UNIVERSITY OF CALIFORNIA SAN DIEGO

Shell-element mesh generation of unconventional aircraft configurations for multidisciplinary structural analysis

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Engineering Sciences (Mechanical Engineering)

by

Xiangbei Liu

Committee in charge:

Professor John Tae Hyeon Hwang, Chair
Professor Michael Joseph Frazier
Professor David M Kamensky

2022

The thesis of Xiangbei Liu is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor John T. Hwang for his support as chair of my committee and director of the Large-Scale Design Optimization (LSDO) Lab. Without his guidance and efforts, none of this would be possible. His support has proved to be invaluable.

I would like to gratefully acknowledge Professor David M Kamensky. His invaluable advice improved my thesis and expanded it in different areas. I would also like to acknowledge the assistance of Professor Michael Joseph Frazier for being on my committee. He provided me with encouragement and patience throughout my defense.

Chapter 3 section 3.2 and section 3.4, in full is currently being prepared for submission for publication of the material. Liu, Xiangbei; Xiang, Ru; Pasetto, Marco; Li, Ning; Kamensky, David M; Hwang, John T. The thesis author was the primary investigator and author of this material.

Chapter 3 section 3.3, in full is currently being prepared for submission for publication of the material. Liu, Xiangbei; Xiang, Ru; Pasetto, Marco; Li, Ning; Kamensky, David M; Hwang, John T. The thesis author was not the primary investigator of this material.

Chapter 3 section 3.3, in full, is also a rephrase of the material as it appears in Automatic generation of global shell-element meshes for large-scale structural design optimization in AIAA AVIATION 2020 FORUM 2020 . Li, Ning; John T. Hwang., 2020. The thesis author was not the primary investigator and author of this paper.

Chapters 4, in full is currently being prepared for submission for publication of the material. Liu, Xiangbei; Xiang, Ru; Pasetto, Marco; Li, Ning; Kamensky, David M; Hwang, John T. The thesis author was the primary investigator and author of this material.

ABSTRACT OF THE THESIS

Shell-element mesh generation of unconventional aircraft configurations for multidisciplinary structural analysis

by

Xiangbei Liu

Master of Science in Engineering Sciences (Mechanical Engineering)

University of California San Diego, 2022

Professor John Tae Hyeon Hwang, Chair

Unconventional aircraft configurations are of substantial interest in aircraft design research. A challenge is that the design of unconventional aircraft is often a multidisciplinary and high-dimensional problem. Therefore, traditional empirical models are often inadequate. Large-scale multidisciplinary design, analysis, and optimization (MDAO) has evolved as a potential solution to this problem. In this thesis, I present a novel mesh generation method that is compatible with a large-scale MDAO framework, which means the generated structural meshes can be easily updated in response to changes in shape design variables. The mesh generation algorithm starts with a B-spline based geometry that defines the aircraft's structural

components, such as the outer skin and the internal members. Initial triangular meshes are generated under the constraint of the intersection connectivity between different components. Next, all the meshes of the components are subjected to a series of mesh quality improvement procedures that involve splitting, merging, and smoothing optimizations. Lastly, the resulting quad-dominant meshes are transformed into fully quadrilateral meshes. I also validate the quality of the generated meshes and their finite-element shell analysis results. The results show that this algorithm is versatile, efficient, effective, and compatible with large-scale MDAO processes.

Chapter 1

Introduction

There is a growing interest in unconventional aircraft configurations in aircraft design [3], e.g., for: hybrid electric aircraft with improved energy efficiency [4]; electric vertical takeoff and landing (eVTOL) in the expanding field of urban air mobility (UAM) [5]; and supersonic airliners [6]. However, the evaluation of unconventional aircraft with a specific set of requirements is often limited by the lack of experience and data, while traditional empirical models are inadequate to apply. Therefore, large-scale multidisciplinary design, analysis, and optimization (MDAO) has evolved as a strategy to address the complicated design trade-offs in unconventional configurations [3]. Large-scale MDAO is an approach that applies optimization to engineering design problems by employing models that span multiple disciplines. Regarding the typical multidisciplinary and high-dimensional engineering design problems for the aforementioned unconventional aircraft, as well as other complex engineering systems (e.g., design of satellites, wind turbines, robotics [7]), large-scale MDAO is significant, because it can represent a single monolithic, multidisciplinary optimization objective of the system's performance with a large number of design variables and constraints (see schematic example in Figure 1.1).

