 2.5y]	[0.5y, 3.1y]	[0.8y, 1.15y]
$C_l^*$	0.178045	0.292885	0.283510
$w_l$	$1.0 \times 10^6$	$1.0 \times 10^6$	$4.0 \times 10^6$
$C_{d_w}^*$			$3.244189 \times 10^{-3}$
$w_d$			$5.0 \times 10^8$

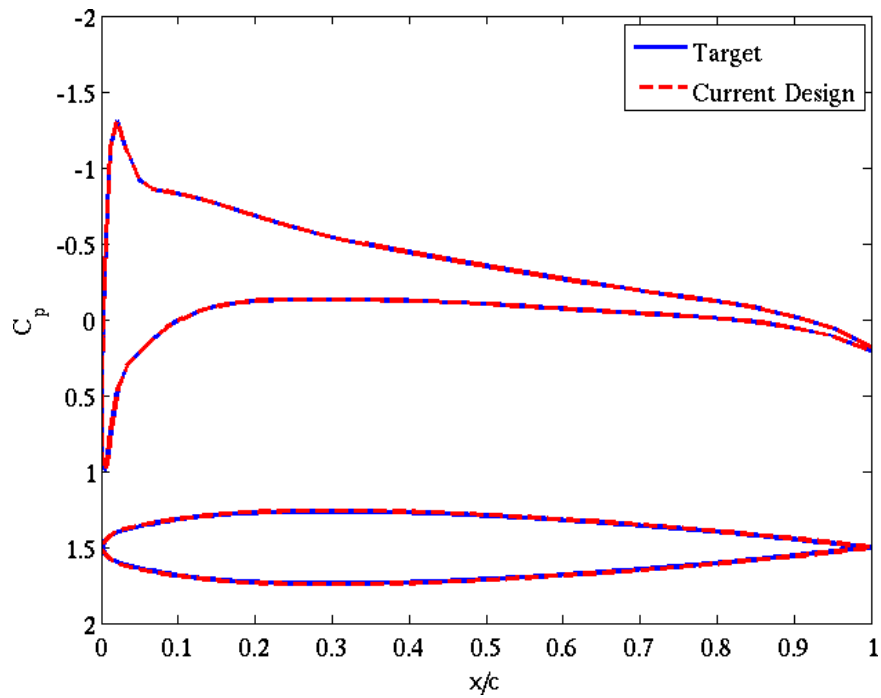
Table 6.2: Setup data for the inverse design problems on simple NACA 0012 and RAE 2822 model geometry. The differentiated panel code is used for each problem except the NACA 0012 with the MSES designation.

An additional 2D inverse-design problem is done using the MSES Euler code [21] on the RAE 2822 geometry. Wave-drag is now included in the objective functional and only side bound constraints on  $\mathbf{x}$  are needed. Tighter bounds on the design space were chosen to avoid additional local minima seen with larger bounds<sup>4</sup>. This did not warrant using the thickness constraints from the previous examples. The optimization problem then became

<sup>4</sup>Other local minima were reached due to the presence of multiple shocks. In some instances one shock diminished in strength while the other increased in strength. The combined effect would null a change in total wave-drag or cause a net-increase. This caused a design trajectory to terminate at local minima away from the intended target. The chosen side bounds in this problem avoided this situation.

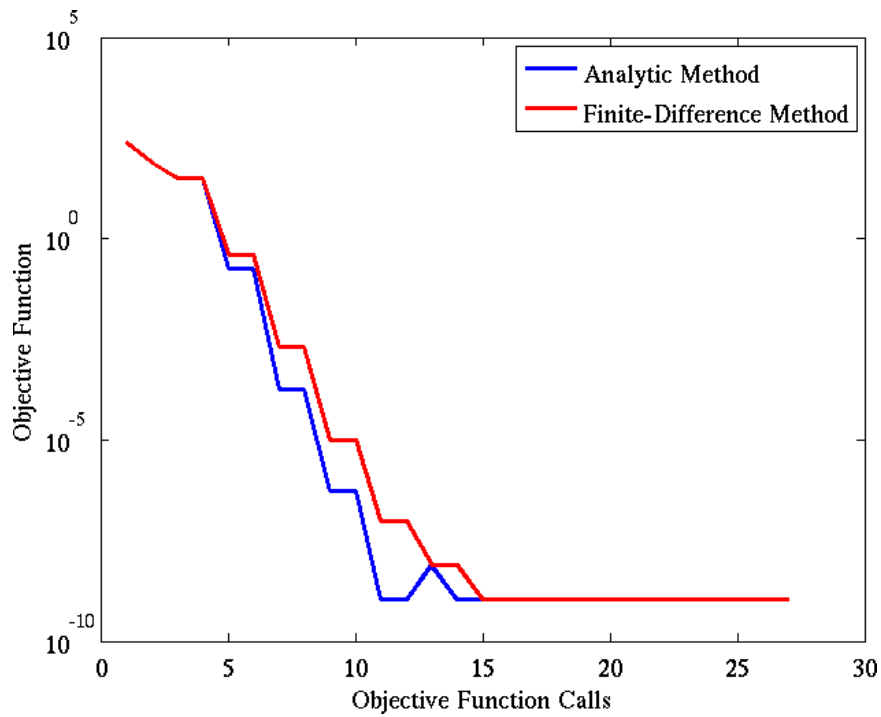


(a) NACA 0012 Initial

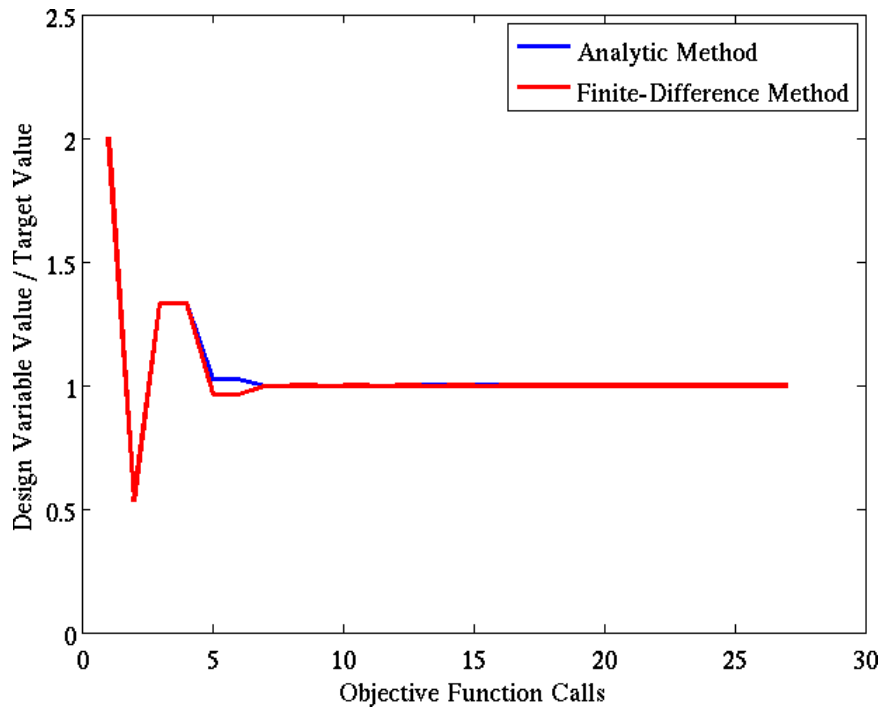


(b) NACA 0012 Final

Figure 6-11: (a)–(b) Inverse-design problem for the NACA 0012 airfoil depicting initial and final design points.

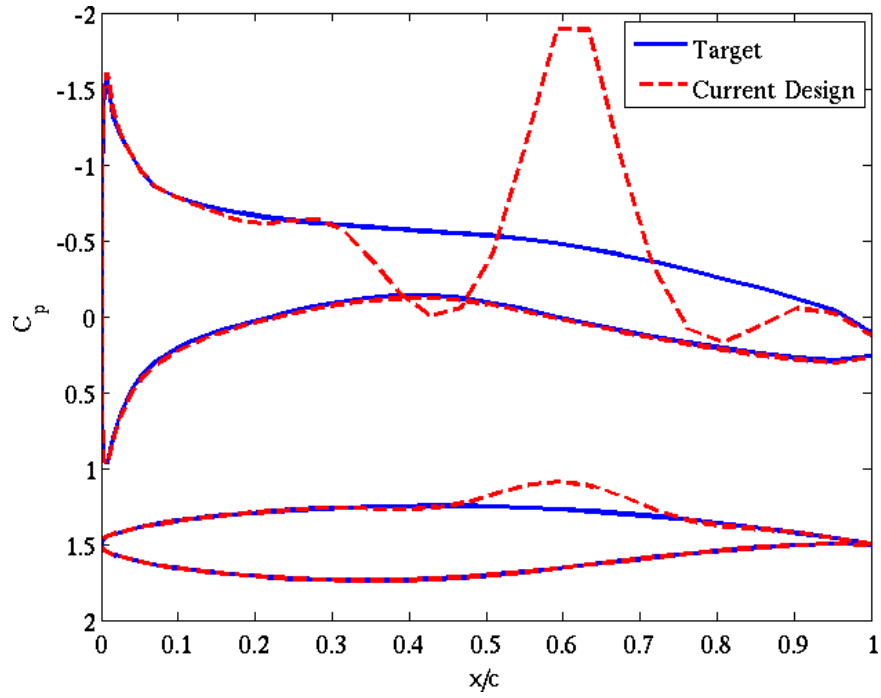


(a)

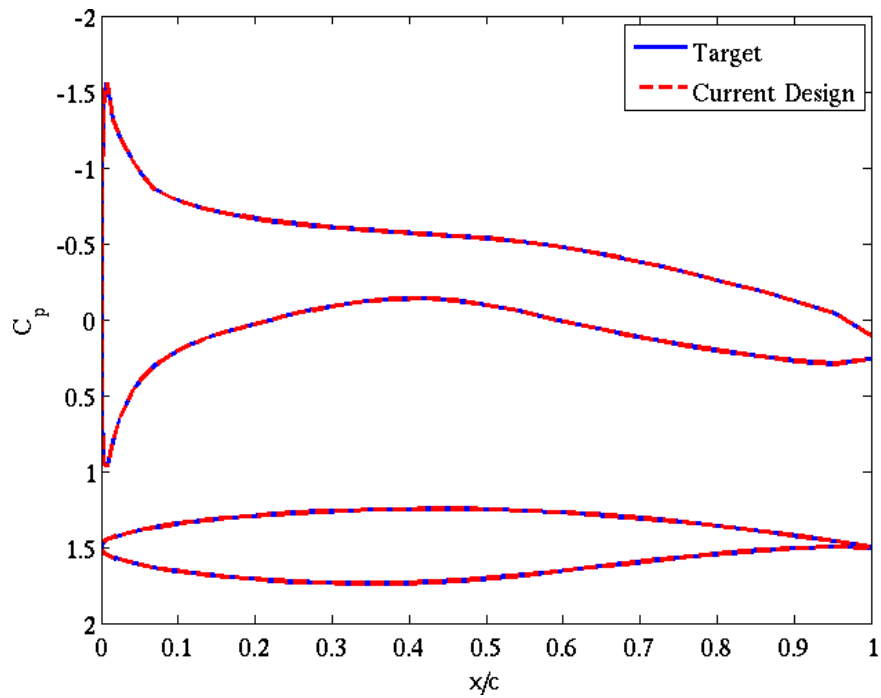


(b)

Figure 6-12: (a) Comparison between the objective function history results using the analytic and finite-difference design velocity methods for the inverse-design problem in Figure 6-11. (b) Comparison between the resulting design trajectory obtained with both methods.

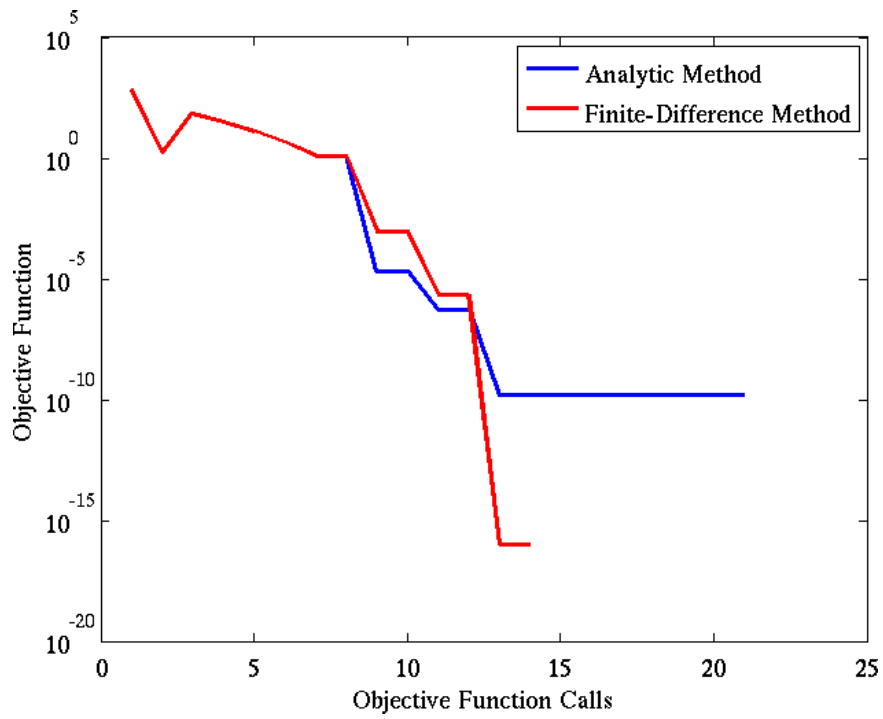


(a) RAE 2822 Initial

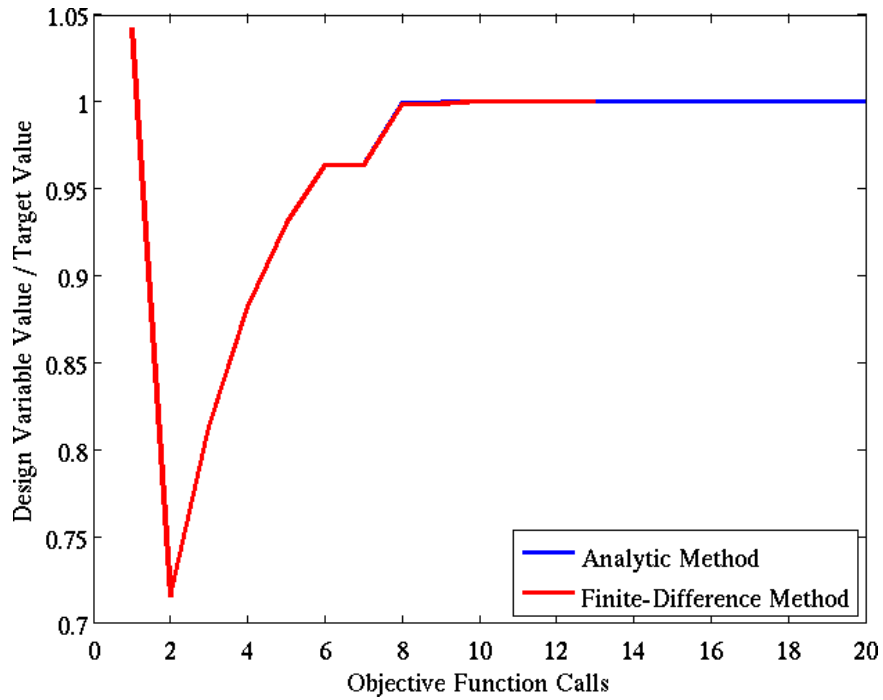


(b) RAE 2822 Final

Figure 6-13: (a)–(b) Inverse-design problem for the RAE 2822 airfoil depicting initial and final design points.



(a)



(b)

Figure 6-14: (a) Comparison between the objective function history results using the analytic and finite-difference design velocity methods for the inverse-design problem in Figure 6-13. (b) Comparison between the resulting design trajectory obtained with both methods.

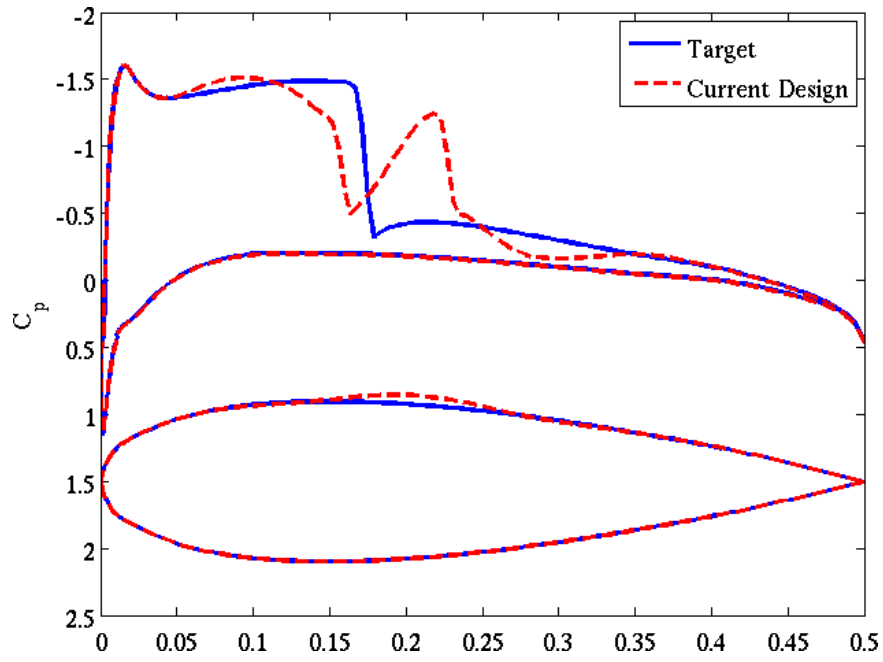
$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathcal{J} = w_l(C_l - C_l^*)^2 + w_d(C_{d_w} - C_{d_w}^*)^2 \\
\text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}}
\end{aligned} \tag{6.4}$$

The complete objective functional gradient is only approximated by the B-spline geometry gradient because the MSES code was not differentiated. This code utilizes the same discretization points obtained from the sub-model in the panel code problem. With this input the MSES suite generates a cubic B-spline interpolation and a subsequent flow-field mesh. An inverse-design problem is run at  $M = 0.7$  and  $\alpha = 3.0^\circ$  using the problem data in Table 6.2. The same optimizer is used as in the panel code problem and the target design is recovered within  $0.99969y$ . This is an excellent result given the gradient circumstances. Figure 6-15 depicts the initial and final design points for this inverse problem. This again validates the quality of the B-spline curve geometry gradient algorithm through a non-ideal case. Comparison with a finite-difference gradient also yields excellent agreement, as seen in the objective function history in Figure 6-16(a) and the design trajectory in Figure 6-16(b). The target design is recovered within  $0.99968y$  using finite-differences.

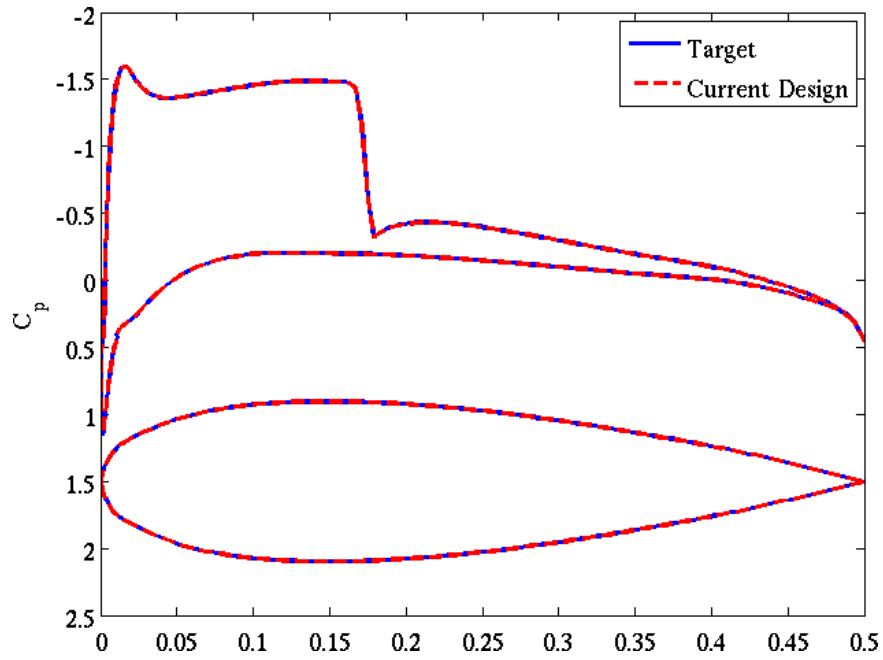
### 6.2.3 2D Single-Discipline Design Problem

A forward design problem is presented here to highlight the impact of errors in the finite-difference gradient on design trajectories of B-spline curves. A sub-model geometry is extracted from the mid-plane cross-section of the BWB model in Figure 6-23, which already contains a  $+1^\circ$  incidence angle. This sub-model consists of cubic B-spline curves on the top and bottom airfoil that meet at the leading/trailing edges. The discretized sub-model is passed to MSES for analysis. The  $y$ -component of two support points are selected as design variables on the top B-spline curve. With the RAE 2822 airfoil as the initial design point shape, a  $M = 0.725$  flow at  $\alpha = 0.125^\circ$  provides conditions for a weak shock on the upper surface that results in a wave-drag coefficient of  $C_{d,w} = 0.000245$ . A constrained gradient-based minimization problem is written for this experiment with the intent of removing wave-drag:



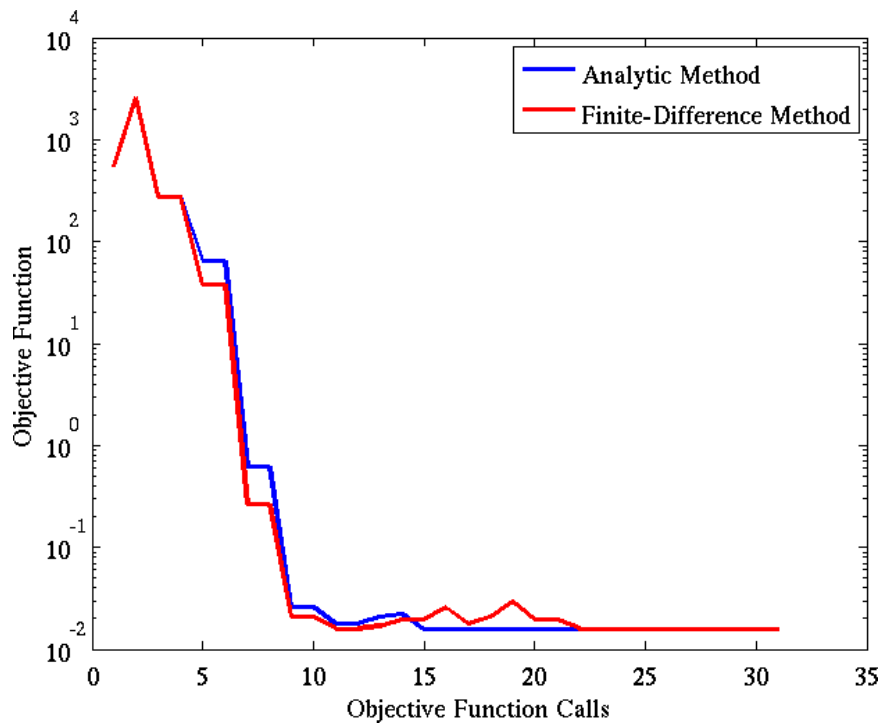


(a) Initial

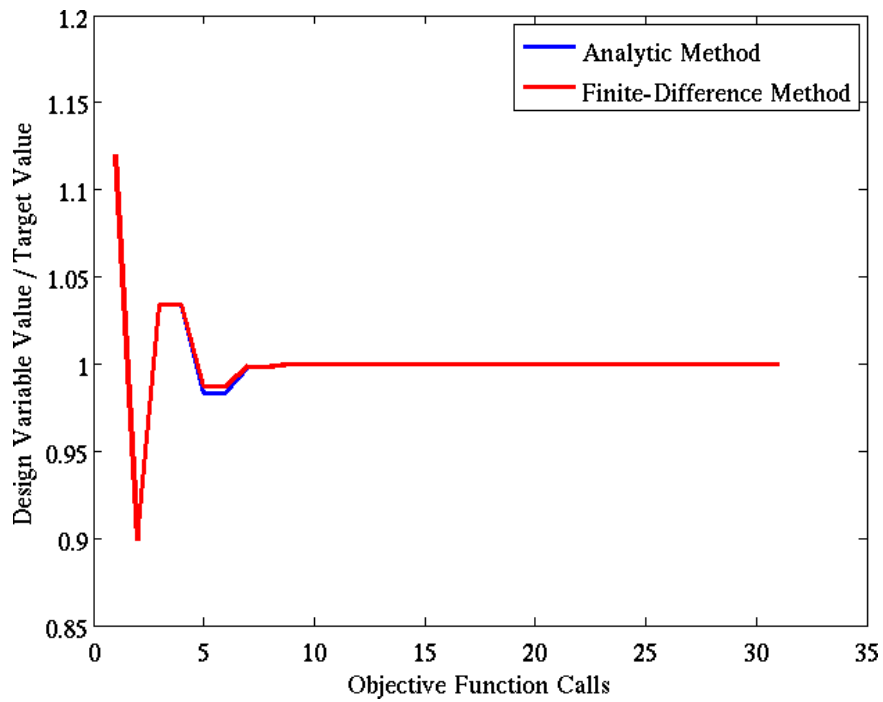


(b) Final

Figure 6-15: In using the MSES analysis tool, (a) the initial design point for a NACA 0012 sub-model is driven to a (b) final design point. The airfoil  $y$ -coordinates are scaled to show greater curvature detail.



(a)



(b)

Figure 6-16: (a) Comparison between the objective function history results using the analytic and finite-difference design velocity methods for the inverse-design problem in Figure 6-15. (b) Comparison between the resulting design trajectory obtained with both methods.

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathcal{J} = w(C_d - C_d^*)^2 \\
\text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}}, \\
& 7\% - (t/c)_1 \leq 0, \\
& 7\% - (t/c)_2 \leq 0, \\
& (t/c) - 25\% \leq 0
\end{aligned} \tag{6.5}$$

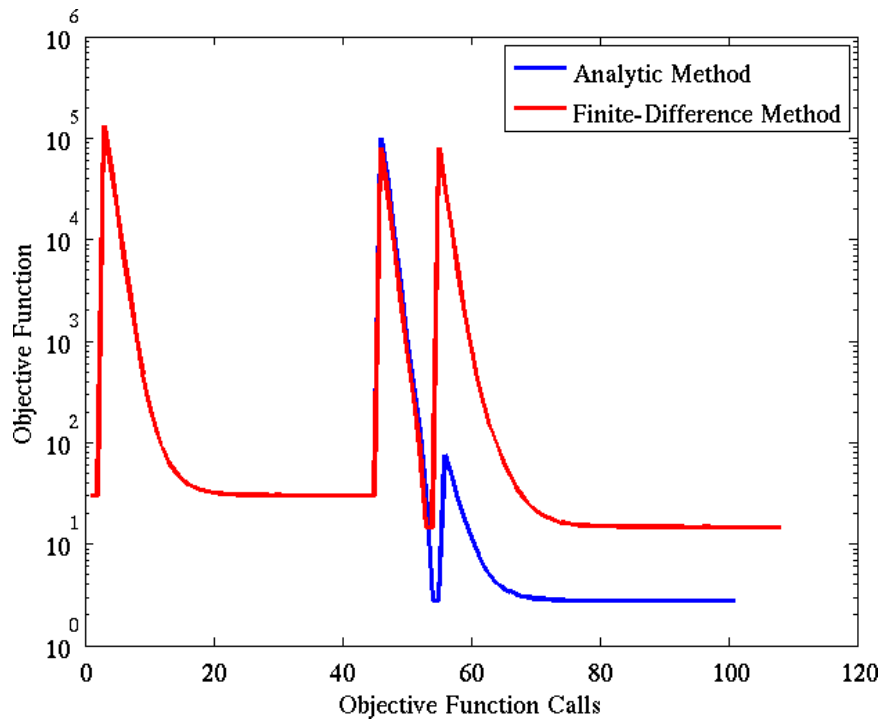
where the target drag coefficient is  $C_d^* = 0.0$  and the weight is  $w = 5.0 \times 10^8$ . The constraints enforce minimum thickness at chord locations  $(x/c)_1 = 15\%$  and  $(x/c)_2 = 65\%$  and a maximum thickness everywhere on the airfoil in the same manner as in Section 6.2.2. Side bounds for both design variables are set as  $[0.8y, 1.12y]$ . Both the analytic design velocity method from Section 5.5 and a finite-difference method are used for comparison.

Table 6.3 summarizes results from cases using both methods. The analytic method provides a design trajectory that results in much lower wave-drag than that found using finite-differences. The execution time for the analytic case is much faster than the finite-difference case as well. Figure 6-17 illustrates the objective function and normalized design variable history, where a design trajectory bifurcation is visible between the analytic and finite-difference cases. Figure 6-18 also shows the initial and final  $C_p$  distributions for both design velocity cases. It appears that two design variables are insufficient to eliminate the shock. Each changing support point has a competing influence on the shock since both contribute to the airfoil design motion fore/aft of the shock. The bifurcation in design

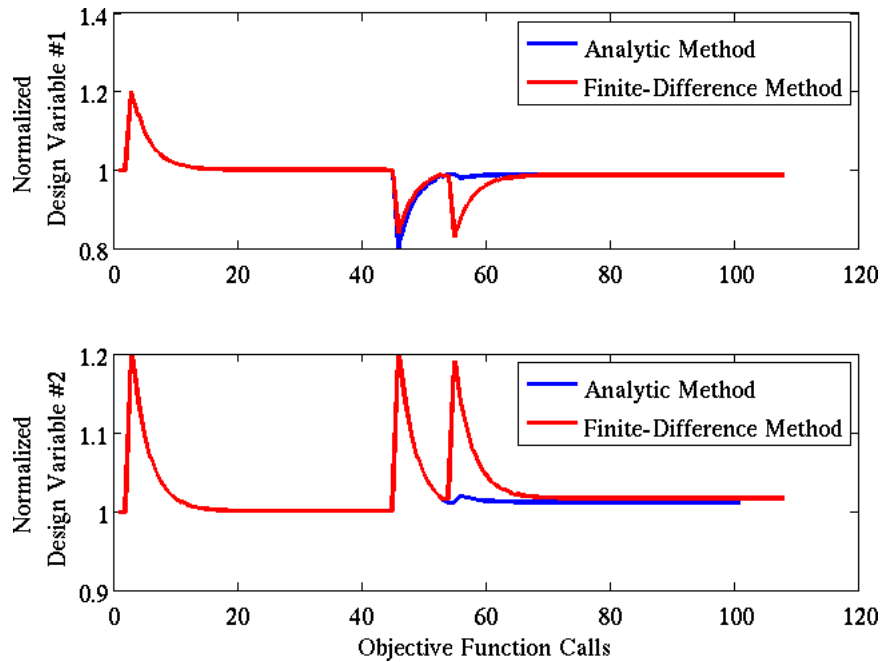
<b>Gradient Method</b>	<b>Initial <math>C_d</math></b>	<b>Final <math>C_d</math></b>	<b>Execution Time [min.]</b>
<i>Analytic</i>	0.000245	0.000074	43.93
<i>Finite-Difference</i>	0.000245	0.000172	203.93

Table 6.3: Results from a design problem for wave-drag minimization wherein analytic and finite-difference design velocity are used.

trajectory between the analytic and finite-difference case is a consequence of the geometry gradient quality. As discussed in Section 5.8, this design problem exhibits inconsistent finite-difference computations throughout its design trajectory. Due to the sensitive nature of the transonic flow-field with weakening shocks along the airfoil upper surface, the geometry gradient errors appear sufficient in magnitude to bifurcate the design trajectory. This is an



(a) Objective Function History



(b) Normalized Design Variable History

Figure 6-17: Comparison of objective function and design variable history between an analytic and finite-difference design velocity method for a drag minimization problem.

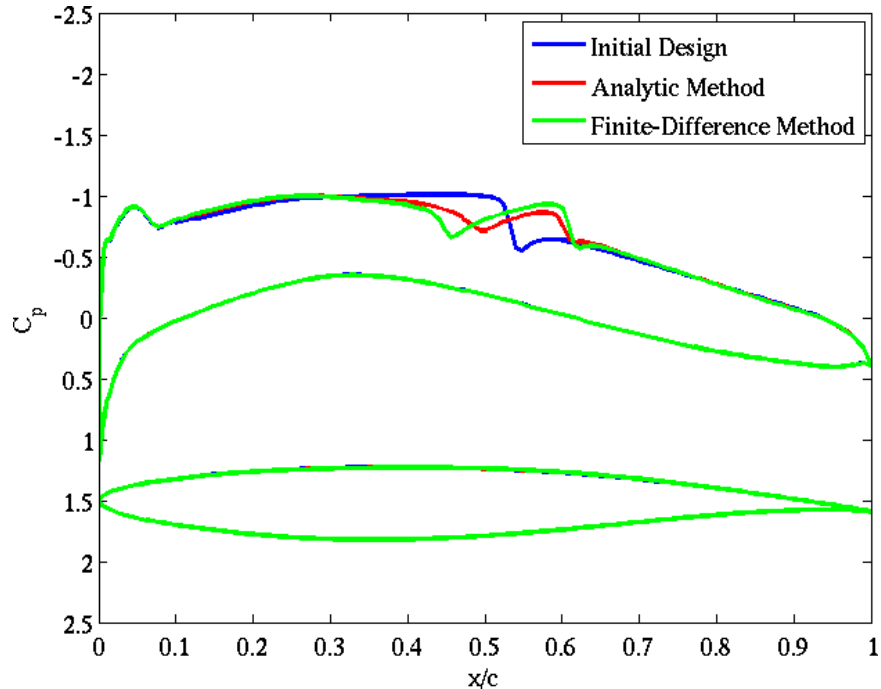


Figure 6-18: Comparison of  $C_p$  distribution between the initial and final design point results using an analytic and finite-difference design velocity method in a drag minimization problem.

important realization since many practical design problems of interest in aerospace require shape design to eliminate shocks. This example shows that finite-differencing support points on CAD-generated B-spline curves can lead to different results since knots and/or support points are reparameterized by the geometry kernel in a non-transparent manner. Without the analytic method to compare, the finite-difference result is typically considered sufficient since the location of local minima are initially unknown in the design space. Scrutiny of such results is necessary when employing finite-differences methods on this type of model geometry.

#### 6.2.4 3D Single-Discipline Inverse-Design Problem

The inverse-problem conducted in 6.2.2 is repeated in a 3D setting using the Cart3D design framework [59] for the NACA 0012 wing (Figure 6-10(a)) at  $M = 0.5$  and  $\alpha = 3.0^\circ$ . In this design framework, volume mesh and residual sensitivities are computed in a non-transparent manner within the framework and an internal SQP optimizer is utilized. Surface tessellation sensitivities are provided using the B-spline surface geometry gradient algorithm introduced in 5.5. The surface tessellation has about 378000 triangular elements and the volume mesh

has about 661000 cells. A modification of the optimization problem is done by specifying a new target  $C_L^*$ , corresponding to the Cart3D baseline result, and removing the thickness constraints to give

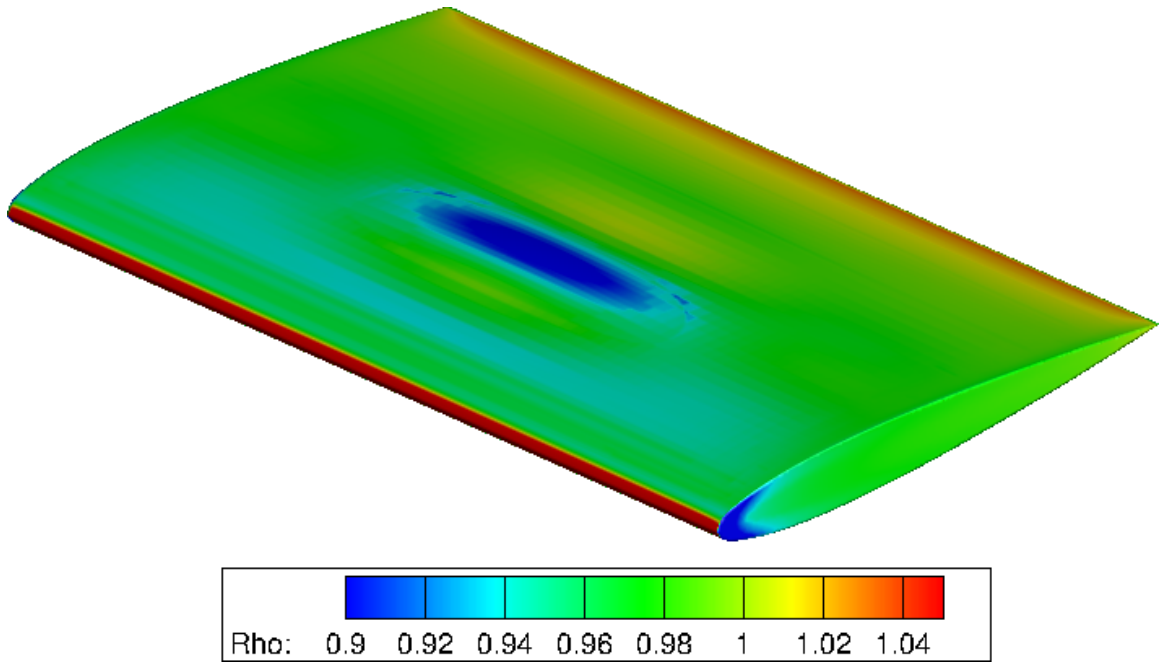
$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{J} = w(C_L - C_L^*)^2, \\ \text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}} \end{aligned} \tag{6.6}$$

where the problem data is defined in Table 6.4. Results for this case are shown in Figures 6-19 through 6-20, where the objective function history demonstrates that the baseline shape and pressure distribution are recovered very well. In this case a local minimum near the target coordinate value is found before the design trajectory reaches the target geometry itself. This test case also serves to validate the quality of the B-spline surface geometry gradient since there is excellent agreement with the finite-difference results.

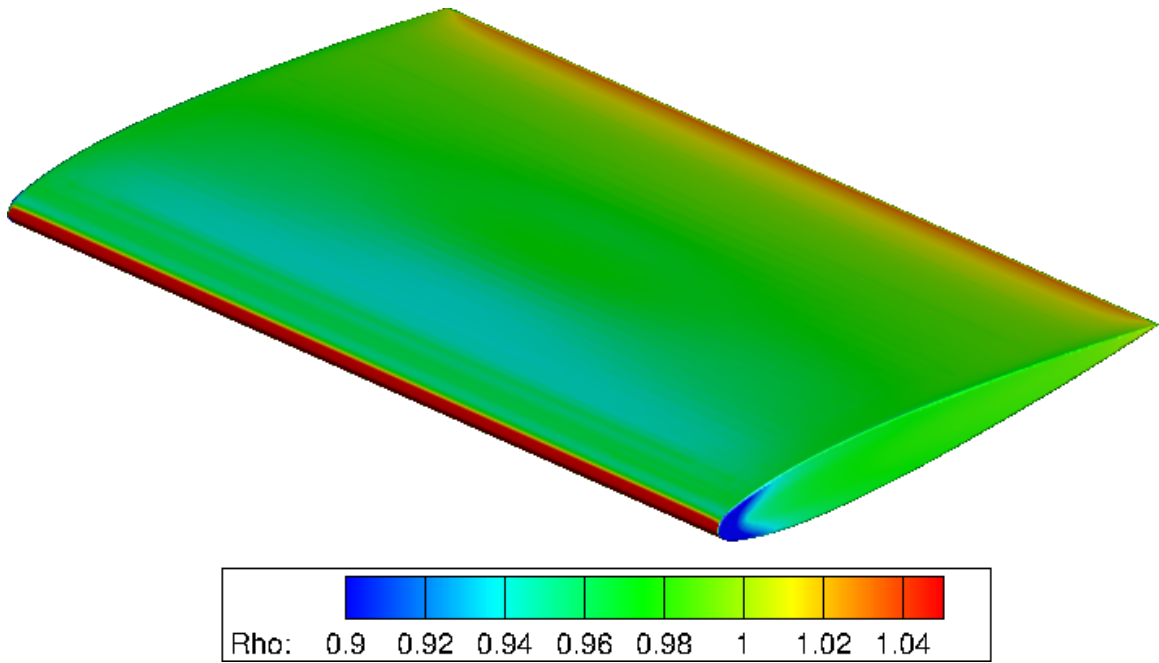
<b>Model Geometry</b>	<b>Initial Perturbation</b>	<b>Side Bounds</b>	$C_L^*$	$w$
<i>NACA 0012</i> (3D)	1.3 $y$	[0.88 $y$ , 1.4 $y$ ]	0.011988	$1.0 \times 10^6$
<i>NACA 0012</i> (2D)	1.09 $y$	[0.88 $y$ , 1.12 $y$ ]	0.266244	$1.0 \times 10^6$

Table 6.4: Setup data for the inverse design problems on a simple 3D NACA 0012 model geometry and 2D cross-section case. The design problems are both executed using the Cart3D design framework.

The same design framework can conduct 2D flow analysis by taking the 3D input geometry and only discretizing the volume mesh along a “planar-cut” 1-cell deep, resulting in a volume mesh with about 5500 cells. This analysis mode is used as a lower-fidelity analysis to repeat the inverse design problem above. The problem uses the same NACA 0012 wing model, but different constraints, target  $C_L^*$  and initial perturbation magnitude, as summarized in Table 6.4. B-spline surface geometry gradients are also provided as before. The results in Figure 6-21 and 6-22 validate the quality of the geometry gradient once again as the baseline shape and target  $C_L^*$  are reached. Agreement is excellent between the analytic and finite-difference design velocity methods as well. With both design velocity methods a local minimum is found before reaching the target support point  $y$ -component value.

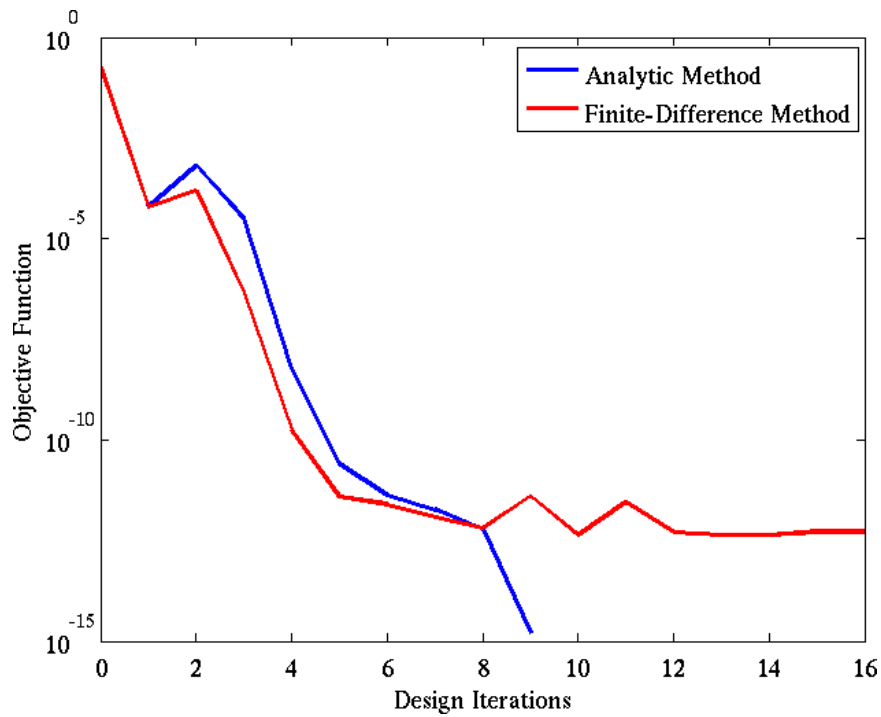


(a) Initial

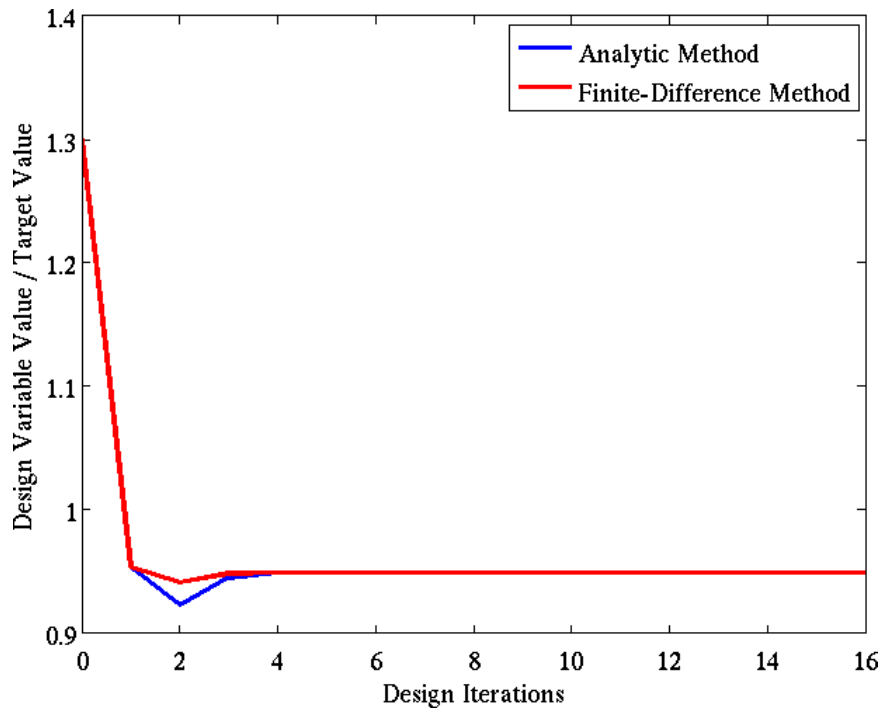


(b) Final

Figure 6-19: Initial and final design points for the 3D NACA 0012 wing via inverse design with the Cart3D design framework.



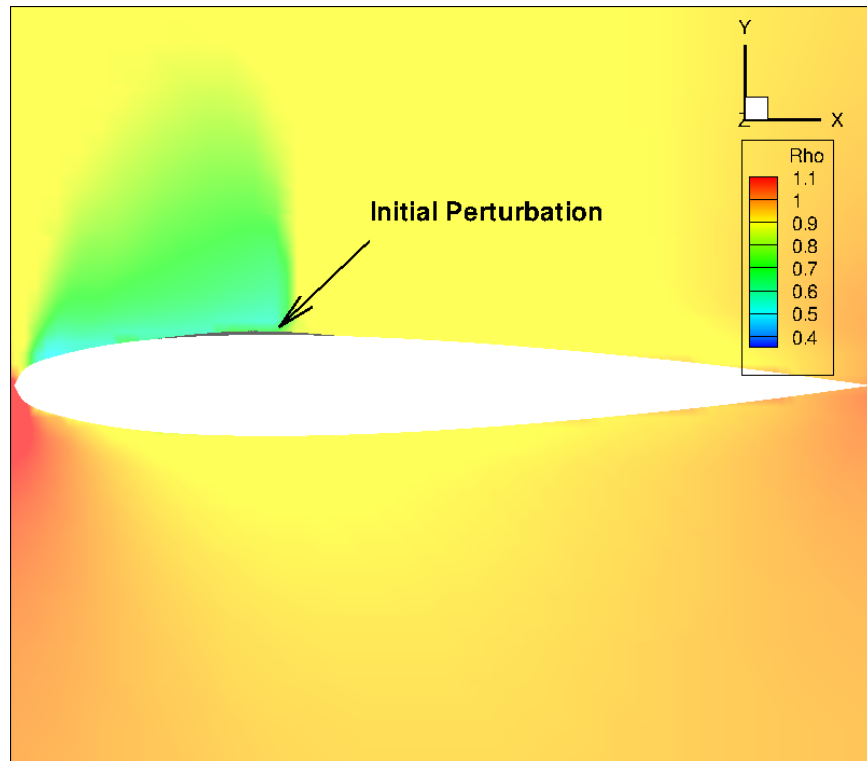
(a)



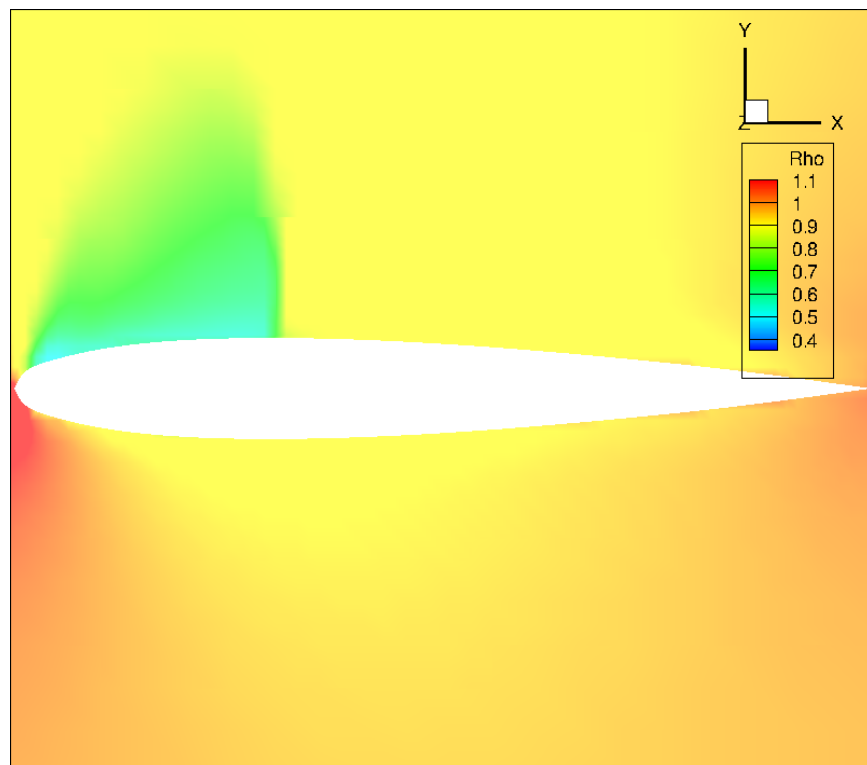
(b)

Figure 6-20: (a) The objective function history for the 3D NACA 0012 inverse design problem comparing the analytic and finite-difference gradient methods; (b) the design trajectory comparison between the two methods.



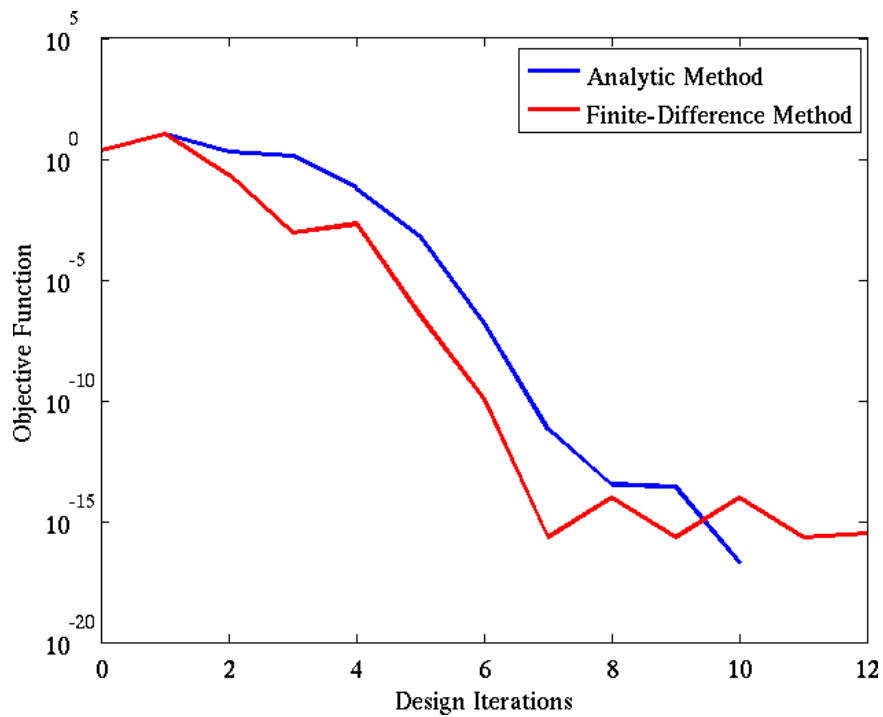


(a) Initial

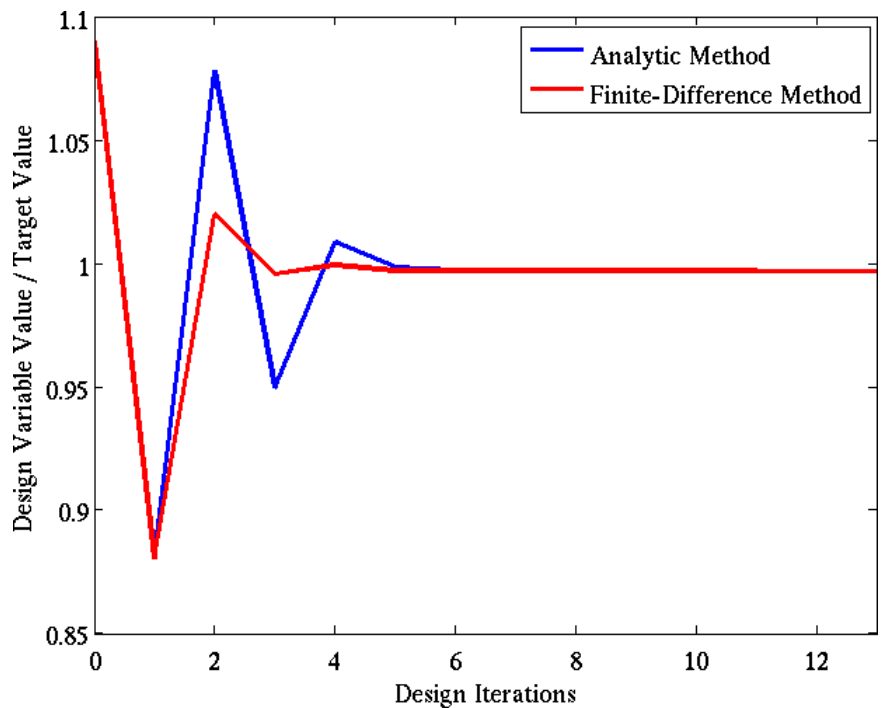


(b) Final

Figure 6-21: (a) Initial 3D NACA 0012 wing with spline point perturbation in a 2D analysis, followed by (b) the final design point obtained via inverse design with the Cart3D design framework.



(a)



(b)

Figure 6-22: (a) The objective function history for the 3D NACA 0012 inverse design problem comparing the analytic and finite-difference gradient methods; (b) the design trajectory comparison between the two methods.

## 6.2.5 3D Single-Discipline Design Problem

Once the geometry gradients are validated via the previous inverse design problems, an unconstrained-lift and constrained-lift forward design problem is tested to further illustrate the usefulness of the analytic gradient approach. A Blended-Wing-Body (BWB) SolidWorks model is created with an airfoil stack consisting of nine RAE 2822 cross-section profiles. A trapezoidal planform area is used to distribute the airfoils and both wing area and aspect ratio are determined from a prior study [50]. In addition to using  $5^\circ$  of dihedral, a linear twist distribution is used from root-to-tip, where the root airfoil contains a  $+1^\circ$  incidence angle and a  $-3^\circ$  washout is set at the wing tips. As in the previous example models, the airfoils are divided at the leading and trailing edge with a cubic B-spline interpolating 12 support points (obtained from RAE 2822 coordinates) in a half-cosine manner, thereby better resolving the leading edge radius of curvature. The resulting lofts interpolating these airfoil sections are a bicubic B-spline surface for the top and bottom. The geometry is tessellated with about 196000 triangulation elements in a small-mesh case and about 596000 elements in a large-mesh case. Mesh parameters are reused for tessellating model geometry at each design point in each design problem.

Both design problems consist of reducing drag in a transonic flow with  $M = 0.8$  and an initial  $\alpha = 5^\circ$ . Two support points on each of the airfoil upper B-spline curves are design variables ( $y$ -coordinate) in addition to  $\alpha$  for a total of 19 design variables. The gradient-based Cart3D design framework is again utilized and loft geometry gradients are calculated using the algorithm presented in this work.

The unconstrained-lift problem is formulated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{J} = w_D(C_D - C_D^*)^2, \\ \text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}} \end{aligned} \tag{6.7}$$

with a weight  $w_D = 1.0 \times 10^3$  and a target drag  $C_D^* = 0.0$ . The set  $\mathbf{x}$  consists of two upper spline support points at each cross-section of the BWB and  $\alpha$ . Upper and lower bounds for the support points are  $[0.7y_i, 1.1y_i]$  for the  $i$ th  $y$ -coordinate and  $\pm 10^\circ$  for  $\alpha$ .

Although a shock initially exists, a local minimum is found that reduces the shock and drag from  $C_D = 0.04614$  to  $C_D = 0.01522$  in the small mesh case. This again confirms the quality of the geometry gradient algorithm for forward design. Figure 6-24 illustrates

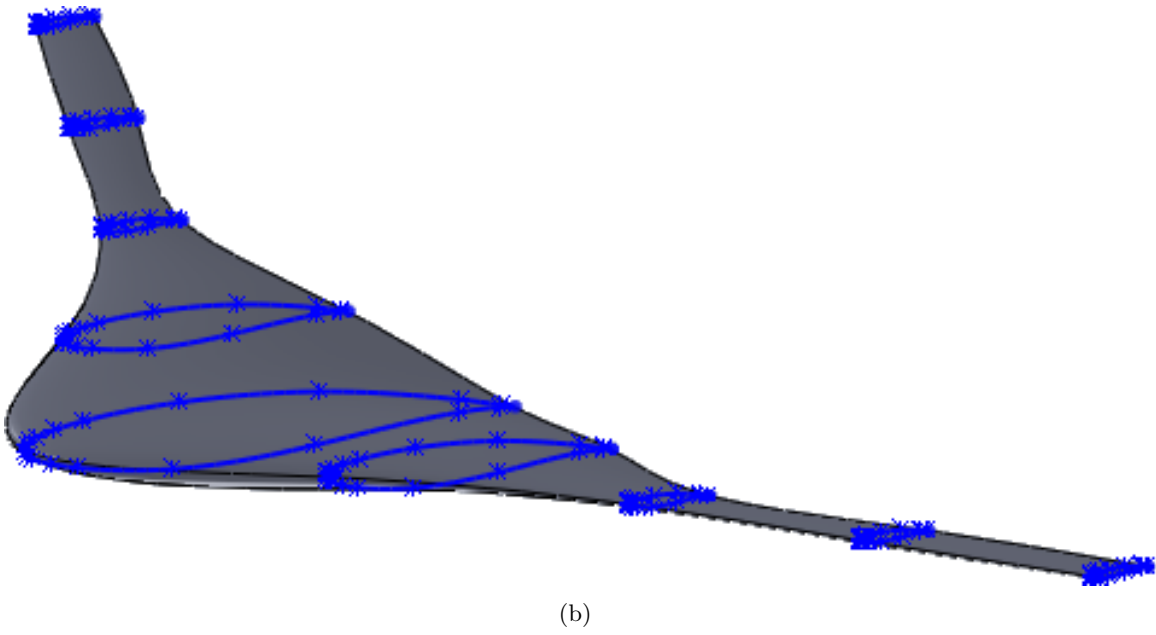
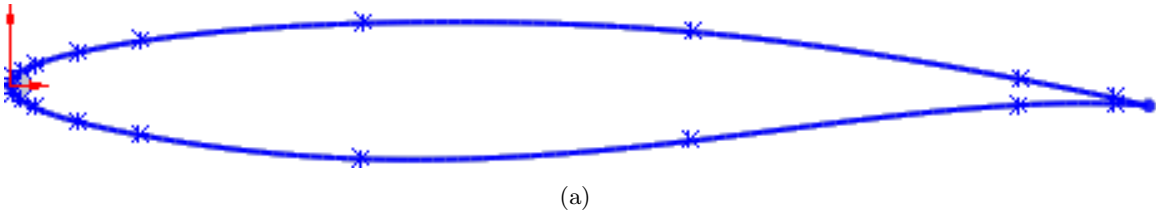


Figure 6-23: Model geometry created using SolidWorks to represent a (a) RAE 2822 airfoil and (b) Blended-Wing-Body aircraft outer mold line.

the initial and final  $C_p$  contours across the top of the BWB loft and Figure 6-25 shows the initial and final cross-section  $C_p$  distributions at select span-wise locations.

The constrained-lift problem is formulated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{J} = w_D(C_D - C_D^*)^2 \\ \text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}}, \\ & C_L - C_L^* = 0, \end{aligned} \tag{6.8}$$

where  $C_L^* = 0.3$ . This is reformulated by augmenting the objective function with a penalty on lift-coefficient as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{J} = w_D(C_D - C_D^*)^2 + w_L(C_L - C_L^*)^2, \\ \text{s.t.} \quad & \mathbf{x}_{\text{LB}} \leq \mathbf{x} \leq \mathbf{x}_{\text{UB}} \end{aligned} \tag{6.9}$$

where the lift-coefficient weight is also set to  $w_L = 1.0 \times 10^3$ . The design variable set and initial flow conditions are exactly the same as in the previous unconstrained-lift problem. With the same starting point and the small mesh case, the initial drag of  $C_D = 0.04614$  is reduced to  $C_D = 0.025998$ . The lift-coefficient is driven from the initial value  $C_L = 0.36896$  to  $C_L = 0.29452$ , which is close to the desired value of  $C_L = 0.3$ . This again confirms the quality of the geometry gradient algorithm. Figure 6-26 illustrates the initial and final  $C_p$  contours across the top of the BWB loft and Figure 6-27 shows the initial and final cross-section  $C_p$  distributions.

In both the constrained-lift and unconstrained-lift design problems a local minimum is found that yields an improved aerodynamic design compared to the initial design. Due to the typically large sensitivity of transonic flow to changes in geometry, the design space is non-linear and other local minima are possible. Choosing different spline support points (or more of them) as design variables results in different local minima as well for the same objective functional. In these cases a “wavy” airfoil shape often emerges with multiple shocks when changing an increasing number of support points. It is possible to see a single support point strengthen some shocks and weaken others since its design velocity field influences design motion upstream and downstream along the airfoil. The changes in shock strength by one design variable in a region may cancel due to the combined design motion effect of multiple design variable changes in another. Such airfoil shapes may be avoided

<b>Design Problem</b>		<b>Initial</b>	<b>Target</b>	<b>Analytic Gradient</b>	<b>Finite-Difference Gradient</b>
<i>Unconstrained-lift</i>	$C_D$	0.04614	0.0	0.01522	0.02496
<i>Constrained-lift</i>	$C_D$	0.04614	0.0	0.025998	0.033496
	$C_L$	0.36896	0.3	0.29452	0.31200
<b>Avg. Sensitivity Computation Time</b> [ <i>min:sec / Design Iteration</i> ]					
<i>Unconstrained-lift</i>				7:22	23:07
<i>Constrained-lift</i>				6:27	22:29

Table 6.5: Results comparison when using the analytic and finite-difference geometry gradients for an unconstrained-lift and constrained-lift design problem. Surface mesh size is about 196000 elements and average volume mesh size is about 935500 cells. Computation time is reported for the full geometry gradient.

by constraining against inflection points along the airfoil.

A last comparison is made between design velocity methods in both the unconstrained-lift and constrained-lift problems. The geometry gradient is first computed using the algorithm developed herein and in a separate run a finite-difference computation provides the gradient. All other geometry and flow conditions are exactly the same for both cases. Results from both types of runs are surprisingly different, where the analytic geometry gradient obtains superior drag results in both the constrained-lift and unconstrained-lift problems and more closely meets the  $C_L$  constraint. In each case the average execution time for the full geometry gradient per design iteration is much less for the analytic method than the finite-difference method. These results are summarized in Table 6.5 for the small-mesh case and Table 6.6 for the large-mesh case. The only significant change due to mesh size is seen in the finite-difference result for  $C_D$  in the unconstrained-lift case. The objective function history for both problems, seen in Figure 6-28, shows the analytic case reaching a different local minimum compared to the finite-difference case. Design trajectory comparisons between the two methods are also seen in Figures 6-29 through 6-32, where each design variable trajectory per cross-section across the semi-span is recorded. It is clear that the varying design velocities between methods causes a different traversal of the design space towards separate local minima.

The difference in results between the finite-difference and analytic geometry gradients are attributed to an inherently different quality in the computed gradient direction. This

<b>Design Problem</b>		<b>Initial</b>	<b>Target</b>	<b>Analytic Gradient</b>	<b>Finite-Difference Gradient</b>
<i>Unconstrained-lift</i>	$C_D$	0.046153	0.0	0.015164	0.032265
<i>Constrained-lift</i>	$C_D$	0.046153	0.0	0.026023	0.033441
	$C_L$	0.369088	0.3	0.294527	0.311856
<b>Avg. Sensitivity Computation Time [hr:min:sec / Design Iteration]</b>					
<i>Unconstrained-lift</i>				0:33:42	1:42:51
<i>Constrained-lift</i>				0:38:51	1:39:24

Table 6.6: Results comparison when using the analytic and finite-difference geometry gradients for an unconstrained-lift and constrained-lift design problem. Surface mesh size is about 596000 elements and average volume mesh size is about 935500 cells. Computation time is reported for the full geometry gradient.

result is consistent with the results seen in the 2D example of Section 6.2.3. As explained in Section 5.8, the finite-difference computation may easily become incorrect if the B-spline surface knot vectors are different when perturbing the model (in these cases it is clear that the finite-difference approach creates non-zero geometry gradients in all three component directions, which is symptomatic of an inconsistency since only one direction should be non-zero). Since the analytic gradient avoids this condition, it appears to provide a higher quality geometry gradient that allows an optimizer to find different local minima for both deterministic problems tested here. In especially sensitive flow regimes, as in the transonic case considered here, a bifurcation in design trajectory is possible based on the design velocity quality. In contrast, the examples in Sections 6.2.2 and 6.2.4 did not show the sensitivity to design velocity required to bifurcate a design trajectory between the analytic and finite-difference cases even though the finite-difference gradients are inconsistent. In those cases the flow-field is less-sensitive to the geometry because the shock is stronger or non-existent. As seen in the example of Section 6.2.3, though, weakening or emerging shocks make the flow-field very sensitive to design velocity. 3D effects may exacerbate this sensitivity further in certain situations.

It is noteworthy to point out that a finite-difference case starting from the final design obtained with analytic gradients also stayed at that same local minimum. However, starting the analytic gradient case from the original finite-difference final design resulted in a new design trajectory to a different local minimum altogether (this was an improved design

compared to the local minimum found previously with analytic gradients). This implies that the finite-difference gradient error may cause a stall in progression along a design trajectory towards a true local minimum.

These scenarios can frequently occur in practical design, thus attention is needed on the design velocity quality when employing finite-difference methods with support points. Potentially adding “noise” to a finite-difference computation may provide insight about the sensitivity of design trajectory to design velocity in a particular problem if the B-spline curve or surface information is unavailable. Finally, a comparison between the results from different surface discretizations show that  $C_D$  appears more sensitive to the discretization. This may impact the finite-difference gradient and the design trajectory in more sensitive flow regimes when  $C_D$  is part of the objective functional. The results presented here show only a minor dependence on mesh size for the analytic method.

These outcomes shed further light on the need to scrutinize optimization results when employing finite-difference geometry gradients of support points on B-spline curves and surfaces, especially when analysis is sensitive to geometry. In the BWB design case considered here the finite-difference results would have been declared sufficient if knowledge of their inconsistent nature were unknown and an analytic comparison were unavailable. Although the finite-difference approach may suffice for other parts of a model BRep, it is clear that this approach must be used cautiously on support points for B-spline curves and surfaces because a geometry kernel may modify their knot vectors and/or support point parameterizations. This leads to inconsistent finite-difference geometry gradients that may create a bifurcation in design trajectory.



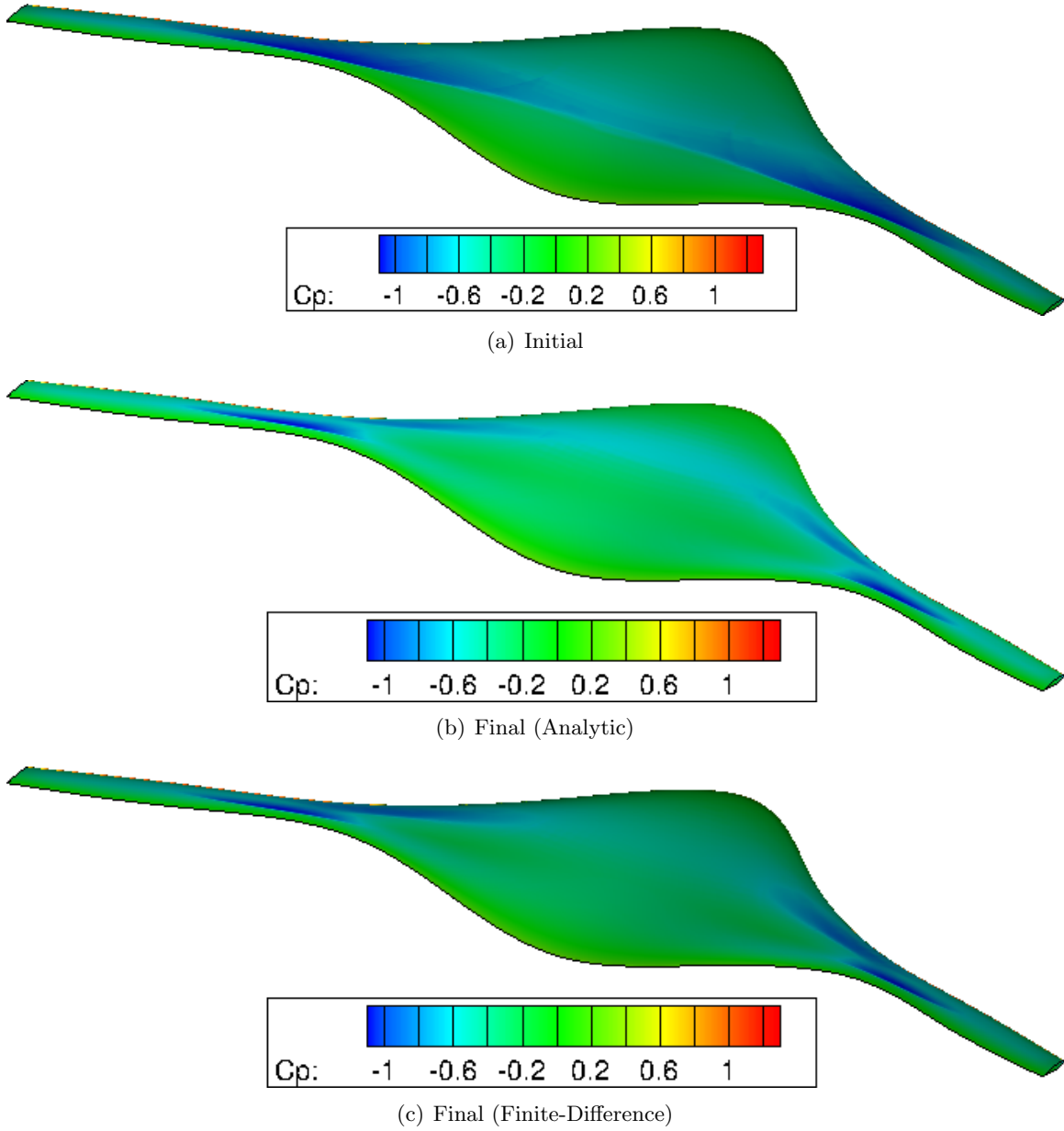


Figure 6-24: Isometric contours of  $C_p$  showing the initial and final distributions obtained with the unconstrained-lift BWB design problem.

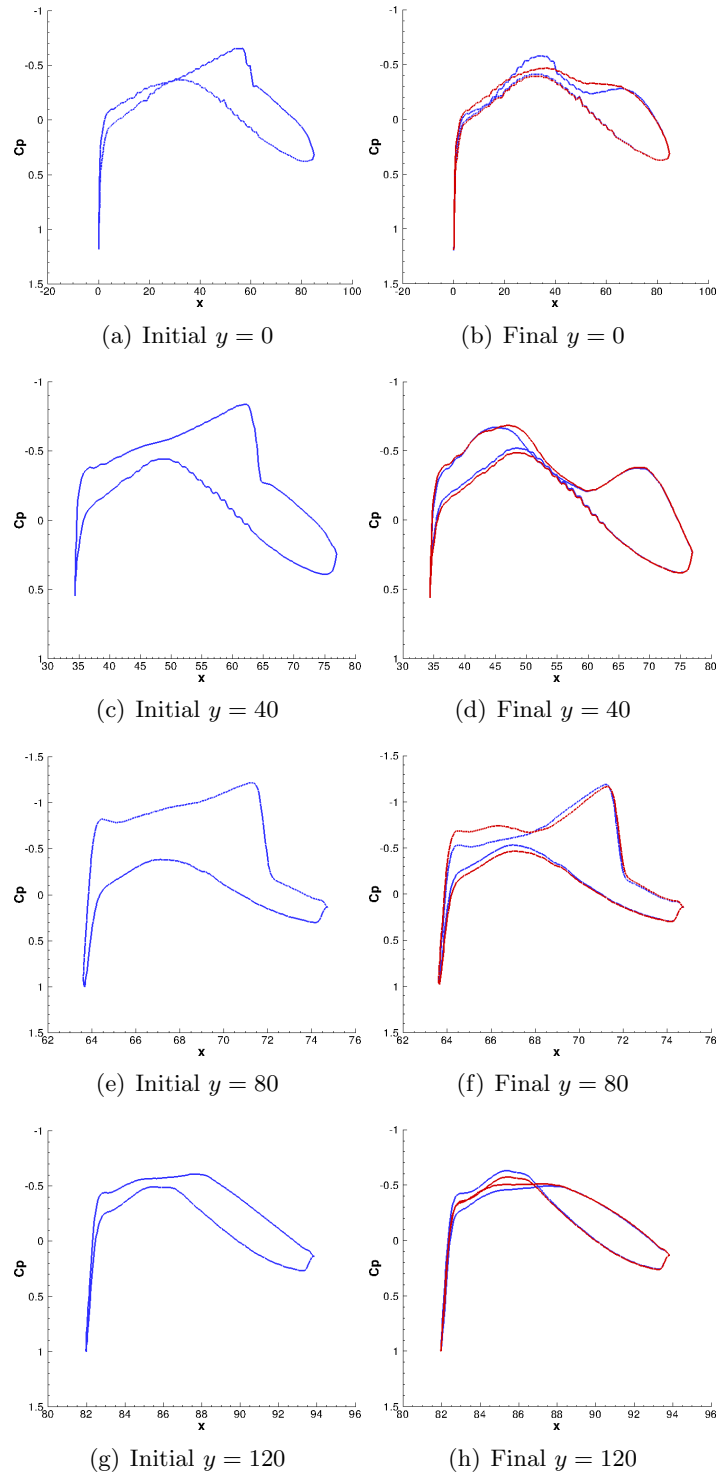
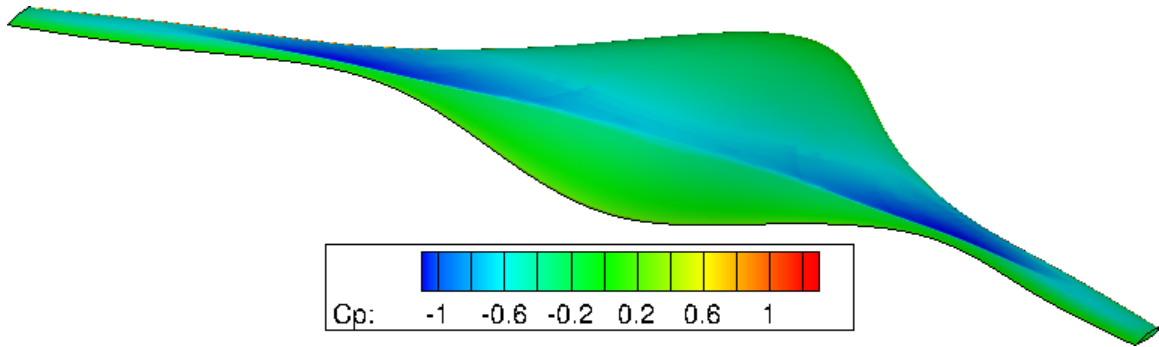
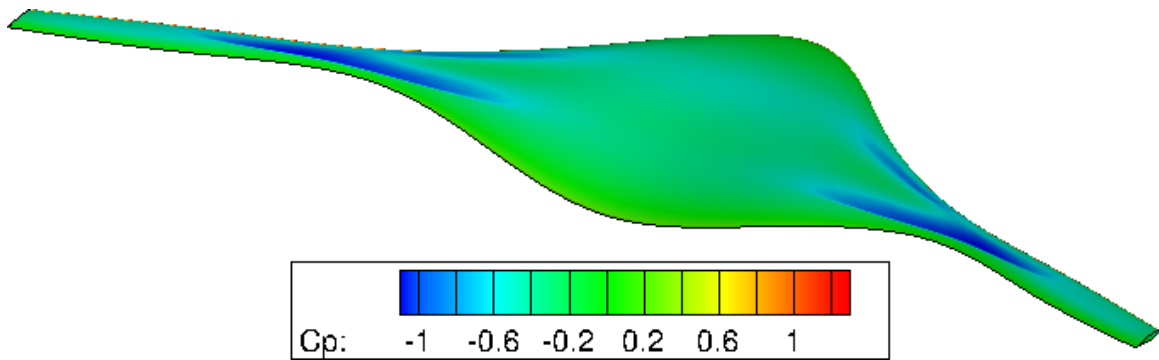


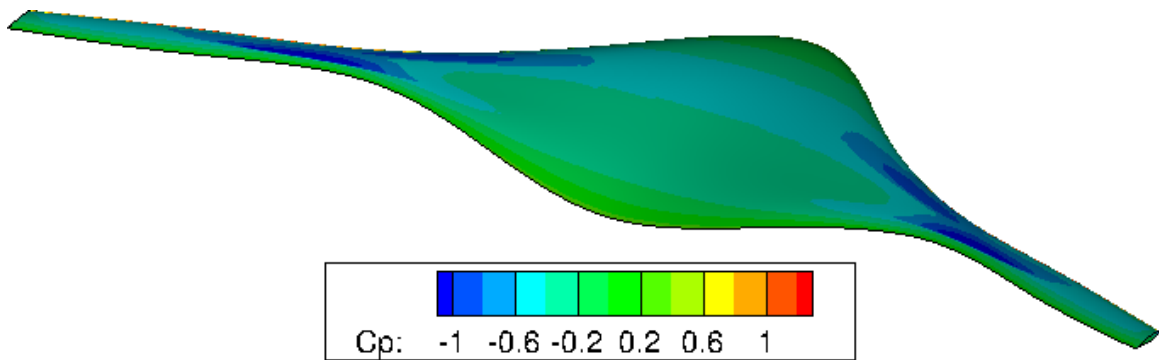
Figure 6-25: Select cross-section  $C_p$  distributions are shown from across the BWB semi-span to compare the impact of initial and final geometry obtained in the unconstrained-lift design problem. Blue denotes results stemming from analytic geometry gradients and Red denotes results using finite-differences. Non-smoothness in the inboard  $C_p$  distributions reflect a locally more coarse mesh spacing.



(a) Initial



(b) Final (Analytic)



(c) Final (Finite-Difference)

Figure 6-26: Isometric contours of  $C_p$  showing the initial and final distributions obtained with the constrained-lift BWB design problem.

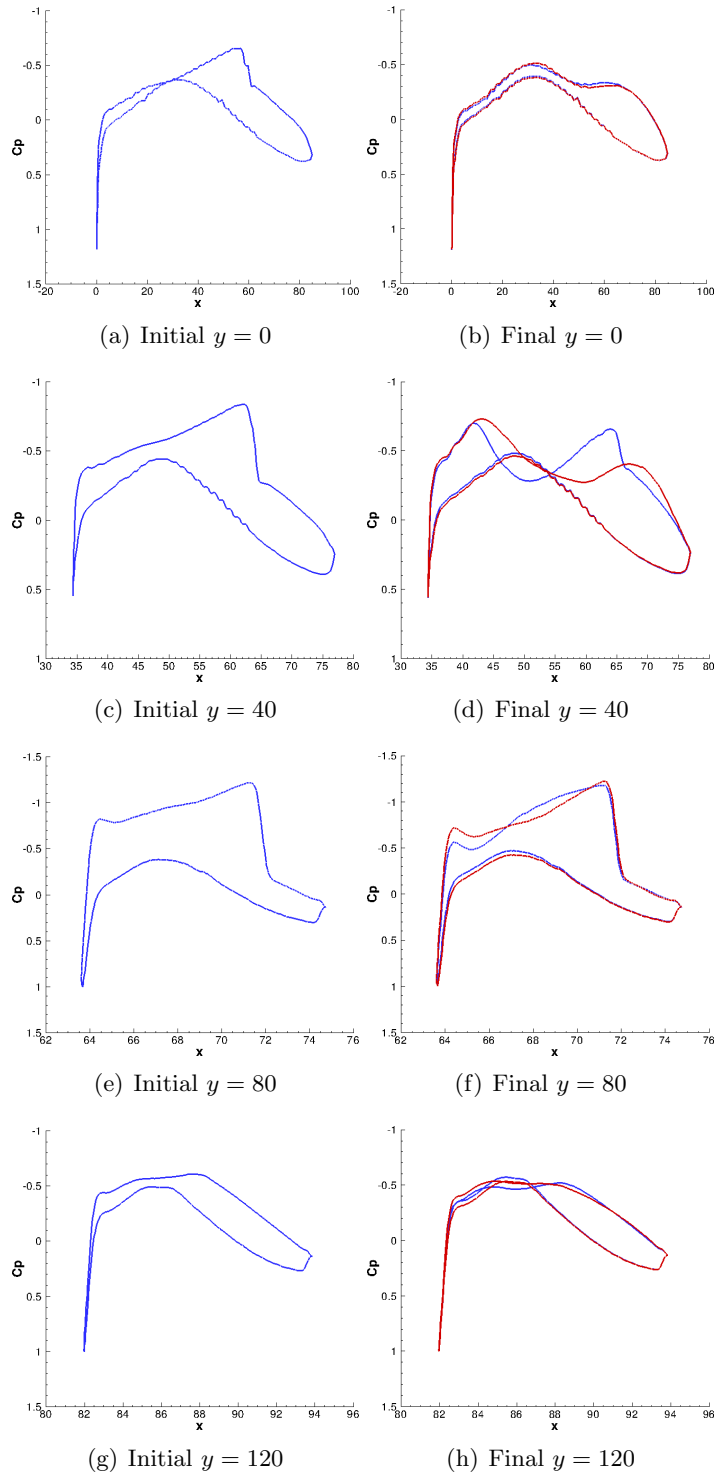
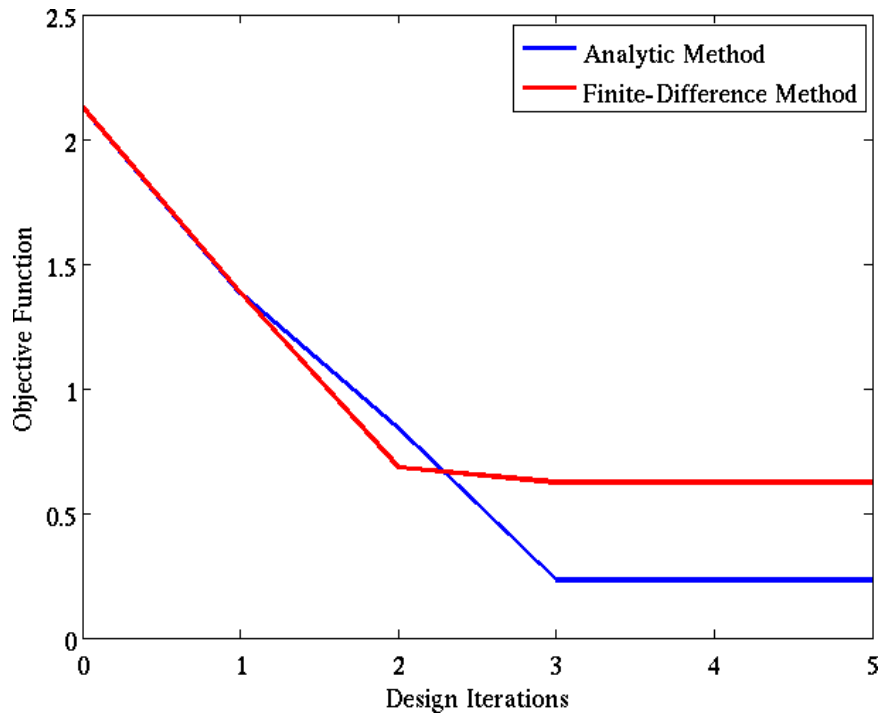
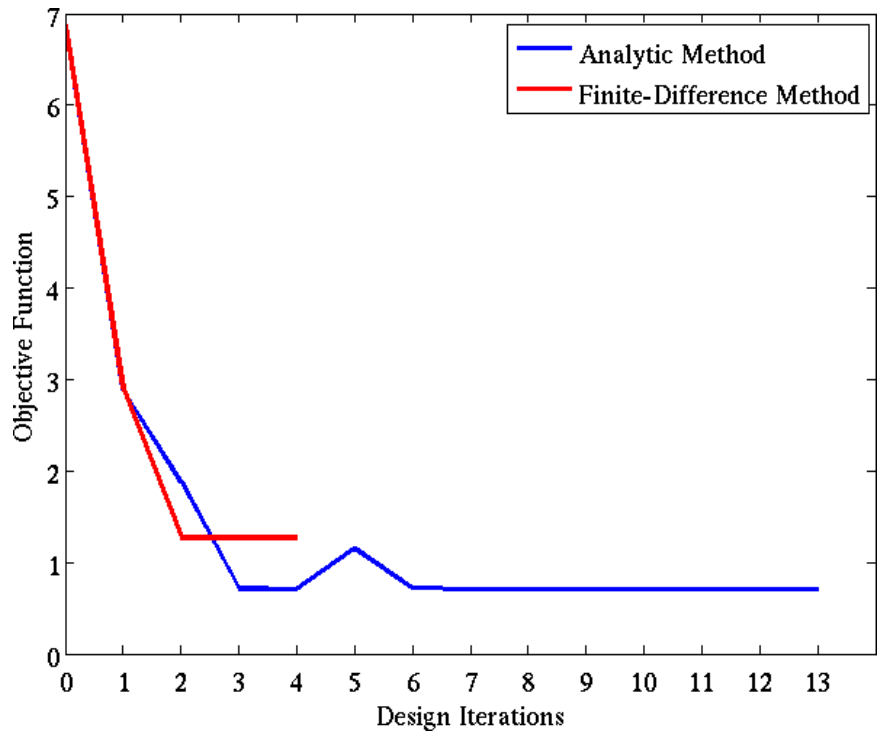


Figure 6-27: Select cross-section  $C_p$  distributions are shown from across the BWB semi-span to compare the impact of initial and final geometry obtained in the constrained-lift design problem. Blue denotes results stemming from analytic geometry gradients and Red denotes results using finite-differences. Non-smoothness in the inboard  $C_p$  distributions reflect a locally more coarse mesh spacing.



(a) Unconstrained-lift Problem



(b) Constrained-lift Problem

Figure 6-28: Comparison of objective function history results stemming from analytic and finite-difference design velocity methods.

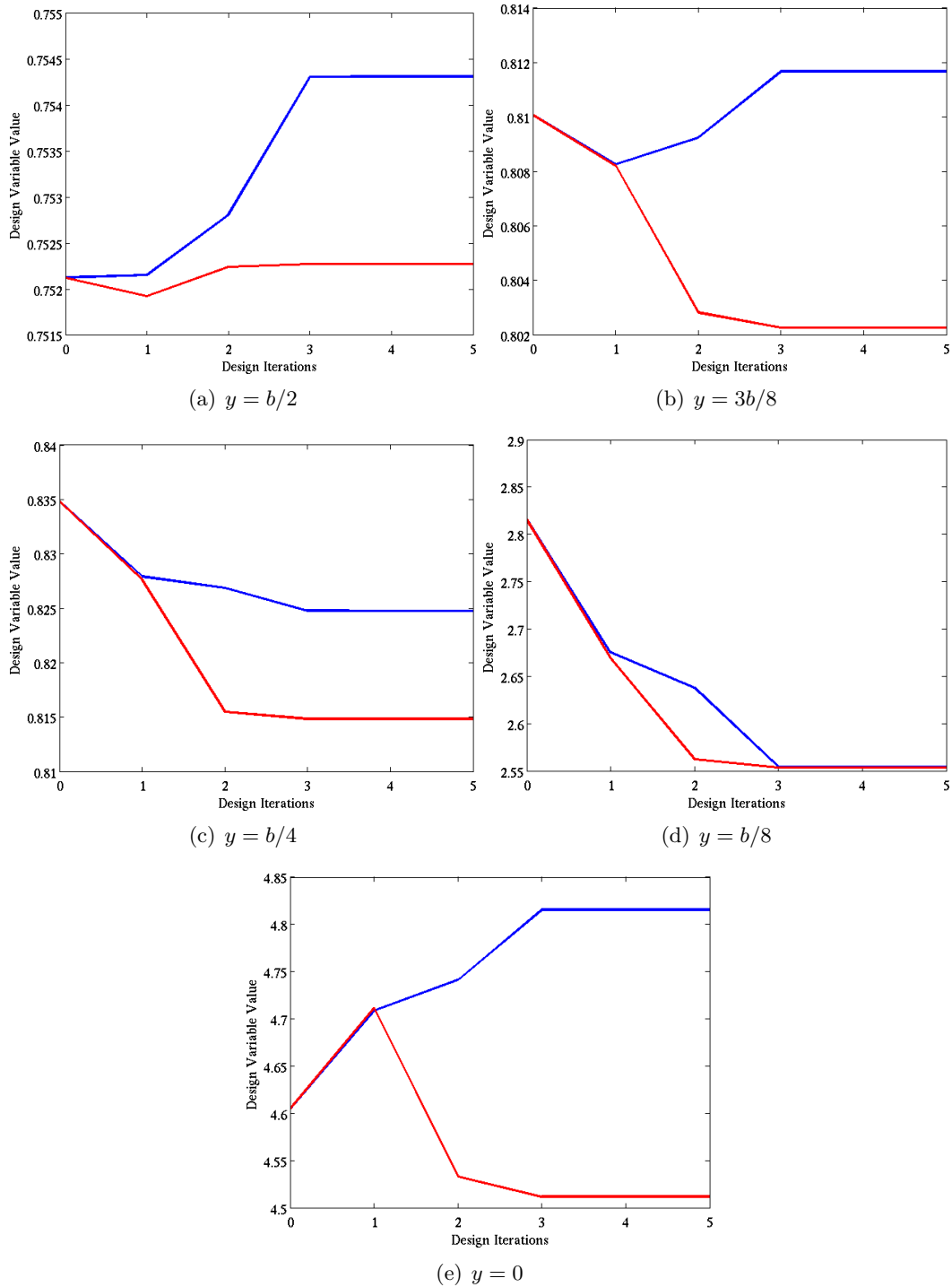


Figure 6-29: The design trajectory of design variable #1 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the unconstrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences.

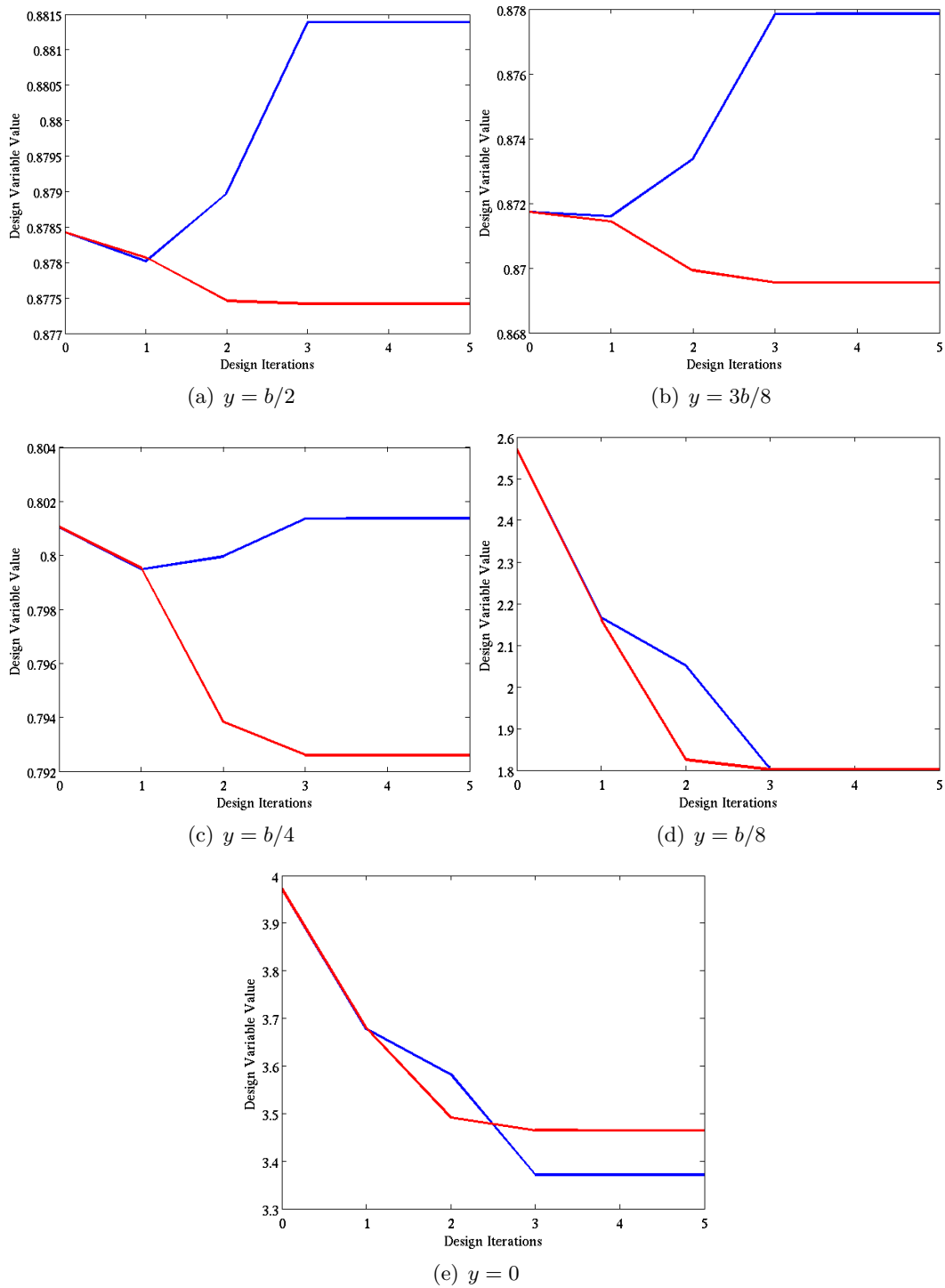


Figure 6-30: The design trajectory of design variable #2 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the unconstrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences.

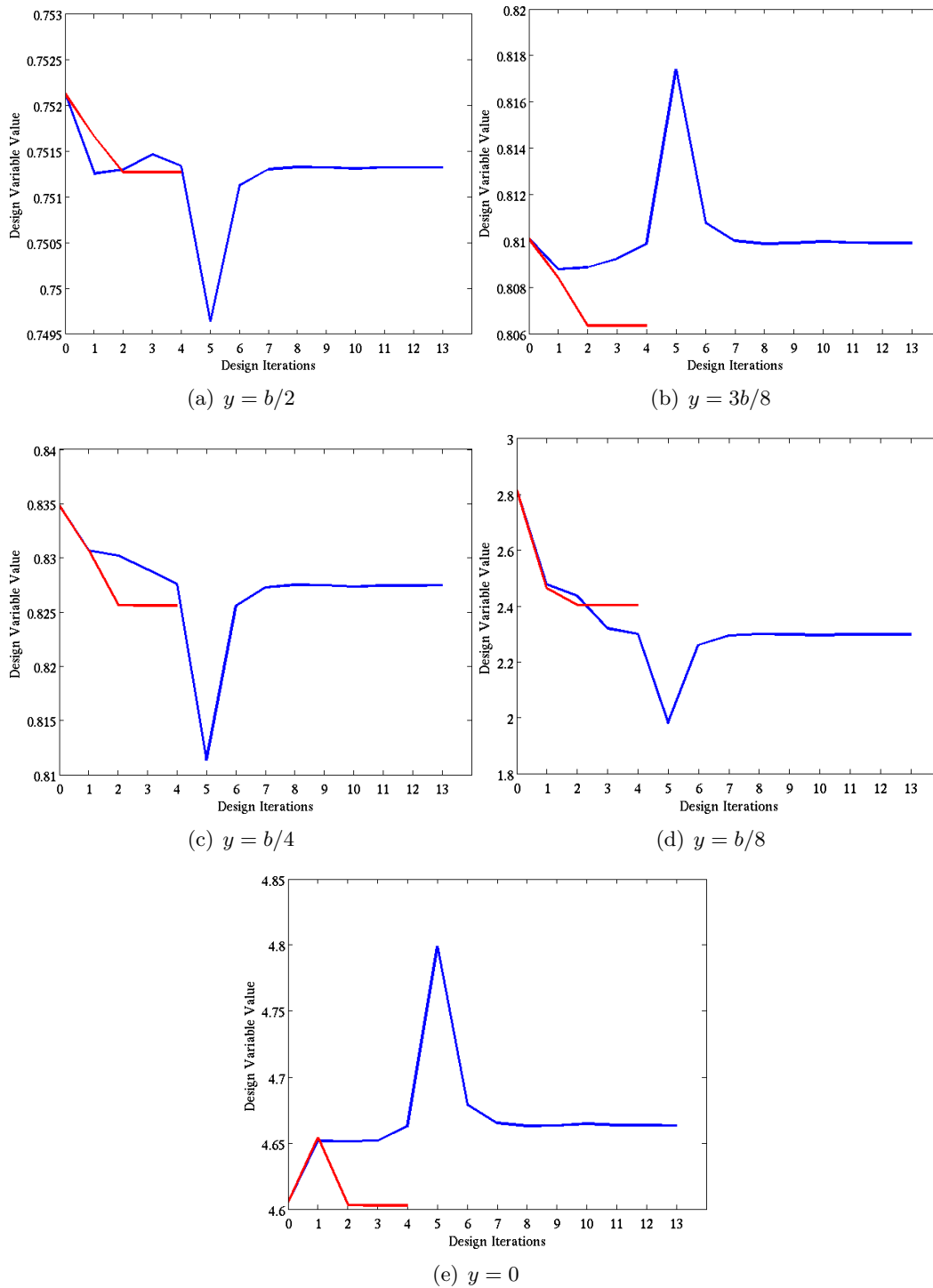


Figure 6-31: The design trajectory of design variable #1 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the constrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences.



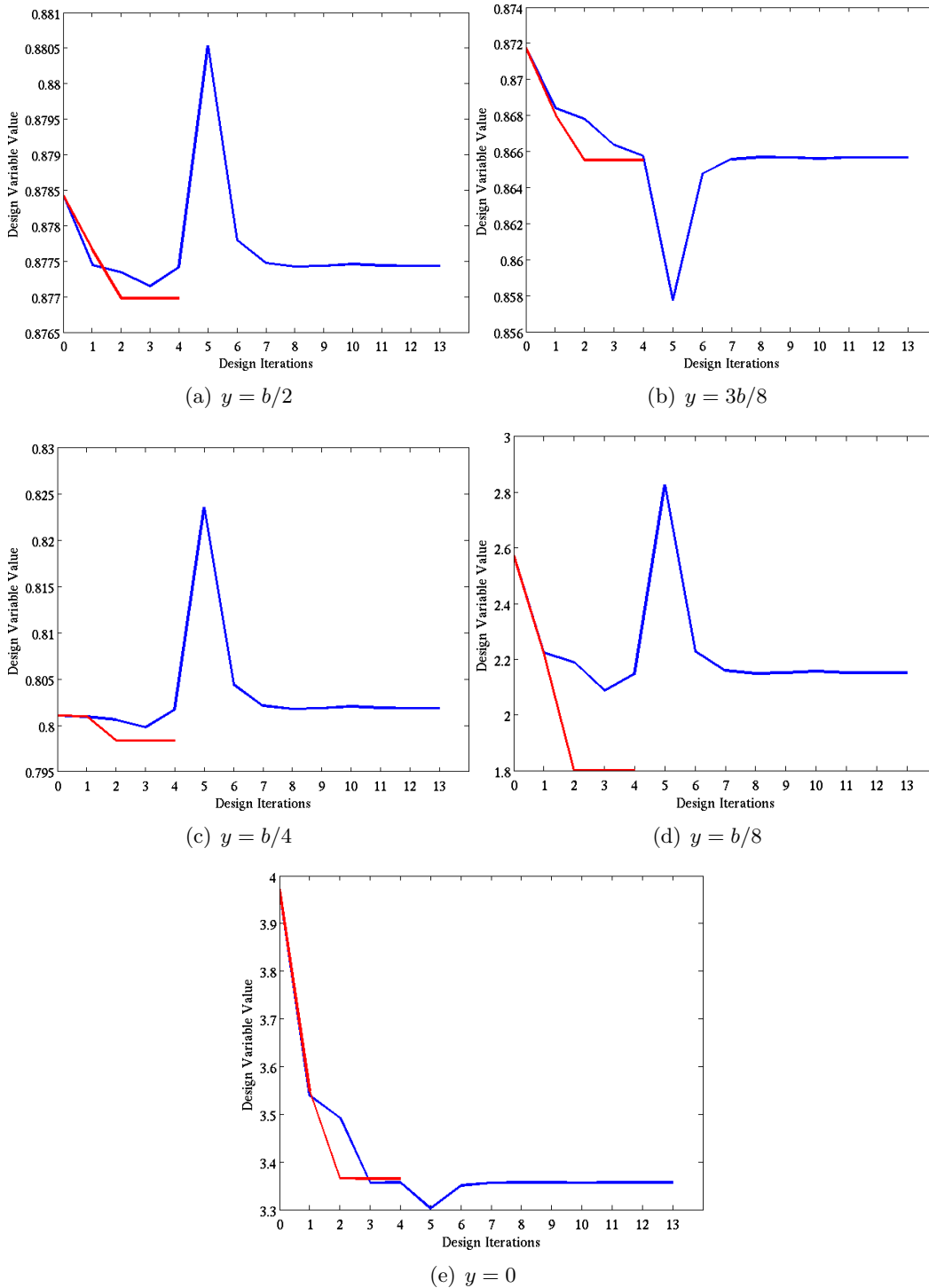


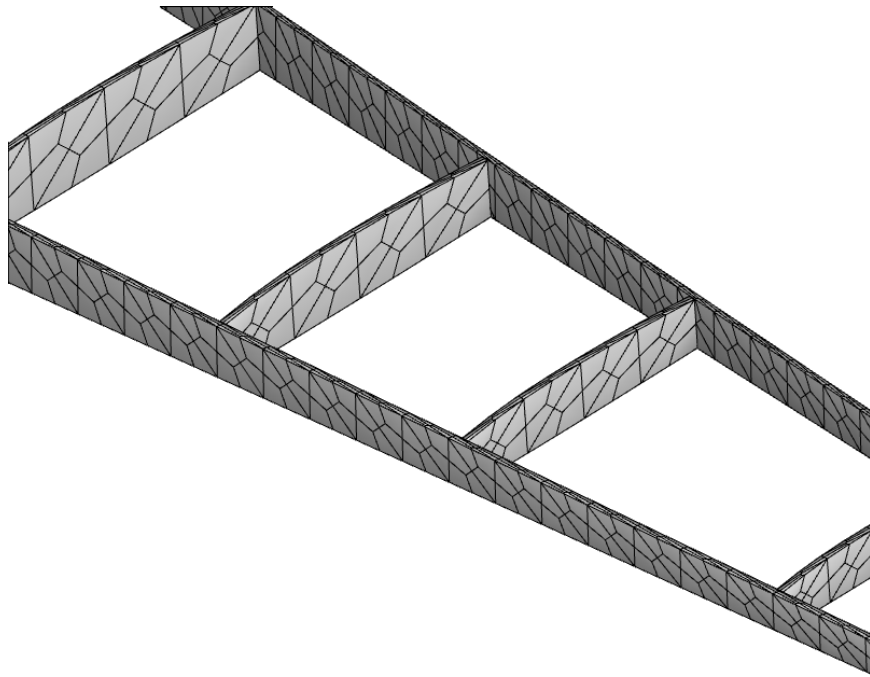
Figure 6-32: The design trajectory of design variable #2 on cross-sections across the BWB semi-span is shown to compare the impact of analytic or finite-difference design velocity in the constrained-lift problem. Blue denotes results stemming from the analytic method and Red denotes results using finite-differences.