Due to the thin-walled structure of aircraft, finite shell-element analysis can accurately simulate their structural performance. One of the bottlenecks in structural shell-element analysis is the construction of the structural mesh, which normally demands a high level of expertise and extensive manual work. We come up with four requirements for an appropriate shell-

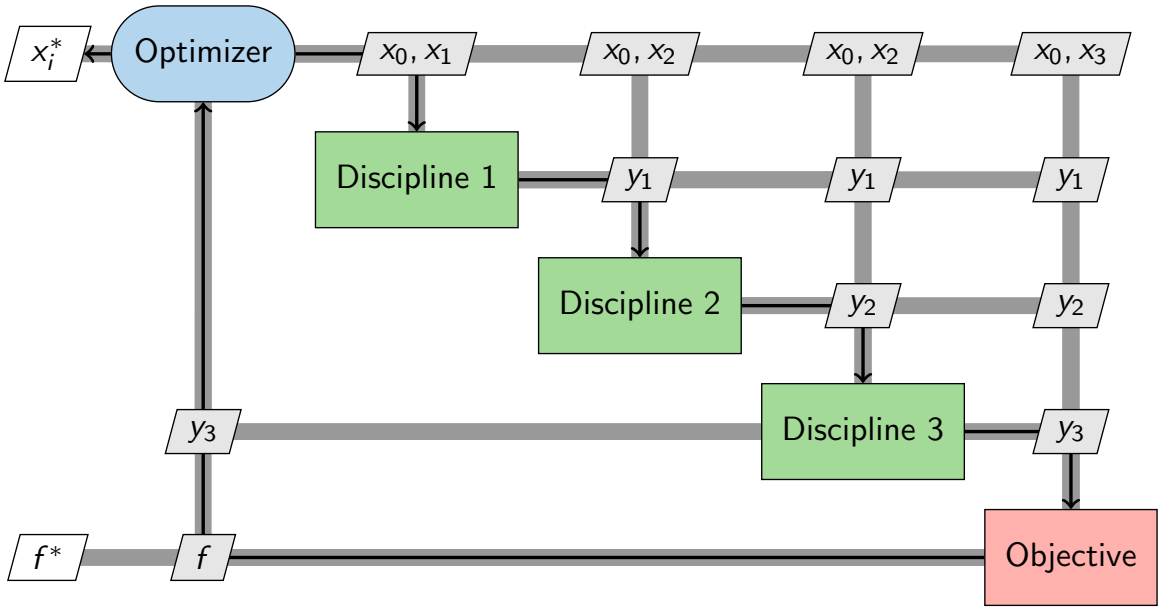


Figure 1.1. Schematic example of an large-scale MDAO process.

element mesh generation algorithm. Firstly, it must result in meshes with global high quality for the overall geometry. This will ensure the simulation's accuracy. First of all, as part of the engineering design process, it should be efficient with a quick turnaround time, and when the user describes the appropriate structural components, the mesh should be generated automatically without the need for additional manual effort. Secondly, it must result in meshes with global high quality for the overall geometry. This will ensure the simulation's accuracy. In addition, the mesh generation tool must be versatile, meaning that it can be used for various design models and can generate the full aircraft configuration, including the fuselage, wings, and tails. Most importantly, there is a priority requirement for the generated mesh to work and interface with a large-scale MDAO framework for the further design and optimization process. The gradient information of the generated mesh nodes is significantly required. Considering the enormous computational cost of gradient-free methods when it comes to optimization problems involving a large number of design variables, gradient-based optimization is the only feasible approach for large-scale MDAO [7]. Therefore, derivatives of the mesh node coordinates with respect to the

shape design variables (in our case, the B-spline control points of the geometry) are required [8].

Based on the above expectations, particularly the high-quality and the differentiability, a structural shell-element mesh generation algorithm has been developed. The thesis is organized as follows: In Chapter 1, I review the existing relevant work in unconventional aircraft mesh generation and mesh smoothing algorithms. Then, in Chapter 2, I present the specifics of the mesh generation algorithm. In Chapter 3, I validate the versatility of this mesh generation algorithm and the quality of the generated mesh. I also compare its analysis performance with commercial ICEMCFD mesh. In Chapter 4, I summarize the accomplishments of this thesis and the future plan.