## 6.2.6 3D Multidisciplinary Design Space Exploration

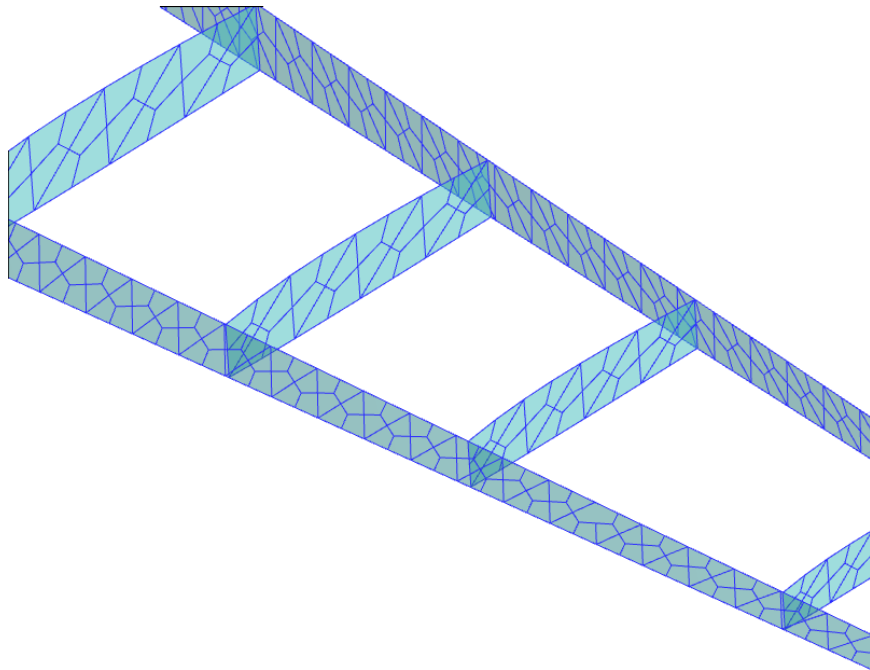
A parametric design space study is presented for wing structural layouts consisting of the wing skin, spars and ribs. Aerodynamics analysis provides the loading for structural analysis and both utilize sub-models extracted from a CAD model. This model is constructed using the principles introduced in Chapter 2. Structural components may be resized, suppressed or un-suppressed to identify various structural layouts within the design motion of the wing. Stresses and displacements throughout the wing structure are evaluated at various design points as well. The selected analysis tools for use in this study include Cart3D for aerodynamics and NASTRAN [73] (a linear and non-linear structural element analysis suite) for structural analysis. As a design space exploration example, this study does not directly include structural or aerodynamic optimization. Instead this is an important step in defining specific optimization problems in promising regions of the global design space that are found via Latin Hypercube sampling.

Both the NASTRAN and Cart3D analyses require a discretized geometry representation of the CAD model components. Multifidelity structural analysis is possible with NASTRAN by importing different quad meshes representing structural elements. High-fidelity analysis is available by utilizing all quad elements on the 3D surface of a structural member, as seen in Figure 6-33(a). Low-fidelity analysis is also an option by employing the quad meshes placed on some faces of the structural members, as shown in Figure 6-33(b). These 2D sub-models are a satisfactory representation of “thin” shell components (meaning that the component cross-section has thickness-to-height ratio much less than 1), where stress distributions are presumed to change substantially along the span of the component and negligibly across its thickness (e.g., spar shear web). The shell thickness is obtained from the 3D model thickness parameter for each structural component.

In this study multiple structural members are analyzed as a structural system, thus the CAD model *assembly* connectivity must be captured by the NASTRAN sub-models. The CAD model structural *assembly* is visualized in Figure 6-34(a). Creating the structural sub-model entails traversing the full structural quad mesh to find the elements on the faces of interest. Another search is done for edges that connect the adjacent structural components and edges that are constrained in some fashion. Given the need for automation, the geometry extraction algorithms are performed in a pre-analysis step. For NASTRAN



(a)



(b)

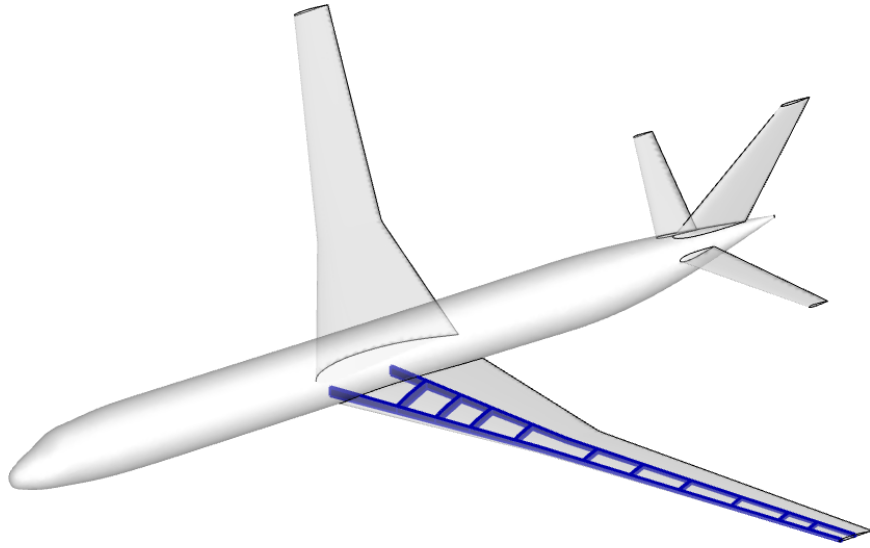
Figure 6-33: The 3D source geometry is tessellated to create (a) a complete quad mesh of the 3D structure surfaces and (b) a sub-model geometry consisting of quad elements from certain faces on the 3D surface. The mesh representing the wing skin is not shown.

input, the components are “welded” by (1) specifying rigid connections between the nodes on connecting component faces and (2) constraining edges connecting the wing and spar root to the fuselage as fixed. Such a definition requires a search for the nearest nodes on adjacent components to create the “weld” connector (connectors are defined for edges-to-face or face-to-face welds). A resulting sub-model *assembly* is seen in Figure 6-34(b), where the top and bottom faces of the wing skin are also extracted. Sub-models for spars are chosen from 3D spar faces that have their normal pointing inside the wing box. Sub-models for ribs are taken from 3D rib faces that have their normal pointing towards the fuselage.

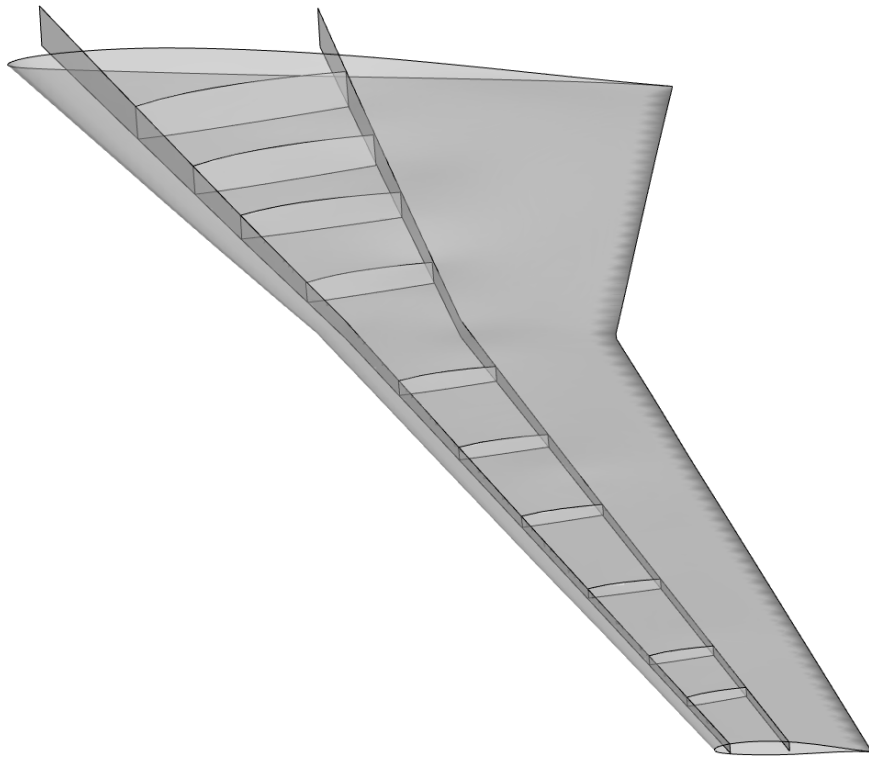
The deflection vectors for the structural sub-model *assembly* are viewed after a structural analysis to determine proper component connectivity (i.e., all components exhibit expected relative displacements). Figure 6-35 illustrates proper sub-model *assembly* connectivity. In this example the structure responds to loading with uniform deflection across the wing span that is consistent with first-principles intuition. If the sub-model connectivity were faulty, the component displacements would violate first-principles (e.g., ribs would not deflect due to missing weld-joints with the wing skin).

Multidisciplinary coupling between disciplinary sub-models can also be modeled intuitively through their parent CAD model. Since the aerodynamic results specify the structural loading condition, the aerodynamic information must be mapped from the aero sub-model to the structural sub-model via their parent geometry. The Cart3D solution contains pressure information at each node of the surface triangulation sub-model. The NASTRAN sub-model for the wing skin, though, is a quad mesh with nodes that do not coincide with the triangulation nodes. Both the triangulation and quad mesh are derived from the same  $(u, v)$  parameterizations of outer mold line faces, thus allowing for interpolation of pressure information in  $uv$ -space. In this case, the pressure at a quad node  $(u, v)_q$  is determined by a linear interpolation of pressure data from the triangle element nodes  $(u_i, v_i)_t$  that contain  $(u, v)_q$ . By doing this for each quad node, an average pressure value could be applied for each quad element (a requirement for NASTRAN loading definitions).

The case in Figure 6-36 is selected to illustrate the results of this multidisciplinary mapping procedure. It depicts a pressure distribution at Mach 0.8 with a lift-coefficient of 0.4. This flow condition simulates a strong loading case that causes substantial deflection (7075-T6 Aluminum is used with typical thickness values for airliners). A strong shock exists at about 70% chord along the wing span, thus creating a 3D, non-uniform loading



(a) Full Model Geometry



(b) Structural Sub-Model

Figure 6-34: The *assembly* connectivity embedded in the full 3D model, seen in (a), is by necessity conserved in (b) the sub-model *assembly* seen by the structural analysis.

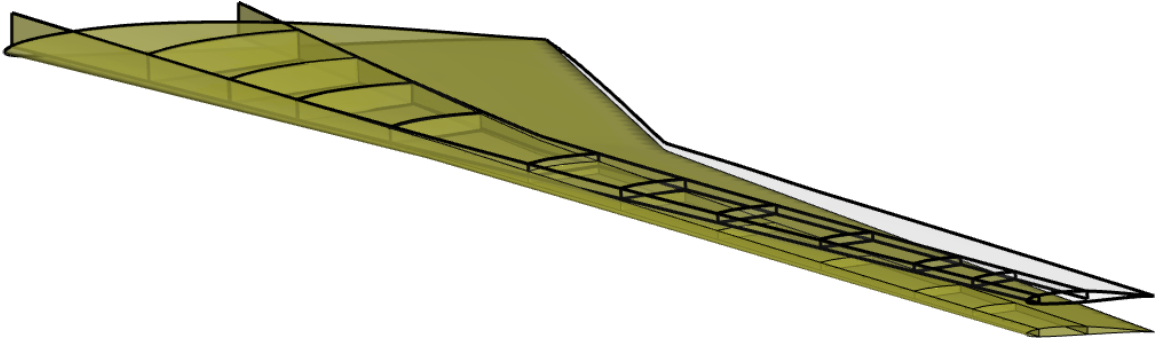


Figure 6-35: An unloaded view of a structural sub-model (dark shade) is overlaid with a deflected sub-model (light shade) under loading to depict proper *assembly* connectivity. Improper connectivity would appear with displacements that violate first-principles.

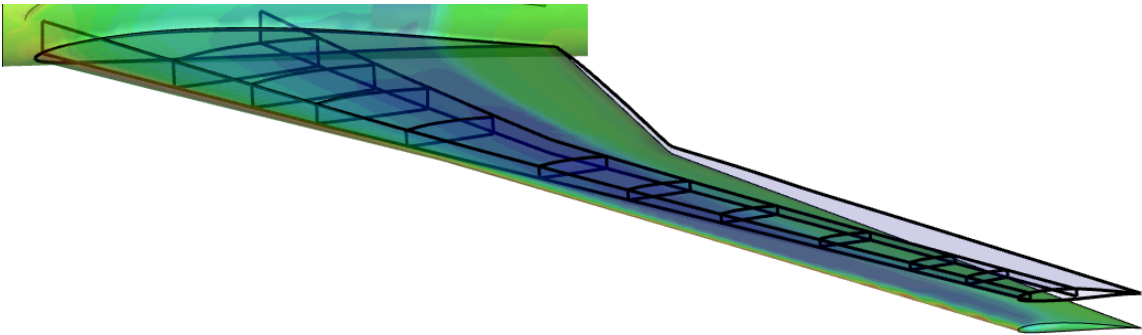


Figure 6-36: Aero-structural coupling is captured by mapping a wing pressure distribution from an aerodynamic sub-model to a structural sub-model. The aerodynamics sub-model (dark shade) shows a wing pressure distribution (Mach 0.8, 0.4 lift-coefficient); the deflected sub-model (light shade) is the static structural response to the pressure loading.

condition that induces bending and torsion loads on the wing skin and internal structure. Deflection of the structural sub-model is also shown in comparison to its initial unloaded state.

An additional comment is needed regarding this mapping procedure. The quad mesh size must be sufficient to adequately resolve the features in the pressure distribution, such as shocks. Otherwise the mapped pressure distribution will “smear” the original pressure distribution and diminish the loading consistency between the two sub-models. Therefore, the quad mesh should be as fine as the surface triangulation to avoid this problem. It is important to also note that the case in Figure 6-36 is for a “static” load. To properly resolve aero/structural coupling, geometry displacement must be sent to the aerodynamics analysis in order to update the flow solution. This is iterated until the difference in displacements is within a user-specified tolerance, meaning the final structural response is consistent with the final pressure distribution. Options for accomplishing this are not pursued here and left for future work efforts.

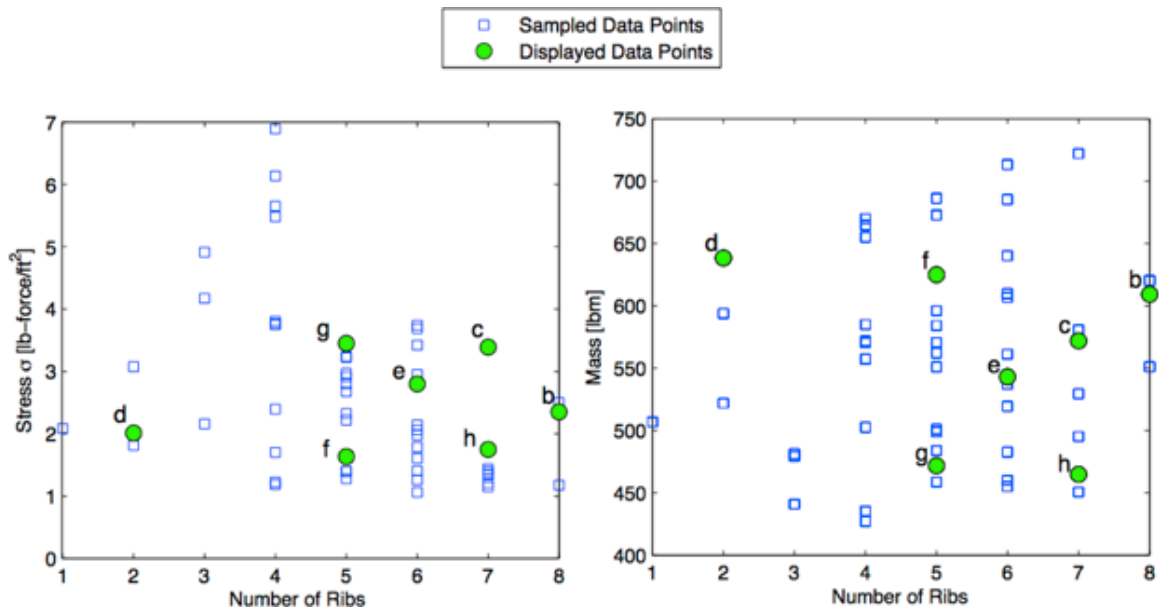


Figure 6-37: Scatter plots of the design space obtained by Latin Hypercube sampling. The labeled points correspond to the structural layouts in Figure 6-38.

The sizing parameters and suppression status for each structural component make up the design variable set. Latin Hypercube sampling of the design space is automated to create a set of design points for analysis. A baseline 3D model is then updated and regenerated at each sampled point to create the new model instances. Since only static loading is

<b>Design Point</b>	<b>Num. of Ribs</b>	<b>Rib Thickness</b> [ft.]	<b>Spar Thickness</b> [ft.]	<b>Skin Thickness</b> [ft.]	<b>Layout Mass</b> [lbm.]	
Figure 6-38	(a)	10	0.145	0.145	0.010	3750
	(b)	8	0.016	0.027	0.020	609
	(c)	7	0.016	0.025	0.018	572
	(d)	2	0.025	0.030	0.023	638
	(e)	6	0.022	0.022	0.020	543
	(f)	5	0.027	0.027	0.022	625
	(g)	5	0.021	0.023	0.016	472
	(h)	7	0.015	0.021	0.023	465

Table 6.7: A listing of rib, spar and wing skin thickness for the design point layouts in Figure 6-38.

considered, a single Cart3D aerodynamics solution provides the loading conditions for the structural analysis using the previously explained mapping procedure. Through the CAPRI API the geometry kernel is queried to obtain volume information for the structural components. The structural mass is then determined using given material densities (Aluminum is used in this case) for the components. Figure 6-37 illustrates a portion of the design points created by Latin Hypercube sampling. The maximum Von Mises stresses and weight for each structural layout are also depicted for comparison.

An excerpt of design point sub-models are depicted in Figure 6-38 to illustrate the variety of geometry obtained in this example by an automated exploration of the design space. Table 6.7 also shows the corresponding component thickness and total layout weight for the design points in 6-38. The baseline structural layout is shown for comparison.

It is clear in this problem that the single CAD model is capable of representing topologically different structural layouts. Since suppression status is a design variable, the CAD model can regenerate across multiple structural design spaces. Each individual design space can serve as the starting point of a structural optimization problem as well. However, we emphasize that if a CAD model is created for one such structural layout, it will not have the capacity to represent other layouts unless a new geometry model is generated each time. As a result of the CAD-based geometry management in this design framework, this issue is successfully circumvented with a single CAD model.



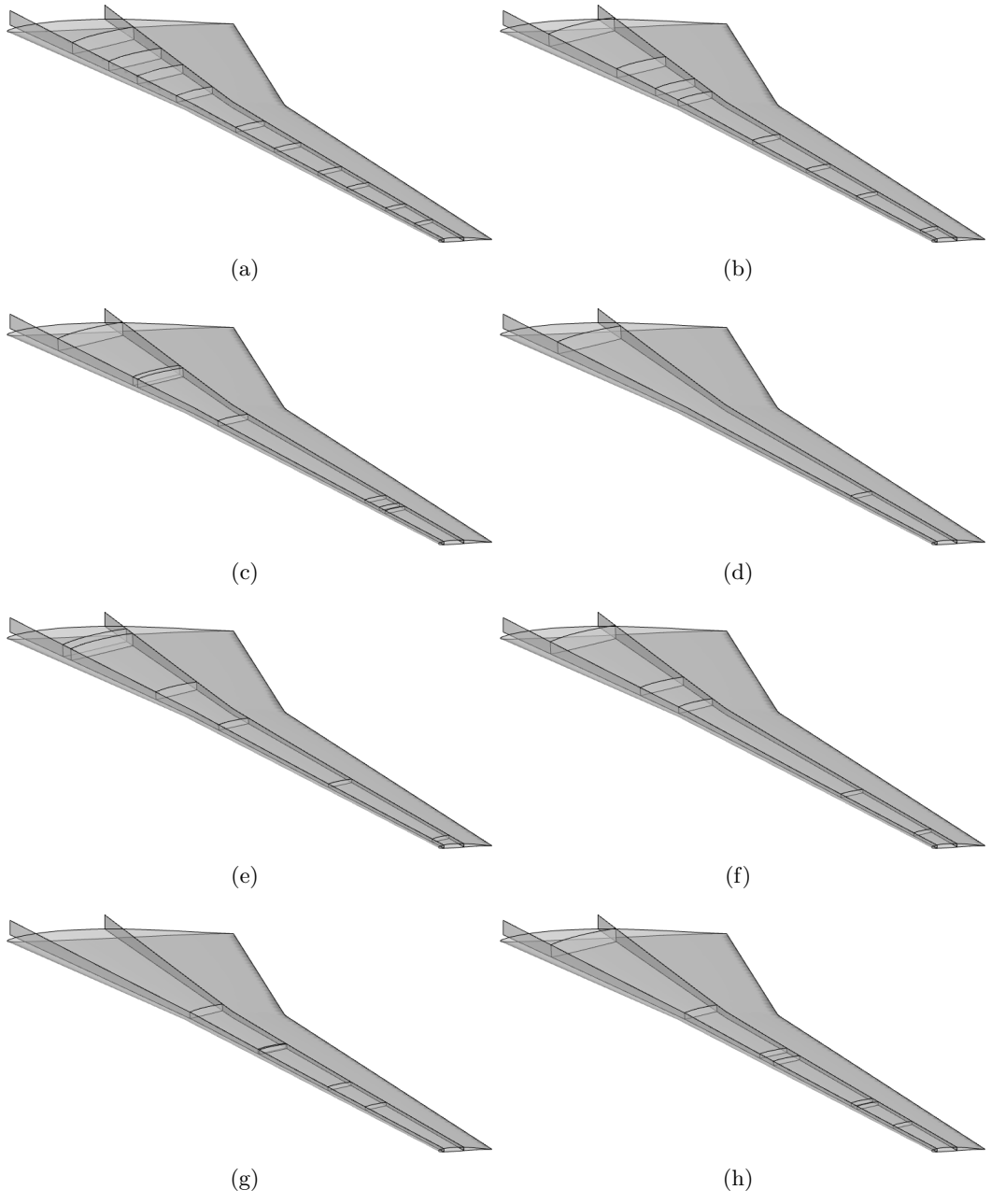


Figure 6-38: (a) A baseline structural layout is compared to various design points (b)–(h) obtained by automated Latin Hypercube sampling.

THIS PAGE INTENTIONALLY LEFT BLANK

## Chapter 7

# Conclusions and Future Work

In Chapter 1 the application of CAD models in aircraft design optimization was introduced and many limitations were discussed that inhibit their widespread implementation. The need to bridge existing computational geometry tools with aerospace design needs became apparent since CAD systems are not explicitly designed for usage in automated design frameworks. Two particular concerns were described as the focal point for advancing the state-of-the-art in CAD-based geometry management in this thesis, namely model construction methods and shape sensitivity methods. These were addressed with solutions using theory and example demonstrations that further enable the use of CAD models in aircraft design optimization.

Three specific research objectives are fulfilled by the thesis contributions presented herein, namely:

1. Create a methodology for CAD-generated model geometry that is suitable for a conceptual design setting.
2. Create an analytic formulation for geometry gradients of CAD-generated model geometry.
3. Demonstrate the implementation of CAD-generated model geometry and analytic geometry gradients within design frameworks to demonstrate the effectiveness of each contribution above.

New perspectives on CAD-generated model geometry are presented in Chapter 2 with notions of multifidelity/multidisciplinary geometry and design motion. With these guiding

principles a formal definition of design intent is given that is more suitable for CAD models used in aircraft conceptual design. Along with suggested construction methods, these notions bring to light the importance of creating CAD-based models that are sufficiently flexible, malleable and robust in regeneration to support automated design optimization.

In the realm of geometry gradients, Chapter 3 presents the “reverse engineering” of highly utilized *features* in a CAD system to provide the analytic design velocity with respect to any sketch dimension for extrusions, revolutions and sweeps. Chapter 4 presents analytic approximations to the design velocity along trim curves and nodes in a BRep to resolve the issues associated with conventional approaches. Chapter 5 shows other analytic methods to differentiate B-spline curves and surfaces for loft *features* as well.

An added contribution in Chapter 5 is made in discovering an error in finite-difference methods applied to B-spline curves and surfaces when linearizing support points. Although this method is still appropriate when differentiating other entities in a model geometry BRep, an error can arise when a geometry kernel regenerates B-spline curves and surfaces because their parameterization may change. This inconsistency is explained by a presentation of theory and examples. Due to the prevalent usage of this gradient method in the literature, this finding calls for an awareness to scrutinize design trajectory results when varying B-spline support points in a regime of strong physics sensitivity to geometry. In these cases the finite-difference gradient error may lead to less-desirable design trajectories compared to those found with analytic gradients.

The sensitivity methods presented in this thesis are applicable for any CAD system if the parameterizations are equivalent to those found here (which correlate with the SolidWorks CAD system). An extension to higher-order derivatives is also possible if those derivatives exist for a given parameterization. However, if the parameterizations are different, the overall algorithms are still applicable given the new parameterization information is used instead. As mentioned in the future-work discussion below, verification of the presented parameterizations is needed for other CAD systems.

Finally, various design framework implementations of CAD models are demonstrated in Chapter 6 using the analytic geometry gradients presented in this work. These serve to validate the geometry gradient algorithms in both inverse and forward design problems. In comparison to finite-difference methods, the analytic approach also demonstrates improved results, both in computational time and in some cases even with improved design trajectory.

To summarize, the scope of this project was narrowed to the geometry perspective of design because it is an evolving field that impacts how effectively CAD models are used in aircraft conceptual design. With the current state of design tools, geometry modelers and design methodologies, these contributions will further efforts towards a seamless aircraft design process that is better suited for conventional and unconventional designs of the future.

Further work is possible to enable widespread application of CAD models in aircraft conceptual design. As new design frameworks continue to apply CAD systems in their geometry management, ample work remains to fully “reverse engineer” other *features* and constraints in order to obtain analytic geometry gradients properly. There is also a need to automatically create an associativity table between design parameters in the master-model and the model BRep. With this development the connectivity between parameters, sketch entities and surfaces is required to generate the differentiation code automatically. These efforts can lead to automated pre-processing of CAD models and thereby free designers from deciphering the CAD system internals. Otherwise, designers need to “reverse-engineer” the CAD model components and construct their sensitivity formulations manually. The additional option also exists to develop new CAD systems that are suitable for automated design optimization frameworks. Unlike the systems available today, a new system can internally incorporate all necessary information for analytic geometry gradients and generate models with desired design motion characteristics. The innovative work presented in this thesis sets the stage for these additional developments, thus making more feasible the ability to streamline aircraft design with a new paradigm in CAD-based geometry management.

THIS PAGE INTENTIONALLY LEFT BLANK

# Appendix A

## CAD Model Generation

A general procedure for developing CAD-generated models is not given here because it is beyond the scope of this work. Such an endeavor is intractable due to a large number of possible design scenarios, model uses, etc. Instead, an approach is detailed that is most relevant for representing aircraft models in a conceptual design setting. This discussion begins with preliminary considerations of CAD system features that will impact the usefulness of the model in an automated design setting. Secondly, a methodology for model construction is explained that provides flexibility, robustness and malleability for use in those design settings as well.

### A.1 Model Feature Considerations

Despite the many ways to generate a model geometry in a CAD system, a detailed look at the effects of *feature* construction on model design motion and model robustness are important to make the model suitable for automated design optimization. Loft *features* are discussed in particular due to their common usage in aircraft design.

#### A.1.1 Perspectives on Generating Loft Features

Most CAD systems support 3D loft features, which are typically cubic B-spline surfaces which interpolate cross-section *primitives*. The CAD geometry kernel ensures these features are “closed” (i.e., watertight) and likewise acceptable for automated tessellation. These features are a natural choice for representing aircraft lifting-surfaces. However, care must be taken in how these features are constructed in order to maintain a completely watertight

model.

Two possible options for setting up a loft feature are:

1. Define a datum plane that is offset and parallel to a plane defined by the global Cartesian axes, such as the  $xy$ -plane. Constrain a spline *primitive* to remain fixed at a predetermined location on its sketch plane. This anchored point can be used to define the spline location with respect to the global origin.
2. Define a spline with 3D points on which all desired sketch planes will be coincident. Create a datum plane that is normal to the 3D-spline at a desired spline point (also referred to as an “anchor point”) and ensure that this point becomes the origin of the datum plane. After ensuring the datum plane sketch axes are in the proper orientation, sketch and constrain a 2D spline *primitive* with a support point coincident to this datum plane origin.

In the first case, as seen in Figure A-1(a), the cross-section spline has a design motion that is limited to translation, rotation or scaling along its sketch datum plane without any out-of-plane motion. The datum plane definition becomes the hindrance if the representation of an out-of-plane geometry mode is desired. To provide the desired orientation for the datum plane, additional datum lines and curves will be needed. These also require angular dimensioning to recreate Euler angles and orient the datum plane in the desired manner. The process for doing this becomes cumbersome and iterative due to the need for careful location of additional datum references and dimensioning. Furthermore, this approach could reduce regeneration robustness due to greater coupling between datum references.

In the second case, as seen in Figure A-1(b), a simpler approach is taken. The spline *primitive* resides on a datum plane that is not constrained to the global Cartesian planes. Instead, the 3D datum spline orients the datum plane passing normal to a tangent vector at the anchor point, which also acts as the normal to the plane. This tangent vector can essentially point anywhere and the airfoil sketch will be oriented as desired. This apparent out-of-plane design motion capability is inherent to the sketch plane and not the spline *primitive* itself. This approach is beneficial, for example, in the case where the initial datum spline represents a structural bending axis on a wing. The undeformed state might have airfoil sections parallel to the plane of symmetry of the wing, yet the deformed state will appear to have airfoil sections translated and rotated out of their initial plane.



Lifting-surfaces are often designated a single sweep angle and taper ratio because the planform is modeled as a trapezoid geometry. These two parameters, though, are a simple case of a lifting-surface with multiple sweep angles and panel taper ratios. In order to represent the higher-dimensional design space, a lifting-surface loft may be constructed with piecewise continuous datum splines. Here a discontinuous derivative exists at the spline end-points instead of creating a single datum spline *primitive* that enforces continuous derivatives at each support point. A potential benefit of this construction is greater loft shape control between the cross-section sketches. Such an option may be useful for modeling a wing break, where dihedral may be discontinuous.

Caution must be had if the piecewise datum spline approach is taken. When piecewise continuous datum splines are used the merged loft features may create splinter faces or trim curves at the location of discontinuous derivative. For example, if a single loft is created between three cross-section sketches, as seen in Figure A-3(a), no splinter faces or trim curves are created. However, if the “thin” attribute is selected for the loft (meaning the loft volume is hollowed-out to within a specified wall thickness), then splinter trim curves are created, as shown in Figure A-3(b), on the outside or the inside of the loft (depending on which direction the thin-attribute was applied). These BRep topology artifacts can be problematic in automatic tessellation because they are small-scale features that a mesher will attempt to tessellate. Tessellation of the “outer” loft surface may be improved by keeping splinter trim curves along the inner loft faces, yet the designer has little to no control of how the CAD geometry kernel determines the location of these BRep entities.

If two separate loft features are created between three cross-section sketches using piecewise continuous datum splines, the thin-attribute will create splinter faces. Even though two loft features are given the same thickness value, the discontinuous derivative at their intersection causes one loft feature to be tangent to the end-point derivative of the other. Figure A-3(c) shows that the inner, or outer, face of one loft is extended beyond the normal to its own spline end-point until it is tangent to the other loft. Therefore, one loft uses the derivative on the *wrong* side of the merge-point. This scenario cannot be avoided when choosing to merge two loft features instead of creating a single loft feature. Lastly, if each loft were created with its sketch profiles normal to the end-point derivatives of their respective datum spline (without being merged), the two lofts would intersect and not result in a single closed volume, as seen in Figure A-3(d).

### A.1.2 Mirroring Features

Many conventional CAD model construction methodologies rely on the inherent symmetry of aircraft components to simplify the construction recipe. There are examples of asymmetric design motion, though, that the model needs to comply with in order to represent additional geometry modes. For example, asymmetric gust loading yields a lifting-surface response that is also asymmetric. Figure A-4(a) illustrates an asymmetric model, whereas Figure A-4(b) demonstrates the design motion limitation of a similar model constructed using a mirrored loft feature. Although easier to construct, the model in Figure A-4(b) *cannot* represent certain geometry modes due to design motion rigidity embedded in its construction. If such geometry modes are not meant to be represented, then the mirror feature will suffice. Prudence in the model construction depends heavily on the a posteriori application of the model.

### A.1.3 Loft Self-Intersection

Self-intersection of loft surfaces becomes an issue if the loft construction has insufficient number of cross-sections across its span, as seen in Figure A-5. “Guiding datum curves” (i.e., guide curves) that span a loft through the point of maximum thickness (top and bottom) on each sketch provide some protection against self-intersection. However, these only constrain a narrow band of the loft surface. If the design motion of anchor points move the cross-sections further apart at some point in a design trajectory, the loft may intersect the top and bottom surfaces. This may be unavoidable at certain design points because the loft is a cubic interpolation that may over/undershoot between its defining cross-sections. In the case of lifting-surfaces, this is often an issue when increasing the loft span between a thick airfoil section to a thin airfoil section. In addition, when the thin-attribute is applied to the inside of a loft the design trajectory may be hindered in regions of minimal thickness along the wing span.

## A.2 Loft Definitions for Lifting-Surfaces

In attempting to control the orientation of loft cross-section sketches in Euclidean space, the concept of “anchor points” can provide sufficient location information without excessive dimensioning of the sketch to a global origin. In some CAD systems, the processing time

for a model may increase significantly with an increase in dimensioned *primitives* because the master-model equation set requires more time to solve (see Section 6.1.5). The anchor points may also serve as a sketch origin for orientation and placement of loft cross-section sketches.

The datum spline mentioned above may refer to the wing quarter-chord line, the wing leading or trailing edge. Depending on its usage, the datum spline (or linear segments) passes through the anchor points<sup>1</sup>, which may be spaced at irregular intervals along a straight line or a space-curve in  $\mathbb{R}^3$ . Regardless of their starting location, the anchor points can be mapped in such a way to satisfy orientation requirements, such as sweep and dihedral angle, for each loft segment across its span. This perspective is relevant to the construction of lofts representing lifting-surfaces.

Given a set of anchor points  $\{\mathbf{r}_i\} \in \mathbb{R}^3$  (where  $\mathbf{r}_i = [x_i, y_i, z_i]^T$  for  $i = 1, 2, \dots$ ) that are written in sequence from one end of a datum curve to the other, the following linear transformation is possible when orienting the vectors  $\mathbf{r}_i - \mathbf{r}_{i-1}$ :

$$\mathbf{r}_{i,new} = \mathbf{T}(\Lambda_{i,new})\mathbf{T}(\delta_{i,new})\mathbf{T}(\delta_{i,old})^{-1}\mathbf{T}(\Lambda_{i,old})^{-1}(\mathbf{r}_{i,old} - \mathbf{r}_{i-1,old}) + \mathbf{r}_{i-1,new}, \quad (\text{A.1})$$

where the rotation matrices are

$$\mathbf{T}(\Lambda_i) = \begin{bmatrix} \cos(\Lambda_i) & \sin(\Lambda_i) & 0 \\ -\sin(\Lambda_i) & \cos(\Lambda_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (\text{A.2})$$

$$\mathbf{T}(\delta_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta_i) & -\sin(\delta_i) \\ 0 & \sin(\delta_i) & \cos(\delta_i) \end{bmatrix}. \quad (\text{A.3})$$

The “new” and “old” subscripts correspond to the desired and original orientation, respectively, of a reference line passing through anchor points. Since only the original anchor

---

<sup>1</sup>In some CAD systems it is important that all anchor points be explicitly coincident to the datum spline. Otherwise, regeneration will not create the desired design motion of all components in a loft. These coincidence constraints may need to be manually set even after generating the datum spline through the anchor points.

point locations are known, the orientation of each segment  $\mathbf{r}_i - \mathbf{r}_{i-1}$  is determined by:

$$\cos(\Lambda_i) = \frac{y_i - y_{i-1}}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}} \quad (\text{A.4})$$

$$\cos(\delta_i) = \sqrt{\frac{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}}. \quad (\text{A.5})$$

These orientation angles correspond to  $\Lambda_{i,old}$  and  $\delta_{i,old}$ , whereas the desired orientation angles are substituted into  $\Lambda_{i,new}$  and  $\delta_{i,new}$  in (A.1).

For lofts representing lifting-surfaces, the common orientation angles of each loft segment corresponds to sweep and dihedral angles. When the usual aircraft body coordinate system is employed (i.e., the  $+y$  axis points out the right wing in planform view,  $+x$  axis points from the aircraft nose to tail,  $+z$  axis points normal to the  $xy$  plane), positive sweep  $\Lambda$  is measured on the  $xy$  plane as the clockwise rotation of a vector about the  $z$  axis (planform view). Positive dihedral  $\delta$  is measured in the  $yz$  plane by clockwise rotation of a vector about the  $z$  axis (front view).

An example of this lifting-surface construction methodology is conducted for an arbitrary flying-wing *configuration*, as seen in Figures A-6 through A-8. First, a continuous spline is created through anchor points across the wing span. These anchor points serve as the sketch origin for each airfoil sketch plane, which are constrained normal to the datum spline at each anchor point. The generic airfoil sketches consist of two continuous splines (top and bottom) created through three spline points each (leading edge, max thickness point, and trailing edge). Tangency control and proportional sizing (meaning the spline *primitives* are scaled by a constant) are chosen as additional simple parameterizations for these airfoils. Chord length, airfoil tangency conditions and airfoil thickness vary across the wing span in a symmetric manner across the wing centerline. No explicit mirroring of *features* or *primitives* is used.

An input set of anchor points, arranged sequentially from the centerline to the wing tip, are mapped according to Equation (A.1). The final desired sweep and dihedral angles are (in degrees):

$$\Lambda_i \in \{35, 35, 35, 35\},$$

$$\delta_i \in \{0, 15, 0, -2\},$$

where each angle value pertains to wing segment  $i = 1, \dots, 4$ . The local wing segments are rotated with respect to the global  $z$  (for sweep) and  $x$  (for dihedral) axes by using the new anchor coordinates obtained in Equation (A.1).

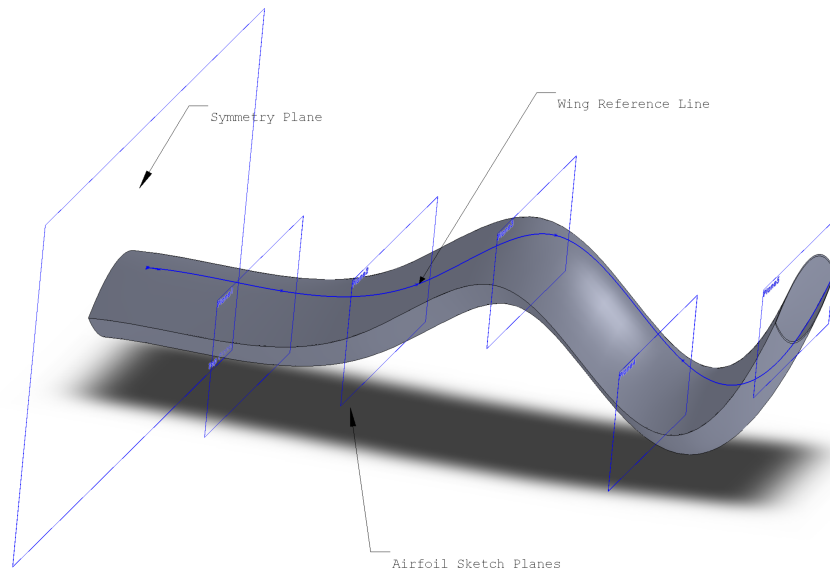
The flying wing images in Figures A-6 through A-8 compare the original model flying-wing to the mapped regenerated model. Since Equation (A.1) is an affine transformation, the Euclidean distance between anchor points is preserved. Figures A-6(a), A-7(a) A-8(a) highlight the initial and mapped anchor point reference line from different viewing perspectives, whereas Figures A-6(b)–(c), A-7(b)–(c) and A-8(b)–(c) show the model response from various views.

The intended mapping of the flying wing *configuration* is accomplished with airfoil sketches remaining normal to the leading edge reference line. In some cases this may be permitted in order for an airfoil section to “see” a component of the freestream flow velocity (in other cases the airfoil section is defined parallel to the aircraft centerline). Another observation in Figure A-6 (c) is that the leading edge is non-linear near the centerline. This occurs because the leading edge is created as a spline with continuous derivatives at the anchor points. The trailing edge is constructed with piecewise linear segments that exhibit no “rounding” as a result of discontinuous derivatives at its anchor points (which correspond to the airfoil trailing edge point). This is another example of a non-intuitive design motion of loft surfaces if a designer is careless in their loft construction approach. Unless the loft construction features are properly selected, this design motion may be problematic due to the leading edge rounding or the top-surface folding seen near the centerline trailing edge (Figure A-8(c)). Lastly, the orientation of airfoil sketches may need to remain parallel to the model centerline, hence constraining their design motion to remain normal to the leading edge may also be an issue if it is not desired.

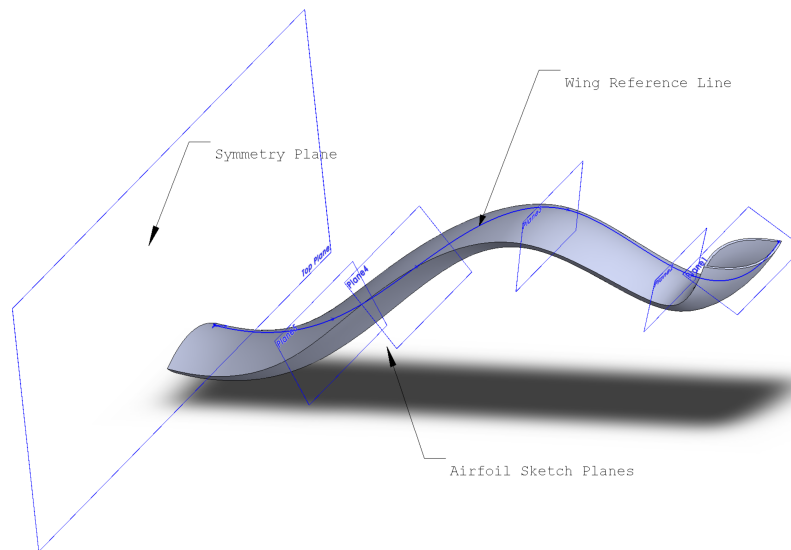
To overcome these potential issues, the datum spline and anchor points can be constrained along the  $y$ -direction, thus allowing the cross-sections to remain parallel to the aircraft centerline. The sweep and dihedral angles are preserved by considering the leading edge of a spline *primitive* as pseudo-anchor points. The relative horizontal/vertical location of this pseudo-anchor point on the sketch plane will result in the appropriate model sweep and dihedral (due to the constraint along  $y$ , this mapping will not be affine).

Figure A-9 shows a three-view of a trapezoidal-planform wing with NACA 0012 airfoil cross-sections approximated by spline *primitives*. Constructed using the modified loft

procedure with a constrained datum spline (not shown), it is evident that cross-sections are parallel to the wing centerline and both sweep/dihedral are preserved (in this case  $\Lambda = 25^\circ$  and  $\delta = 5^\circ$  everywhere). The positioning of cross-sections is driven by the PASS parameterization for wings seen in Tables 2.3 and 2.6. Furthermore, the cross-sections are positioned via pseudo-anchor points at the spline *primitive* leading edges without dimensioning, thereby keeping each cross-section decoupled from the rest of the model. This attribute permits controlling the cross-sections via *any* chosen parameterization, minimizes regeneration time and increases regeneration robustness.



(a) Global reference plane used.



(b) Spline datum reference used.

Figure A-1: A comparison of two methods used to create cross-section sketch planes along a loft span. Figure (a) shows sketch planes constrained parallel to a global Cartesian coordinate plane. Figure (b) shows sketch planes constrained normal to a 3D spline reference datum at its support points; in this case the cross-sections exhibit out-of-plane design motion.

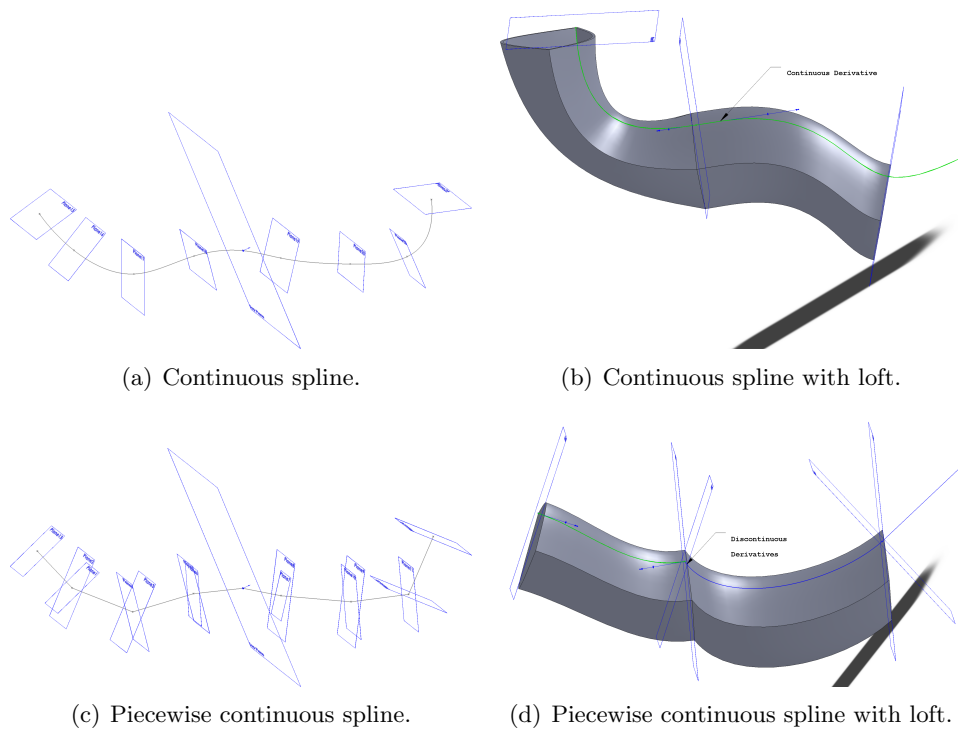


Figure A-2: A comparison between continuous and piecewise continuous (discontinuous derivatives) datum splines. Figures (a)-(b) show sketch planes normal to a continuous spline at its spline points. Figures (c)-(d) show points of discontinuous derivatives on the piecewise spline, where two sketch planes pass through each spline point normal to a tangent vector on each side of the spline point.



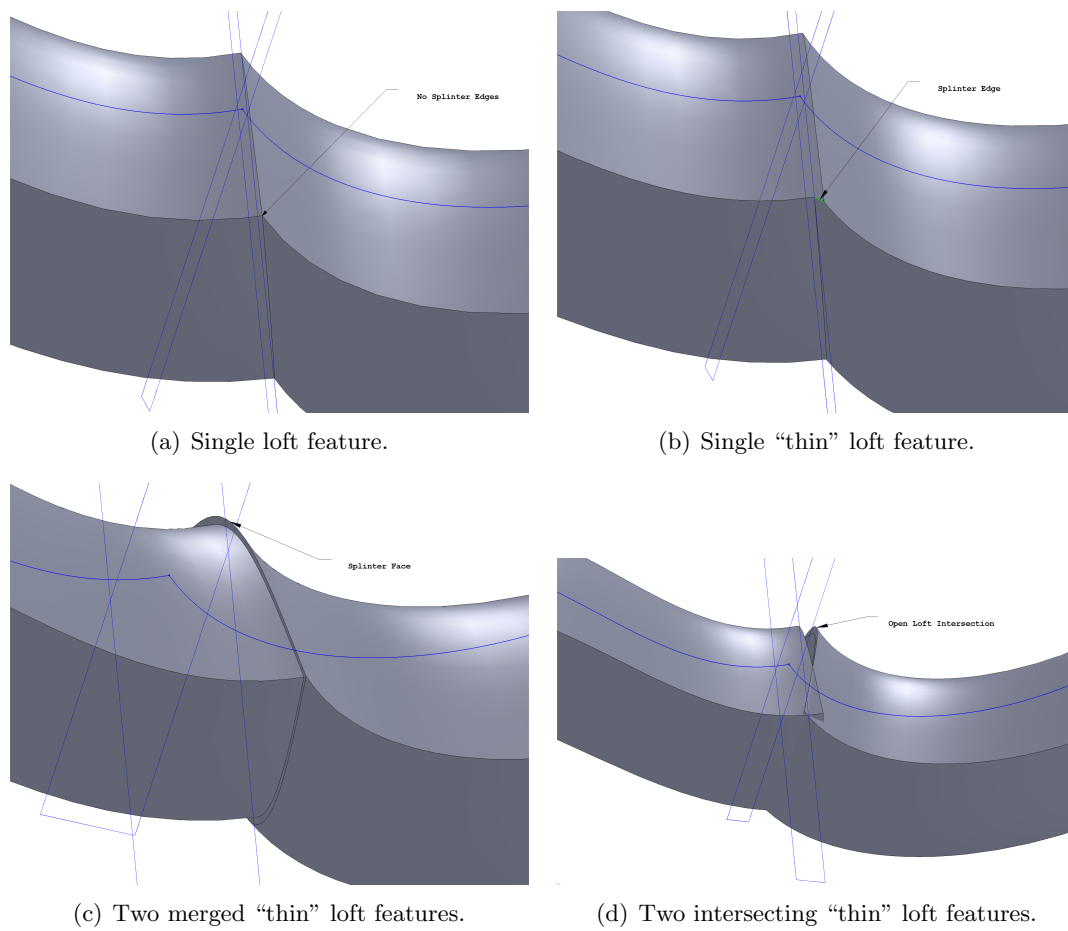


Figure A-3: With the exception of (a), these loft construction approaches illustrate limited usefulness for automated surface tessellation downstream of model construction. In (a) there are no splinter trim curves or faces; however, (b) contains splinter trim curves and (c) results in splinter faces. Clearly (d) highlights poor loft construction because a single closed volume is not obtained as desired.

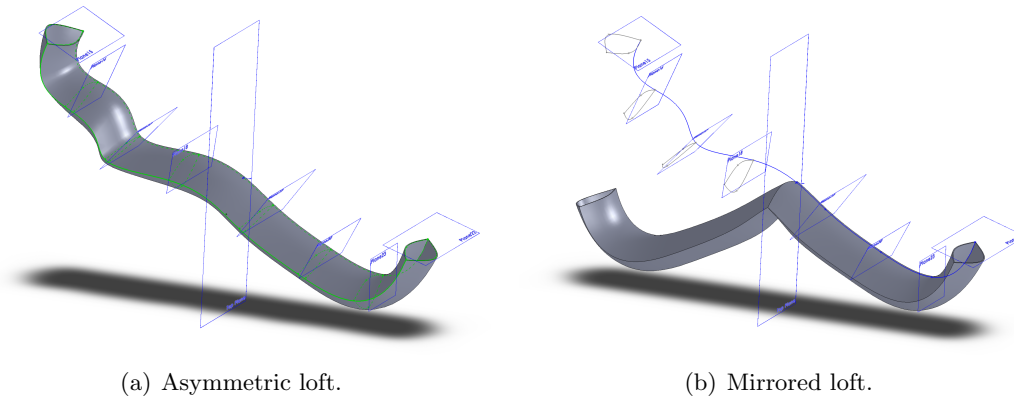


Figure A-4: In (a) the loft can represent asymmetric geometry modes; however, (b) a loft generated using a mirror feature cannot model asymmetric geometry (a skeleton of the asymmetric loft is also portrayed for comparison). The loft construction must embed support for each geometry mode needed in a model.

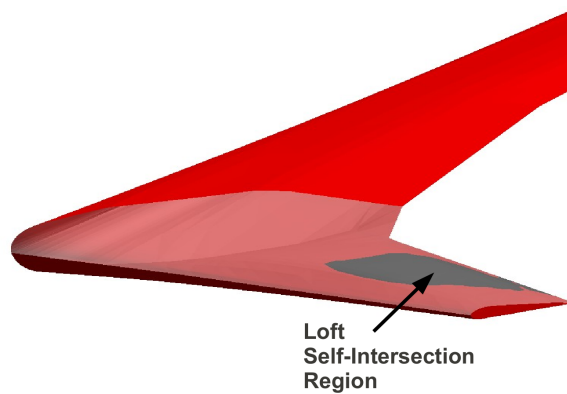
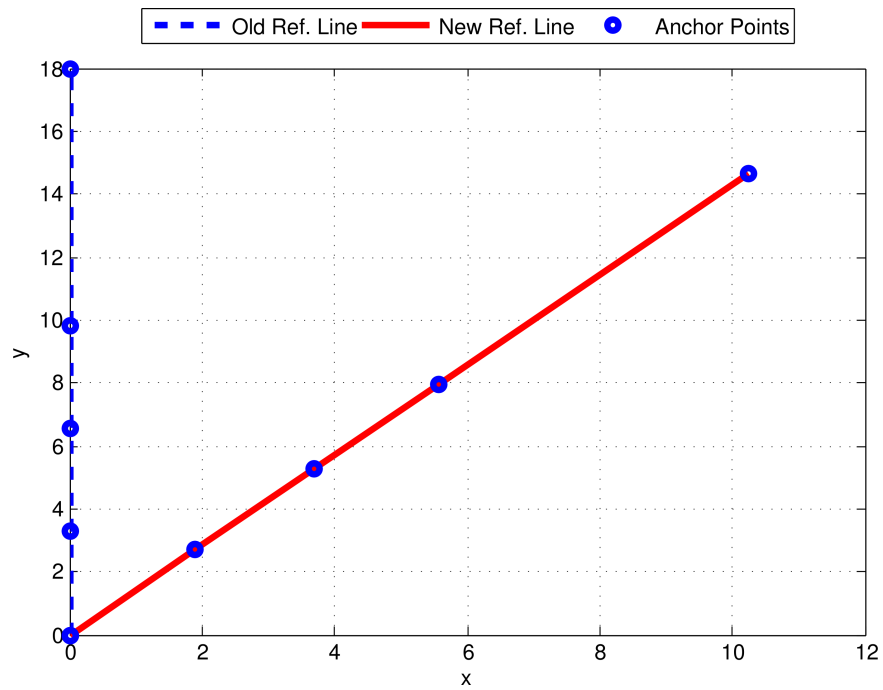
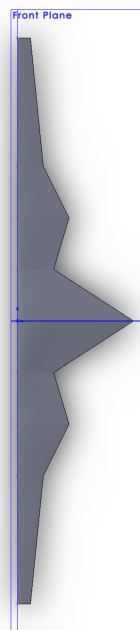


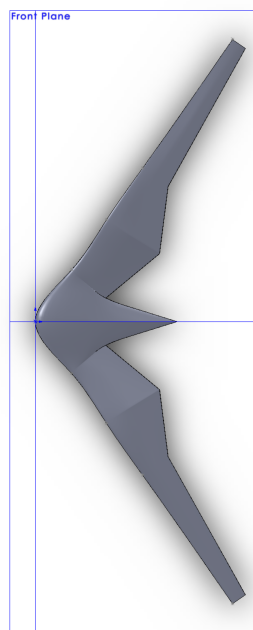
Figure A-5: A loft is hollowed-out to demonstrate a region of self-intersection (grey) that results from a model design point that spreads two cross-sections too far apart. The large discrepancy between cross-section thickness also exacerbated this design motion for the loft.



(a) Anchor point reference line.

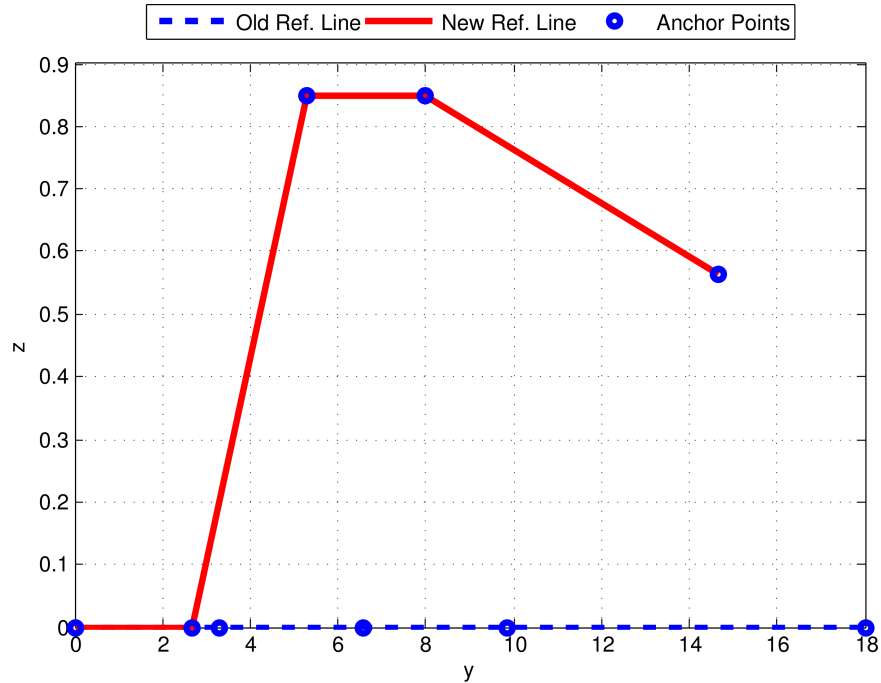


(b) Baseline.

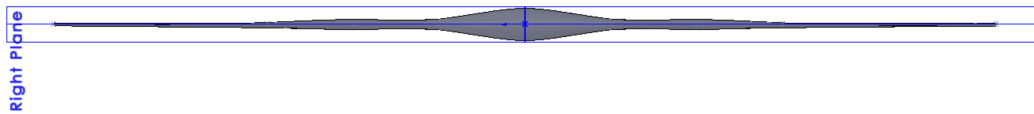


(c) After mapping.

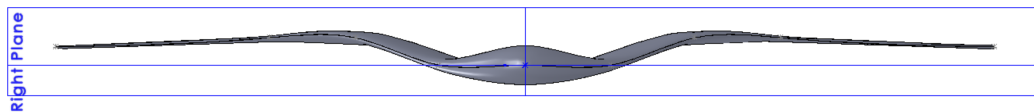
Figure A-6: A planform view ( $xy$  plane) showing the application of the anchor point mapping in (a), where the reference line and anchor points are on the wing leading edge.



(a) Anchor point reference line.



(b) Baseline.



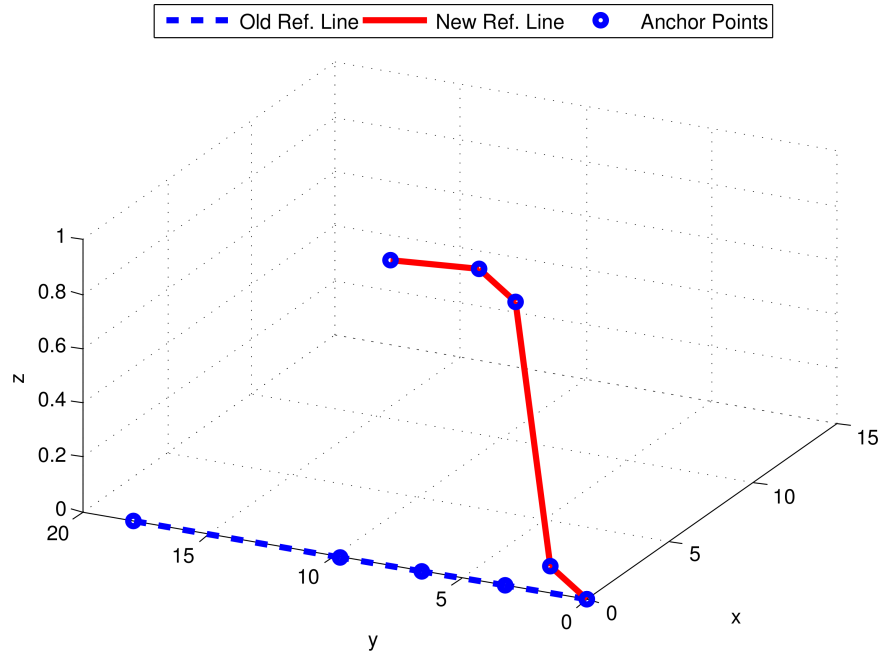
(c) After mapping.

Figure A-7: A front view ( $yz$  plane) showing the application of the anchor point mapping in (a), where the reference line and anchor points are on the wing leading edge.

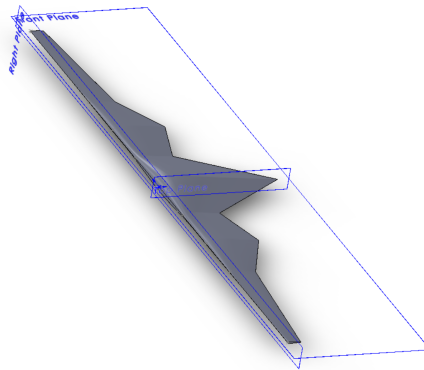
### A.3 Loft Definitions for Fuselage-Type Models

Fundamentally a fuselage-type loft is different from a lifting-surface loft only in its cross-sectional shape. Its construction differs because it generally needs a different design motion. Various approaches surrounding the construction of fuselage-type lofts are presented to highlight potential issues.

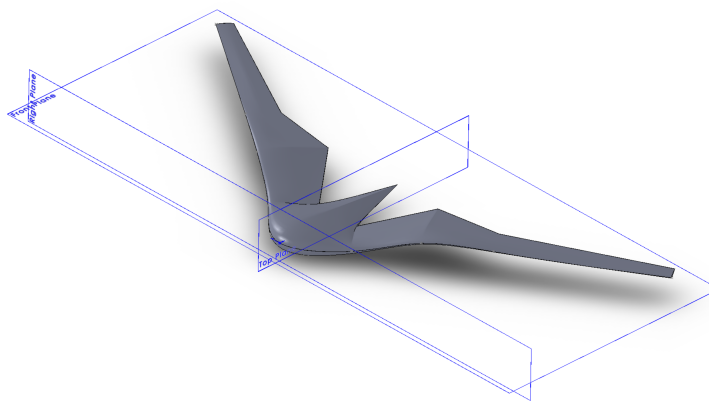
In one case, piecewise continuous guide curves, such as linear or elliptical *primitives*, serve to confine the extent of cross-section shapes from a planform or side-view. Cross-sections are also defined using classic geometry *primitives* of specific shape (e.g., ellipses



(a) Anchor point reference line.



(b) Baseline.



(c) After mapping.

Figure A-8: Perspective views showing the application of the anchor point mapping in (a), where the reference line and anchor points are on the wing leading edge.

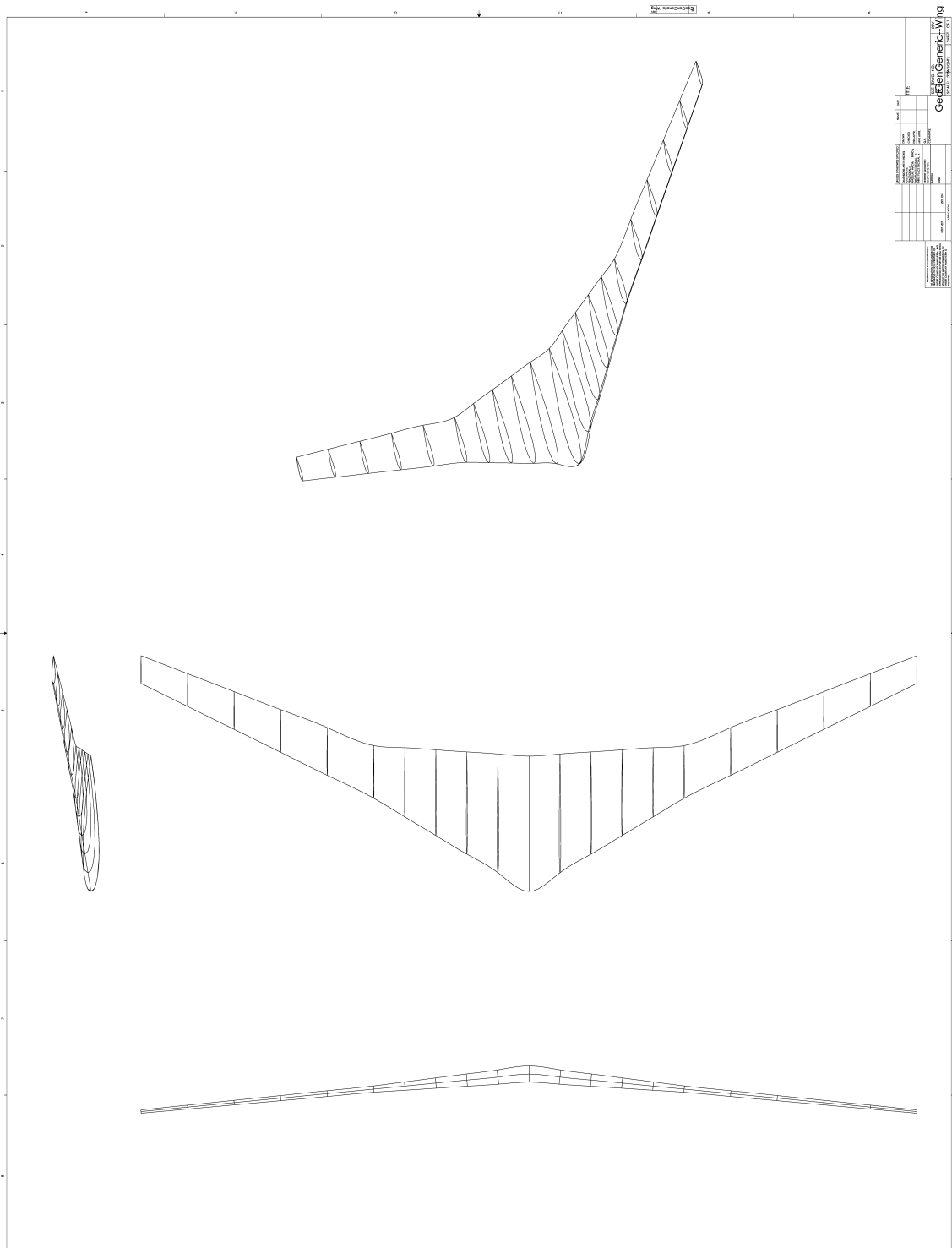


Figure A-9: A three-view of a wing with constrained datum spline and NACA 0012 airfoil cross-sections approximated by spline *primitives*, each of which are independent of all other components in the model.

or circles). This restricts the loft design motion to that of classical shapes due to their geometric similarity. The advantage in this scenario is that the loft has “natural” geometry constraints that enforce a specific cross-section shape, regardless of the driving parameter values. The disadvantage is that other cross-section profiles cannot be modeled, even if such a design trajectory would yield an improved objective function value during optimization.

In another approach no guide curves are used and the positioning of cross-sections is calculated outside of the CAD system by the designer. Classical *primitives* are replaced by spline *primitives* in defining cross-section shapes as well. This requires additional effort upfront in order to ensure that the fuselage cross-sections have the intended design motion as driving parameters are varied (i.e., the cross-sections will not maintain geometric similarity). The advantage is that the model is very malleable because any desired surface curvature is possible by properly choosing a profile and re-orienting it in  $\mathbb{R}^3$ . Spacing between cross-sections can be varied to better resolve regions of higher curvature, such as near the fuselage nose. In addition, the design space is not restricted to a single parameterization because the designer can change the cross-section definition at any location along the fuselage. This permits usage of custom shape functions, for example.

Another model construction issue arises when considering how cross-section sketches are placed. For example, a straight spline datum line can be set along the  $x$ -axis (body axis) in order to constrain sketch planes normal to it. This permits proper out-of-plane deformation, as in the lifting-surface case. It is important to properly understand the design motion of the cross-section if the elliptic *primitive* center-point is offset from the sketch plane origin (which may be constrained to the datum spline points for convenience). Otherwise, by initializing the model with the datum spline passing through cross-section center-points that are not collinear (which can occur if ellipse center-points are not coincident to the sketch origin), the sketch planes will remain normal to an undulating datum spline. Any *primitives* sketched on these particular sketch planes will not result in a loft surface with the intended planform or side-view geometry. This is an issue to consider because designers may define shape functions, external to the CAD system, by assuming the sketch planes are always parallel.

Without guide curves, a potential issue arises if sketch planes are too close in a region of quickly changing derivatives. The loft may overshoot between sketch profiles because a sudden change in the local tangent vector occurs in a small region. Either the cross-

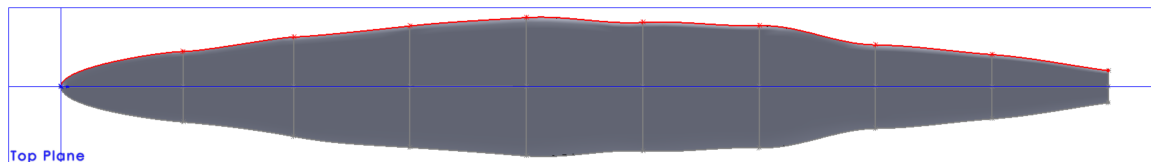
sections need to be spaced further apart or the change in derivative needs to be reduced (e.g., smaller taper angle for the tail-cone portion of the fuselage). These loft overshoots will prove problematic in high-fidelity aerodynamic analysis results. Another option is to divide the single problematic loft into two lofts that merge at the point of quickly changing derivative (similar to that discussed in Section A.1). As seen before, merging lofts in this manner may create sliver trim curves and faces, especially if the thin-attribute is used.

Other fuselage construction methodologies include utilizing guide curves that constrain the top/bottom and side contours after cross-sections are defined. One potential benefit of this approach is that the nose cap may be resolved properly in the context of the first cross-section. Otherwise, constraining a separate nose cap loft to be coincident and tangent to the fuselage loft can be troublesome. Problems sometimes arise when attempting to make all of the nose cap splines tangent to the loft face and coincident to the adjacent cross-section *primitive*. When the model centerline (datum spline) is straight there is no apparent problem with this technique. However, perturbing the datum spline point adjacent to the nose can cause regeneration problems because the tangent-matching constraints are violated.

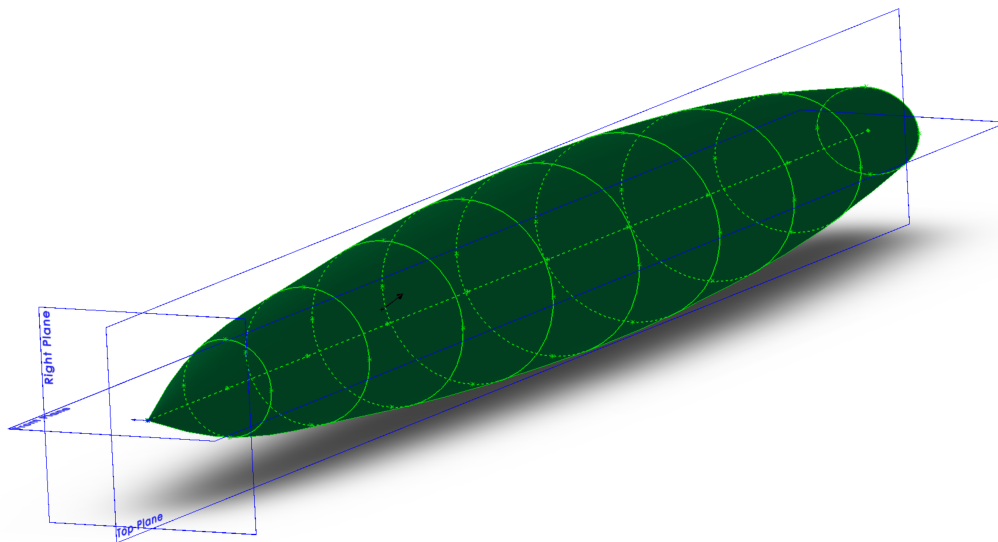
The loft feature can be fragile (in a regeneration sense) when using guide curves to constrain its perimeter shape. The loft may not follow all guide curves properly, meaning the loft surface is only influenced locally by the guide curve. However, using a “boss/base boundary” feature (instead of a loft) may circumvent this issue. With this feature the guide curves serve as profiles in one direction while the cross-sections serve as profiles in a second direction. The thin-attribute can also be applied successfully when terminating this feature with a sharp nose cap.

An issue with the guide curve approach appears when a cross-section parameter is changed. The guide curve will be modified due to the change in location of the cross-section point it is constrained to. However, the previous or following spline point will not translate to a new location and the tangent vector at those spline points will remain unchanged. Therefore, as seen in Figure A-10(a), points of inflection may occur between two cross-sections. It may be possible to avoid this by constructing the guide curve with piecewise continuous splines that have discontinuous derivatives at the spline points. This adds extra complexity to the model since each tangent vector must be accounted for by the designer. Instead, it may be better to only allow tangency control at certain spline





(a)



(b)

Figure A-10: (a) A side-view of a simple fuselage concept with a rounded nose cap (accomplished using guide curves along the fuselage top/bottom and sides). The top guide curve is shown in red to demonstrate inflection-points that occur after regenerating with smaller aft cross-sections. This occurs because the tangency vectors at the mid and aft spline points maintain their original orientation. (b) A perspective view for a simple fuselage concept with a sharp nose cap, as seen in supersonic aircraft. In this case the elliptical cross-sections are visible and centered on a datum spline with fixed spacing between sketch planes.

points, thus removing the need for the designer to calculate appropriate tangent vectors that conform with the changing cross-section geometry. Points without tangency control will have their local tangent vector varied by the CAD system when cross-sections are perturbed. This situation is also evident when perturbing a centerline guide curve (or datum spline) because all cross-sections and perimeter guide curve tangency are modified. This is a concern if the tangent vector orientation is not constrained with the intended local taper-angle, as mentioned above, and surface undulations are thus possible. If a shape function drives the guide curve, it can be differentiated to control the local tangent.

The fuselage loft may be terminated with a rounded or sharp nose cap, depending on the application. Rounded nose caps can be created using a “dome” feature. This requires the fuselage loft to terminate prior to a datum spline end-point with a solid face (this feature will likely not work if the loft terminates with an open face created by the thin-attribute). Dome features attempt to match the surface tangent of the adjacent loft surface and then terminate coincident to a point or have a specified radius. Sharp nose caps can be created by including the datum spline end-point in the loft profile sequence. An example is seen in Figure A-10(b). Attempting to round a sharp loft, which is possible by controlling the tangency of a perimeter guide curve, is difficult since the tangency condition does not behave like that of a revolved guide curve (which creates a surface of revolution). The guide curves often only impose local conditions instead of driving the entire loft surface. As expected, sharp nose caps may give regeneration problems when coupled with a thin-attribute due to the degeneracy at the tip.

An example fuselage model is shown in three-view format in Figure A-11. This model is constructed using classical elliptical *primitives* as cross-sections, the extent of which are bounded by a planform and side-view parameterization defined by linear and elliptical distributions. Parameterization is also accomplished using the PASS parameters in Tables 2.3 and 2.4. As done in the wing case, each cross-section is independent of all other model components and is only coupled with a datum spline (not shown) spanning the  $x$ -direction (the body axis). As before, this permits flexibility in changing the parameterization and increases robustness with minimal dimensioning and constraints.



## A.4 Assembly Model Considerations

When multiple parts are assembled in a model their individual *and* collective design motion must be taken into account. For an aircraft representation, the wing and fuselage lofts can be assembled using various options. Since aircraft are predominantly symmetric about their centerline, the plane of symmetry for the wing and fuselage can be constrained coincident. Also, conventional design often places the global coordinate system origin at the fuselage nose, hence the wing loft can be positioned with respect to that. A third option is to mate a wing anchor point on the datum spline for the fuselage. However, this does not provide for variation in wing height with respect to the fuselage, as seen in Figure A-12.

It becomes clear that planform design motion of the wing relative to the fuselage is best obtained by *not* explicitly mating the two lofts at some topological entity or datum object. The wing loft has the most flexible design motion if its own origin is used as the anchor point for the entire loft. Assembly variables then become the position coordinates of the wing origin with respect to the fuselage origin.

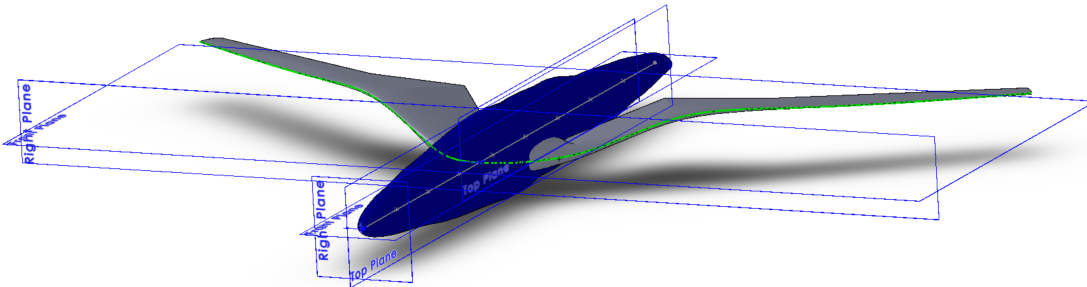


Figure A-12: A perspective view of a simple fuselage concept assembled with a wing. A “path mate” is utilized, where the wing longitudinal position along the fuselage datum spline (shown in grey) is set by the central anchor point at the wing leading edge (shown in green). A relative vertical wing design motion is not possible in this setup.

Three-views of a generic tube-and-wing model assembled in this manner are seen in Figures A-13 through A-15, where the wing from Figure A-9 and the fuselage from Figure A-11 are assembled. These figures depict the main steps of model generation within the CAD system: (1) establish a reference datum system, (2) allocate cross-section geometry *primitives* and (3) loft the corresponding sets of cross-section geometry. The wing and empennage positions are parameterized by anchor point coordinates relative to the fuselage length and height, as shown in Figure A-13. Doing this for each part permits independent design motion among the four components. Furthermore, the cross-section *primitives* of

each component are defined by part-level parameters relative to their corresponding reference datums, as implied in Figure A-14. The final single-assembly *configuration* in Figure A-15 thus maintains the individual design motion of each part and allows for the relative design motion among parts. This means the part and assembly design motion are uncoupled (which is consistent with the hierarchy viewpoint of parameterization levels). As discussed in Sections A.2 and A.3, the wing and fuselage parts are parameterized at the part level for uncoupled design motion from the assembly-level parameterization.



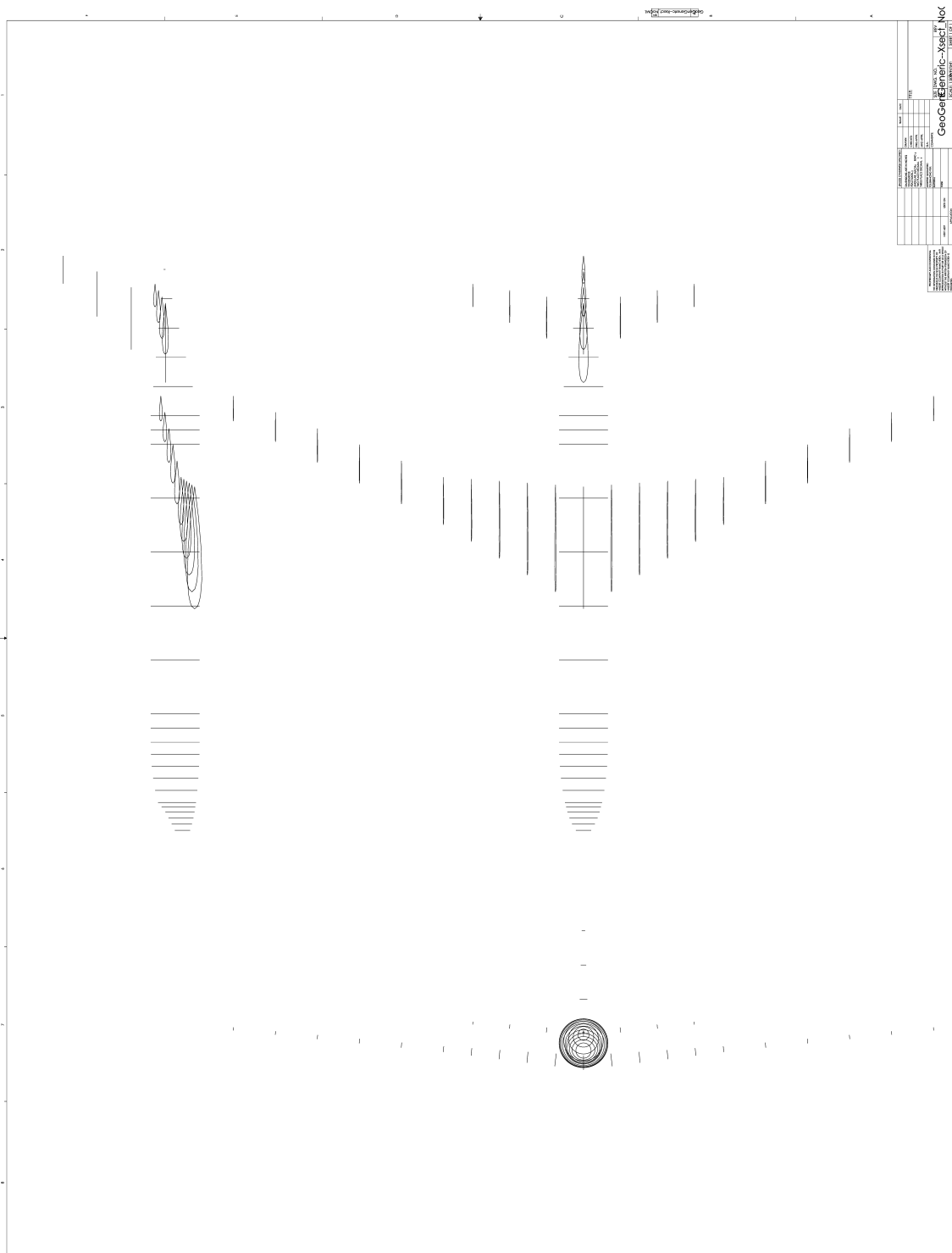


Figure A-14: A three-view of the independent cross-section geometry *primitives* initialized after placing sketch planes along each reference datum in Figure A-13.





## A.5 Multidisciplinary Geometry Considerations

Multidisciplinary CAD models have additional considerations to take into account. The most important consideration for these models is avoiding surface intersections between parts. When part sizing permits placement of parts within the interior volume of another, this is of less concern unless the interior parts have a design trajectory that eventually cause surface intersections. In other cases, sub-discipline parts are constructed in the context of parent parts pertaining to another discipline. For example, a wing spar is typically constrained within the interior volume of a wing outer mold line, hence the structure shape is locally driven by the wing loft shape. If constructed poorly, this scenario can easily lead to surface intersections that are non-intuitive.

There are two ways in which parts constructed in the context of another can lead to unexpected surface intersections. Both require an understanding of loft and spline generation in a geometry kernel. First, the internal part must reside within the envelope of the external part in order to avoid intersection. To accomplish this with lofts generated in the context of other lofts, it is necessary that both lofts pass through the *same* cross-section plane and have the *same* span. Figure A-16 demonstrates how avoiding this suggestion is problematic in the example of 2D splines passing through the same cross-section planes across different wing spans. The underlying cubic B-spline formulation will not generate the same space-curve in both instances, which appears as over- and under-shoots of the half-span spline compared to the full-span spline. It is obvious that this also occurs if the two splines did not share the same support points.

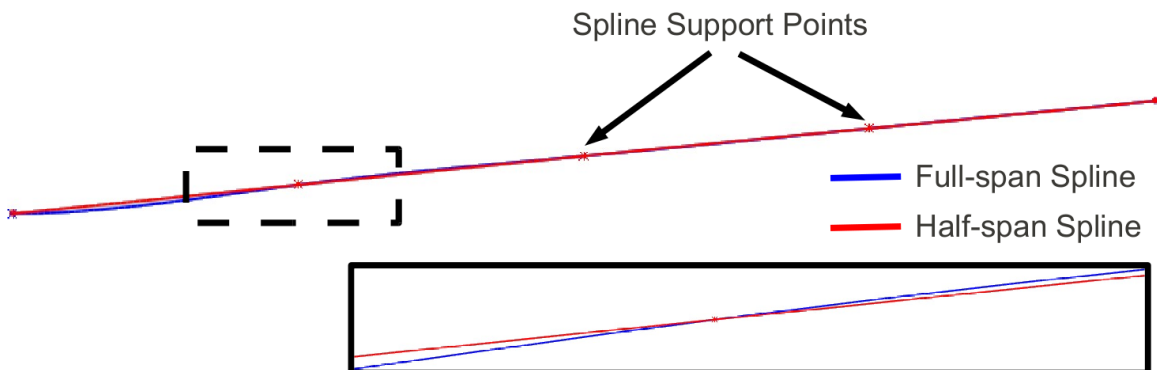


Figure A-16: Two 2D splines sharing the same support points, but not the same span, cannot generate the same space-curve.

Since a B-spline surface contains B-splines spanning two-directions as isoparameter lines,

the same outcome will occur if the scenario in Figure A-16 is extended to B-spline surfaces, as seen in Figure A-17. In this case the internal spar structure loft passes through the same cross-section planes as the wing outer mold line loft, but only at half-span (wing root to tip). It is clear that the spar loft cannot reside within the envelope of the wing outer mold line and surface intersections occur. In other instances, this may not be a problem until some later point in the model design trajectory. For example, Figure A-18 demonstrates how the design point modeling a wing-tip deflection creates a surface intersection. This is due to design motion discrepancies between the wing loft and internal spar loft. Earlier design points for this model did not experience such surface intersections. The deflection was modeled by setting the  $z$ -coordinate of each wing datum spline point to

$$z = Ay^2 + d_{z,\text{offset}},$$

where  $y$  is the span-wise coordinate,  $d_{z,\text{offset}}$  is the wing  $z$ -offset from the global origin and  $A$  is a constant. Therefore, the loft construction suggestions given here are necessary, but not sufficient to guarantee zero surface intersections generally. These suggestions reduce the likelihood of surface intersections occurring along a design trajectory. Further improvements are possible if the internal lofts span many more cross-sections constrained between the outer loft sections, thereby reducing the possibility of surface over/under-shoots in regions of fast derivative changes.

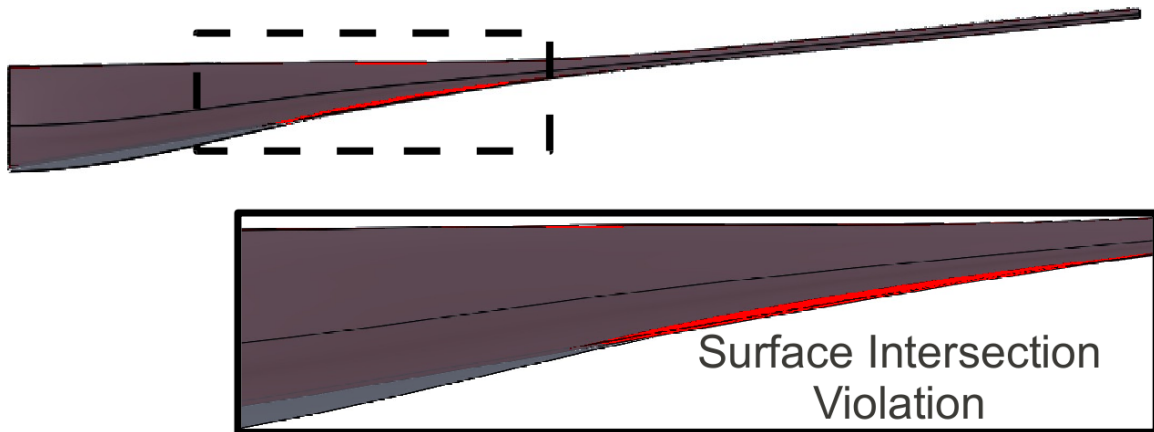


Figure A-17: The red loft (spar structure) passes through the same cross-section planes as the grey loft (wing outer mold line), but does not share the same span. Hence, the spar loft cannot remain inside the envelope of the wing loft.

The second manner in which unexpected surface intersections may occur deals with

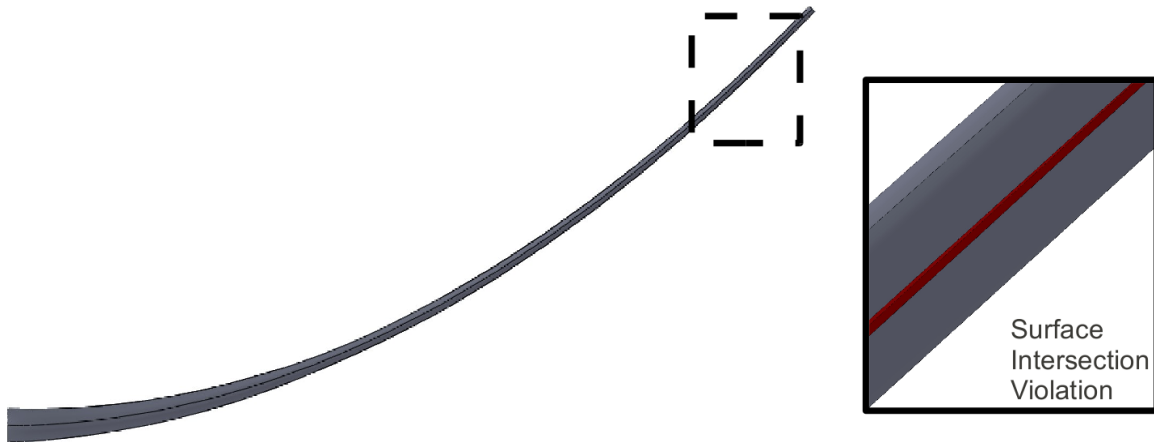


Figure A-18: The design point for a wing-tip deflection exacerbates discrepancies between the design motion of the red loft (spar structure) and grey loft (wing outer mold line), leading to surface intersections that may not have occurred earlier on the model design trajectory.

poor model construction. Figure A-19 shows a flying-wing model with two internal spar structures penetrating the outer mold line. The internal spar lofts use the same cross-section planes as the wing loft and also share the same span. However, it is clear in the zoomed-in images of Figure A-19 that the spar penetrates the wing loft to a “small” extent. A tessellation of this model will generate sliver elements that disrupt the wing surface curvature (akin to a boundary layer trip) and lead to spurious aerodynamic results in an automated design setting. Unlike the top image in Figure A-20, the spar profile must reference an internal offset of the wing contour, as shown in the bottom image of Figure A-20. This will maintain the spar loft within the envelope of the wing loft.

The geometry explanation for the problem seen in Figure A-19 arises in how the geometry kernel creates new profiles that reference other features. In the case of Figure A-19, a cross-section of the wing loft is created at the intersection of the wing and a datum plane. Subsequently the local airfoil is referenced on a new sketch as a constrained spline *primitive*. Added sketch entities may reference this spline since it will be driven by its parent loft. The spar in Figure A-19 uses only a portion of the reference spline to define the spar caps (the intended design motion is that the spar top/bottom always follows the contour of the local airfoil regardless of its chord-wise placement). In creating these caps, the reference spline is “cut” into a smaller spline that spans the spar web (modeled by two vertical line segments). The geometry kernel essentially re-splines the “un-cut” portion of the reference spline (shown as the solid black line contrasting against the broken-line of the reference

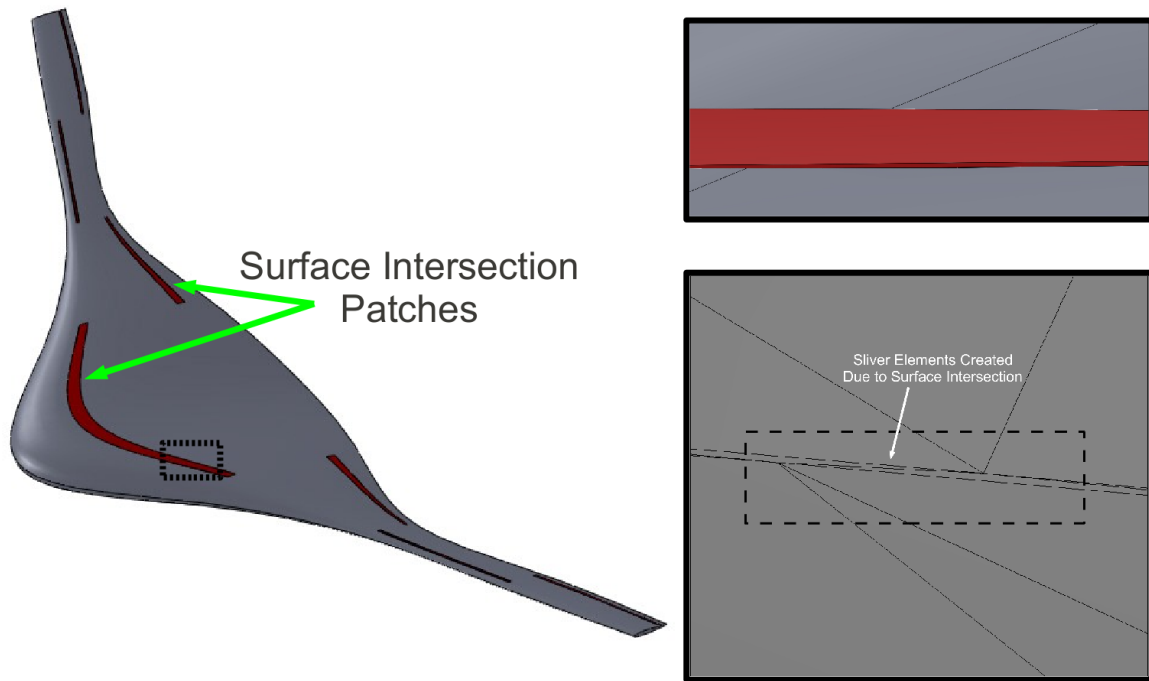


Figure A-19: The red loft (spar structure) passes through the same cross-section planes as the grey loft (wing outer mold line) and shares the same span. However, the spar loft still penetrates the envelope of the wing loft due to its construction method.

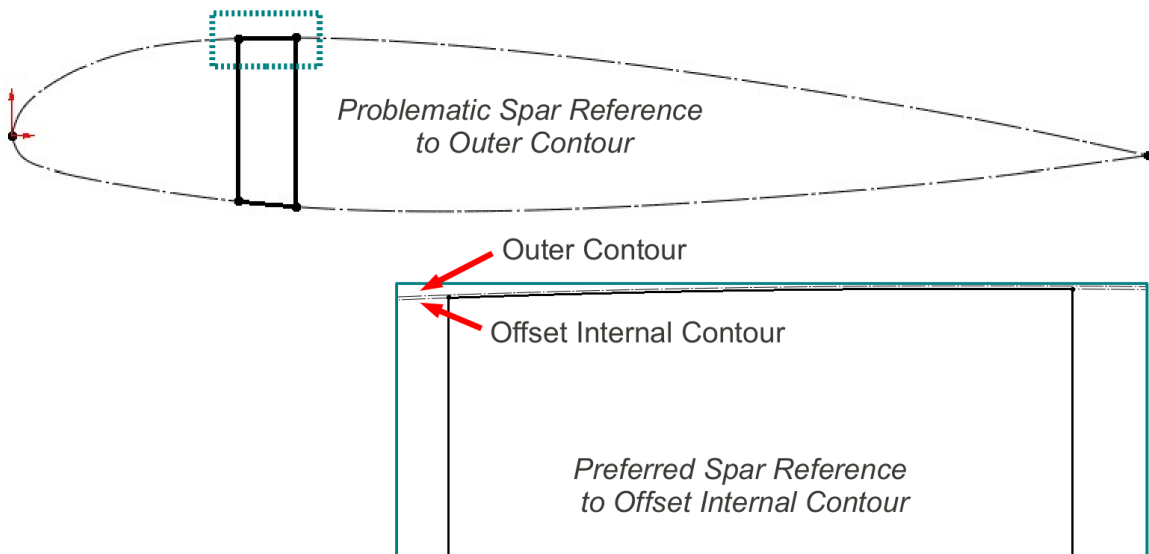


Figure A-20: The spar is constructed using the wing airfoil profile as a constraint in the top figure; however, unless it is constructed with an *offset* from the wing contour as in the bottom image, the spar loft will more easily create a surface intersection.

airfoil in Figure A-20) as a local *approximation* of the airfoil. Since the approximation is within a tolerance (internal to the geometry kernel), lofting through the spar profiles (albeit at the same cross-section planes as the wing profiles) implies *not* lofting through the actual airfoil contours, hence a minor surface intersection is created. Table A.1 compares the B-spline surface parameters (except for the  $U$  knot vector) for the wing face  $\mathcal{F}_4$ , intersection face  $\mathcal{F}_{12}$  and  $\mathcal{F}_5$  on both the thin and box spar of Figure A-21. It is evident that in the chord-wise direction ( $u$ -direction) the approximated airfoil shape is done with fewer control points than on the wing loft, although this improves for the box spar. The  $V$  knot vector does not match between wing face  $\mathcal{F}_4$  and the others, which explains why the span-wise loft distribution does not match for the thin spar and wing. Even though these share the same span and cross-section planes, the B-spline surface knot vectors differ span-wise, meaning that these parameters fundamentally express different isoparameter lines in the  $v$ -direction and intersection will occur.

	<b>Wing</b> <i>Face 4</i>	<b>Intersection</b> <i>Face 12</i>	<b>Thin Spar</b> <i>Face 5</i>	<b>Box Spar</b> <i>Face 5</i>
$u$ degree	3	3	3	3
$v$ degree	3	3	3	3
$n_{\text{CPs},u}$	20	4	4	42
$n_{\text{CPs},v}$	11	11	11	11
$n_{\text{knots},u}$	24	8	8	46
$n_{\text{knots},v}$	15	15	15	15
$v$ knots	0.000000000000	0.000000000000	0.000000000000	0.000000000000
	0.000000000000	0.000000000000	0.000000000000	0.000000000000
	0.000000000000	0.000000000000	0.000000000000	0.000000000000
	0.000000000000	0.000000000000	0.000000000000	0.000000000000
	0.115811274145	<u>0.113456193256</u>	<u>0.113456193256</u>	<u>0.115037954177</u>
	0.231653293268	<u>0.226913437165</u>	<u>0.226913437165</u>	<u>0.230277505523</u>
	0.348544315764	<u>0.347137775904</u>	<u>0.347137775904</u>	<u>0.347264243894</u>
	0.465484487420	<u>0.467081892501</u>	<u>0.467081892501</u>	<u>0.465810371185</u>
	0.582447859963	<u>0.586646240182</u>	<u>0.586646240182</u>	<u>0.584227365889</u>
	0.706375691400	<u>0.712996164702</u>	<u>0.712996164702</u>	<u>0.708449030395</u>
	0.843981652496	<u>0.847525717338</u>	<u>0.847525717338</u>	<u>0.845009777551</u>
	1.000000000000	1.000000000000	1.000000000000	1.000000000000
	1.000000000000	1.000000000000	1.000000000000	1.000000000000
	1.000000000000	1.000000000000	1.000000000000	1.000000000000
	1.000000000000	1.000000000000	1.000000000000	1.000000000000

Table A.1: A comparison between the B-spline surface parameters of the four faces highlighted in Figure A-21.

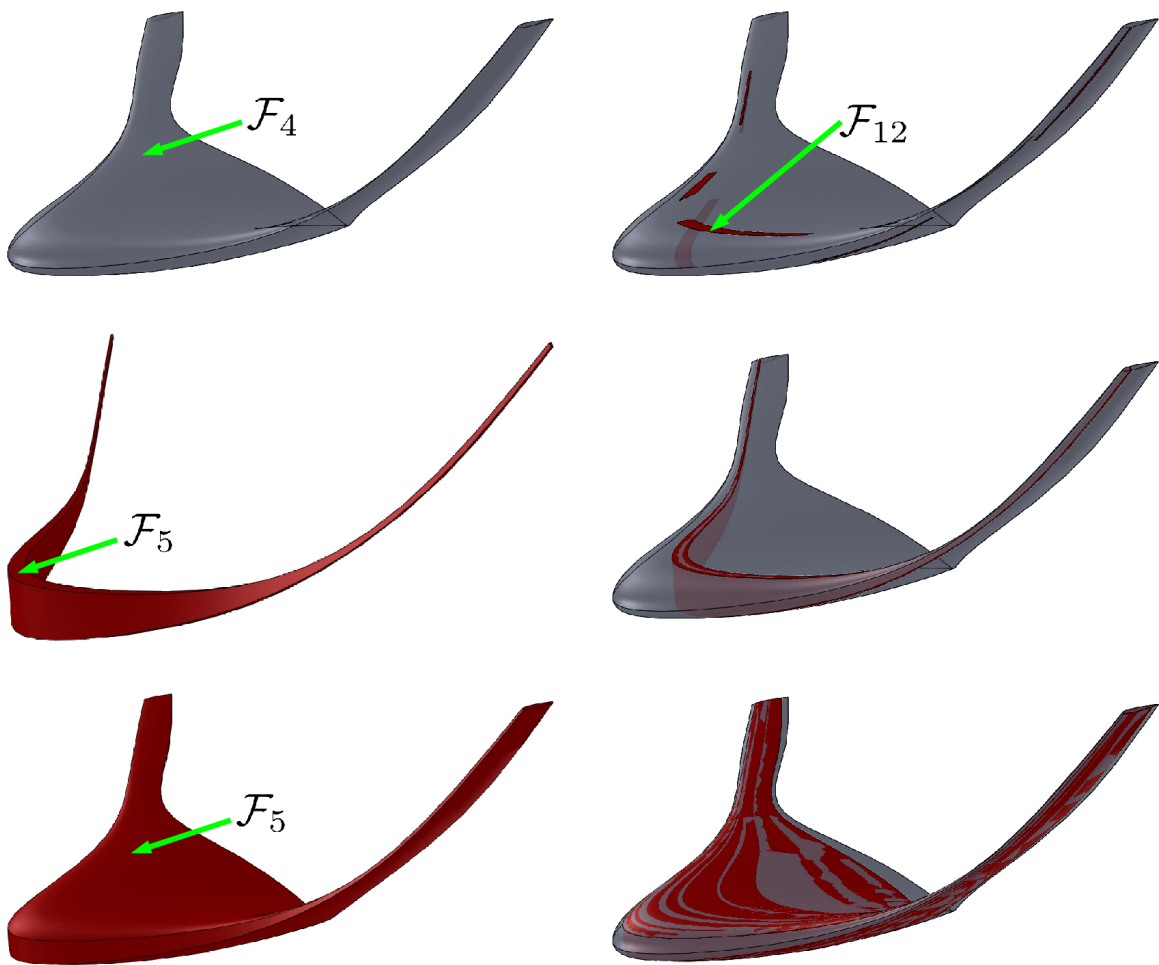


Figure A-21: The effect of approximating a loft swath with another loft is shown, where the wing upper face is approximated by the caps of a thin and box spar loft. As the spar width increases, the approximation improves and the CAD system cannot determine the intersection trim curve.

In this scenario the designer’s intent is not captured by the geometry kernel and the proposed solution in Figure A-20 must be considered. It is interesting to note that extending the spar width to cover a greater percentage of the chord, as in the case of the box spar, allows the cut-spline approximation to improve. The resulting spar loft better fits within the wing loft envelope and the CAD system cannot calculate an actual surface interference since the model becomes non-manifold (i.e., two faces approach being coincident). The designer must determine what offset value is appropriate to avoid this issue in a particular CAD system<sup>2</sup>.

## A.6 Model Initialization and Automatic Generation

By synthesizing the considerations explained thus far, a set of programmable “rules” become apparent that facilitate the automatic generation of CAD models of aircraft. CAD systems often support API programming in various languages in order to accommodate this. There are many benefits to capturing the model construction methods discussed here and embedding an appropriate design intent in a model. This serves multiple purposes, some of which are listed here:

1. Manual creation of the model using the CAD GUI is greatly reduced (a time savings).
2. Parameter distributions are easily handled.
  - (a) Chord and thickness distributions; span-wise airfoil distribution.
  - (b) Distribution of cross-section extent in planform or side-view.
3. Model generation code is similar to model parameter-update code.
4. Easily switch through various parameterizations when generating models.
5. Similar geometry is easily created (e.g., lifting-surfaces).
6. Fast turn-around time when adjustments are needed for new models.

---

<sup>2</sup>The spar caps may be closed via line segments, instead of the airfoil spline approximation, as another option. If the spar web end-points are coincident to the reference spline the same surface intersection issues will occur. This again stems from the discrepancy in the  $V$  knot vectors between the wing loft and spar cap face.

Creating CAD models manually through a GUI requires more time as the model complexity increases. This approach is intractable for an aircraft conceptual design setting because numerous *configurations* need evaluation during a small time frame. Using pre-built models for new designs is inefficient because the designer may not know what the intended design trajectory is for the model components, hence time is spent reverse-engineering the design intent. Only an automated geometry generator utilizing the CAD system API will permit the fastest model production time for conceptual design. In this work all CAD models are created with an automated geometry generator using the API in the SolidWorks CAD system. Manual work on a model is only done for post-generation steps if a particular feature is not executed properly through the CAD API.

Whether creating a model manually or through automatic generation, the designer must decide the level of geometry detail it will capture in the first iterations of a model. The geometry representation in aircraft conceptual design usually lacks the information available in later design phases. Thus an initial estimate of certain geometry information is necessary in order to construct a full 3D model. For example, by choosing the parameterization options found in Tables 2.4 and 2.5 the designer is left with a choice of airfoil for each lifting-surface. Since the wing airfoil stack will generally require further design iterations, the designer may approximate a known airfoil (via spline *primitives*) that is sufficient for the mission at hand in order to initialize a model. The designer's past experience with such *configurations*, coupled with any additional expert opinion, allows this initial airfoil selection to be adequate for early trade studies. Otherwise, a poorly modeled airfoil stack will result in poor drag predictions that may dominate and misrepresent the performance of a candidate *configuration*.

There is a tradeoff when initializing an airfoil stack for lifting-surfaces. On one hand, the spline *primitive* approximation of each airfoil must produce the  $C_p$  distribution of the "true" airfoil shape, which is typically done by using about  $\mathcal{O}(100)$  spline support points to interpolate the airfoil. On the other hand, shape optimization of an airfoil with  $\mathcal{O}(100)$  support points becomes problematic primarily for two reasons: (1) the number of design variables becomes prohibitively large for optimization, and (2) numerous sign-fluctuations in the surface curvature occur as a result of the local-support property inherent to splines. Overcoming this drawback requires *fewer* spline support points, which may result in a poor  $C_p$  approximation of the "true" airfoil.