Chapter 2

Literature Review

Many methods and techniques have been presented over the span of time for the purpose of generating shell-element meshes for aircraft. Fuhrer [9] presented a structural finite-element generator in a lightweight structure design environment, but the location of the internal members can only be added according to the parametric domain instead of the precise global Cartesian coordinate system of the aircraft. They did not compute the mesh nodes as a differentiable function of shape changes. In [8], the authors came up with several requirements for generated meshes in order to be used in an MDAO framework. The structural mesh should be global, with no disconnected meshes for individual aircraft components, and also versatile for the full aircraft configuration. The mesh should be computed based on shape design variables and should be differentiable, which means derivatives of the structural mesh coordinates with respect to the shape design variables should be computed efficiently. They created the GeoMACH tool suite to create geometry and meshes with the architecture of internal structures while enabling users to maintain the coincidence of junction nodes between members and skin. However, the generated mesh is only smooth locally in the parametric domain without globally smoothing. The mesh generation algorithm in GeoMACH is also time-consuming and requires the geometry to be developed within GeoMACH; it cannot work with externally created geometries such as CAD files [10]. Moreover, some algorithms lack automation because they use an external tool to generate an unstructured quad mesh [11]. Besides, NASA's Open Vehicle Sketch Pad (OpenVSP)

[12] is a parametrically driven, open-source airplane geometry design tool developed by NASA. However, OpenVSP can only build triangular meshes of aircraft structures with thin walls, and its capacity to generate quadrilateral meshes is restricted to organized wireframes instead of the full configuration. To sum up, existing previous research cannot simultaneously satisfy all four of the aforementioned requirements. Therefore, we desire a simple and robust method capable of automatically generating unstructured quadrilateral shell-element meshes for the entire aircraft configuration, including the skin, the internal members of wings, and other aircraft components.

When it comes to the specific problem of generating meshes of high quality, several different approaches that minimize smoothness energy have been developed in order to increase the mesh quality. This idea resulted in the high-quality quadrilateral mesh in [13], [14], and [15]. In addition, Knoppel [16] presents a method for calculating the stripe patterns on a surface, which can also be used to generate a mesh on the surface. In Huang's paper [17], they improve the idea of smooth orientation and position fields in [13] and eliminate the singularity mostly by solving a minimum-cost network flow problem. Unfortunately, all of these methods are fairly difficult to imply caused by incorporating nonlinear programming, which does not ensure convergence or a global minimum of the objective function. Therefore, more robust methods are required. [18], [19], and [20] adapt the advancing front techniques to generate quadrilateral meshes. However, the advanced front methods generally require more time than other global mesh generation algorithms.

In addition to the previously stated methods that rely on generating smoothing meshes directly, alternative approaches to smoothing an existing mesh have been developed. For example, in [21] a local optimization approach is applied to improve the topology of unstructured quadrilateral finite element meshes, and in [22] a constrained optimization is employed to generate a smooth mesh. Additionally, quadrilateral mesh quality has been increased based on the parametrization of the surfaces. The mesh is initialized in the two-dimensional parametric domain, and then smoothed and projected to the three-dimensional physical domain [23] [24] [25]. Furthermore, a post-processing step could be performed to improve the quality of the mesh.

As the smoothing of the mesh is performed in the parametric domain, it is possible that the mesh created after projection will be deformed in the physical domain. To prevent this issue and preserve mesh quality, it may be desirable to optimize the mesh directly in the physical domain. Some meshing approaches capable of generating meshes from geometries with gaps and overlaps [26][27], or able to produce inter-domain boundaries having good geometric properties [28] have also been proposed.