An optimization problem can be setup off-line to determine where  $\mathcal{O}(10)$  spline support points should be placed to best approximate the  $C_p$  distribution of a given airfoil. With a known “true” distribution  $C_p^*(\hat{\mathbf{x}})$  as the target at locations  $\hat{\mathbf{x}} = \{\hat{x}_j, \hat{y}_j\}_{j=0}^{\hat{n}}$  across the airfoil, an inverse design problem is setup as follows:

$$\begin{aligned} \arg \min_{\mathbf{x}} \quad & (C_p^*(\hat{\mathbf{x}}) - C_p(\hat{\mathbf{x}}))^2 \\ \text{s.t.} \quad & \mathbf{g}(\bar{\mathbf{x}}) \leq \mathbf{0} , \end{aligned} \tag{A.6}$$

where the design variables  $\mathbf{x} = \{x_i, y_i\}_{i=0}^n$  are the spline support point coordinates and  $n \ll \hat{n}$ . The constraints  $\mathbf{g}(\bar{\mathbf{x}})$  pertain to minimum thickness at two chord fractions  $f_1$  and  $f_2$  such that  $\bar{\mathbf{x}} = \{f_1c, f_2c\}$  becomes position along the chord. This can be solved using a gradient-based optimization algorithm with the starting coordinates  $\mathbf{x}_0 \subset \hat{\mathbf{x}}$ .

THIS PAGE INTENTIONALLY LEFT BLANK

## Appendix B

# PASS Parameterization Definitions

The following tables list the parameters describing a monoplane geometry representation as utilized in the PASS aircraft design software.

<b>Wing</b>		
<b>Parameter</b>	<b>Definition</b>	<b>Units</b>
sref	The reference trapezoidal (trap) wing area.	ft <sup>2</sup>
arw	The wing aspect ratio based on the reference area.	n.d.
sweepw	The sweep of the trapezoidal wing quarter chord.	Deg.
tovercw	The average wing thickness to chord ratio.	n.d.
taperw	Tip to root chord ratio for the trapezoidal reference wing.	n.d.
supercritical	(0) peaky or (1) supercritical section properties	0,1
lex	Leading edge extension added forward of the trapezoidal wing root chord.	Units of trap root chord
tex	Trailing edge extension added aft of the trapezoidal wing root chord.	Units of trap root chord.
chordextspan	The span of the leading and trailing edge extensions.	Semi-span percentage
wingdihedral	Wing dihedral angle.	Deg.
wingheight	(0) Low wing or (1) high wing	0,1
wingxposition	Wing root leading edge (actual wing, not reference wing) relative location on fuselage.	Percentage of fuselage length
alphalimit	Maximum angle of attack limit at $C_{l,max}$ .	Deg.
x/ctransition	Fraction of chord on lifting-surfaces with laminar flow.	n.d.

Table B.1: Wing parameterization used in PASS.

### Horizontal Tail

Parameter	Definition	Units
sh/sref	The ratio of gross horizontal tail area to wing reference area.	n.d.
arh	Horizontal tail thickness to chord ratio.	n.d.
sweep <sub>h</sub>	Sweep of horizontal tail quarter chord.	Deg.
tover <sub>h</sub>	Horizontal tail thickness to chord ratio.	n.d.
taper <sub>h</sub>	Horizontal tail tip chord/root chord.	n.d.
dihedral <sub>h</sub>	Horizontal tail dihedral.	Deg.
cl <sub>h</sub> max	Cl <sub>max</sub> of horizontal tail.	n.d.

Table B.2: Horizontal tail parameterization used in PASS.

### Vertical Tail

Parameter	Definition	Units
ttail	(0) low tail and (1) for T-tail, or anything in between.	0,1
sv/sref	Ratio of vertical tail area to wing reference area.	n.d.
ar <sub>v</sub>	Aspect ratio of vertical tail: height <sup>2</sup> /area	n.d.
sweep <sub>v</sub>	Sweep of vertical tail quarter chord line.	Deg.
tover <sub>v</sub>	Thickness-to-chord ratio of vertical tail.	n.d.
taper <sub>v</sub>	Vertical tail taper ratio.	n.d.

Table B.3: Vertical tail parameterization used in PASS.

### Fuselage

Parameter	Definition	Units
fuse <sub>h</sub> /w	Ratio of fuselage height to width.	n.d.
nose <sub>h</sub> fineness	Nose fineness ratio (nose length/fuselage width)	n.d.
tail <sub>h</sub> fineness	Tailcone fineness ratio.	n.d.
windshield <sub>h</sub> ht	Height of windshield.	ft.
pilot <sub>h</sub> length	Length of pilot station.	ft.
fwd <sub>h</sub> space	Extra space forward of seats in constant section.	ft.
aft <sub>h</sub> space	Extra space aft of seats in constant section.	ft.

Table B.4: Fuselage parameterization used in PASS.

## Appendix C

# Geometry Sensitivity Data for BRep Faces

Surface plots depicting the difference between analytic geometry sensitivities and finite-difference results are shown.

- Figures C-1 through C-11 correspond to the extrude feature in Section 3.3.
- Figures C-12 through C-21 correspond to the revolve feature in Section 3.4.
- Figures C-22 through C-31 correspond to the sweep feature in Section 3.5.

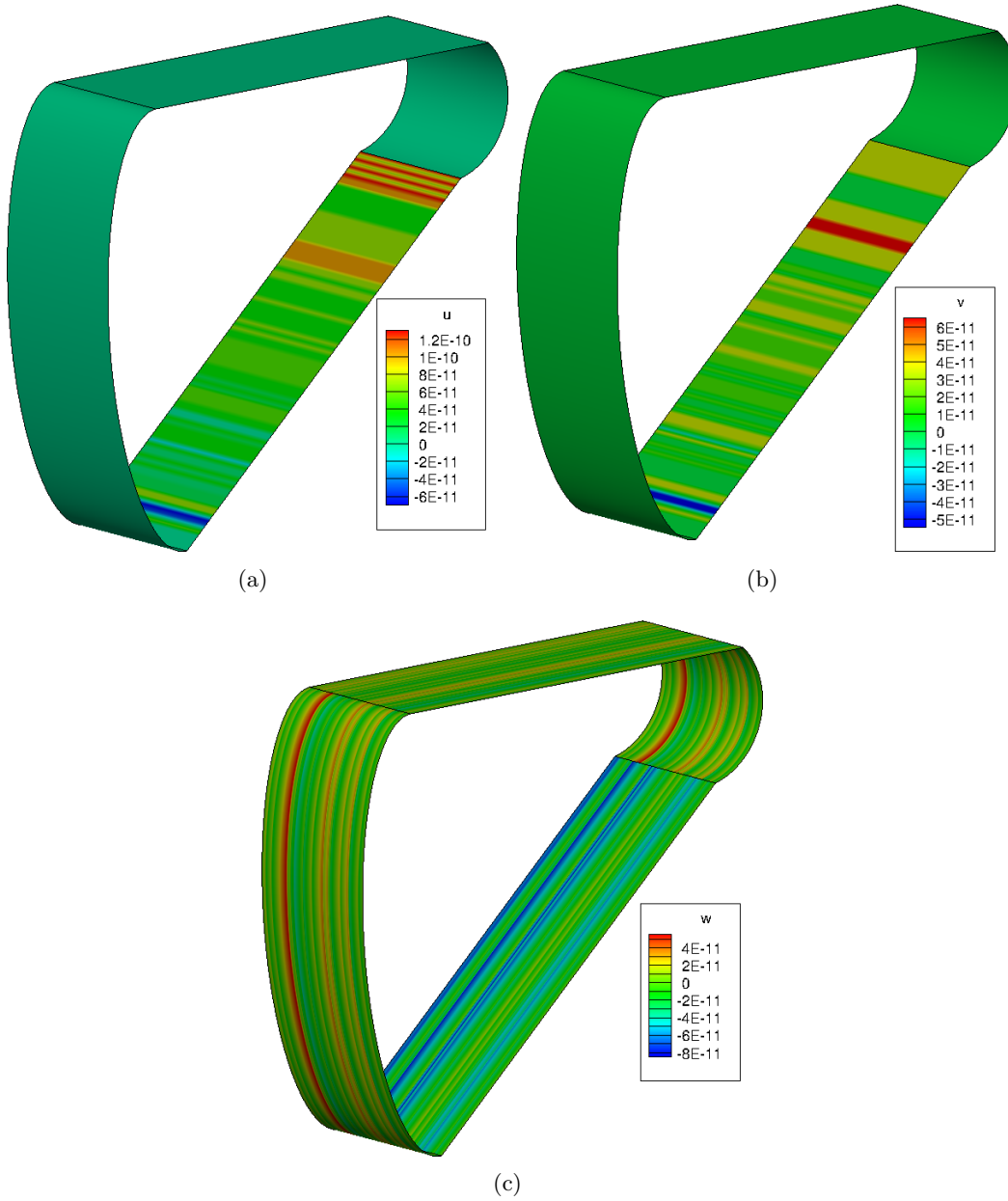


Figure C-1: Relative offset between the extrude *feature* differentiation method and centered-differencing for the extrusion length parameter.

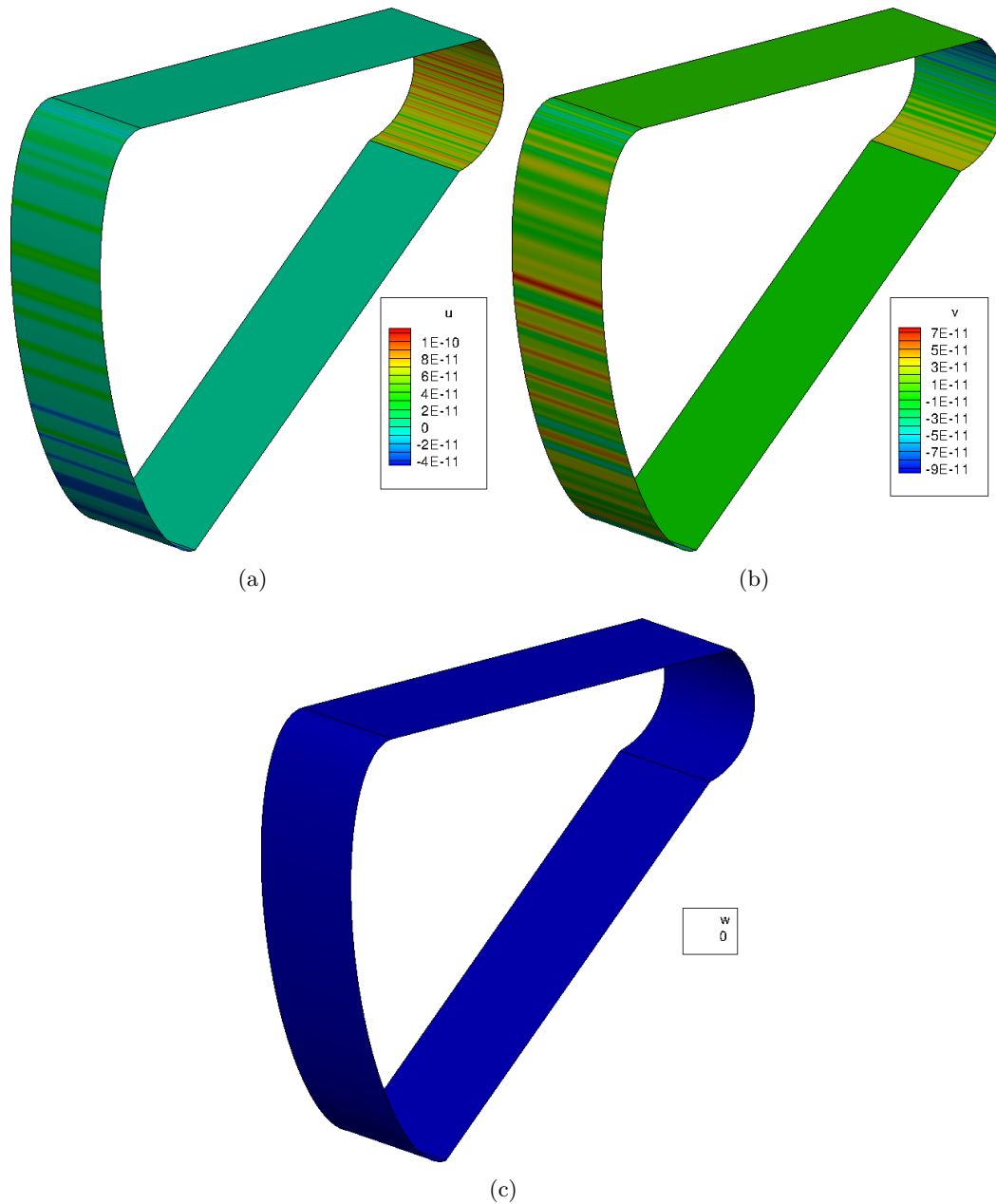


Figure C-2: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{21}$ . The  $w$ -component data is exactly zero in this case.

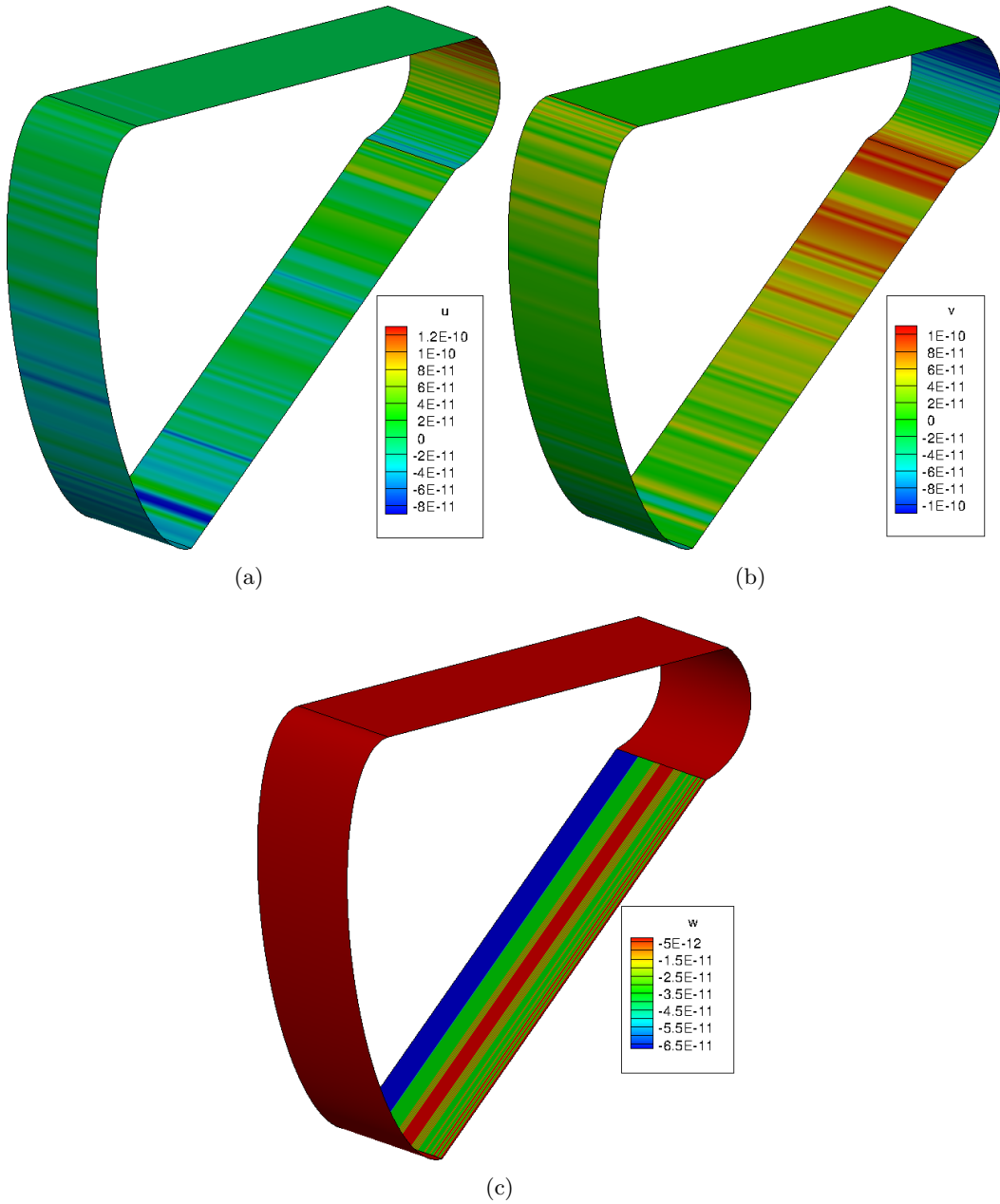


Figure C-3: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{22}$ .



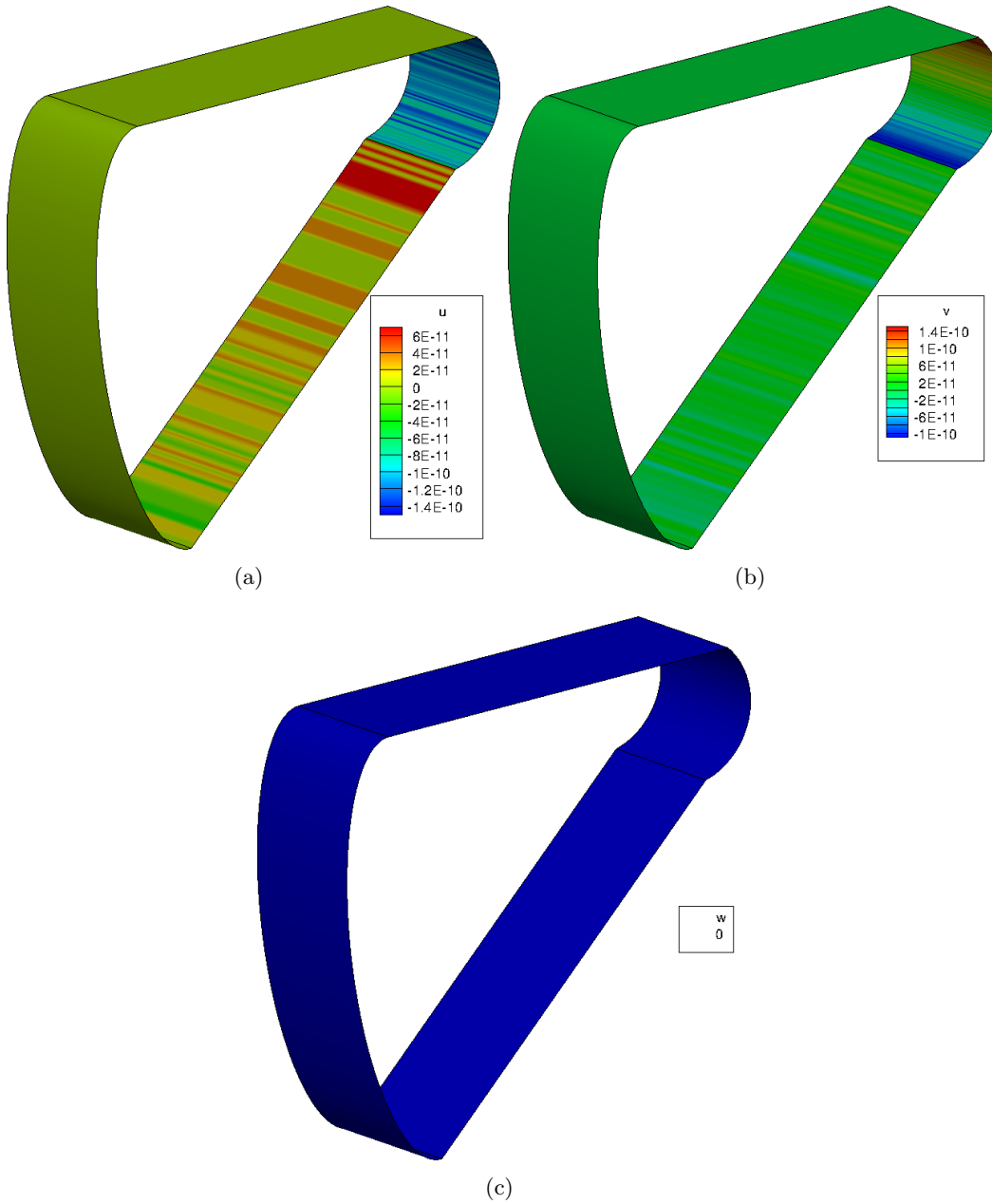


Figure C-4: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{23}$ . The  $w$ -component data is exactly zero in this case.

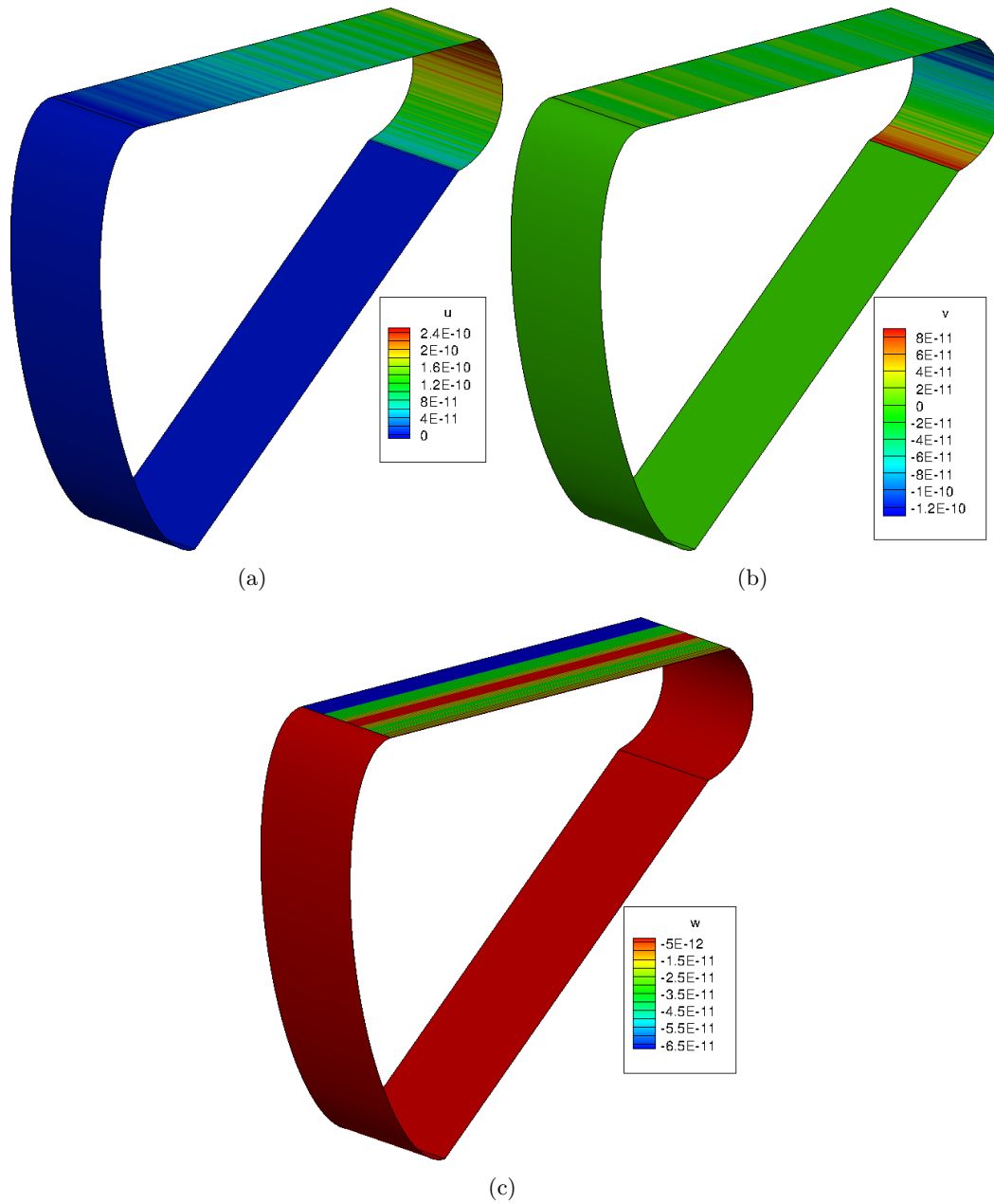


Figure C-5: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{25}$ .

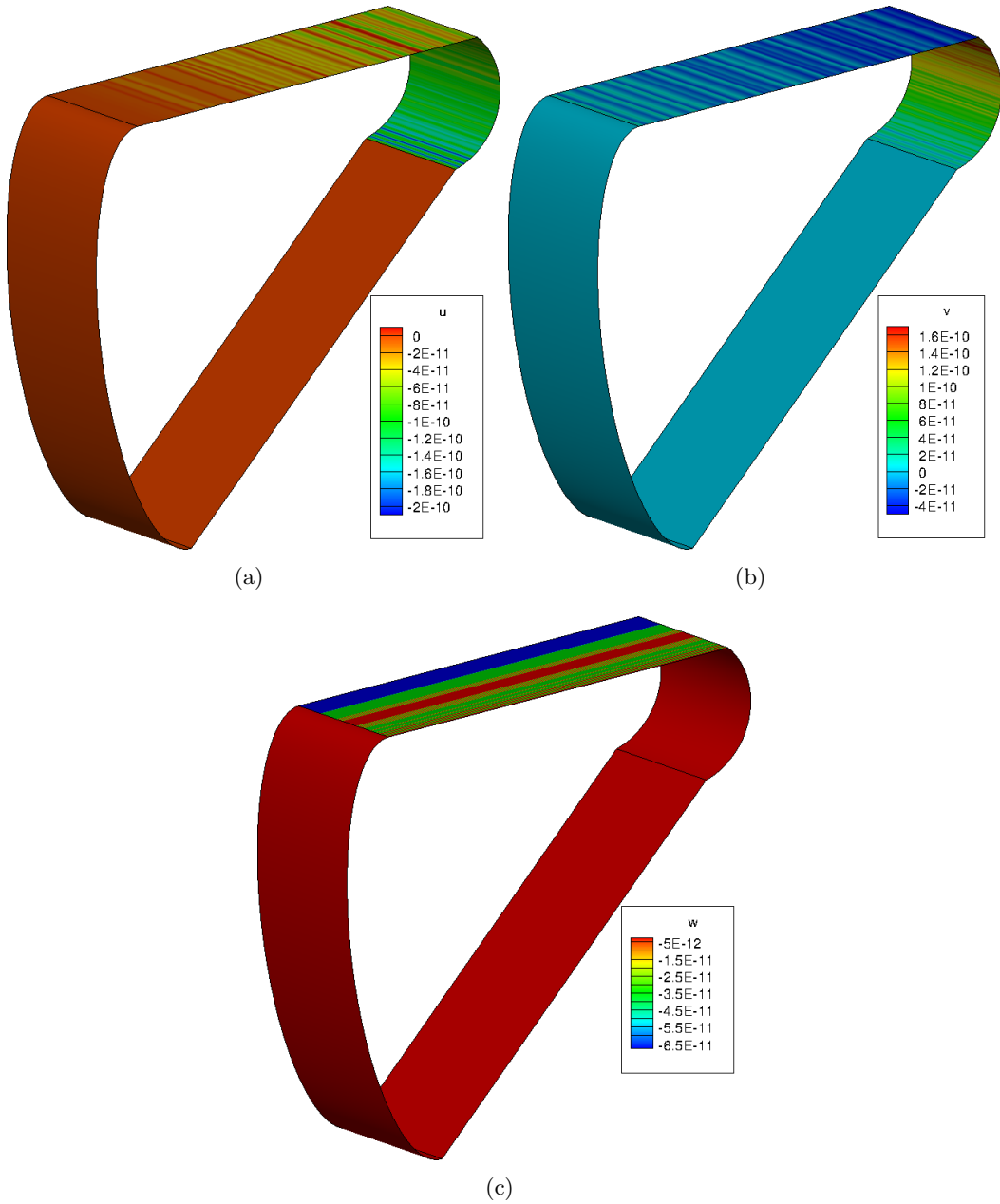


Figure C-6: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{26}$ .

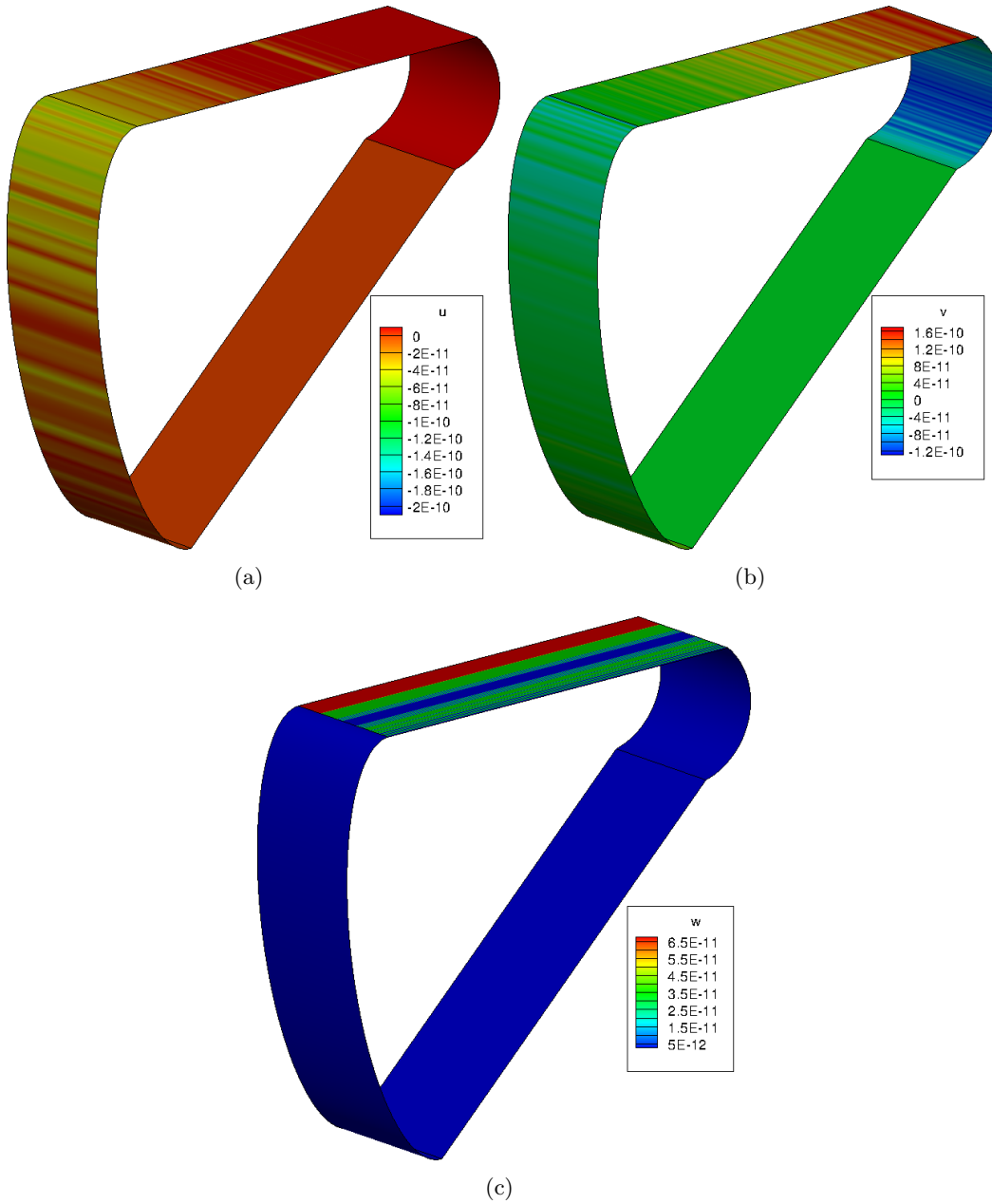


Figure C-7: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{27}$ .

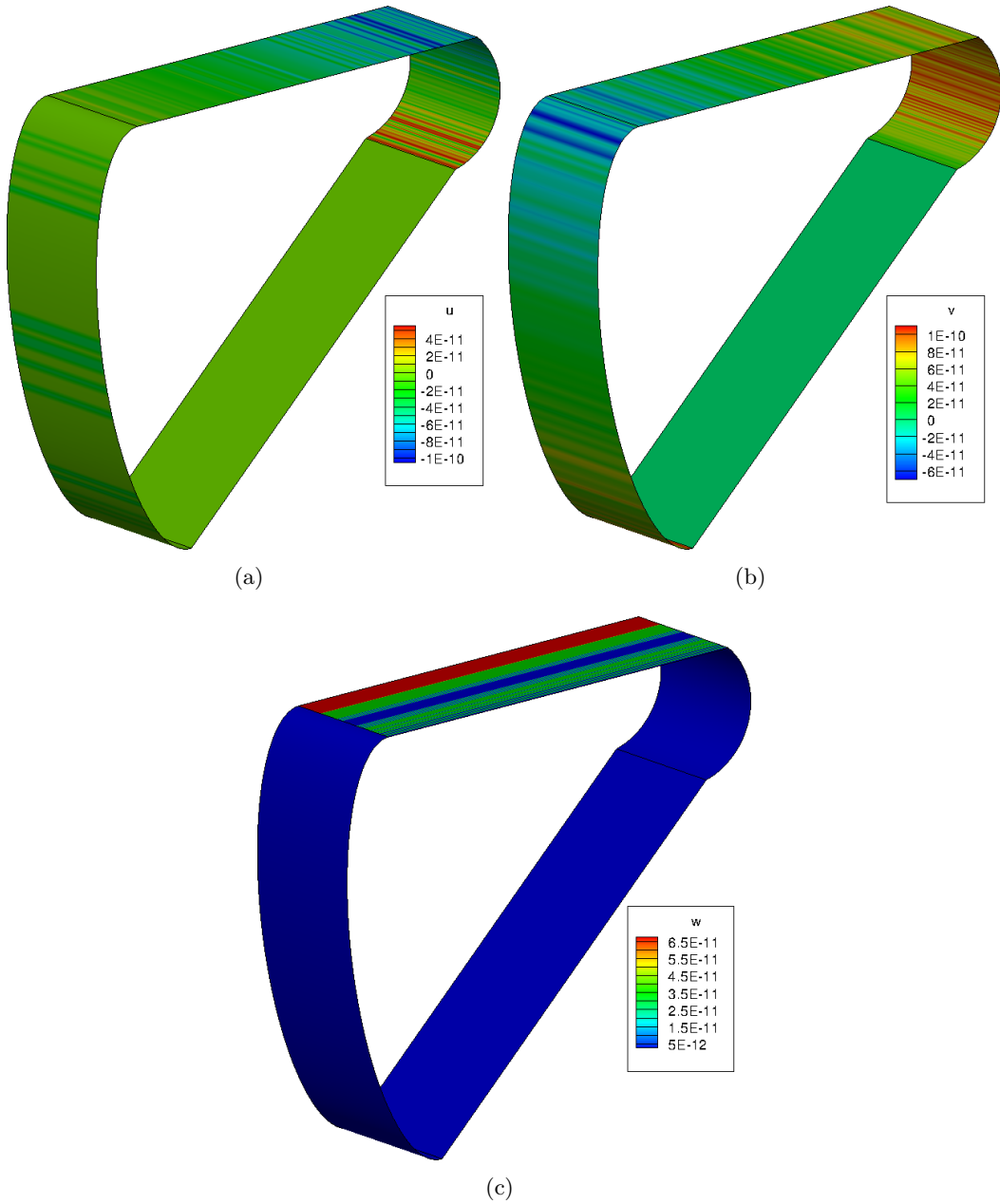


Figure C-8: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{28}$ .

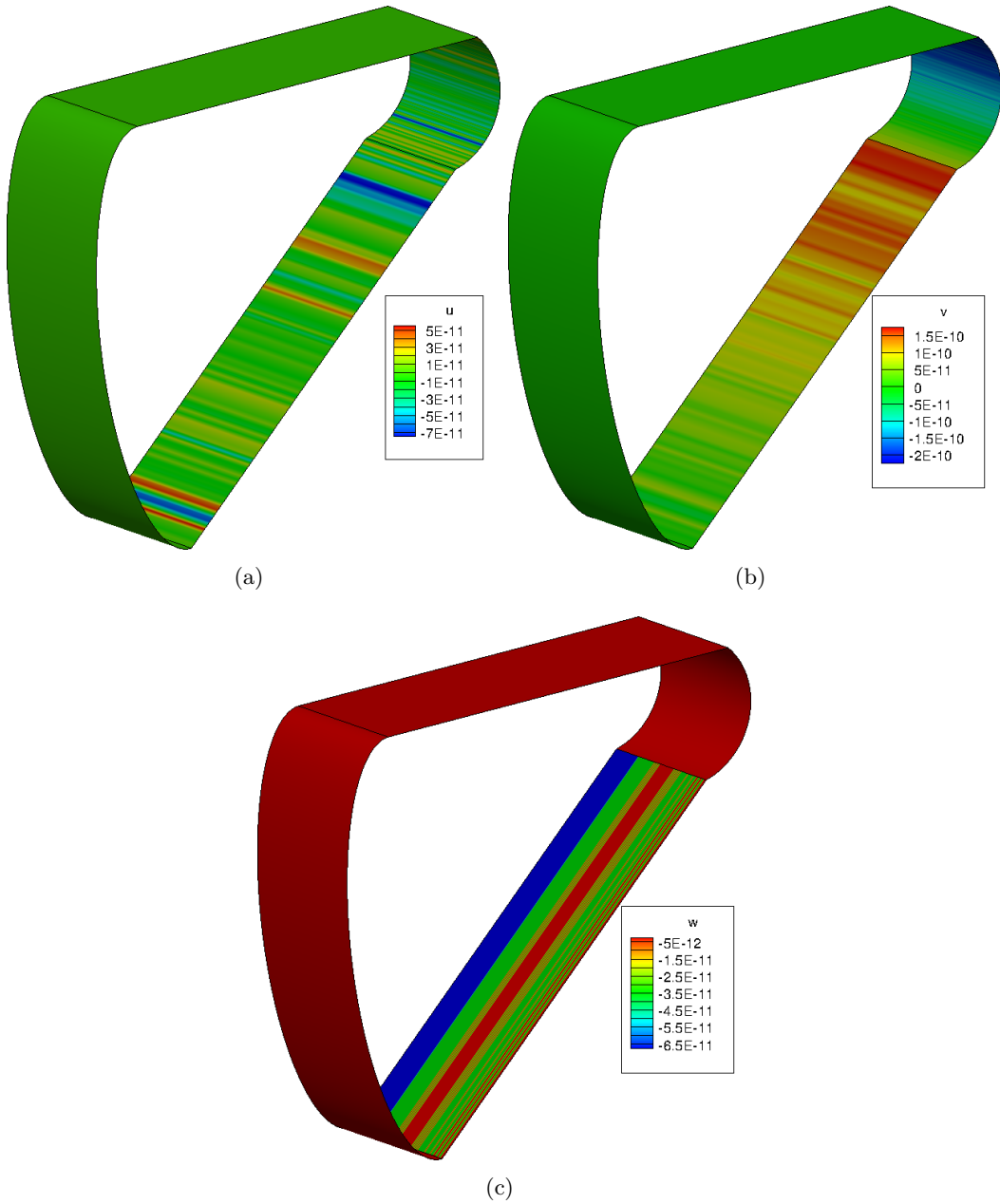


Figure C-9: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{29}$ .

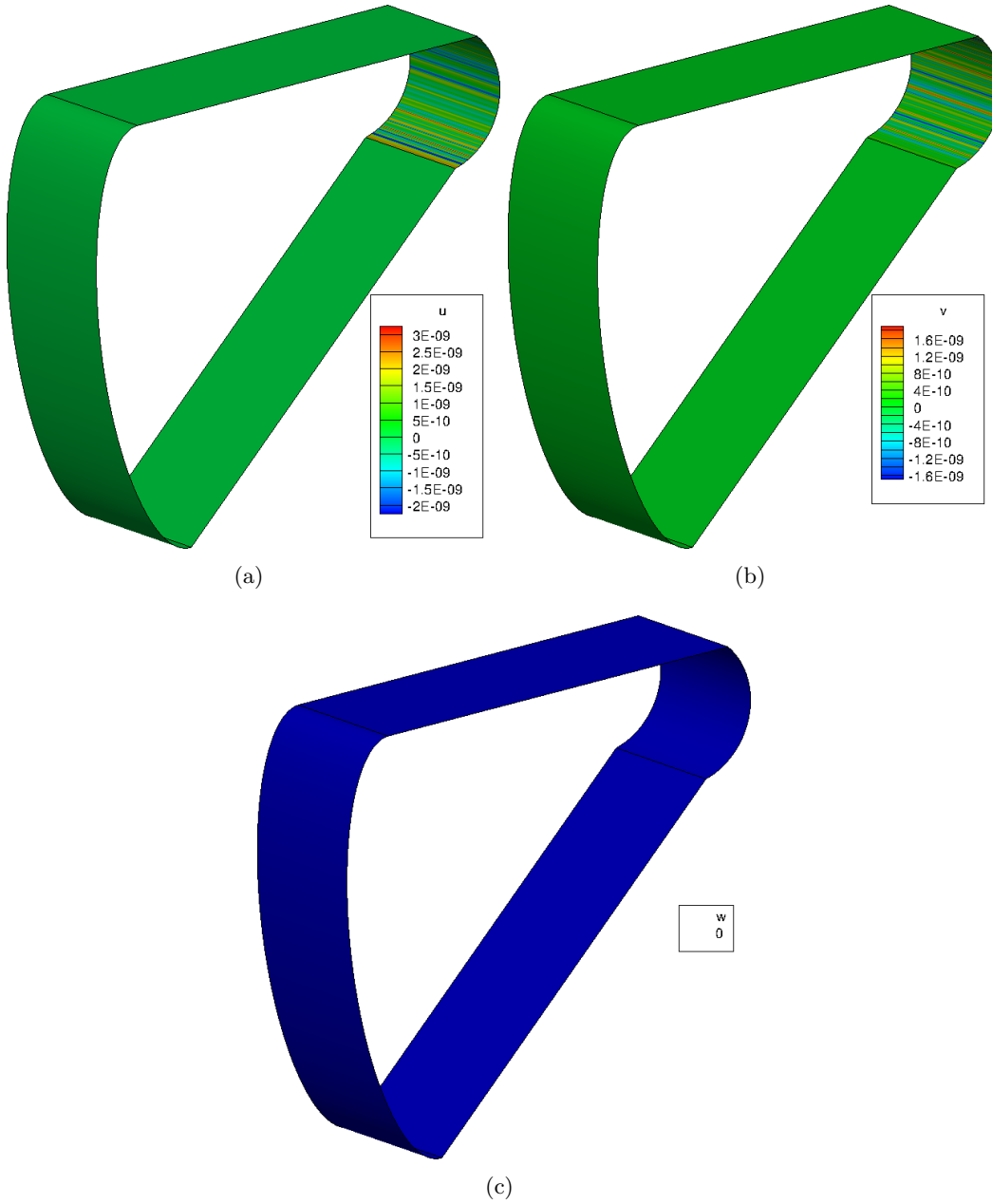


Figure C-10: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{30}$ . The  $w$ -component data is exactly zero in this case.

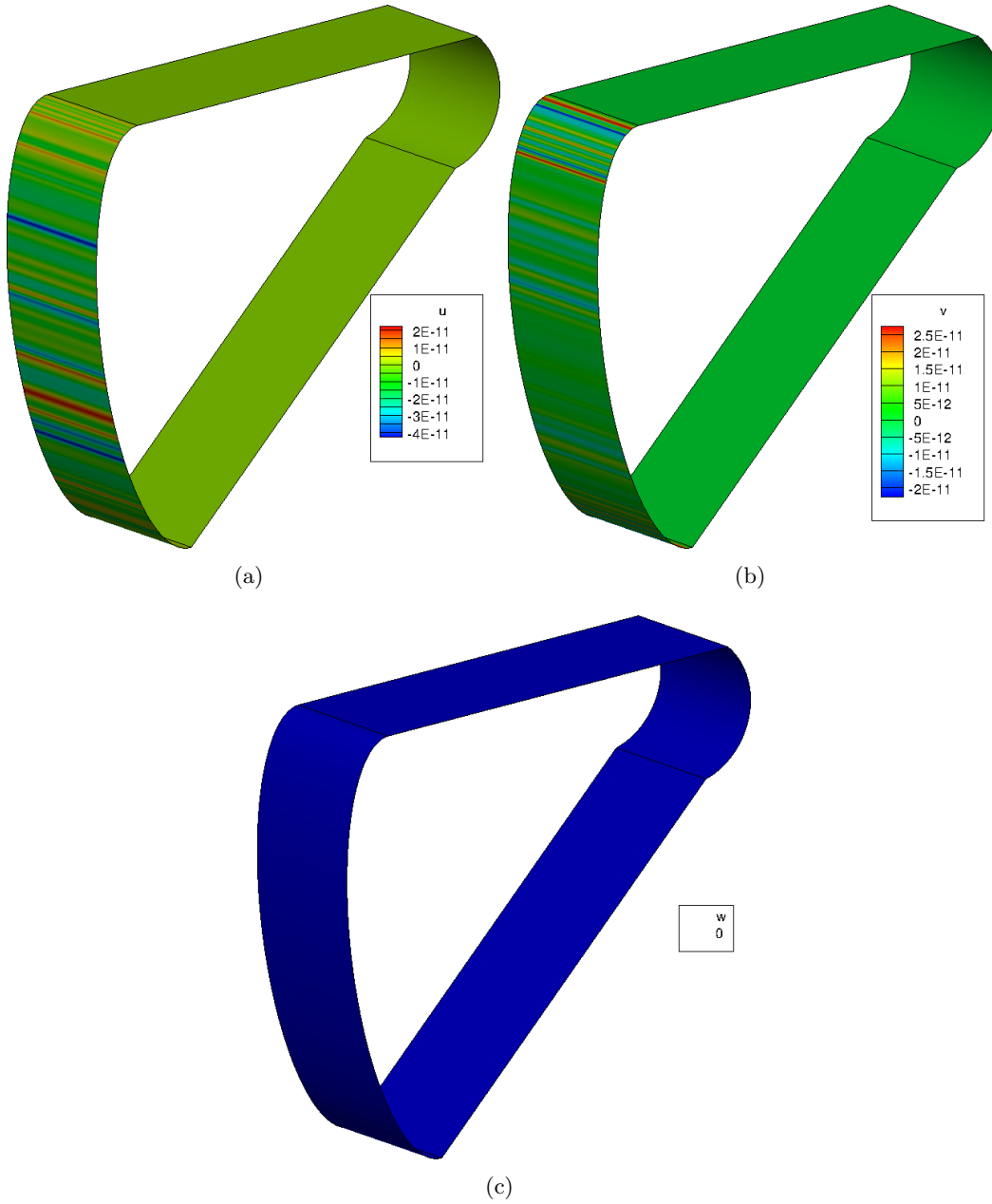
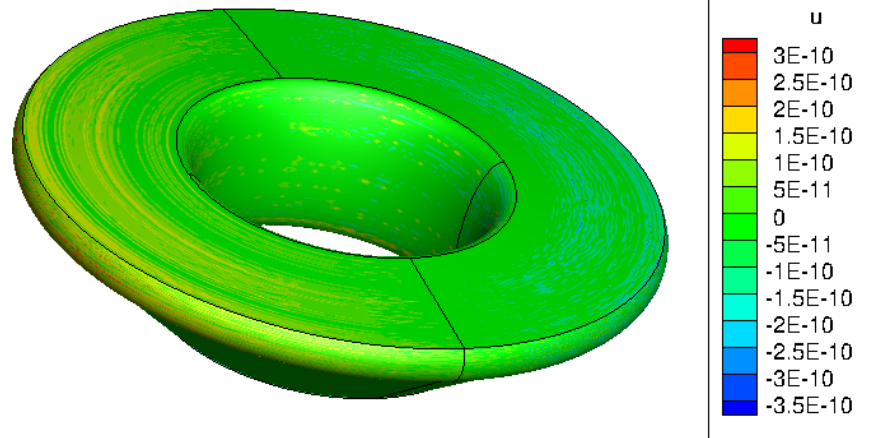
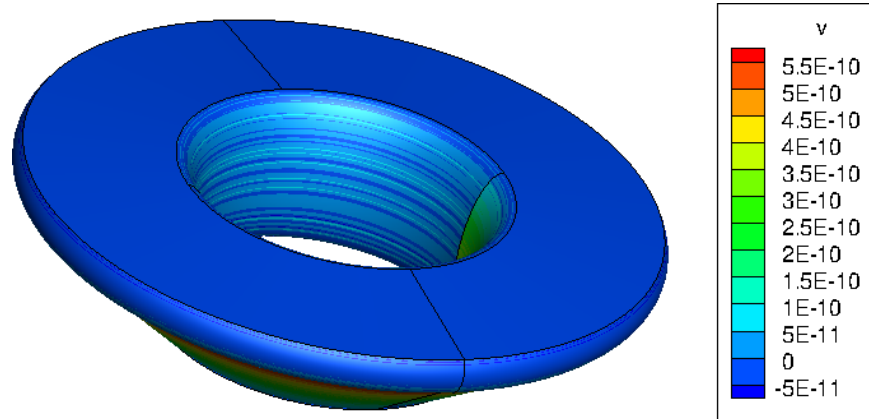


Figure C-11: Relative offset between the extrude *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{31}$ . The  $w$ -component data is exactly zero in this case.

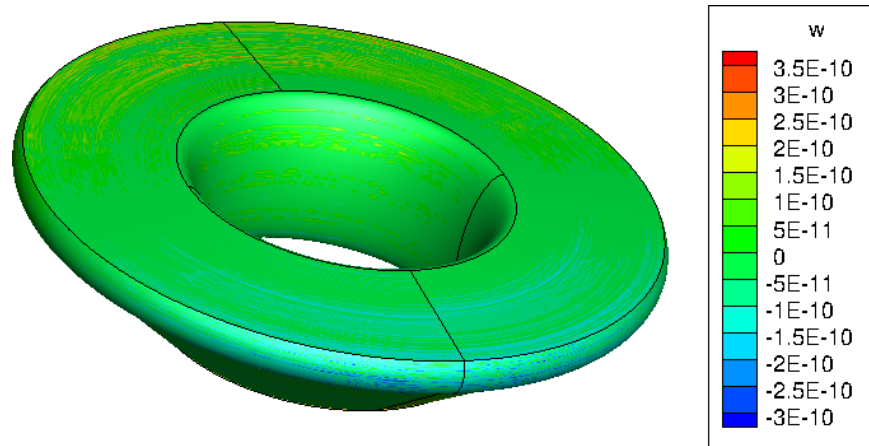




(a)

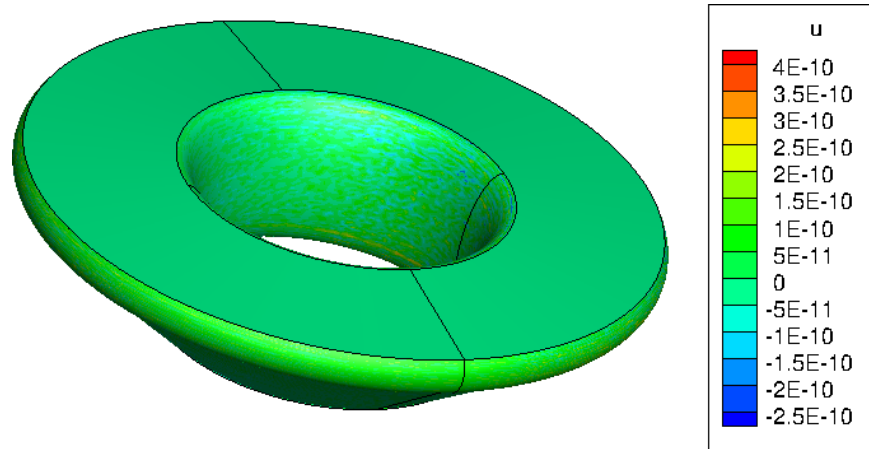


(b)

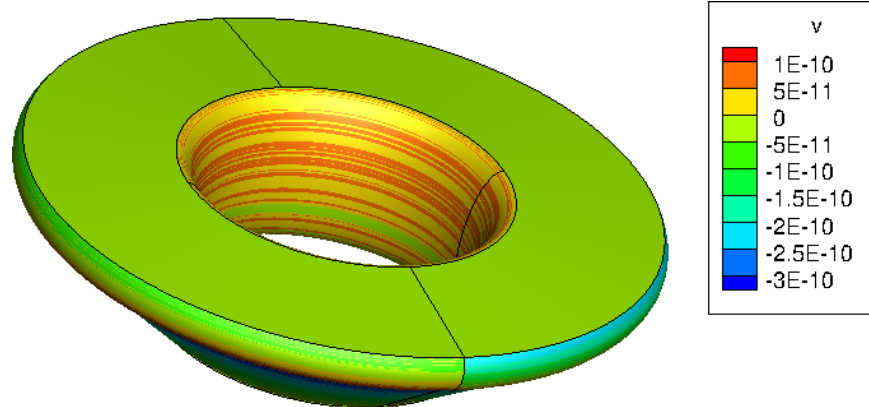


(c)

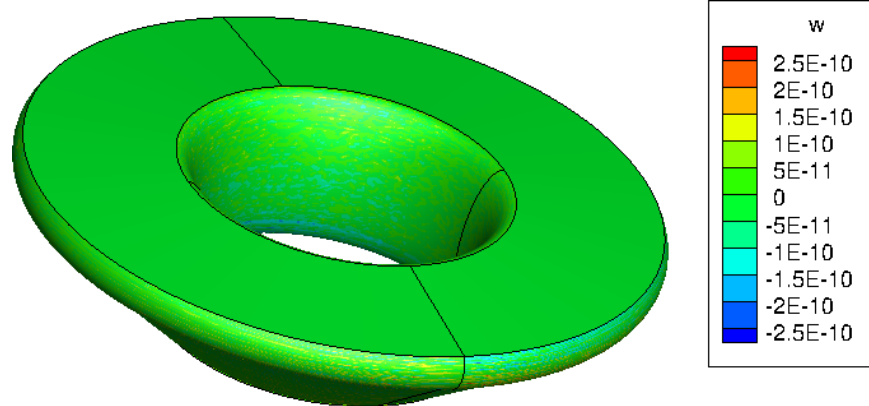
Figure C-12: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{21}$ .



(a)

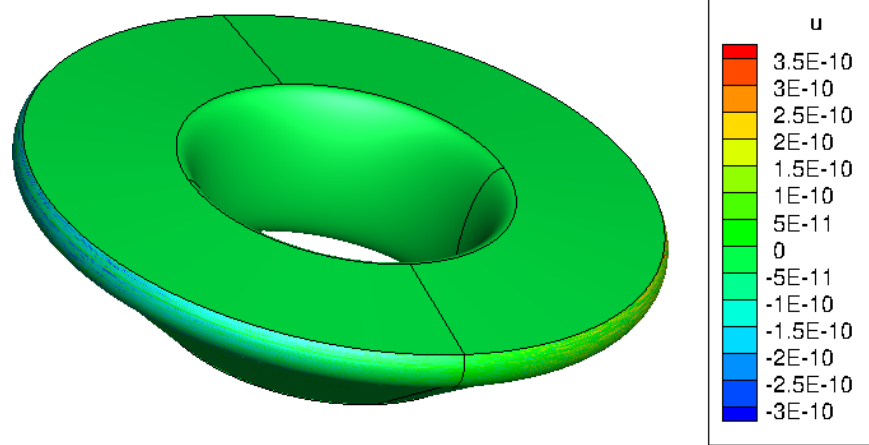


(b)

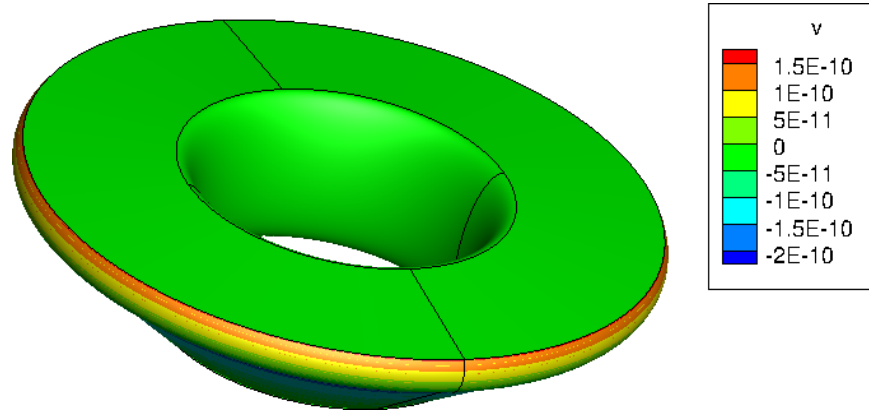


(c)

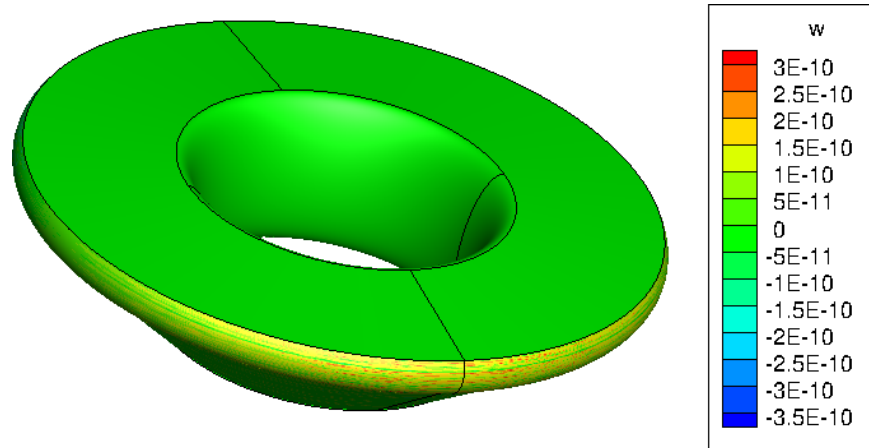
Figure C-13: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{22}$ .



(a)

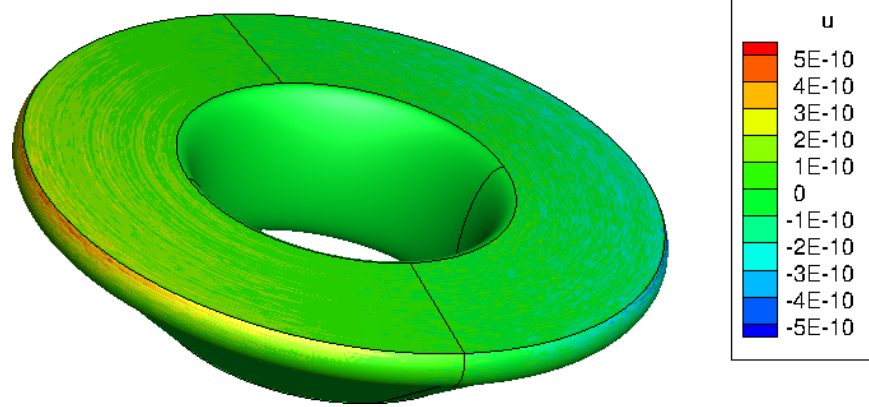


(b)

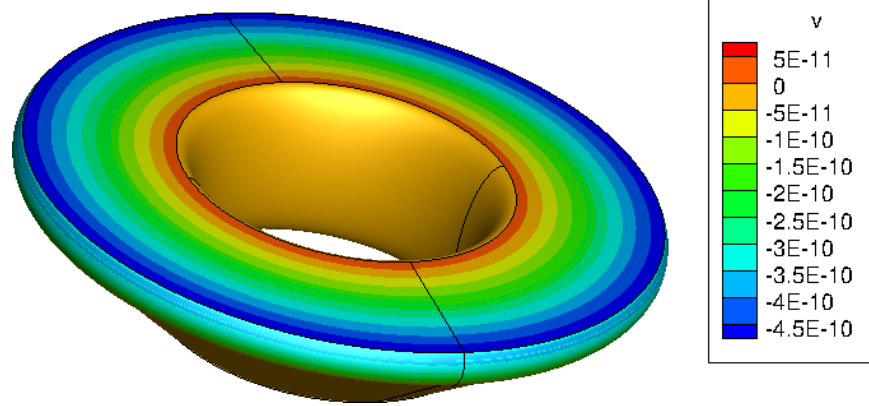


(c)

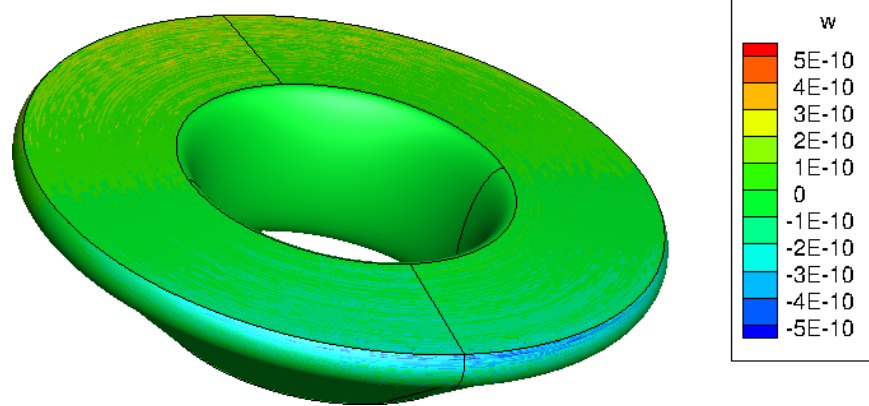
Figure C-14: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{23}$ .



(a)

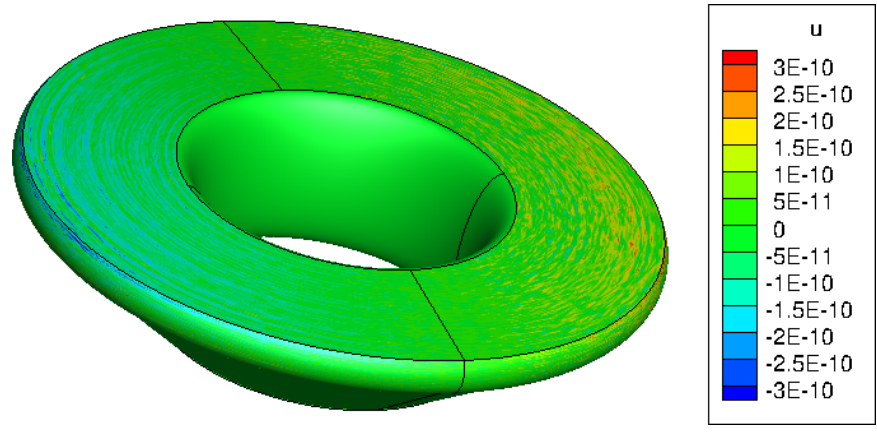


(b)

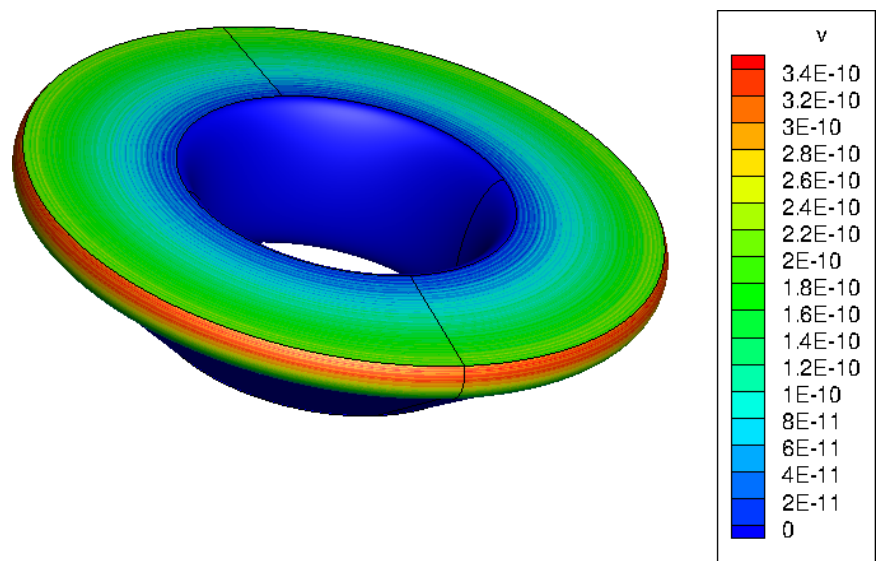


(c)

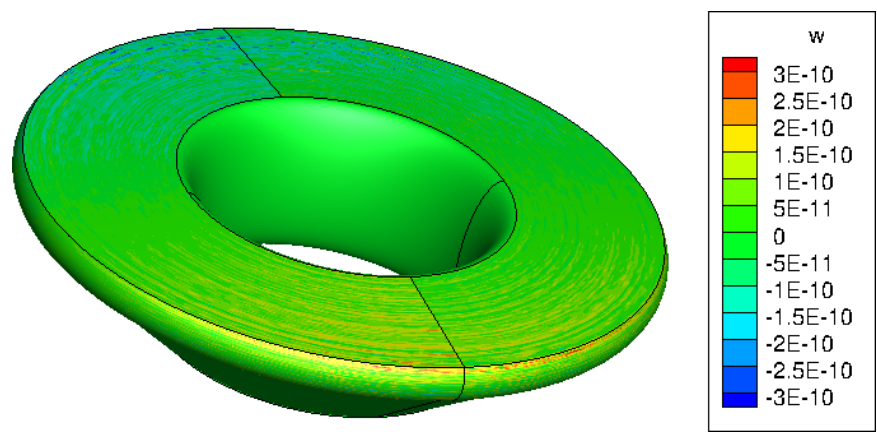
Figure C-15: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{25}$ .



(a)

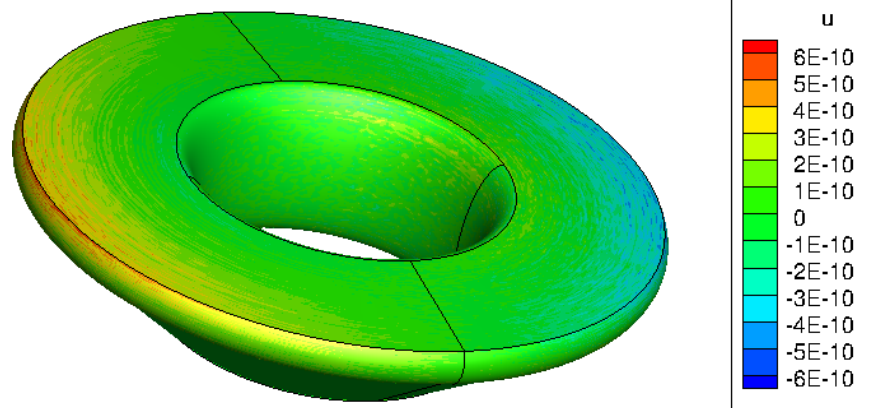


(b)

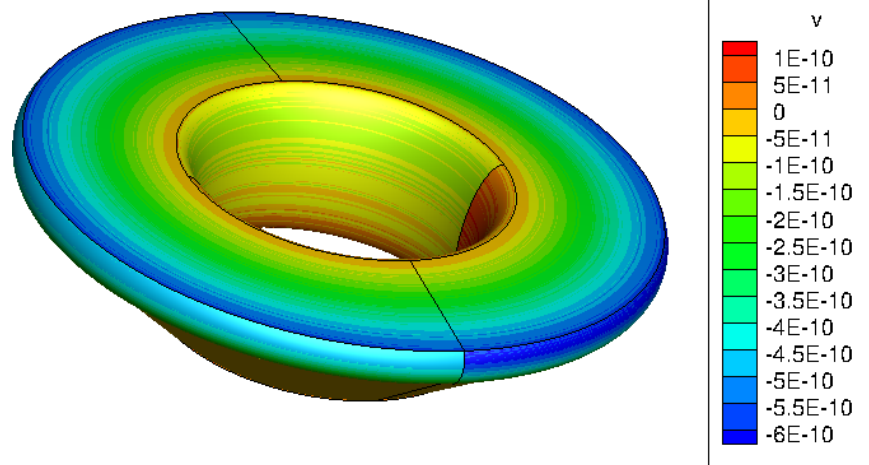


(c)

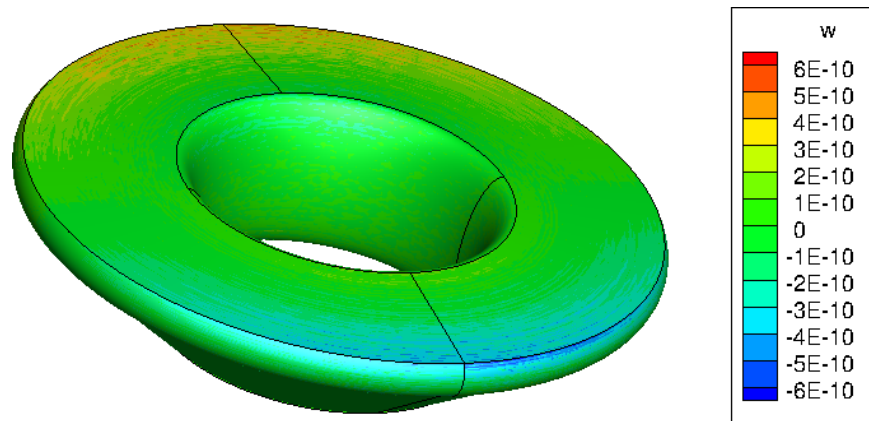
Figure C-16: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{26}$ .



(a)



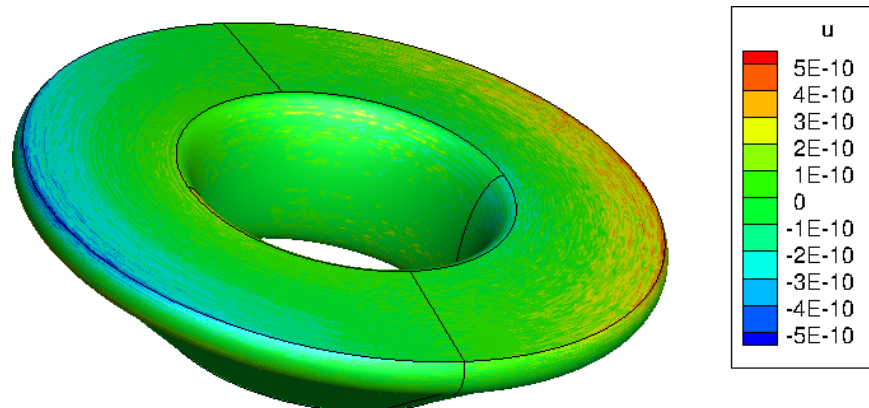
(b)



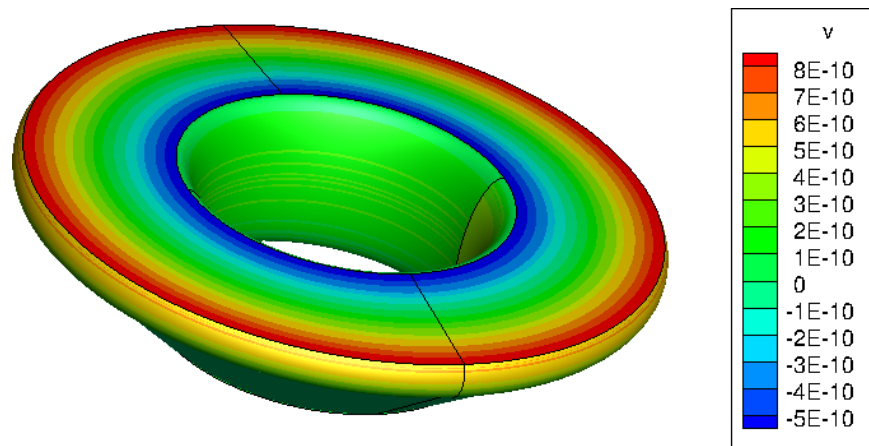
(c)

Figure C-17: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{27}$ .

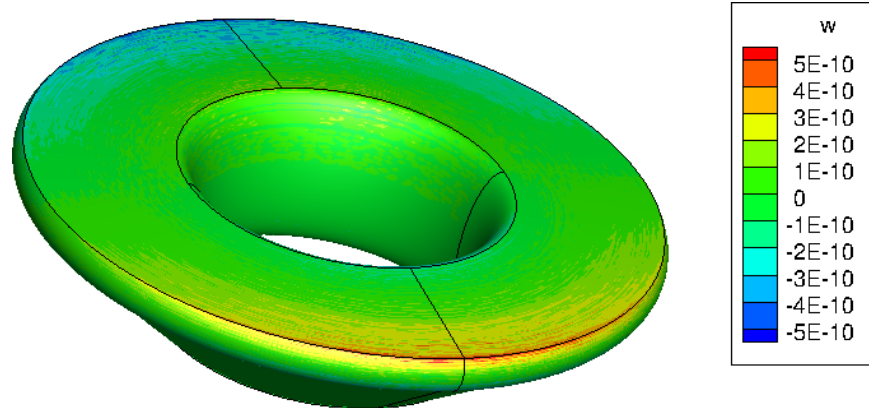




(a)

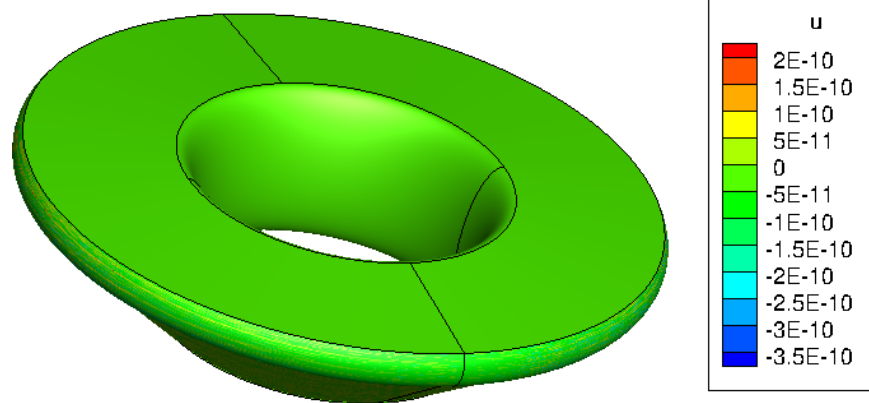


(b)

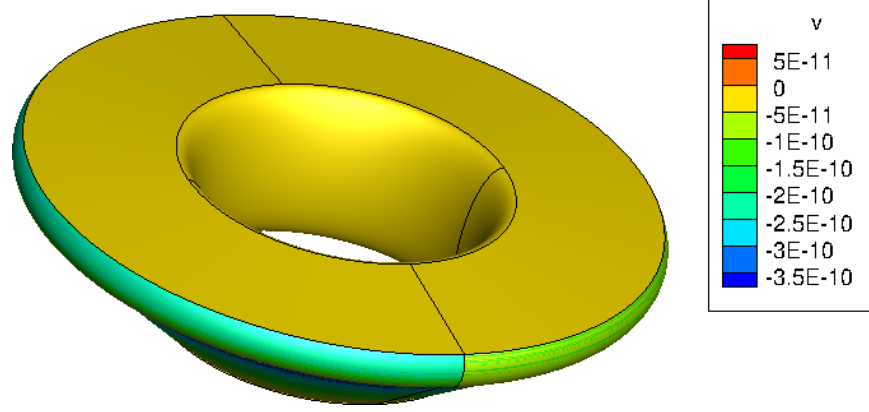


(c)

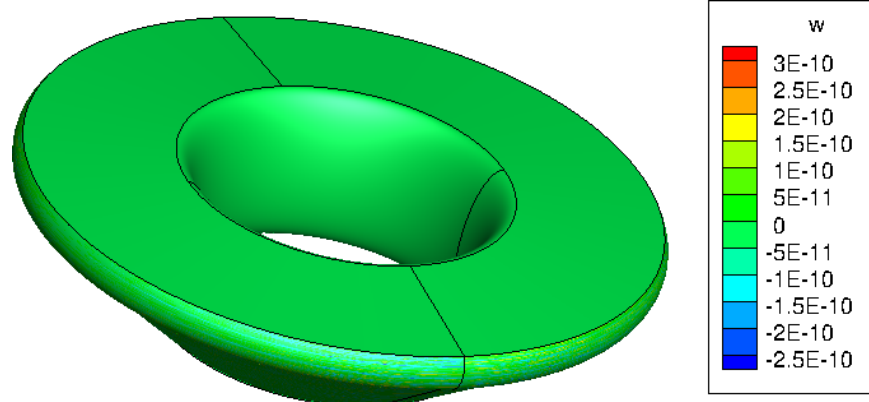
Figure C-18: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{28}$ .



(a)



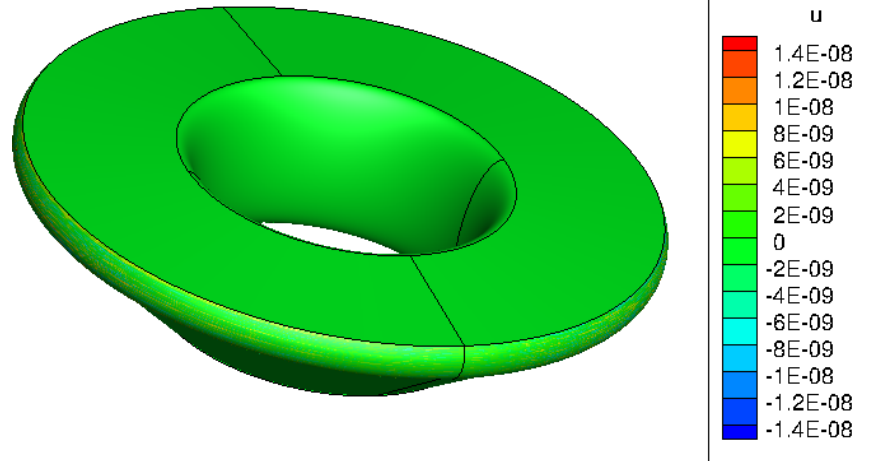
(b)



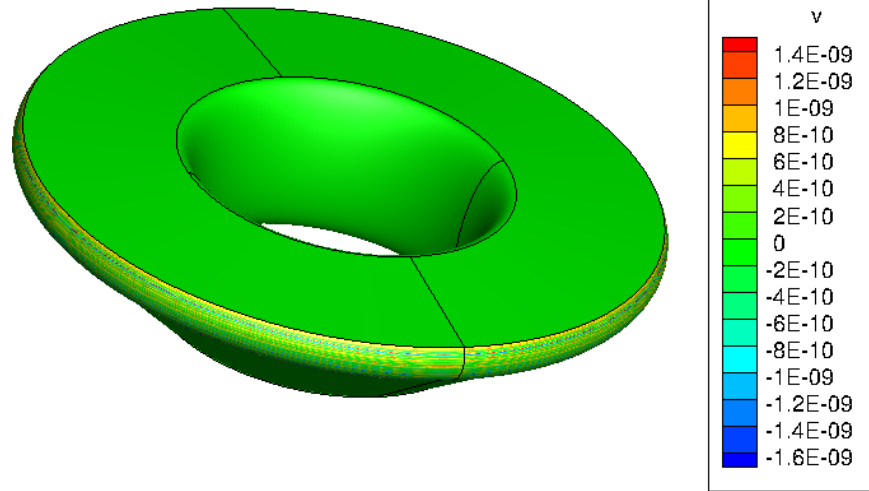
(c)

Figure C-19: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{29}$ .

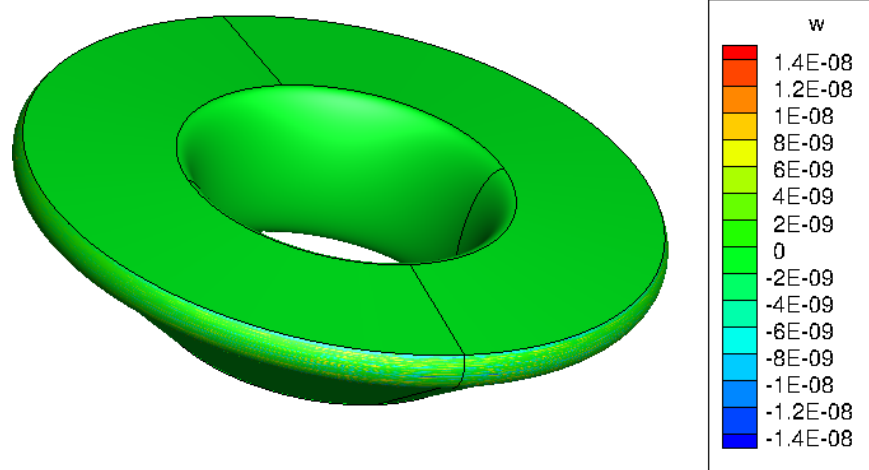




(a)



(b)



(c)

Figure C-20: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{30}$ .

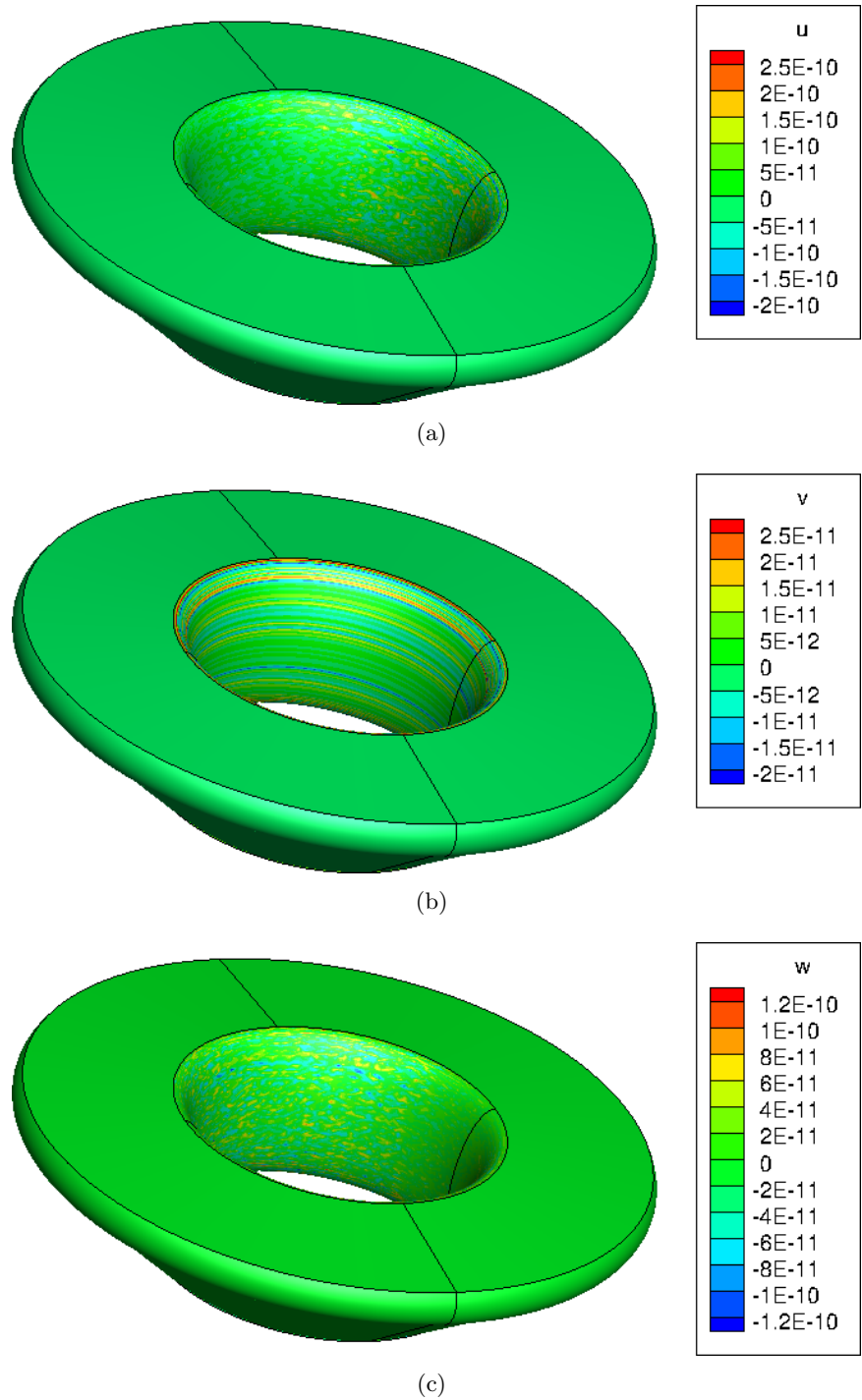
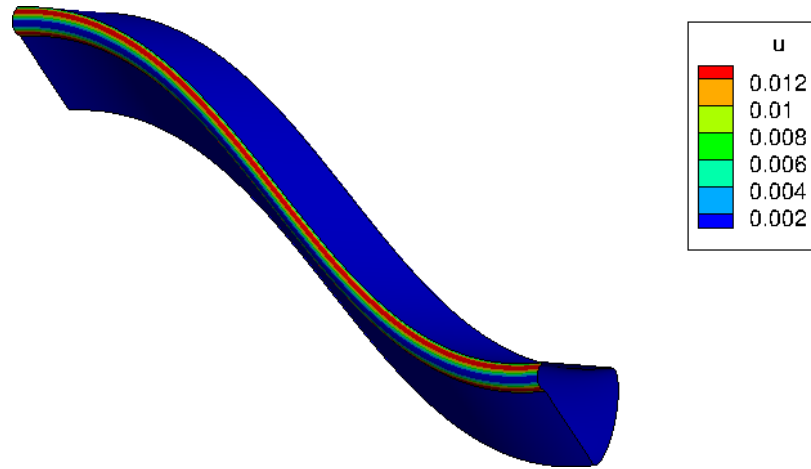
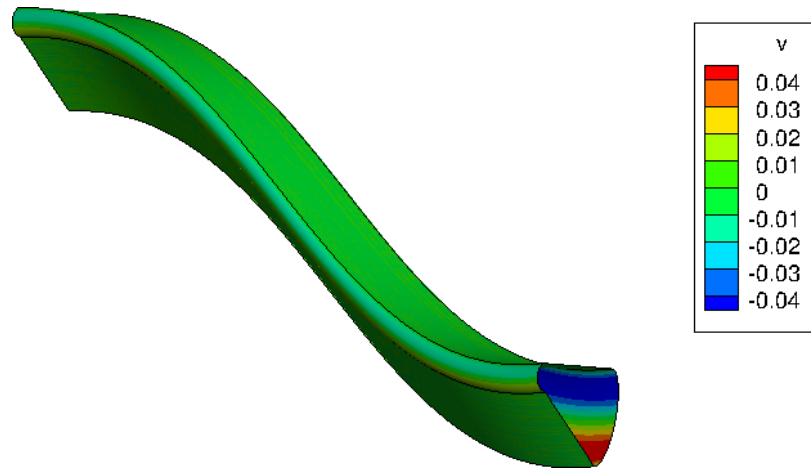


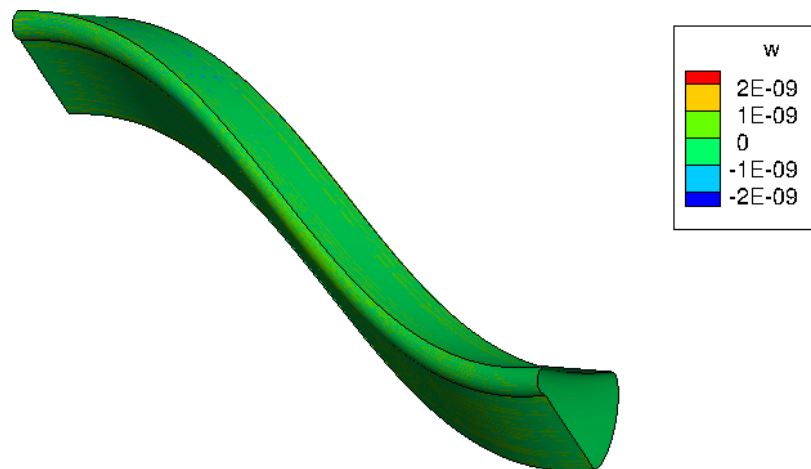
Figure C-21: Relative offset between the revolve *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{31}$ .



(a)

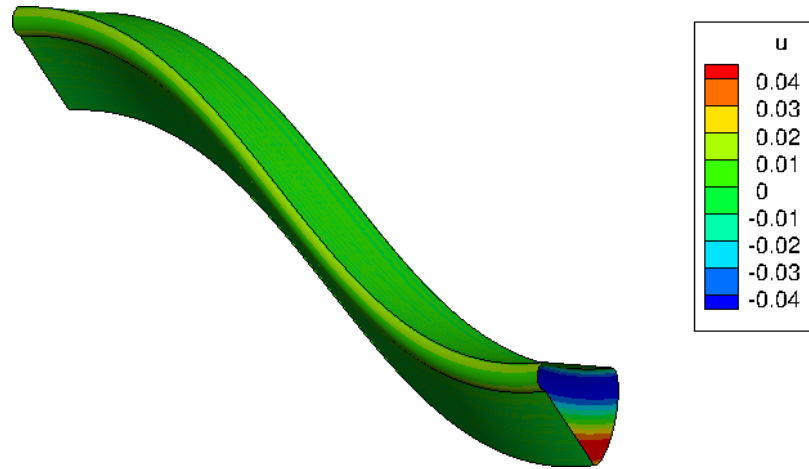


(b)

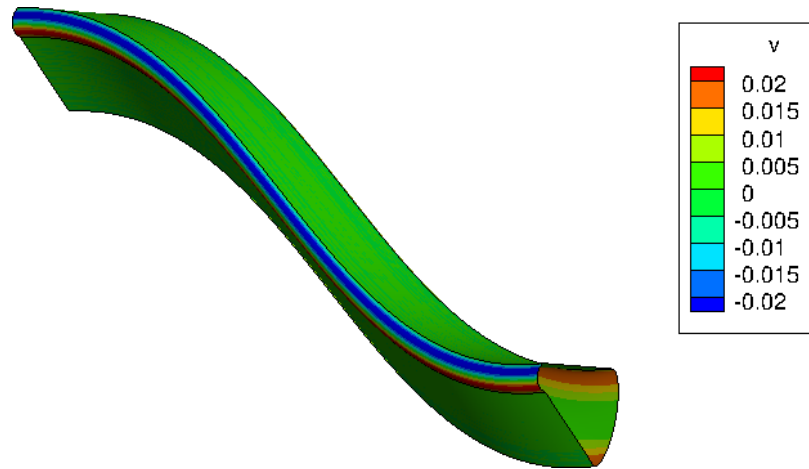


(c)

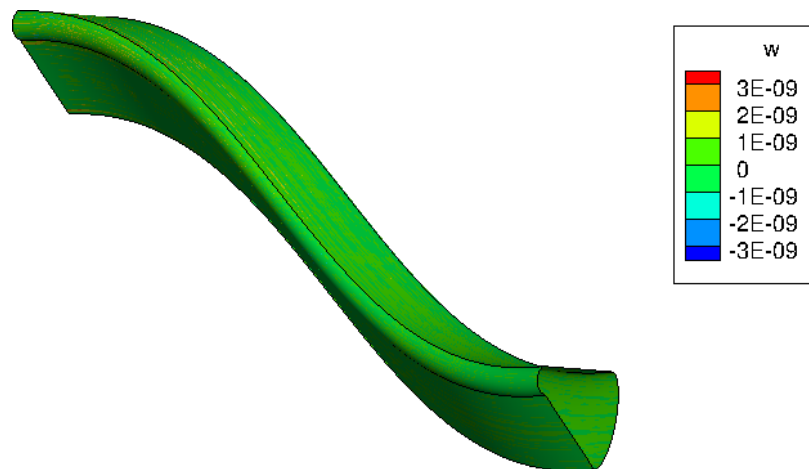
Figure C-22: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{21}$ .



(a)

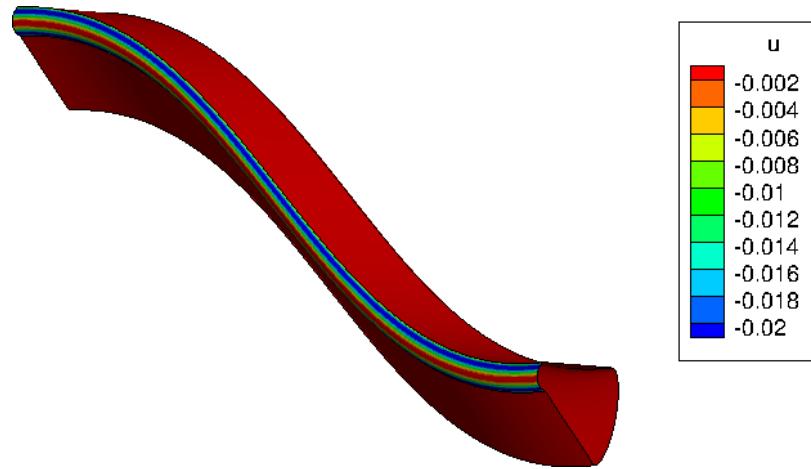


(b)

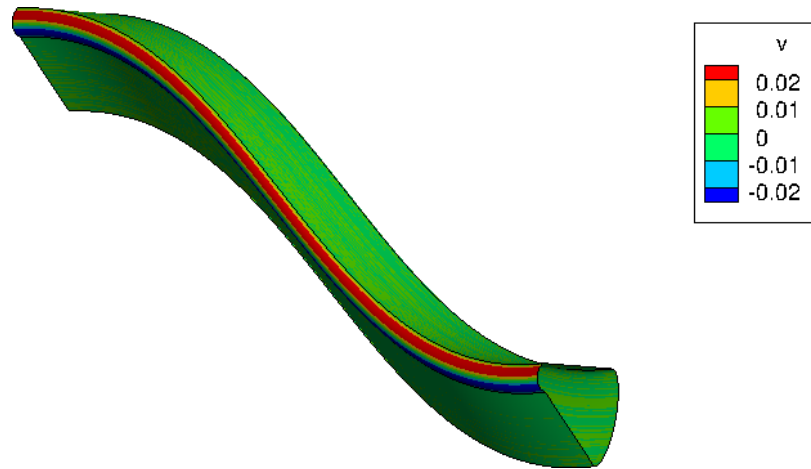


(c)

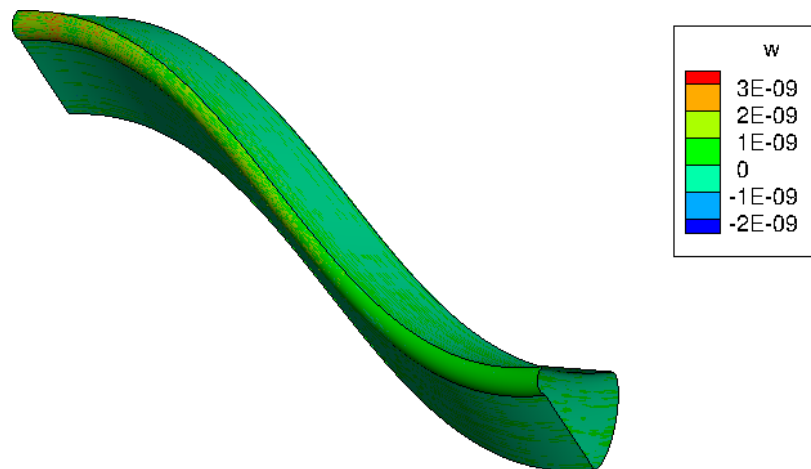
Figure C-23: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{22}$ .



(a)

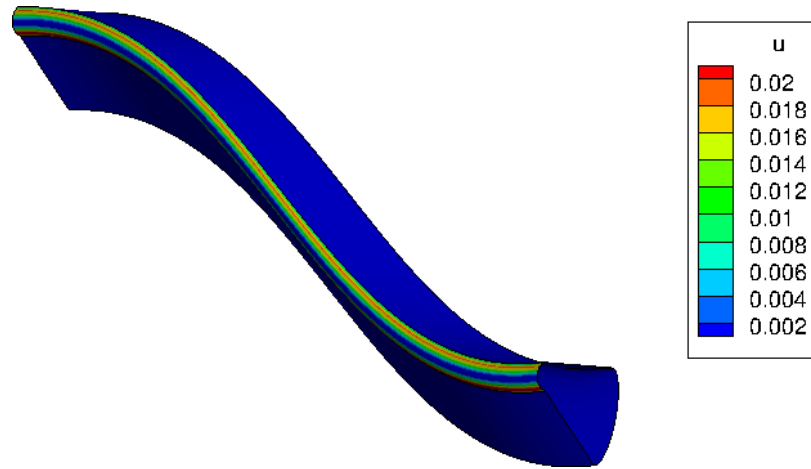


(b)

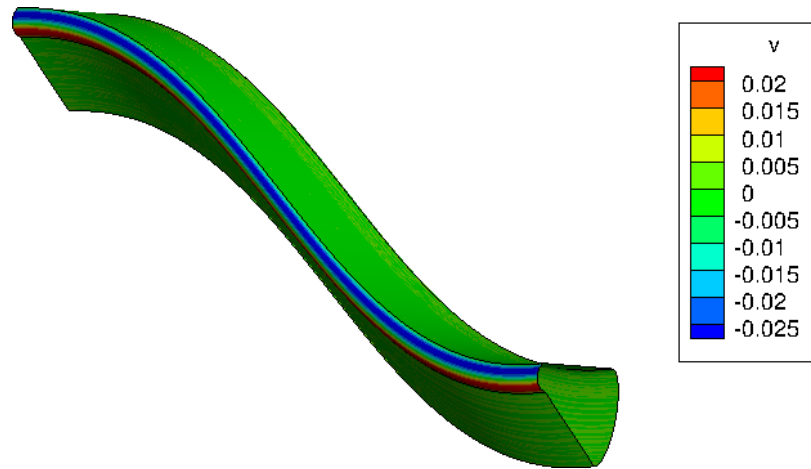


(c)

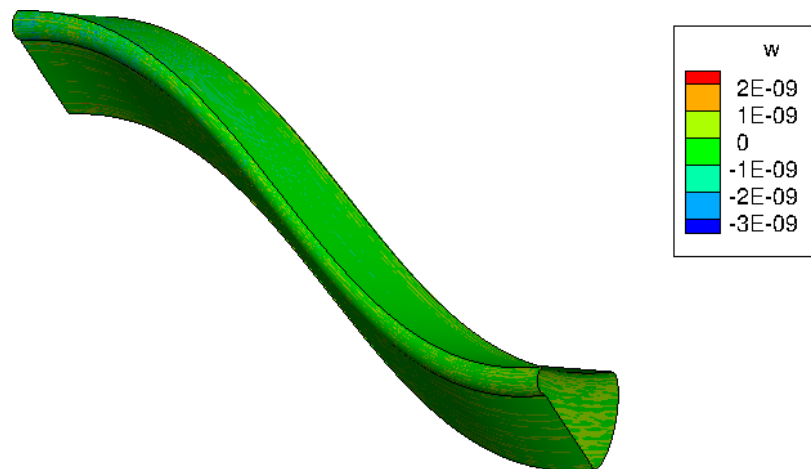
Figure C-24: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{23}$ .



(a)

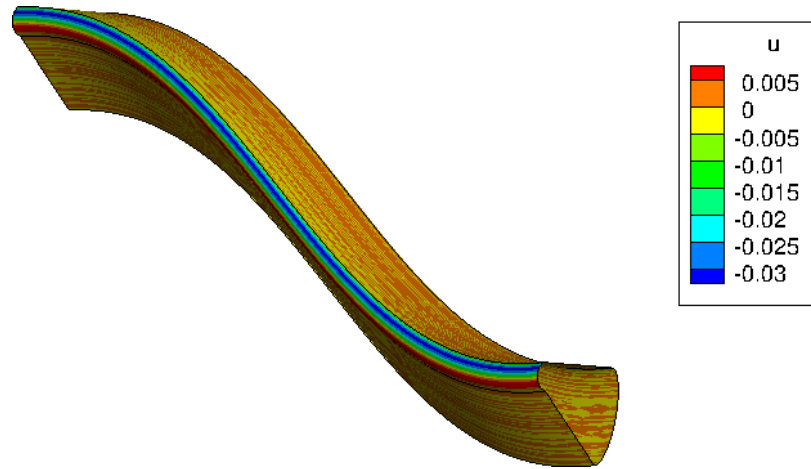


(b)

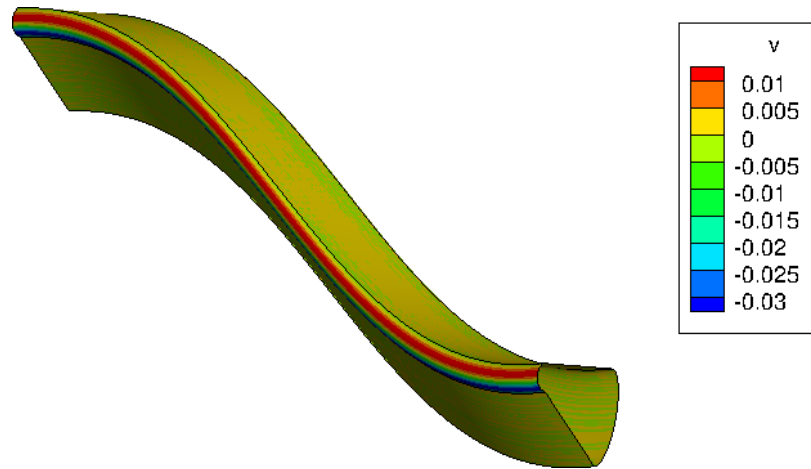


(c)

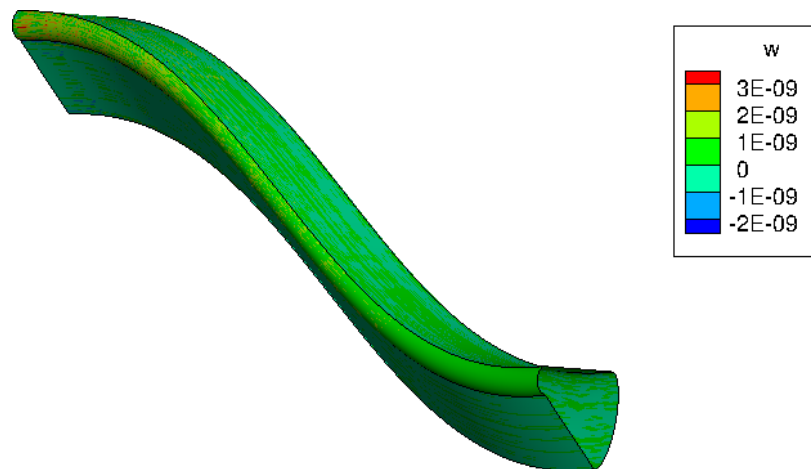
Figure C-25: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{25}$ .



(a)

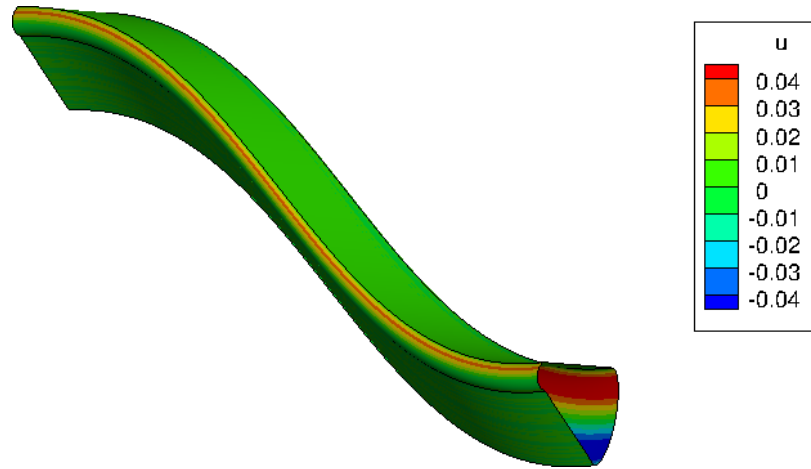


(b)

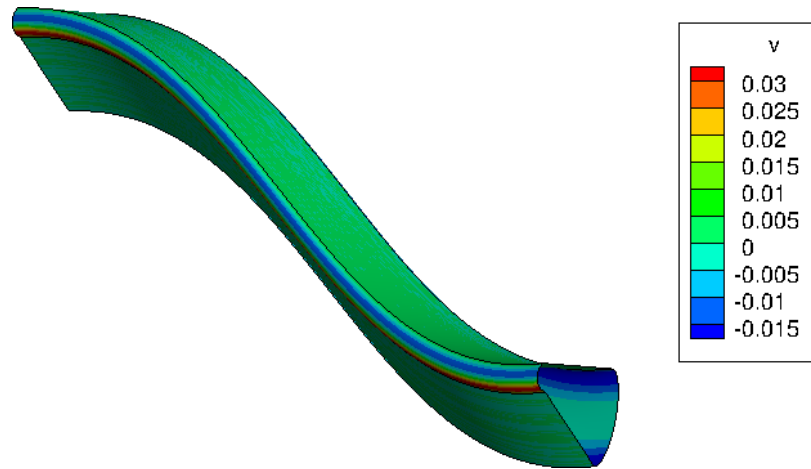


(c)

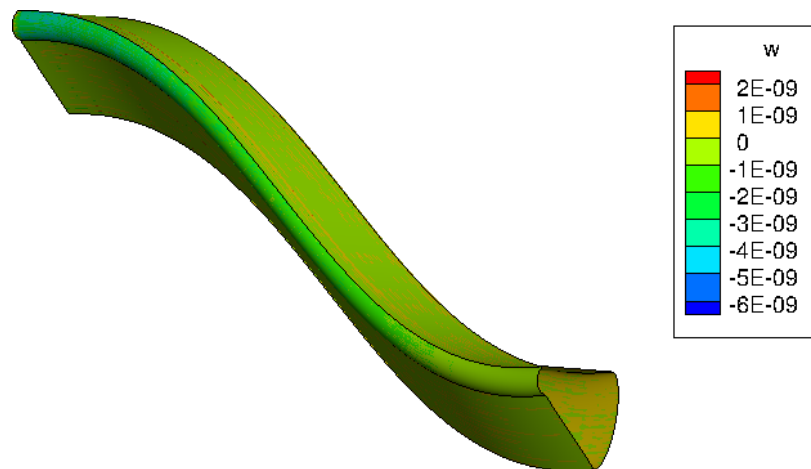
Figure C-26: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{26}$ .