Chapter 3

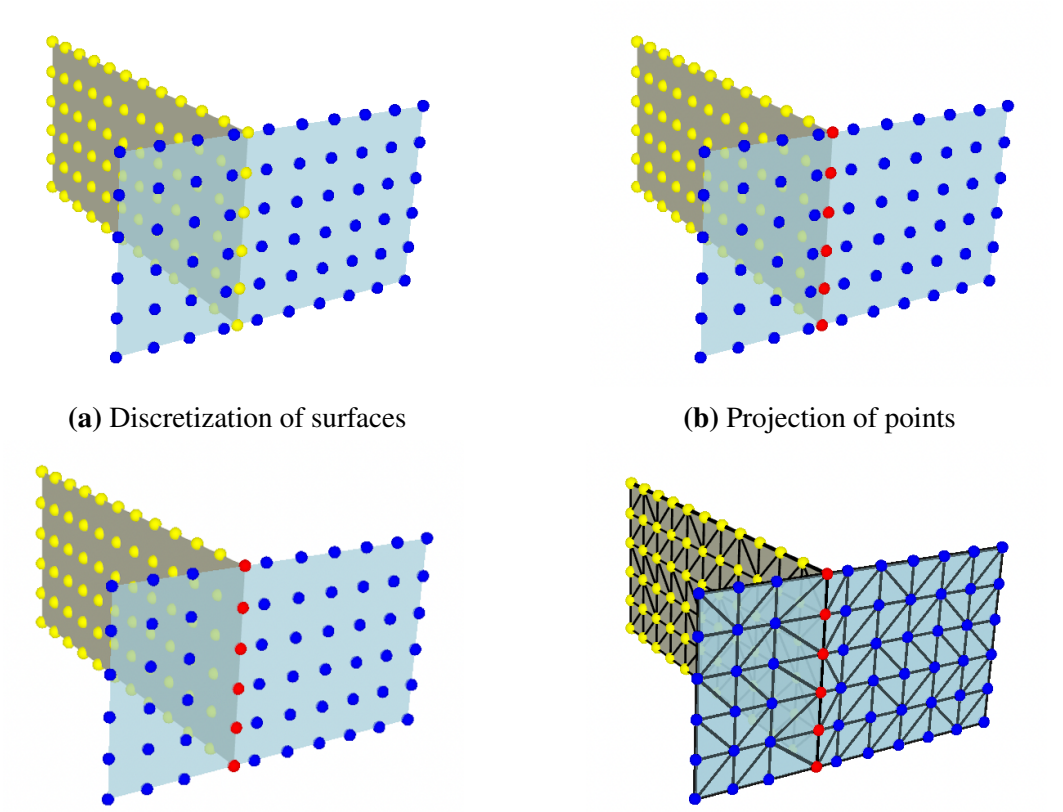
Methodology

3.1 Overview of the algorithm

The algorithm begins with the geometry consisting of a set of B-spline surfaces that represent different components of the aircraft, including the aircraft outer mold line (OML) and the internal structural members of the aircraft. The geometry can either be provided directly by an external database such as the VSP Hanger from NASA's Open Vehicle Sketch Pad (OpenVSP) [12] or obtained via an auxiliary geometry tool. Then, the B-spline surfaces are discretized and the intersection curves between different components of the aircraft are identified. Next, initial triangular meshes are generated by constrained Delaunay Triangulation (CDT) [29] with the constraint that the connectivity of the intersection curves is maintained. Afterwards, the mesh quality optimization algorithms are applied to the meshes of the OML and internal members separately. These optimization steps generate the quad-dominant meshes, which are transformed into a fully quadrilateral mesh with the final step of the Catmull–Clark subdivision [30].

3.2 Initial triangle mesh generation

Our mesh generation algorithm starts with a series of B-spline surfaces consisting of the aircraft outer mold line (OML) and the internal members, which can either come from an external database or from an affiliated geometry parametrization tool in a large-scale MDAO framework.



(a) Discretization of surfaces

(b) Projection of points

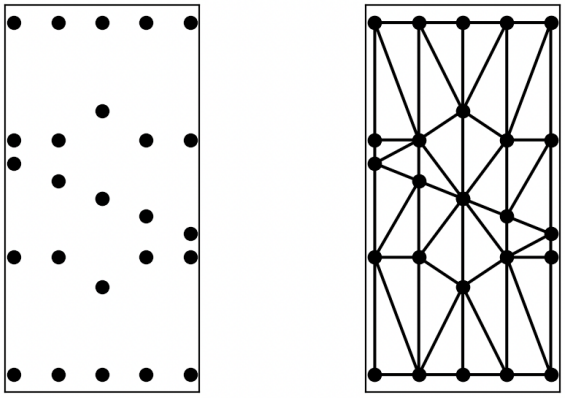
(c) Replacement of projecting points

(d) Construction of initial triangle meshes

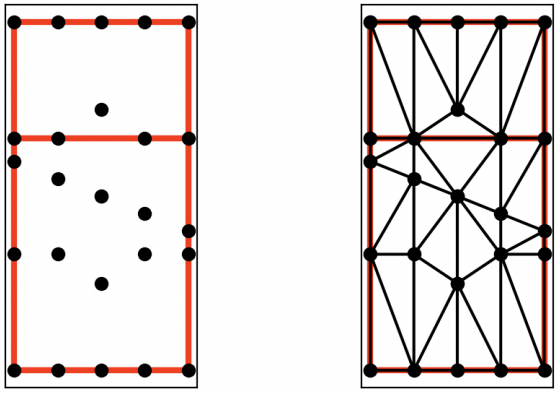
Figure 3.1. Workflow of the initial triangle mesh generation.