(a)



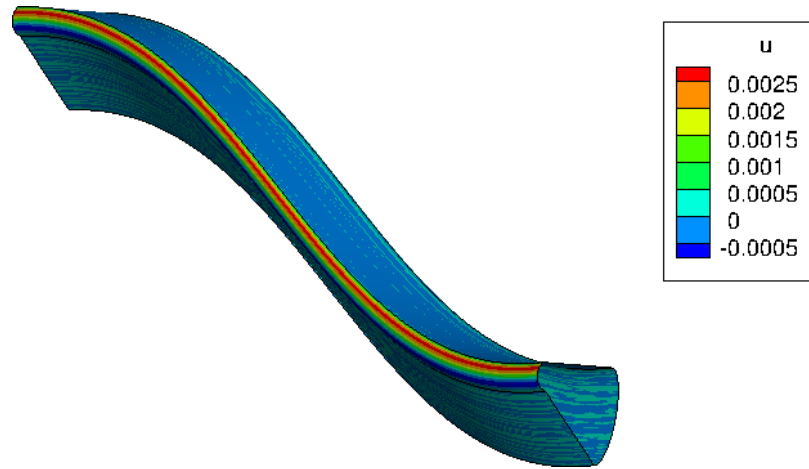
(b)



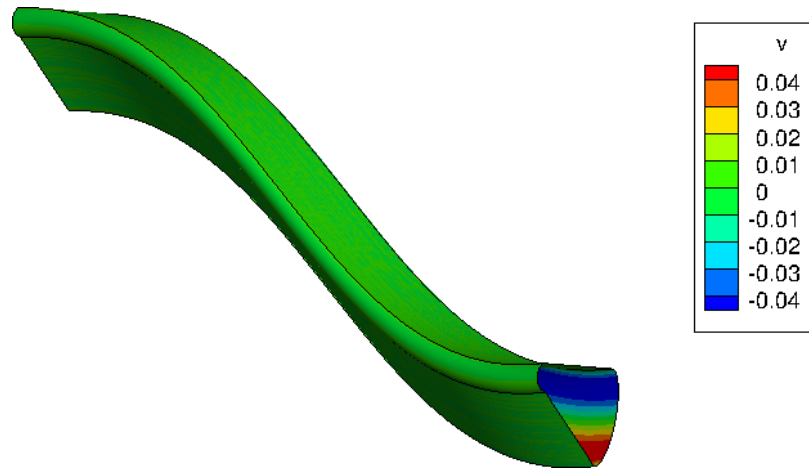
(c)

Figure C-27: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{27}$ .

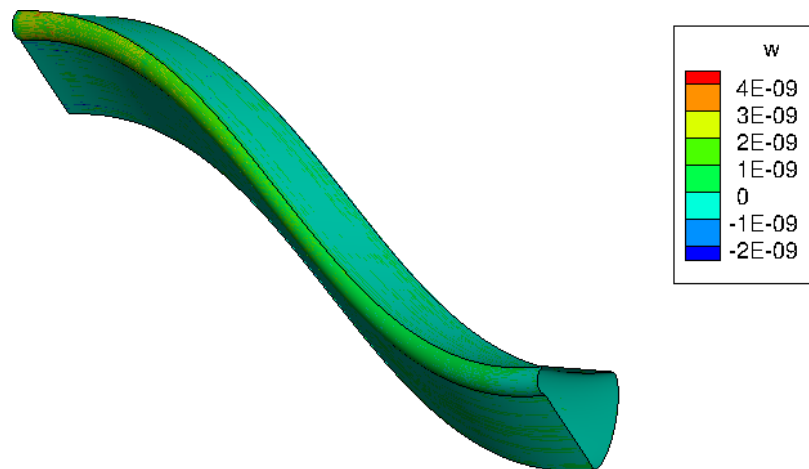




(a)

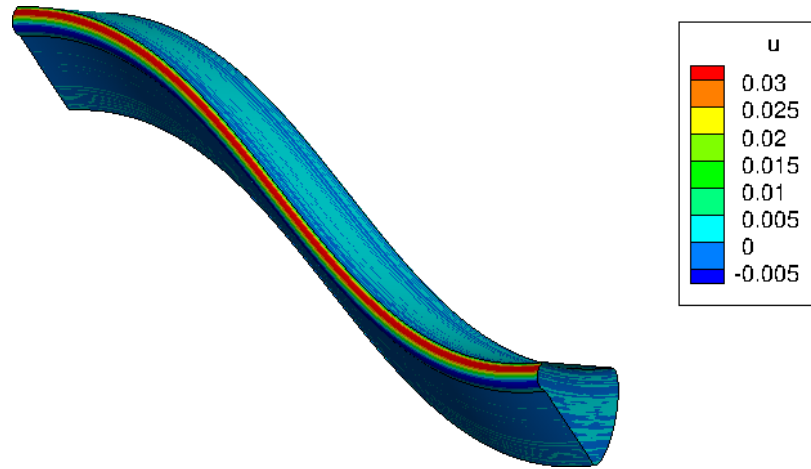


(b)

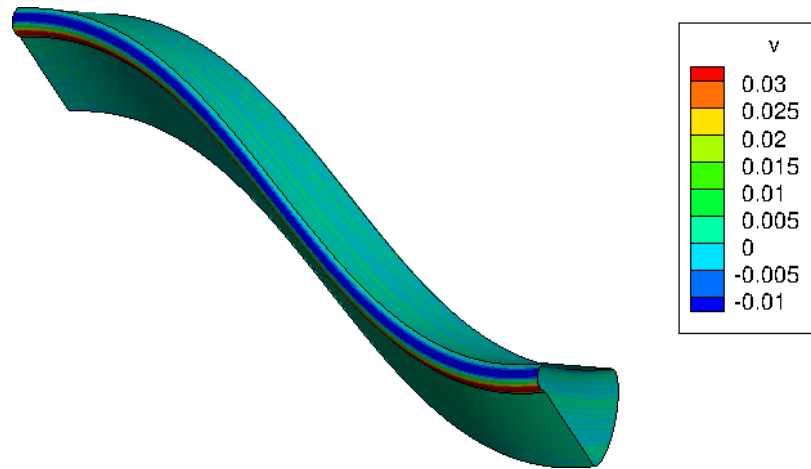


(c)

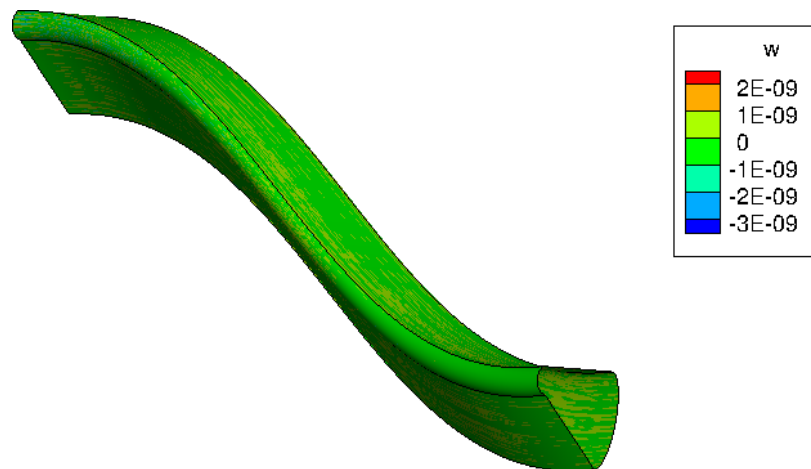
Figure C-28: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{28}$ .



(a)

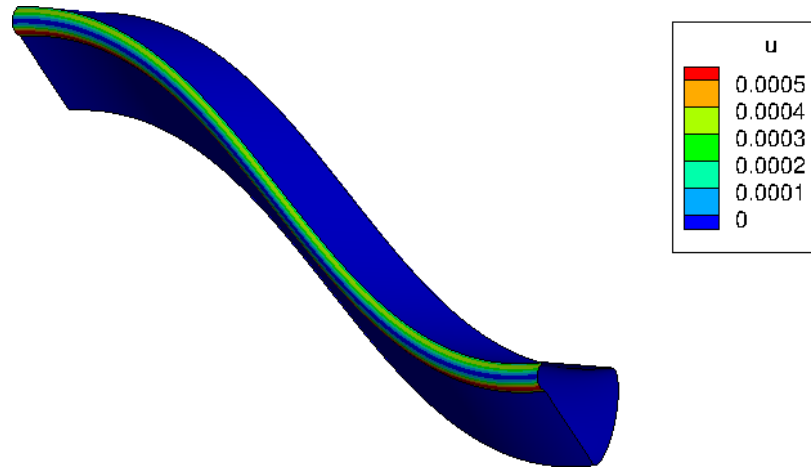


(b)

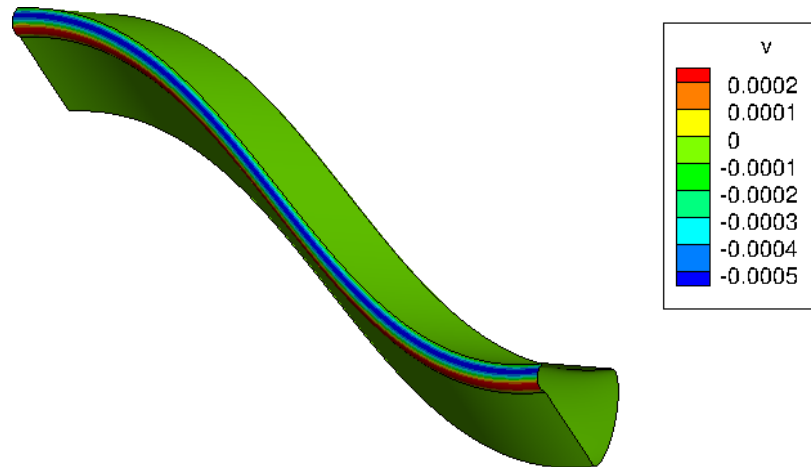


(c)

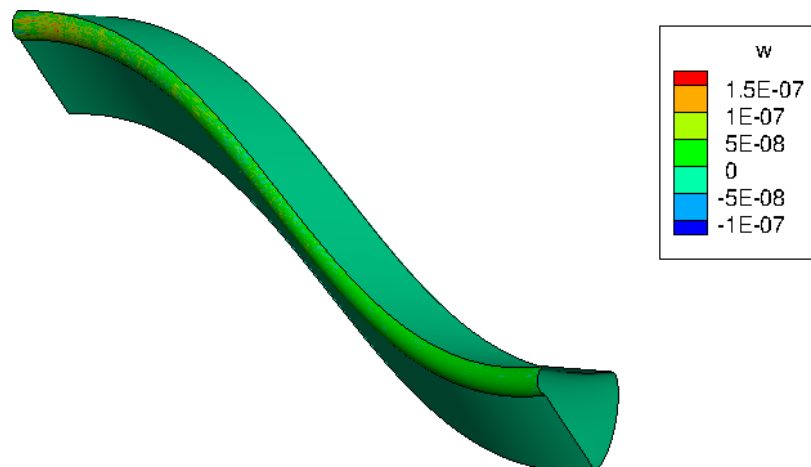
Figure C-29: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{29}$ .



(a)

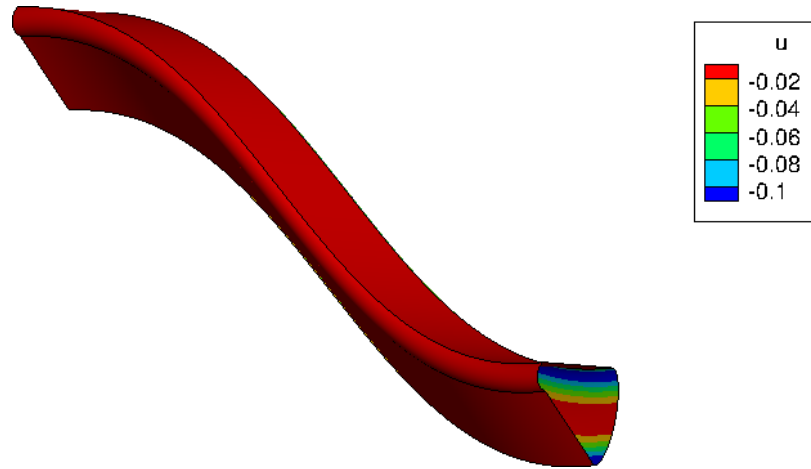


(b)

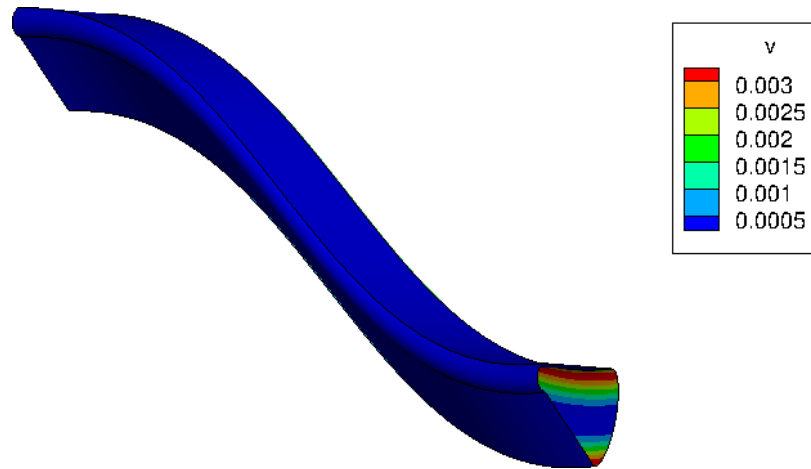


(c)

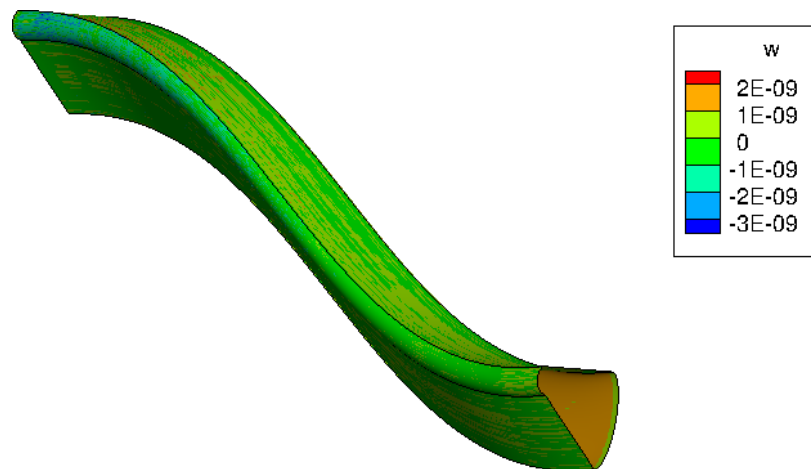
Figure C-30: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{30}$ .



(a)



(b)



(c)

Figure C-31: Relative offset between the sweep *feature* differentiation method and centered-differencing for parameter  $\mathcal{P} = d_{31}$ .

## Appendix D

# B-Spline Surface

# Perturbation Results

The following tables refer to changes in knot vectors for faces on a sweep feature analyzed in Section 3.5. The changing knot vectors appear in finite-difference perturbations that lead to spurious design velocity results.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.46689 <u>1102701769</u>	0.466890 <u>566530759</u>
	0.466890834616315	0.46689 <u>1102701769</u>	0.466890 <u>566530759</u>
	0.466890834616315	0.46689 <u>1102701769</u>	0.466890 <u>566530759</u>
	0.466890834616315	0.46689 <u>1102701769</u>	0.466890 <u>566530759</u>
	0.596167182504661	0.596167 <u>385475848</u>	0.596166979533398
	0.596167182504661	0.596167 <u>385475848</u>	0.596166979533398
	0.596167182504661	0.596167 <u>385475848</u>	0.596166979533398
	0.596167182504661	0.596167 <u>385475848</u>	0.596166979533398
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.116902 <u>836780837</u>	0.116901948793038
	0.116902392787021	0.116902 <u>836780837</u>	0.116901948793038
	0.116902392787021	0.116902 <u>836780837</u>	0.116901948793038
	0.116902392787021	0.116902 <u>836780837</u>	0.116901948793038

Table D.1: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{21}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.46689 <u>1173803889</u>	0.466890 <u>495429981</u>
	0.466890834616315	0.46689 <u>1173803889</u>	0.466890 <u>495429981</u>
	0.466890834616315	0.46689 <u>1173803889</u>	0.466890 <u>495429981</u>
	0.466890834616315	0.46689 <u>1173803889</u>	0.466890 <u>495429981</u>
	0.596167182504661	0.596167 <u>443216163</u>	0.596166 <u>921794282</u>
	0.596167182504661	0.596167 <u>443216163</u>	0.596166 <u>921794282</u>
	0.596167182504661	0.596167 <u>443216163</u>	0.596166 <u>921794282</u>
	0.596167182504661	0.596167 <u>443216163</u>	0.596166 <u>921794282</u>
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.11690295792456 <u>3</u>	0.11690182765169 <u>3</u>
	0.116902392787021	0.11690295792456 <u>3</u>	0.11690182765169 <u>3</u>
	0.116902392787021	0.11690295792456 <u>3</u>	0.11690182765169 <u>3</u>
	0.116902392787021	0.11690295792456 <u>3</u>	0.11690182765169 <u>3</u>

Table D.2: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{22}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.466890 <u>149094674</u>	0.46689 <u>1520138963</u>
	0.466890834616315	0.466890 <u>149094674</u>	0.46689 <u>1520138963</u>
	0.466890834616315	0.466890 <u>149094674</u>	0.46689 <u>1520138963</u>
	0.466890834616315	0.466890 <u>149094674</u>	0.46689 <u>1520138963</u>
	0.596167182504661	0.596166 <u>532648647</u>	0.596167 <u>832361366</u>
	0.596167182504661	0.596166 <u>532648647</u>	0.596167 <u>832361366</u>
	0.596167182504661	0.596166 <u>532648647</u>	0.596167 <u>832361366</u>
	0.596167182504661	0.596166 <u>532648647</u>	0.596167 <u>832361366</u>
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.11690 <u>1610708219</u>	0.11690 <u>3174867686</u>
	0.116902392787021	0.11690 <u>1610708219</u>	0.11690 <u>3174867686</u>
	0.116902392787021	0.11690 <u>1610708219</u>	0.11690 <u>3174867686</u>
	0.116902392787021	0.11690 <u>1610708219</u>	0.11690 <u>3174867686</u>

Table D.3: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{23}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.



	Baseline $U$	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.46689 <u>1233345886</u>	0.466890 <u>43588624</u>
	0.466890834616315	0.46689 <u>1233345886</u>	0.466890 <u>43588624</u>
	0.466890834616315	0.46689 <u>1233345886</u>	0.466890 <u>43588624</u>
	0.466890834616315	0.46689 <u>1233345886</u>	0.466890 <u>43588624</u>
	0.596167182504661	0.596167 <u>484544275</u>	0.596166880464667
	0.596167182504661	0.596167 <u>484544275</u>	0.596166880464667
	0.596167182504661	0.596167 <u>484544275</u>	0.596166880464667
	0.596167182504661	0.596167 <u>484544275</u>	0.596166880464667
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.11690 <u>3053284258</u>	0.11690 <u>1732288951</u>
	0.116902392787021	0.11690 <u>3053284258</u>	0.11690 <u>1732288951</u>
	0.116902392787021	0.11690 <u>3053284258</u>	0.11690 <u>1732288951</u>
	0.116902392787021	0.11690 <u>3053284258</u>	0.11690 <u>1732288951</u>

Table D.4: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{25}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.466890913622128	0.466890755611384
	0.466890834616315	0.466890913622128	0.466890755611384
	0.466890834616315	0.466890913622128	0.466890755611384
	0.466890834616315	0.466890913622128	0.466890755611384
	0.596167182504661	0.596167242351953	0.596167122658037
	0.596167182504661	0.596167242351953	0.596167122658037
	0.596167182504661	0.596167242351953	0.596167122658037
	0.596167182504661	0.596167242351953	0.596167122658037
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.116901985216810	0.116902800358852
	0.116902392787021	0.116901985216810	0.116902800358852
	0.116902392787021	0.116901985216810	0.116902800358852
	0.116902392787021	0.116901985216810	0.116902800358852

Table D.5: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{26}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	Baseline $U$	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.46689 <u>1044261623</u>	0.466890624971686
	0.466890834616315	0.46689 <u>1044261623</u>	0.466890624971686
	0.466890834616315	0.46689 <u>1044261623</u>	0.466890624971686
	0.466890834616315	0.46689 <u>1044261623</u>	0.466890624971686
	0.596167182504661	0.596167 <u>337536957</u>	0.596167027473071
	0.596167182504661	0.596167 <u>337536957</u>	0.596167027473071
	0.596167182504661	0.596167 <u>337536957</u>	0.596167027473071
	0.596167182504661	0.596167 <u>337536957</u>	0.596167027473071
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.116902 <u>736793652</u>	0.116902048781681
	0.116902392787021	0.116902 <u>736793652</u>	0.116902048781681
	0.116902392787021	0.116902 <u>736793652</u>	0.116902048781681
	0.116902392787021	0.116902 <u>736793652</u>	0.116902048781681

Table D.6: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{27}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.4668908 <u>52043850</u>	0.4668908 <u>17189048</u>
	0.466890834616315	0.4668908 <u>52043850</u>	0.4668908 <u>17189048</u>
	0.466890834616315	0.4668908 <u>52043850</u>	0.4668908 <u>17189048</u>
	0.466890834616315	0.4668908 <u>52043850</u>	0.4668908 <u>17189048</u>
	0.596167182504661	0.5961671 <u>95601239</u>	0.5961671 <u>69408286</u>
	0.596167182504661	0.5961671 <u>95601239</u>	0.5961671 <u>69408286</u>
	0.596167182504661	0.5961671 <u>95601239</u>	0.5961671 <u>69408286</u>
	0.596167182504661	0.5961671 <u>95601239</u>	0.5961671 <u>69408286</u>
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.116902 <u>421564924</u>	0.116902 <u>364009563</u>
	0.116902392787021	0.116902 <u>421564924</u>	0.116902 <u>364009563</u>
	0.116902392787021	0.116902 <u>421564924</u>	0.116902 <u>364009563</u>
	0.116902392787021	0.116902 <u>421564924</u>	0.116902 <u>364009563</u>

Table D.7: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{28}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.46689 <u>1073359879</u>	0.466890 <u>59587661</u>
	0.466890834616315	0.46689 <u>1073359879</u>	0.466890 <u>59587661</u>
	0.466890834616315	0.46689 <u>1073359879</u>	0.466890 <u>59587661</u>
	0.466890834616315	0.46689 <u>1073359879</u>	0.466890 <u>59587661</u>
	0.596167182504661	0.596167 <u>363354087</u>	0.596167 <u>001658159</u>
	0.596167182504661	0.596167 <u>363354087</u>	0.596167 <u>001658159</u>
	0.596167182504661	0.596167 <u>363354087</u>	0.596167 <u>001658159</u>
	0.596167182504661	0.596167 <u>363354087</u>	0.596167 <u>001658159</u>
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.11690278826675 <u>4</u>	0.11690199731368 <u>1</u>
	0.116902392787021	0.11690278826675 <u>4</u>	0.11690199731368 <u>1</u>
	0.116902392787021	0.11690278826675 <u>4</u>	0.11690199731368 <u>1</u>
	0.116902392787021	0.11690278826675 <u>4</u>	0.11690199731368 <u>1</u>

Table D.8: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{29}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.466890834616315	0.466890834616315
	0.466890834616315	0.466890834616315	0.466890834616315
	0.466890834616315	0.466890834616315	0.466890834616315
	0.466890834616315	0.466890834616315	0.466890834616315
	0.596167182504661	0.596167182504661	0.596167182504661
	0.596167182504661	0.596167182504661	0.596167182504661
	0.596167182504661	0.596167182504661	0.596167182504661
	0.596167182504661	0.596167182504661	0.596167182504661
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.116902392787021	0.116902392787021
	0.116902392787021	0.116902392787021	0.116902392787021
	0.116902392787021	0.116902392787021	0.116902392787021
	0.116902392787021	0.116902392787021	0.116902392787021

Table D.9: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{30}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

	<b>Baseline <math>U</math></b>	$U_{+h}$	$U_{-h}$
$\mathcal{F}_4$	0.466890834616315	0.466890 <u>331634007</u>	0.46689 <u>1337599707</u>
	0.466890834616315	0.466890 <u>331634007</u>	0.46689 <u>1337599707</u>
	0.466890834616315	0.466890 <u>331634007</u>	0.46689 <u>1337599707</u>
	0.466890834616315	0.466890 <u>331634007</u>	0.46689 <u>1337599707</u>
	0.596167182504661	0.596167 <u>617554458</u>	0.596166747453928
	0.596167182504661	0.596167 <u>617554458</u>	0.596166747453928
	0.596167182504661	0.596167 <u>617554458</u>	0.596166747453928
	0.596167182504661	0.596167 <u>617554458</u>	0.596166747453928
$\mathcal{F}_6$	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.000000000000000	0.000000000000000	0.000000000000000
	0.116902392787021	0.116902 <u>266847871</u>	0.116902518726444
	0.116902392787021	0.116902 <u>266847871</u>	0.116902518726444
	0.116902392787021	0.116902 <u>266847871</u>	0.116902518726444
	0.116902392787021	0.116902 <u>266847871</u>	0.116902518726444

Table D.10: Changing  $U$  knot vectors on Faces  $\mathcal{F}_4$  (semi-ellipse) and  $\mathcal{F}_6$  (circular-arc) when finite-differencing with parameter  $\mathcal{P} = d_{31}$  and step-size  $h = 1.0 \times 10^{-4}$ . Underlined digits denote a deviation from the baseline knot values.

THIS PAGE INTENTIONALLY LEFT BLANK

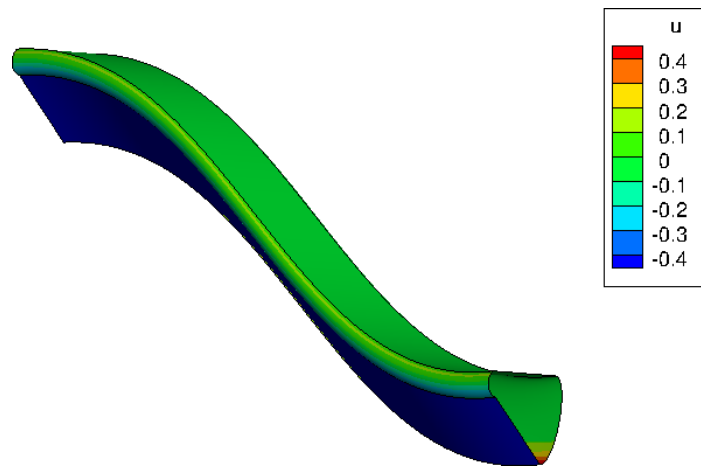


## Appendix E

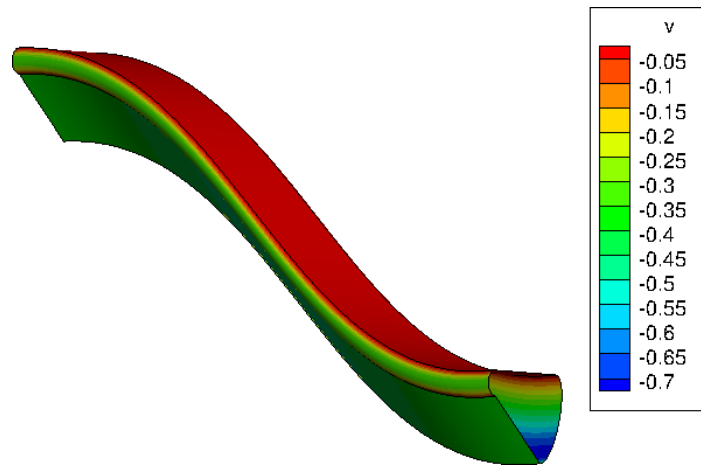
### “Snap” Grid

## Design Velocity Results

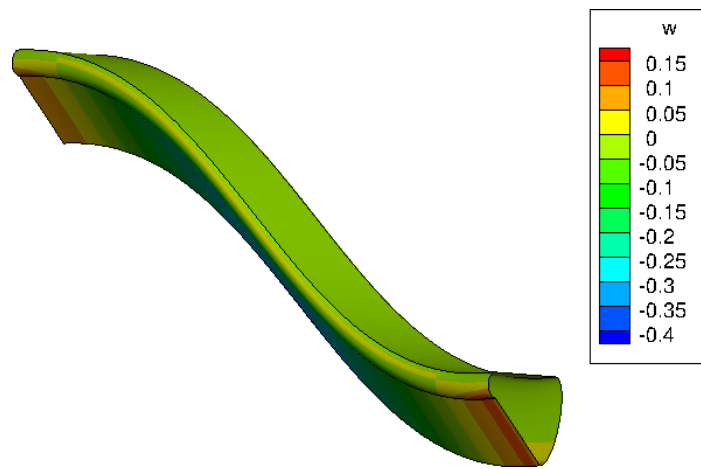
The figures presented here compare design velocity results from the analytic method in Section 3.5 and finite-differencing with “snap” grids. These two methods do not compare well, as seen in Figures E-1–E-10, because the “snap” grid approach does not model design velocity rigorously. Spurious design velocity distributions are seen in Figures E-11–E-19 when using a “snap” grid. As concluded in Section 3.5, these results confirm that the nearest-point projection is generally not related to surface design motion. Therefore, it is recommended that “snapped” grids be avoided as a point-tracking strategy for finite-differencing.



(a)

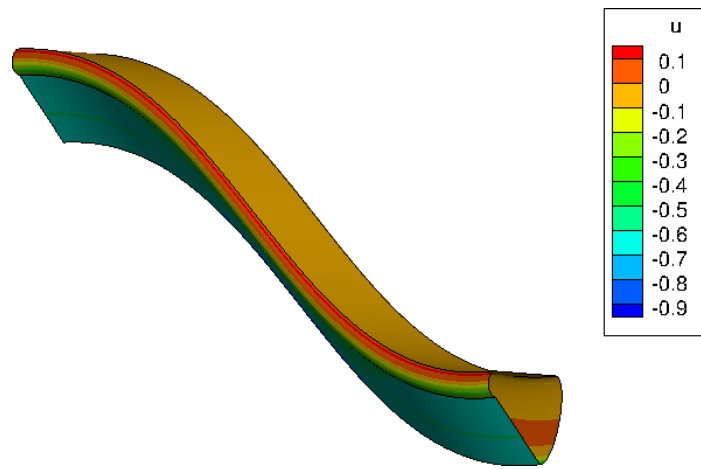


(b)

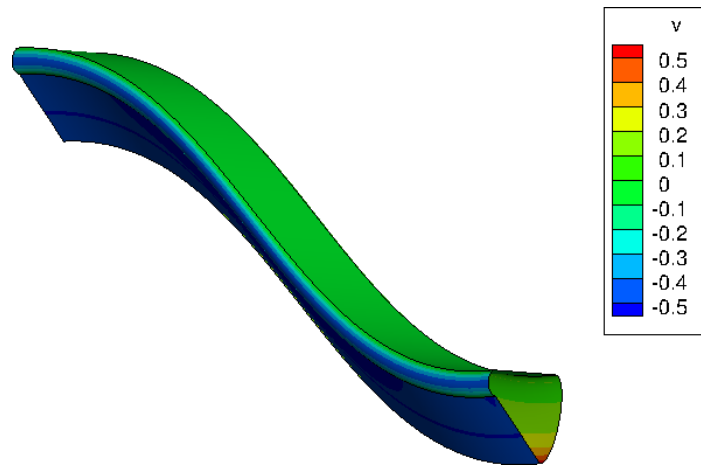


(c)

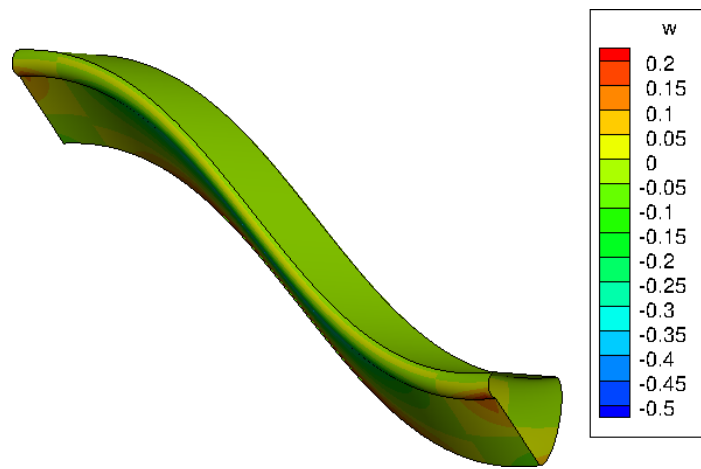
Figure E-1: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{21}$ .



(a)

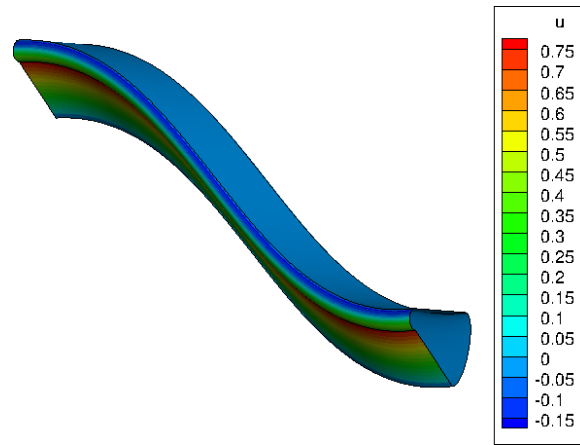


(b)

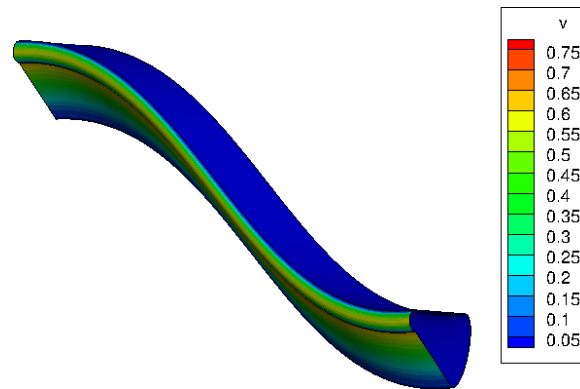


(c)

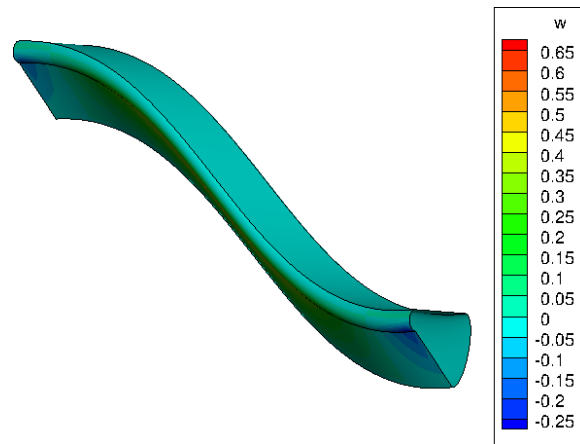
Figure E-2: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{22}$ .



(a)

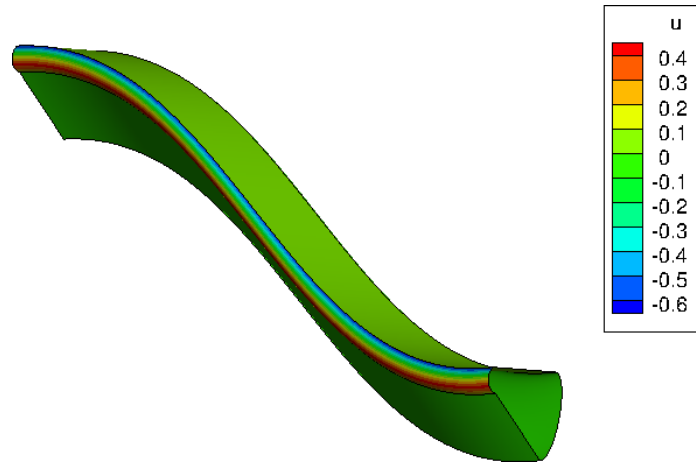


(b)

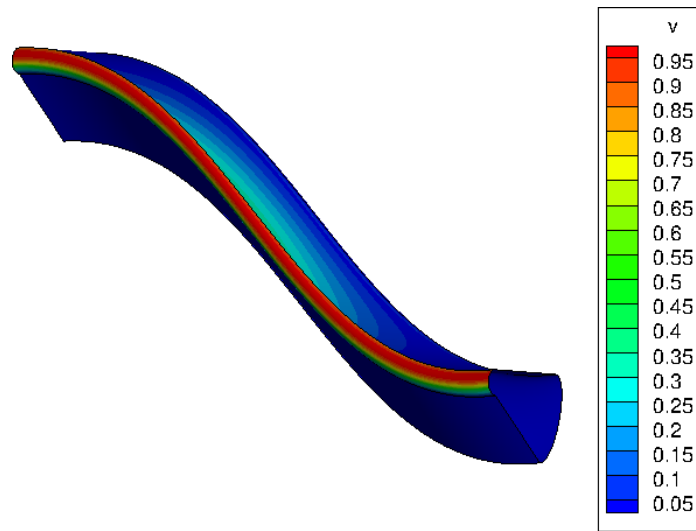


(c)

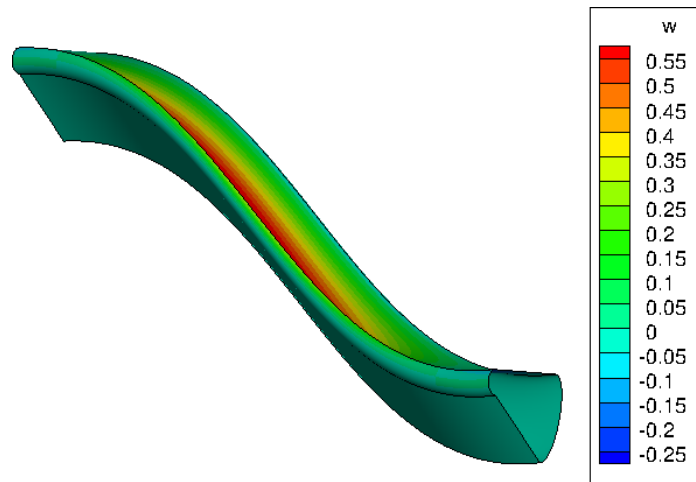
Figure E-3: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{23}$ .



(a)

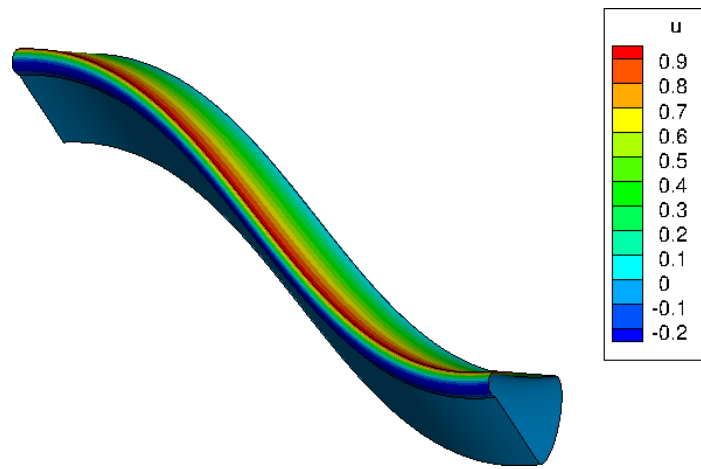


(b)

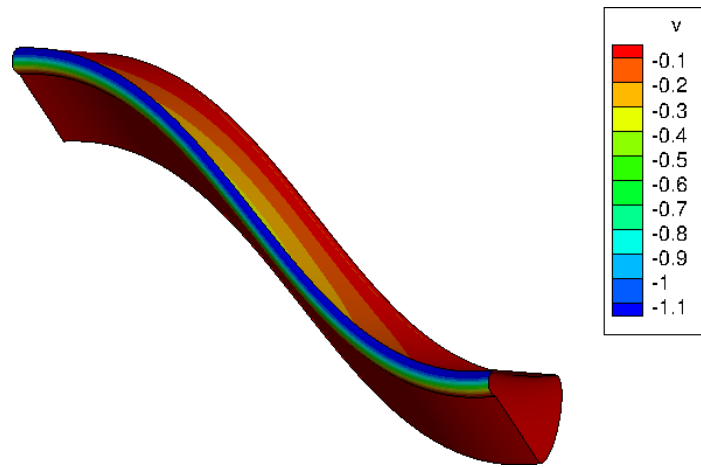


(c)

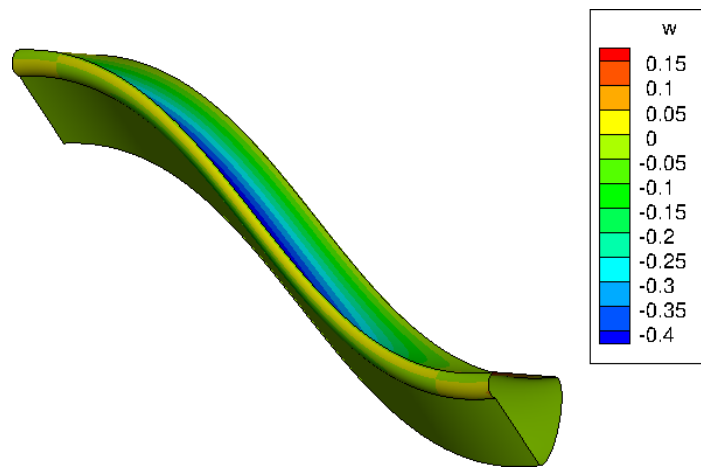
Figure E-4: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{25}$ .



(a)

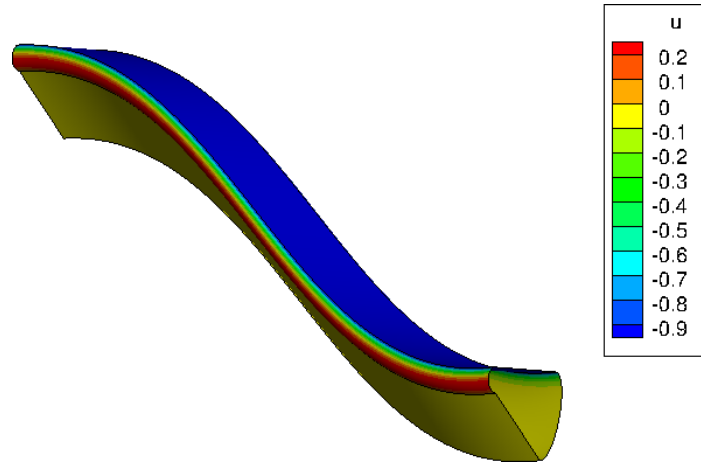


(b)

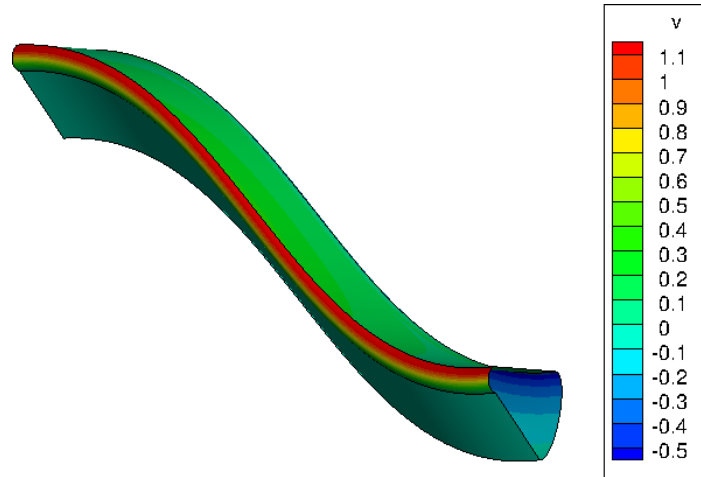


(c)

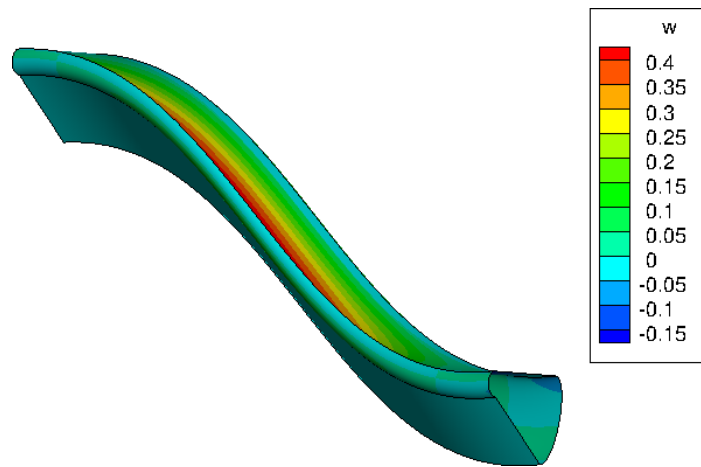
Figure E-5: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{26}$ .



(a)

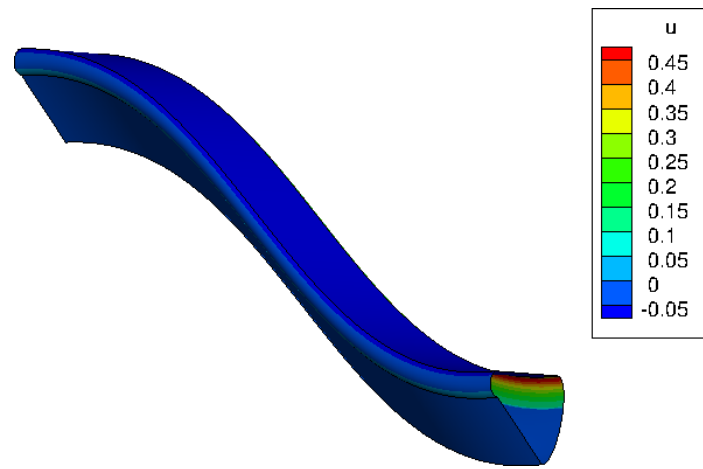


(b)

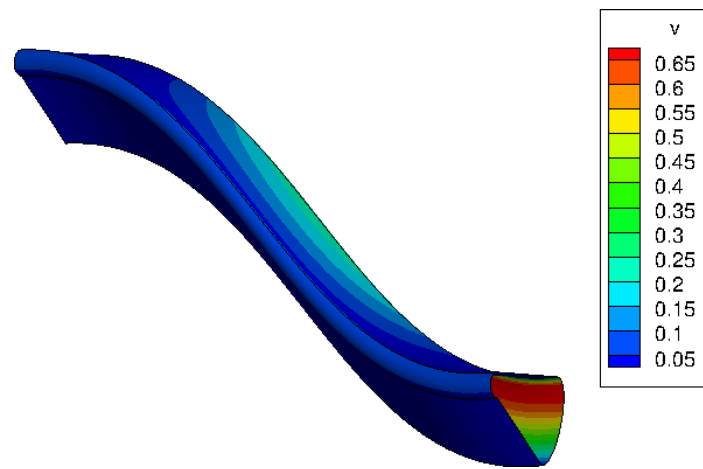


(c)

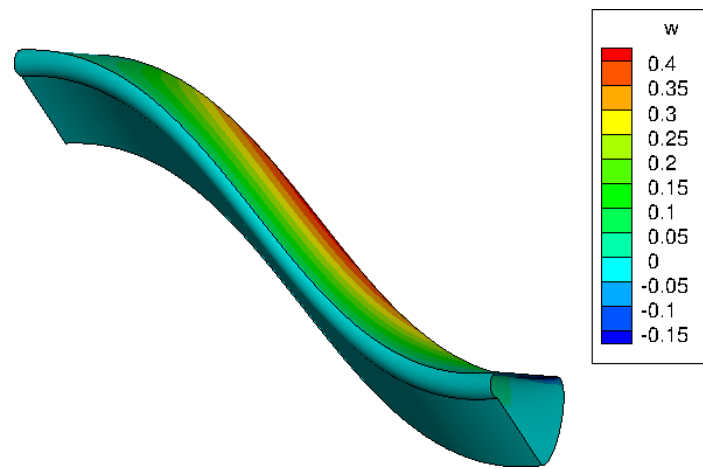
Figure E-6: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{27}$ .



(a)



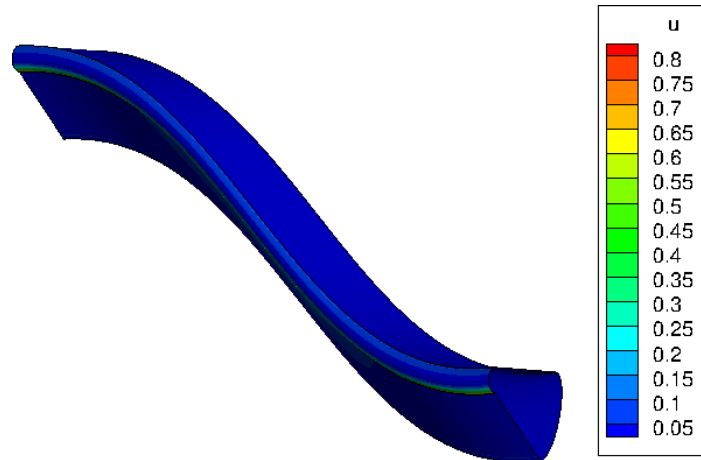
(b)



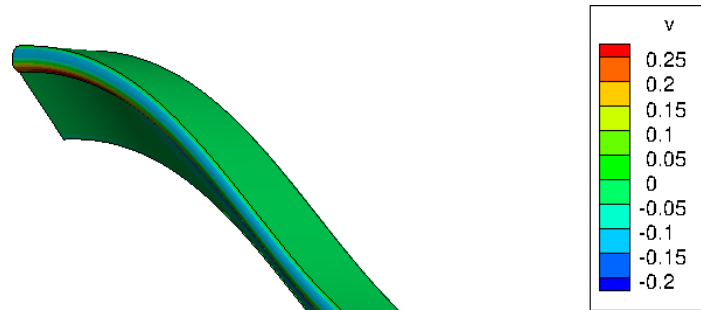
(c)

Figure E-7: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{28}$ .

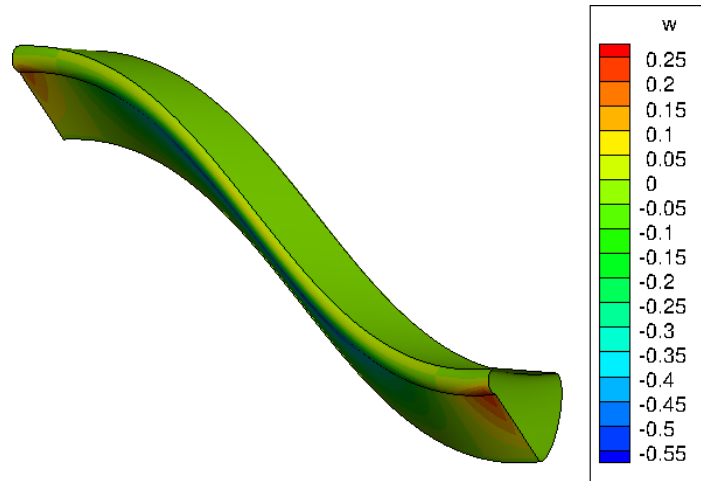




(a)

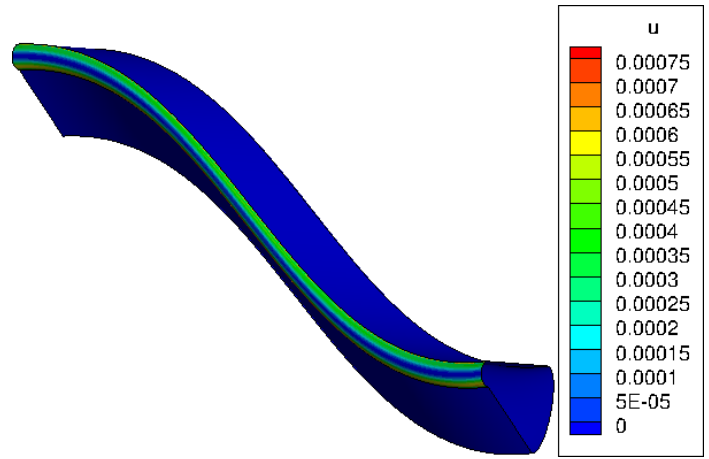


(b)

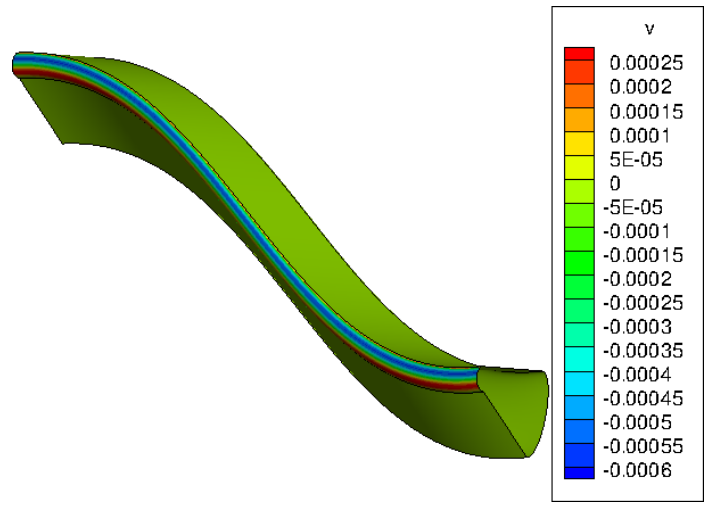


(c)

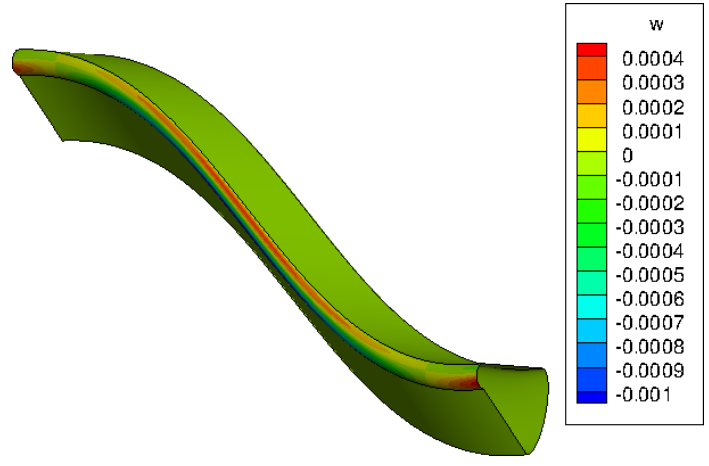
Figure E-8: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{29}$ .