For B-spline surfaces directly from the outer database, there would be the situation that one member is divided into several B-spline surfaces. Then, we implement a preprocessing step of merging these surfaces by deleting the repeated points and refitting a new B-spline surface based on the remaining unique discretized points.

To maintain the connectivity of the edge-surface intersection between different surfaces, the nodes on the intersection curves are projected on their intersecting surface as shown in Fig.3.7b. The two-dimensional parameterization of these projected points can also be achieved in the projection process. Then, these projected points will respectively replace the closest original nodes near them (Fig.3.1c). Lastly, two-dimensional constrained Delaunay triangulation is implemented [31] to build the initial triangle meshes (Fig.3.1d) with the constraint of the sequential connectivity of the nodes on the intersection curve.



(a) Delaunay triangulation



(b) constrained Delaunay triangulation

Figure 3.2. Comparison between Delaunay triangulation and constrained Delaunay triangulation.

Delaunay triangulations are widely utilized in mathematics and computational geometry because of their advantageous geometric characteristics. The essential property in two-dimensional triangulation is the empty circumcircle condition. A Delaunay triangulation of a set of two-dimensional points assures that the circumcircle associated with each triangle contains no additional points. It also maximizes the minimum angle of all the angles of the triangles in the triangulation in order to avoid triangles with one or two exceptionally acute angles, thus enhancing the quality of the resulting triangular mesh. Since our surface is in three-dimensional space and is not sufficiently flat to be considered a plane, we employ constrained Delaunay triangulation in the parametric domain. There is also a need to perform constrained Delaunay

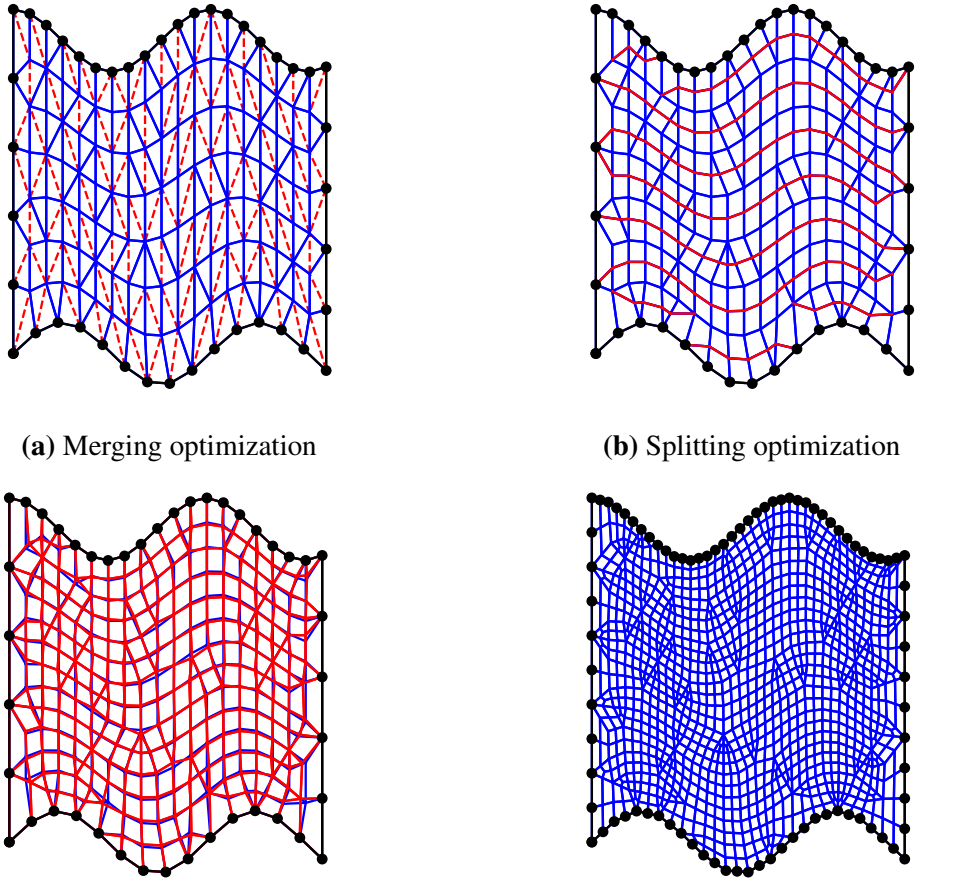
triangulation (CDT) to ensure the proper connection of the boundary curves and the intersecting curves between different members. A constrained Delaunay triangulation is a generalization of the Delaunay triangulation that forces certain required segments into the triangulation as edges [29]. The input to the constrained Delaunay triangulation problem is a planar straight-line graph consisting of a set of points and line segments in the plane that do not intersect. For each extra edge added to this input to form a triangle, there should be a circle through the endpoints such that any vertex within the circle is obscured from at least one endpoint. And there is always a triangulation satisfying these conditions. After CDT, the basic mesh is the result of mapping the vertices back onto the physical domain.