(a)

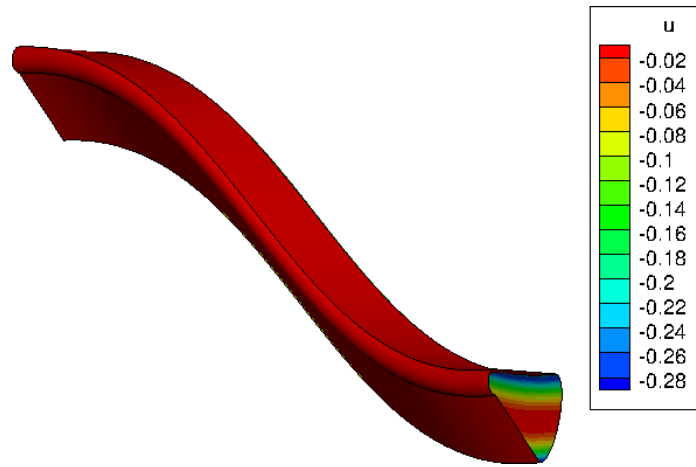


(b)

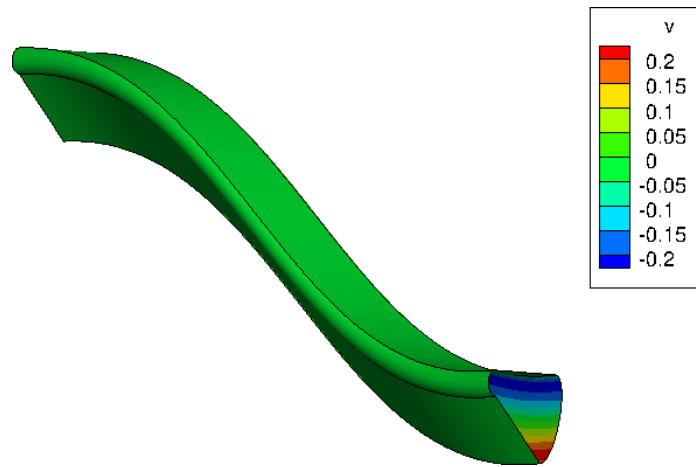


(c)

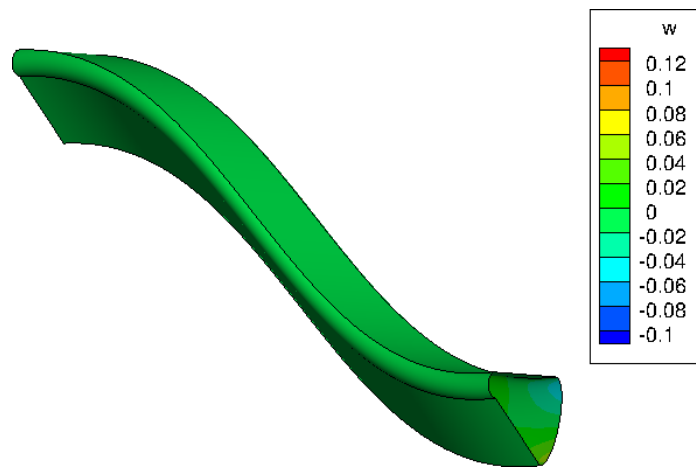
Figure E-9: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{30}$ .



(a)

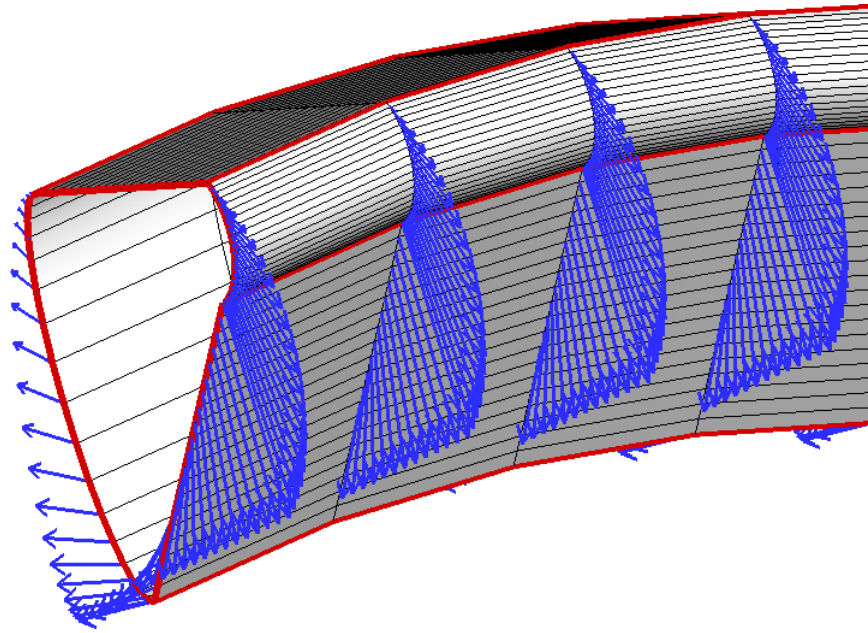


(b)

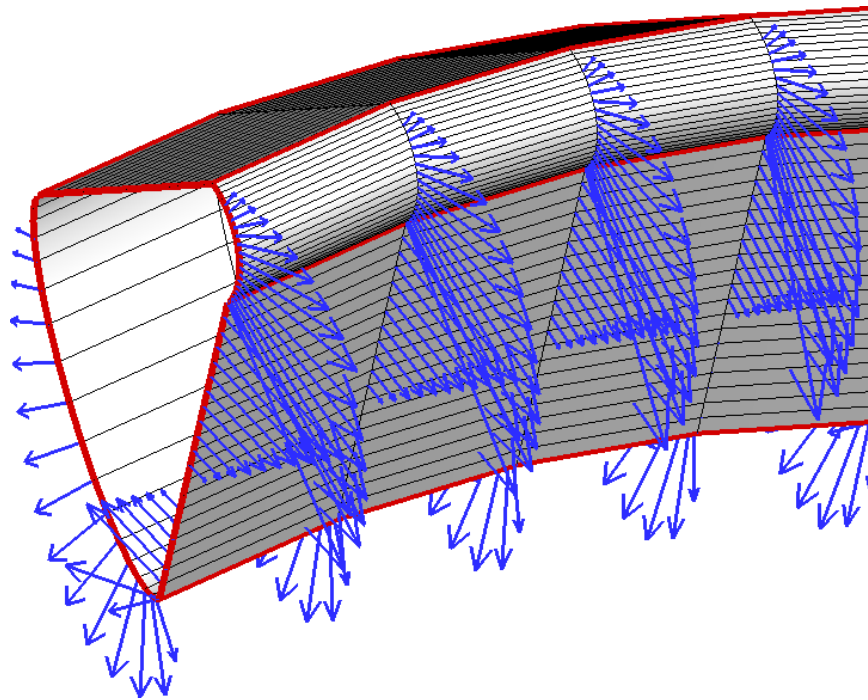


(c)

Figure E-10: Relative offset between the sweep *feature* differentiation method and centered-differencing on a “snap” grid for parameter  $\mathcal{P} = d_{31}$ .

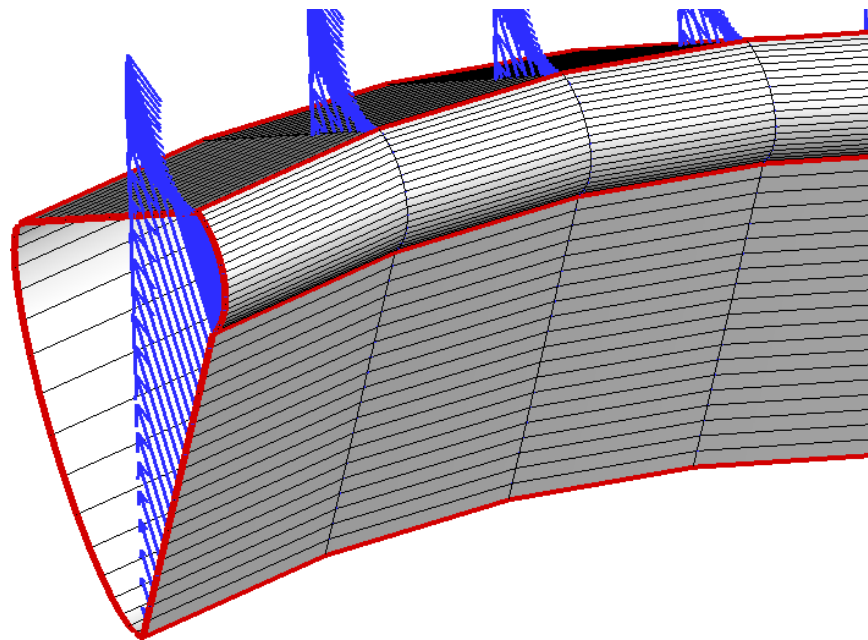


(a) Fixed-spacing Grid

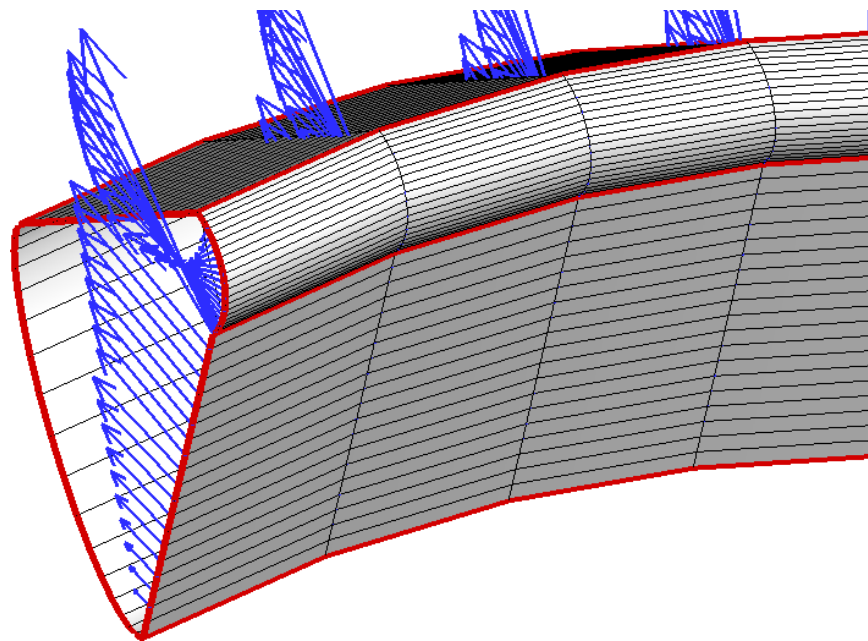


(b) "Snap" Grid

Figure E-11: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-12). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{22}$  and step-size  $h = 1.0 \times 10^{-4}$ .

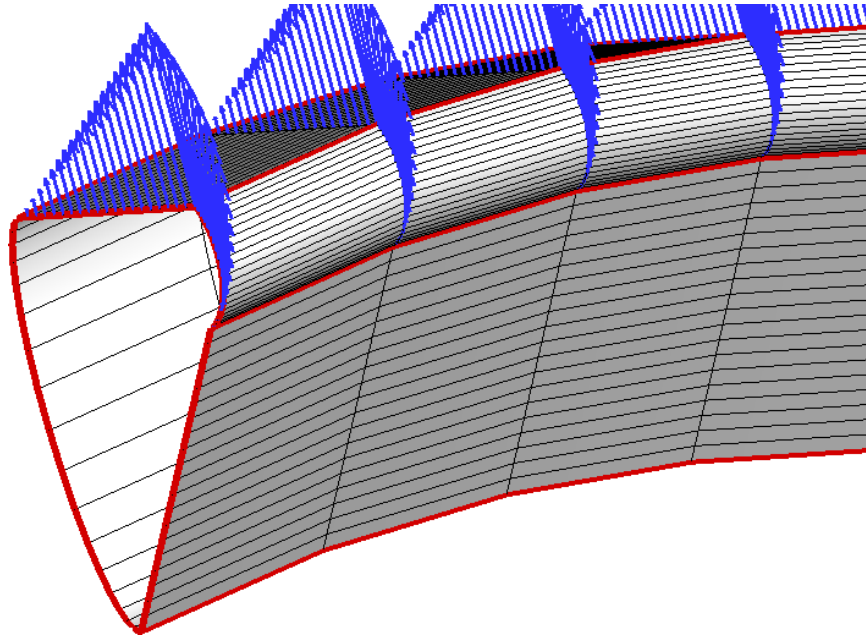


(a) Fixed-spacing Grid

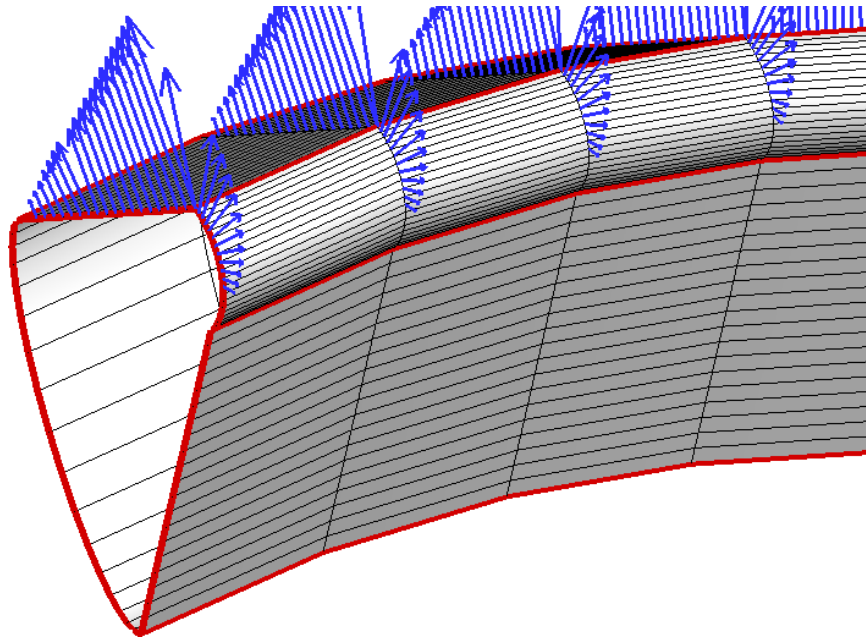


(b) "Snap" Grid

Figure E-12: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-13). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{23}$  and step-size  $h = 1.0 \times 10^{-4}$ .

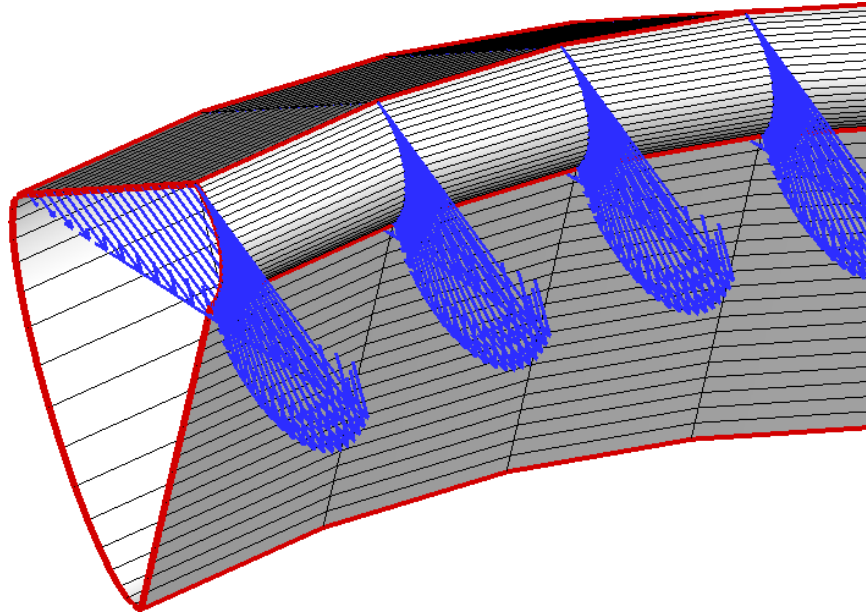


(a) Fixed-spacing Grid

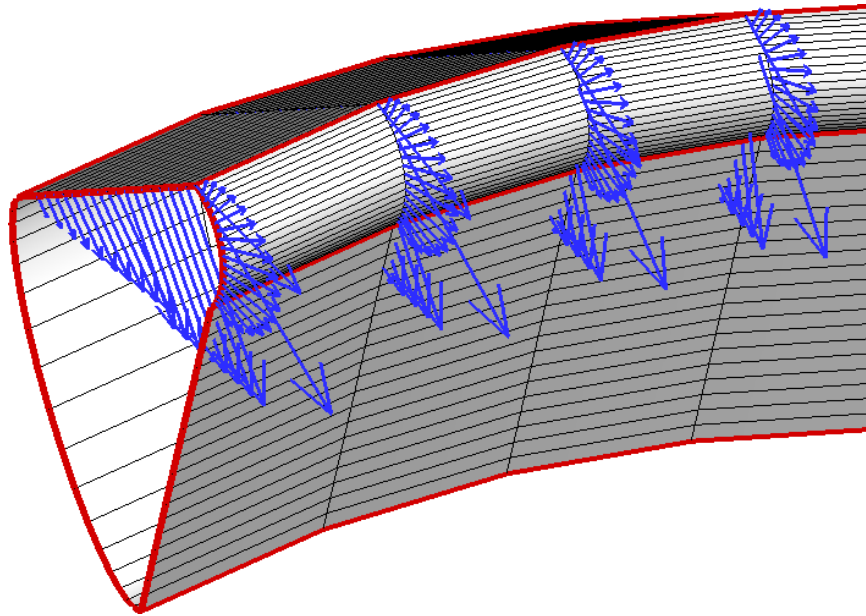


(b) "Snap" Grid

Figure E-13: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-14). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{25}$  and step-size  $h = 1.0 \times 10^{-4}$ .



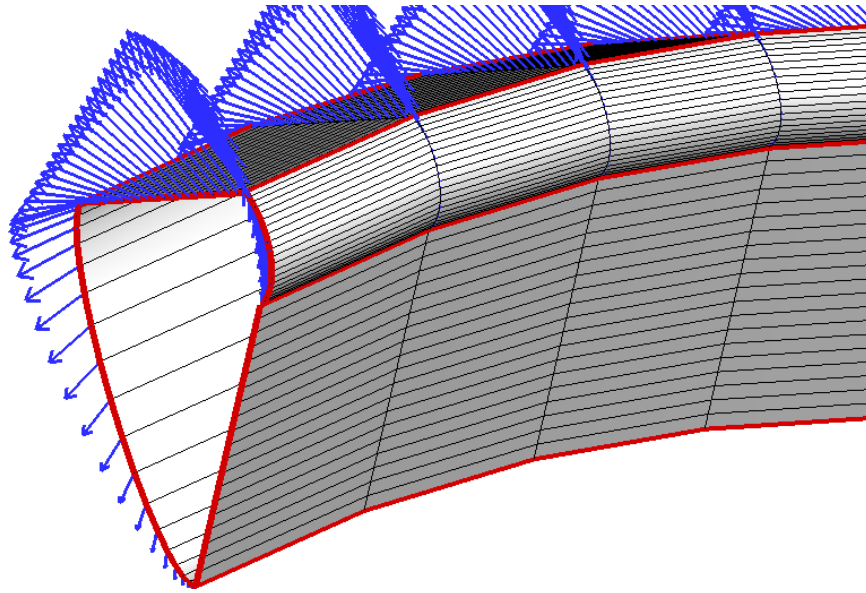
(a) Fixed-spacing Grid



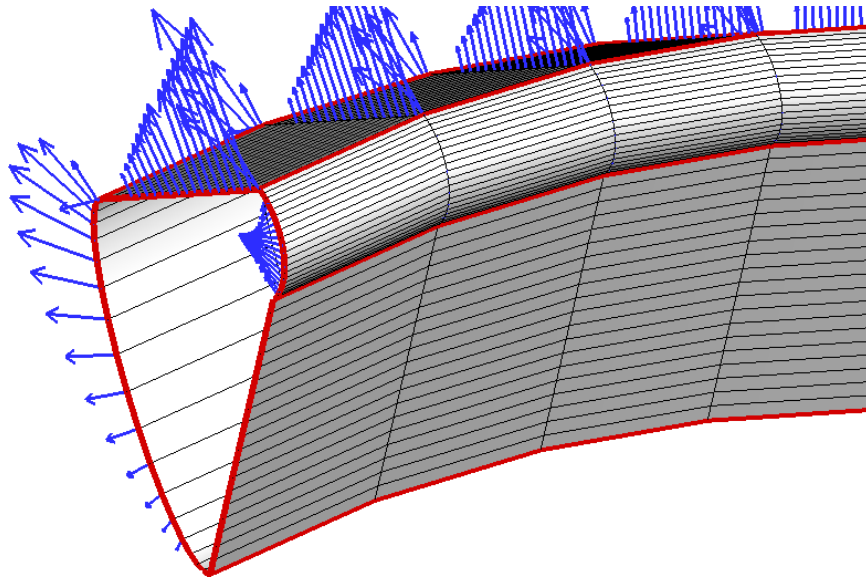
(b) "Snap" Grid

Figure E-14: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-15). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{26}$  and step-size  $h = 1.0 \times 10^{-4}$ .





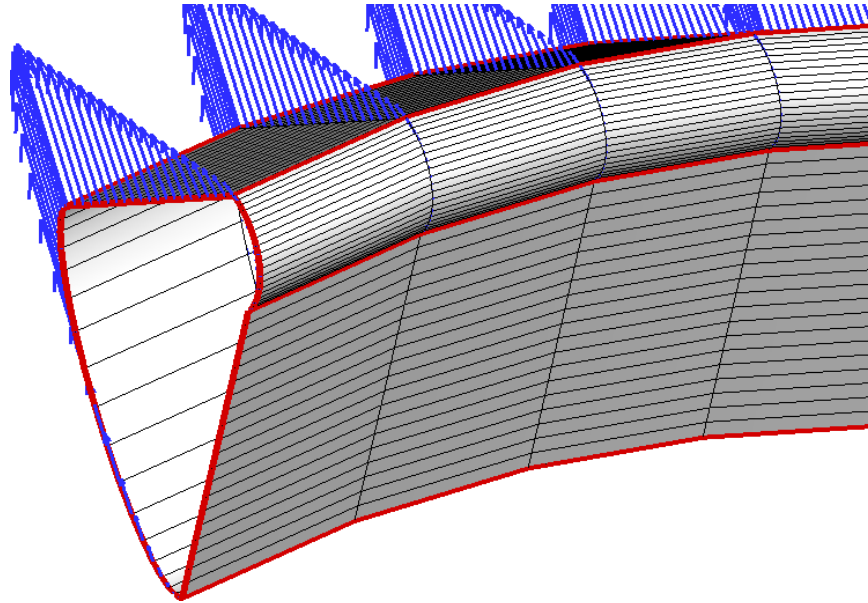
(a) Fixed-spacing Grid



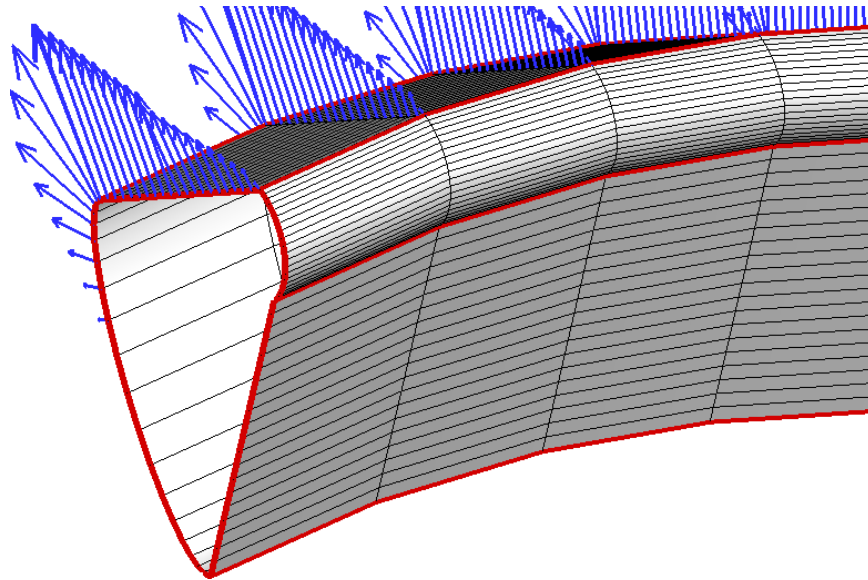
(b) "Snap" Grid

Figure E-15: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-16). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{27}$  and step-size  $h = 1.0 \times 10^{-4}$ .



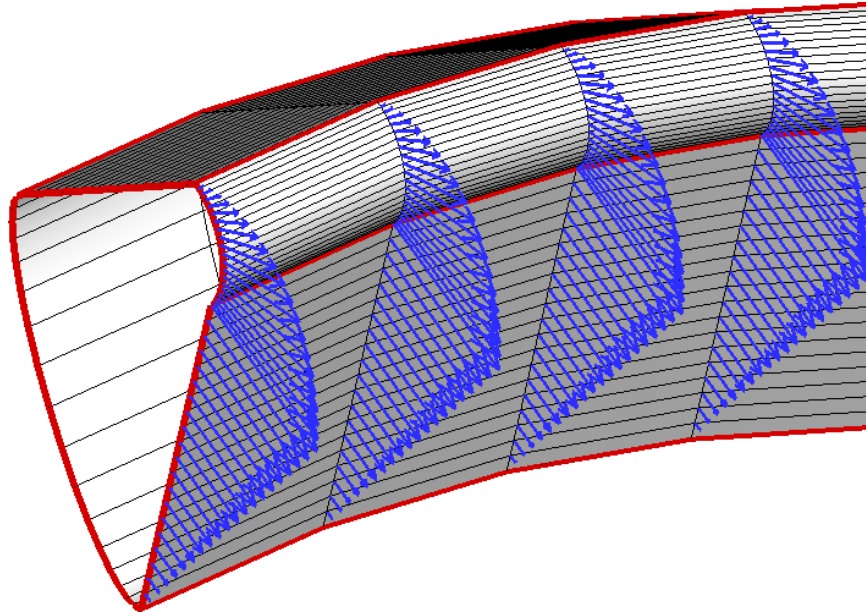


(a) Fixed-spacing Grid

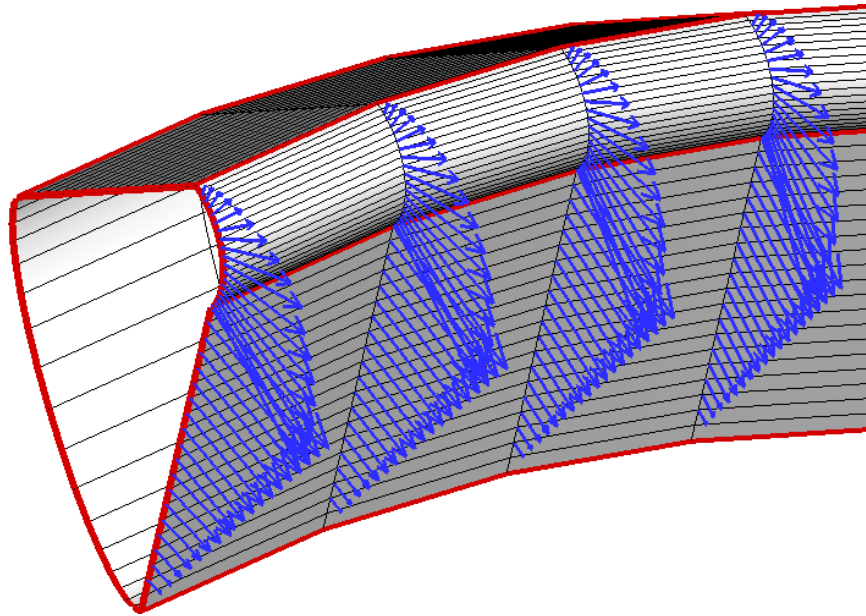


(b) "Snap" Grid

Figure E-16: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-17). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{28}$  and step-size  $h = 1.0 \times 10^{-4}$ .

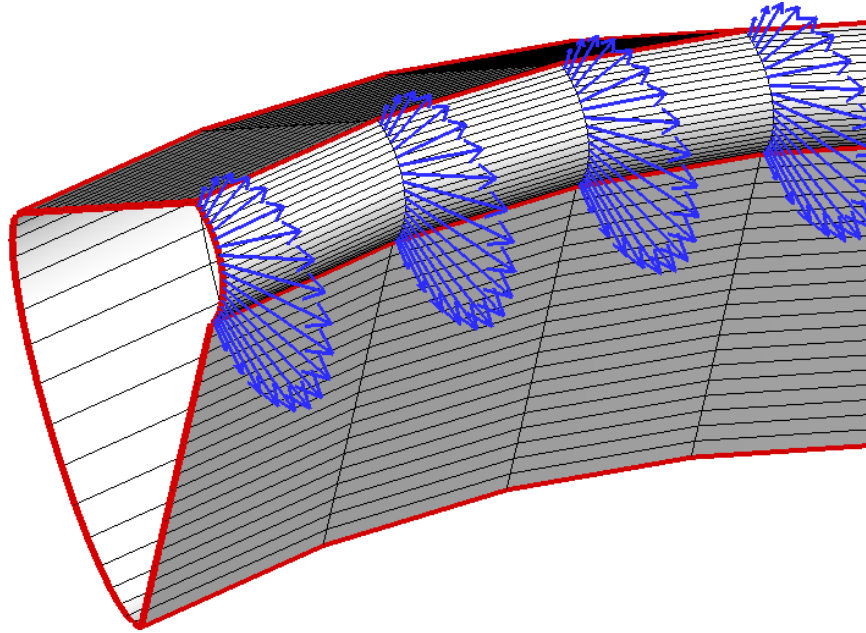


(a) Fixed-spacing Grid

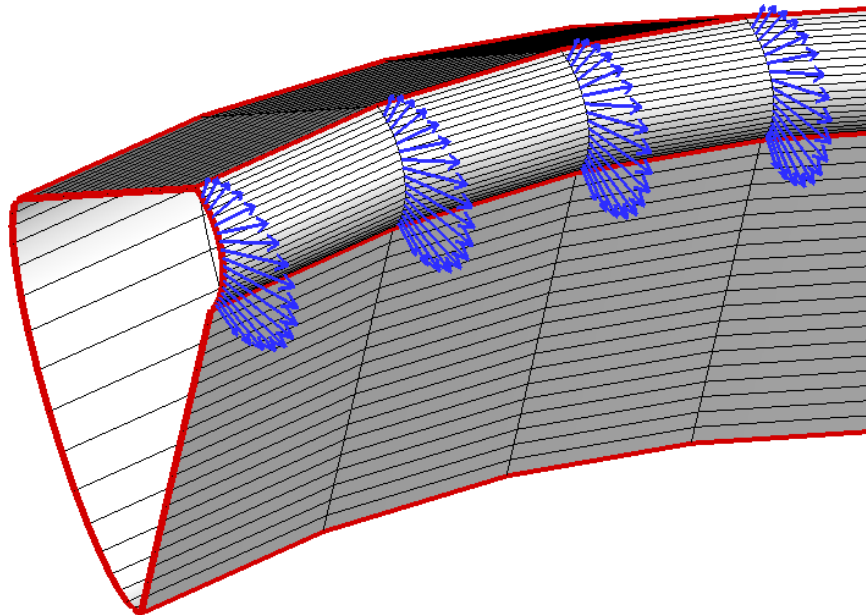


(b) "Snap" Grid

Figure E-17: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-18). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{29}$  and step-size  $h = 1.0 \times 10^{-4}$ .

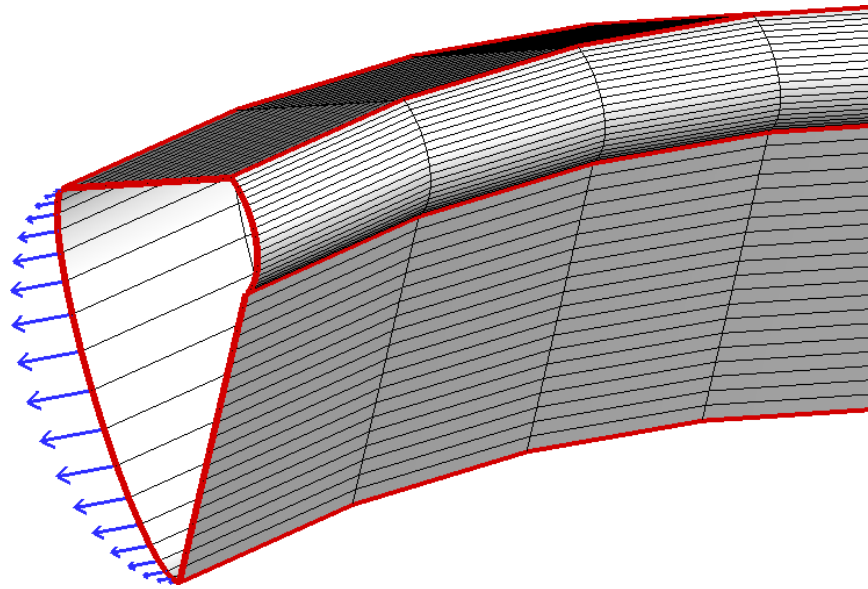


(a) Fixed-spacing Grid

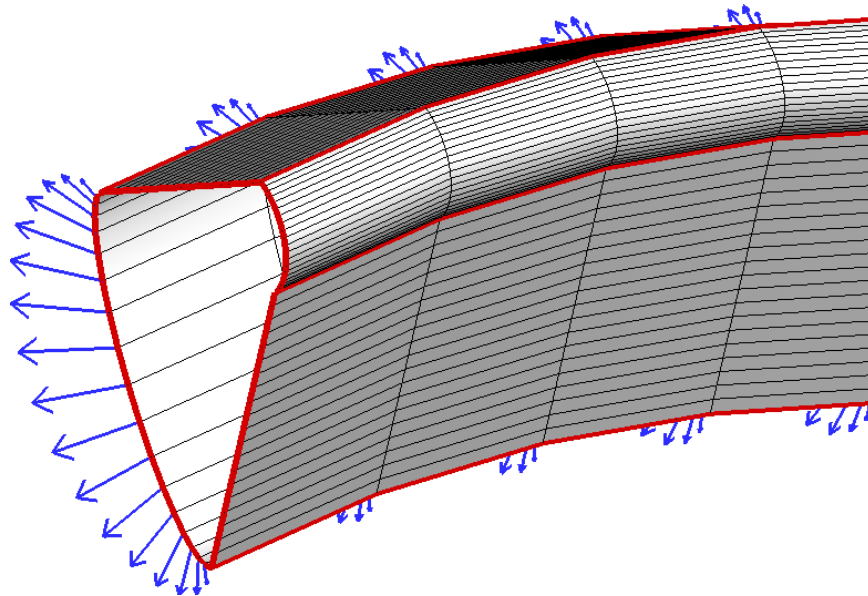


(b) "Snap" Grid

Figure E-18: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-19). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{30}$  and step-size  $h = 1.0 \times 10^{-2}(\pi/180)$ . A larger step-size (greater than  $1.0 \times 10^{-3}$ ) was needed to ensure a non-zero design velocity in this case.



(a) Fixed-spacing Grid



(b) "Snap" Grid

Figure E-19: The design velocity vectors in (a) correctly display a continuous derivative along the  $u$ -isocline (as seen in Figure 3-20). In (b), however, the "snap" grid yields incorrect discontinuous design velocities along the  $u$ -isocline. These design velocities reflect finite-differencing with the parameter  $\mathcal{P} = d_{31}$  and step-size  $h = 1.0 \times 10^{-4}$ .

# Bibliography

- [1] Amadori, K. and Johansson, B. and Krus, P. Using CAD-Tools and Aerodynamics Codes in a Distributed Conceptual Design Framework. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 8-11 January 2007. American Institute of Aeronautics and Astronautics. AIAA 2007-964.
- [2] Amadori, K. and Jouannet, C. A Framework for Aerodynamic and Structural Optimization in Conceptual Design. In *25th AIAA Applied Aerodynamics Conference*, Miami, Florida, 25-28 June 2007. American Institute of Aeronautics and Astronautics. AIAA 2007-4061.
- [3] Armstrong, C. G. and Robinson, T. T. and Ou, H. and Othmer, C. Linking Adjoint Sensitivity Maps with CAD Parameters. In Neittaanaki, P. and Périaux, J. and Tuovinen, T., editor, *Evolutionary Methods for Design, Optimization and Control*, Barcelona, Spain, 2007. CIMNE.
- [4] Badufle, C. and Homsy, P. Value Improvement through a Virtual Aeronautical Collaborative Enterprise. In *Symposium on Applied Aerodynamics and Design of Aerospace Vehicles SAROD 2005*, Bangalore, 8-9 December 2005.
- [5] Bartels, Richard H. and Beatty, John C. and Barsky, Brian A. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Inc., 1987.
- [6] Belegundu, A.D. and Rajan, S.D. A Shape Optimization Approach Based on Natural Design Variables and Shape Functions. *Computer Methods in Applied Mechanics and Engineering*, 66:87–106, 1988. Elsevier Science Publishers B.V. (North-Holland).
- [7] Bischof, C. and Carle, A. and Khademi, P. and Mauer, A. ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs. *IEEE Computational Science and Engineering*, 3(3):18–32, 1996.
- [8] Bischof, C. and Roh, L. and Mauer, A. ADIC—an Extensible Automatic Differentiation Tool for ANSI-C. *Software—Practice and Experience*, 27(12):1427–1456, 1997.
- [9] Botkin, M. E. Three-Dimensional Shape Optimization Using Fully Automatic Mesh Generation. *AIAA Journal*, 30(7):1932–1934, 1991.
- [10] Bowcutt, K. G. A Perspective on the Future of Aerospace Vehicle Design. In *12th AIAA International Space Planes and Hypersonic Systems and Technologies*, Norfolk, Virginia, 15-19 December 2003. American Institute of Aeronautics and Astronautics. AIAA 2003-6957.

- [11] Brandt, S. A. and Stiles, R. J. and Bertin, J. J. and Whitford, R. *Introduction to Aeronautics: A Design Perspective*. AIAA Education Series, 2 edition, 2004.
- [12] Carty, A. and Davies, C. Fusion of Aircraft Synthesis and Computer Aided Design. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, 30 August - 1 September 2004. American Institute of Aeronautics and Astronautics. AIAA 2004-4433.
- [13] Chen, J. and Freytag, M. and Shapiro, V. Shape Sensitivity of Constructive Representations. In *SPM*, pages 85–96, Beijing, China, 4-6 June 2007. Association for Computing Machinery, Inc.
- [14] Chen, S. and Tortorelli, D.A. Three-dimensional shape optimization with variational geometry. *Structural Optimization*, 13:81–94, 1997. Springer-Verlag.
- [15] Choi, K. K. and Kim, N. *Structural Sensitivity Analysis and Optimization*, chapter 6. Springer, 2005.
- [16] Choi, S. and Alonso, J. J. and Kroo, I. M. and Wintzer, M. Multi-Fidelity Design Optimization of Low-Boom Supersonic Business Jets. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, 30 August - 1 September 2004. American Institute of Aeronautics and Astronautics. AIAA 2004-4371.
- [17] G. B. Cosentino and T. L. Holst. Numerical optimization design of advanced transonic wing configurations. *Journal of Aircraft*, 23(3):192–199, 1986.
- [18] Crawford, C. A. and Haimes, R. Synthesizing an MDO Architecture in CAD. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 5-8 January 2004. American Institute of Aeronautics and Astronautics. AIAA 2004-281.
- [19] Dassault, Systemes. SolidWorks. <http://www.solidworks.com/>.
- [20] Drela, M. Xfoil. <http://web.mit.edu/drela/Public/web/xfoil/>.
- [21] Drela, M. Newton solution of coupled viscous/inviscid multielement airfoil flows. In *AIAA, Fluid Dynamics, Plasma Dynamics and Lasers Conference*, Seattle, WA, June 1990. American Institute of Aeronautics and Astronautics. AIAA 90-1470.
- [22] Fudge, D. M. and Zingg, D. W. A CAD-Free and a CAD-Based Geometry Control System for Aerodynamic Shape Optimization. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 10-13 January 2005. American Institute of Aeronautics and Astronautics. AIAA 2005-451.
- [23] Fudos, I. and Hoffmann, C. M. Constraint-based parametric conics for CAD. *Computer-Aided Design*, 28(2):91–100, 1996.
- [24] G. Farin and G. Rein and N. Sapidis and A.J. Worsey. Fairing cubic B-spline curves. *Computer Aided Geometric Design*, 4(12):91 – 103, 1987.
- [25] Giles, Michael B. and Pierce, Niles A. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, 65:393–415, 2000.

- [26] Goraj, Z. and Frydrychewicz, A. and Ransom, E.C.P. and Self, A. and Wagstaff, P. Aerodynamic, dynamic and conceptual design of a fire-fighting aircraft. *IMechE*, 215(G):125–146, 25 April 2001.
- [27] Haimes, R. and Follen, G. Computational Analysis PRogramming Interface. In Cross, Eiseman, Hauser, Soni and Thompson, editor, *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, July 1998.
- [28] Haimes, R. and Merchant, A. Direct CAD Access for Analysis and Design. In Schilling R., Haase, W., Periaux, H. Baier, and Bugada, G., editor, *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, FLM, Munich, 2005. EUROGEN.
- [29] Hardee, E. and Chang, K. and Tu, J. and Choi, K. K. and Grindeanu, I. and Yu, X. A CAD-based design parameterization for shape optimization of elastic solids. *Advances in Engineering Software*, 30:185–199, 1999.
- [30] Hoffmann, C. M. Constraint-Based Computer-Aided Design. *JCISE*, 5(3):182–187, 2005.
- [31] Howe, D. *Aircraft Conceptual Design Synthesis*. Professional Engineering Publishing Limited, 2000.
- [32] Iyer, G. R. and Mills, J. J. Design Intent in 2D CAD: Definition and Survey. *Computer-Aided Design & Applications*, 3(1-4):259–267, 2006.
- [33] Jamaludin, M. A. and Said, H. B. and Majid, A. A. Shape control of parametric cubic curves. In J. Zhou, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 2644 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 128–133, March 1996.
- [34] Jameson, A. and Sriram and Martinelli, L. and Haimes, R. Aerodynamic Shape Optimization of Complete Aircraft Configurations using Unstructured Grids. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 5-8 January 2004. American Institute of Aeronautics and Astronautics. AIAA 2004-533.
- [35] Jameson, Antony. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.
- [36] Jameson, Antony. Automatic Design of Transonic Airfoils to Reduce the Shock Induced Pressure Drag. In *Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics*, Tel Aviv, Israel, February 1990.
- [37] Jenkinson, L. R. and Simpkin, P. and Rhodes, D. *Civil Jet Aircraft Design*. AIAA Education Series, 1999.
- [38] Joaquim R. R. A. Martins and Juan J. Alonso and James Reuther. Aero-Structural Wing Design Optimization Using High-Fidelity Sensitivity Analysis. In *In Proceedings CEAS Conference on Multidisciplinary Aircraft Design Optimization*, pages 211–226, 2001.

- [39] Jouannet, C. and Krus, P. Direct Simulation Based Optimization for Aircraft Conceptual Design. In *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland, 18-20 September 2007. American Institute of Aeronautics and Astronautics. AIAA 2007-7827.
- [40] Kessler, E. Advancing the State-of-the-Art in the Civil Aircraft Design: A Knowledge-Based Multidisciplinary Engineering Approach. In Wesseling, P., Oñate, J. Périaux, editor, *European Conference on Computational Fluid Dynamics*, TU Delft, Delft The Netherlands, 2006.
- [41] Kleinveld, S. and Roge, G. and Daumas, L. and Dinh, Q. Differentiated parametric CAD used within the context of automatic aerodynamic design optimization. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia Canada, 10-12 September 2008. American Institute of Aeronautics and Astronautics. AIAA 2008-5952.
- [42] Kodiyalam, S. and Kumar, V. and Finnigan, P. M. Constructive Solid Geometry Approach to Three-Dimensional Structural Shape Optimization. *AIAA Journal*, 30(5):1408–1415, May 1992.
- [43] Kondo, K. Algebraic method for manipulation of dimensional relationships in geometric models. *Computer-Aided Design*, 24(3):141–147, 1992.
- [44] Kroo, I. M. An Interactive System for Aircraft Design and Optimization. In *Aerospace Design Conference*, Irvine, California, 3-6 February 1992. American Institute of Aeronautics and Astronautics. AIAA 92-1190.
- [45] Kulfan, B. M. Universal Parametric Geometry Representation Model. *Journal of Aircraft*, 45(1):142–158, January-February 2008.
- [46] Kulfan, B. M. and Bussoletti, J. E. "Fundamental" Parametric Geometry Representations for Aircraft Component Shapes. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, Virginia, 6-8 September 2006. American Institute of Aeronautics and Astronautics. AIAA 2006-6948.
- [47] La Rocca, G. and van Tooren, M. Development of Design and Engineering Engines to support multidisciplinary design and analysis of aircraft. In *Delft Science in Design—A congress on interdisciplinary Design*, Delft, Netherlands, 2005. Delft University of Technology.
- [48] Lamit, L. *Moving from 2D to 3D CAD for Engineering Design: Challenges and Opportunities*. BookSurge Publishing, 2007.
- [49] Leslie Piegl and Wayne Tiller. Curve and surface constructions using rational B-splines. *Computer-Aided Design*, 19(9):485 – 498, 1987.
- [50] Liebeck, R. H. and Page, M. A. and Rawdon, B. K. Blended-Wing-Body Subsonic Commercial Transport. In *36th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 12-15 January 1998. American Institute of Aeronautics and Astronautics. AIAA 98-16309.
- [51] Light, Robert and Gossard, David. Modification of geometric models through variational geometry. *Computer-Aided Design*, 14(4):209–214, 1982.



- [52] Lombard, M. *SolidWorks Surfacing and Complex Shape Modeling Bible*. Wiley, 2008.
- [53] Lombard, M. *SolidWorks 2009 Bible*. Wiley, 2009.
- [54] Mavriplis, Dimitri J. A Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Reno, Nevada, January 2006. American Institute of Aeronautics and Astronautics.
- [55] Mawhinney, P. and Price, M. and Armstrong, C. and Raghunathan, S. and Ou, H. and Murphy, A. and Curran, R. Using Idealised Models to Enable Analysis Driven Design. In *AIAA's 3rd Annual Aviation Technology, Integration, and Operations (ATIO) Tech*, Denver, Colorado, 17-19 November 2003. American Institute of Aeronautics and Astronautics. AIAA 2003-6747.
- [56] Mortenson, M. E. *Geometric Modeling*. John Wiley & Sons, 1985.
- [57] Nemec, M. and Aftosmis, M. J. Adjoint Algorithm for CAD-Based Shape Optimization Using a Cartesian Method. In *17th Computational Fluid Dynamics Conference*, Toronto, Ontario Canada, 6-9 June 2005. American Institute of Aeronautics and Astronautics. NAS-05-014.
- [58] Nemec, M. and Aftosmis, M. J. Aerodynamic Shape Optimization Using a Cartesian Adjoint Method and CAD Geometry. In *24th AIAA Applied Aerodynamics Conference*, San Francisco, California, 5-8 June 2006. American Institute of Aeronautics and Astronautics. NAS-06-007.
- [59] Nemec, M. and Aftosmis, M. J. Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry. In *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, 4-7 January 2011. American Institute of Aeronautics and Astronautics. AIAA 2011-1249.
- [60] Nemec, M. and Aftosmis, M. J. and Pulliam, T. H. CAD-Based Aerodynamic Design of Complex Configurations Using a Cartesian Method. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 5-8 January 2004. American Institute of Aeronautics and Astronautics. AIAA 2004-113.
- [61] Othmer, C. CFD Topology and Shape Optimization with Adjoint Methods. In *Internationaler Kongress Berechnung und Simulation im Fahrzeugbau*. VDI Fahrzeug- und Verkehrstechnik, September 2006.
- [62] Parametric Technology Corporation. *Pro/TOOLKIT Reference Manual, Release 18.0*. PTC, 1997.
- [63] Pardessus, T. Concurrent engineering development and practices for aircraft design at Airbus. *ICAS2004*, 2004.
- [64] Piegl, L. and Tiller, W. *The NURBS Book*. Springer, 2 edition, 1997.
- [65] Raymer, D. P. Notional Design of an Advanced Strike Fighter. In *1st AIAA Aircraft Engineering, Technology, and Operations Congress*. American Institute of Aeronautics and Astronautics, 1995. AIAA 95-3922.

- [66] Raymer, D. P. *Aircraft Design: A Conceptual Approach*. AIAA Education Series, 4th edition, 2006.
- [67] Robinson, T. T. and Armstrong, C. G. Collaborative investigation by QUB and VW into the automatic calculation of parametric sensitivity, 2007.
- [68] Roskam, J. *Airplane design, Part VIII: Airplane cost estimation: design development, manufacturing and operating*. RAEC, Ottawa, 1991.
- [69] Samareh, J. A. Survey of Shape Parameterization Techniques for High-Fidelity Multi-disciplinary Shape Optimization. *AIAA Journal*, 39(5):877–884, May 2001.
- [70] Samareh, J. A. Aerodynamic Shape Optimization Based on Free-form Deformation. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, 30 August - 1 September 2004. American Institute of Aeronautics and Astronautics. AIAA 2004-4630.
- [71] Samuel, S. and Rymarz, P. and Kelley, M. and Pragada, A. *Practical Unigraphics NX2 Modeling for Engineers: A Project Oriented Learning Manual*. BookSurge Publishing, 2007.
- [72] Sarfraz, M. and Habib, Z. Conic representation of a rational cubic spline. In *Information Visualization, 1999. Proceedings. 1999 IEEE International Conference on*, pages 232–237, 1999.
- [73] Siemens. *NX Nastran 6 Release Guide*. Siemens Product Lifecycle Management Software Inc., 2008.
- [74] Sobester, A. and Kean, A. J. Airfoil Design via Cubic Splines–Ferguson’s Curves Revisited. In *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, Rohnert Park, California, 7-10 May 2007. American Institute of Aeronautics and Astronautics. AIAA 2007-2881.
- [75] Sträter, O. *Cognition and safety*. Ashgate, 2005.
- [76] Takenaka, K. and Obayashi, S. and Nakahashi, K. and Matsushima, K. The Application of MDO Technologies to the Design of a High Performance Small Jet Aircraft–Lessons learned and some practical concerns. In *Fluid Dynamics Conference and Exhibit*, Toronto, Ontario Canada, 6-9 June 2005. American Institute of Aeronautics and Astronautics. AIAA 2005-4797.
- [77] Tickoo, S. *CATIA V5R18 for Designers*. CAD/CIM Technologies, 2008.
- [78] Townsend, R. and Schmidt, G. R. *Pro/ENGINEER Solutions Advanced Techniques and Workarounds*. OnWord Press, 1999.
- [79] Vandenbrande, J. H. and Grandine, T. A. and Hogan, T. The search for the perfect body: Shape control for multidisciplinary design optimization. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 9-12 January 2006 2006. American Institute of Aeronautics and Astronautics. AIAA 2006-928.
- [80] William, B. and Fudos, I. and Hoffmann, C. M. and Cai, J. and Paige, R. Geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, 1995.

- [81] Yamazaki, W. and Mouton, S. and Carrier, G. Efficient Design Optimization by Physics-Based Direct Manipulation Free-Form Deformation. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia Canada, 10-12 September 2008. American Institute of Aeronautics and Astronautics. AIAA 2008-5953.
- [82] Zang, T. A. and Green, L. L. Multidisciplinary Design Optimization Techniques: Implications and Opportunities for Fluid Dynamics Research. In *AIAA Fluid Dynamics Conference*, Norfolk, Virginia, 28 June - 1 July 1999. American Institute of Aeronautics and Astronautics. AIAA 99-3798.