By now, the initial triangulation is completed, but a few following optimization steps are still required to achieve the quadrilateral-dominant mesh and the fully quadrilateral mesh.

3.3 Mesh quality optimization and fully quadrilateral transformation

In this part, the described methods are from another publication material as it may appear in Automatic generation of global shell-element meshes for large-scale structural design optimization in AIAA AVIATION 2020 FORUM 2020 . Li, Ning; John T. Hwang, 2020. The thesis author was not the primary investigator and author of this paper.

Three optimization problems are formulated to improve mesh quality and generate quadrilateral-dominant meshes before the final process of fully quadrilateral transformation. The splitting and merging optimization problems update the meshes topologically by adding or removing the edges of the meshes, while the smoothing optimization modifies the meshes geometrically by changing the node location of the meshes. By applying the splitting optimization to divide a quadrilateral or a triangle, the number of vertices in the mesh and the total number of polygons will grow. The merging optimization will combine two nearby triangles and reduce the total number of polygons. The smoothing optimization, on the other hand, does not change the



(a) Merging optimization

(b) Splitting optimization

(c) Smoothing optimization

(d) Fully quadrilateral transformation

Figure 3.3. The black line represents the fixed edges. In (a), the red dashed lines are removed edges resulting in the blue mesh. In (b), the red lines are splitting lines for the blue original mesh. In (c), the blue mesh are the original mesh, red mesh are the updated mesh after splitting. In (d), the mesh is transformed into a fully quadrilateral mesh using Catmull–Clark subdivision [1].

number of vertices or connectivity of polygons but rather adjusts the positions of the vertices. All three optimization methods intend to reduce the deviation of polygon edges from having equal lengths or make polygons more regular shaped with approximately equivalent angles.

Splitting optimization

Generally, the nodes generated by discretizing a b-spline surface are uniformly distributed. However, after projecting the points on intersection curves between different surfaces, the initial structured nodes will be moved to the location of the projected points. Thus, the following

constrained Delaunay triangulation may result in triangles with large aspect ratios and poor quality at the intersection regions. To address these irregular triangles and also extend the implementation to quadrilateral elements, the splitting optimization is devised to allow a triangle to be divided into a triangle and a quadrilateral or two triangles, as well as a quadrilateral into a triangle and a quadrilateral or two quadrilaterals. This will result in more regular polygons in the mesh. Due to the robustness and efficiency of the linear programming problem [32], we formulate the splitting optimization as follows:

$$\begin{aligned} \textbf{minimize} \quad & f_s = c_s^\top x_s \\ \textbf{with respect to} \quad & lb_s \leq x_s \leq ub_s \\ \textbf{subject to} \quad & A_s x_s = b_s \end{aligned} \tag{3.1}$$

The design variable x_s is a vector that indicates whether each splitting option for each polygon is chosen or not, i.e., 1 for being selected and 0 for not. Thus, its length is the number of splitting options times the number of polygons in the mesh. As shown in Fig. 3.4, there are eleven splitting options for a quadrilateral, including the option of not splitting. Splitting a triangle has fewer seven options as in Fig. 3.5, but we still assign eleven options to it since we want to make the number of splitting options the same for a triangle and a quadrilateral to make the assembling of coefficient vector c_s easier. The lower bounds lb_s and the upper bounds ub_s for each option are usually 0 and 1, except for two situations. One is that the upper bounds ub_s of the last four triangle splitting options are constrained to 0 to keep the uniformity of x_s between triangle and quadrilateral. The other is that for the polygons with the edges consisting of intersection curves between the internal members and the model outer skin, the upper bound of every option is set to 0.

To make the polygons as regular as possible after splitting, we define the indicator i_s to represent the regularity for each triangle or quadrilateral based on the aspect ratio and the difference between the interior angles in each polygon and the standard angle in an equilateral triangle

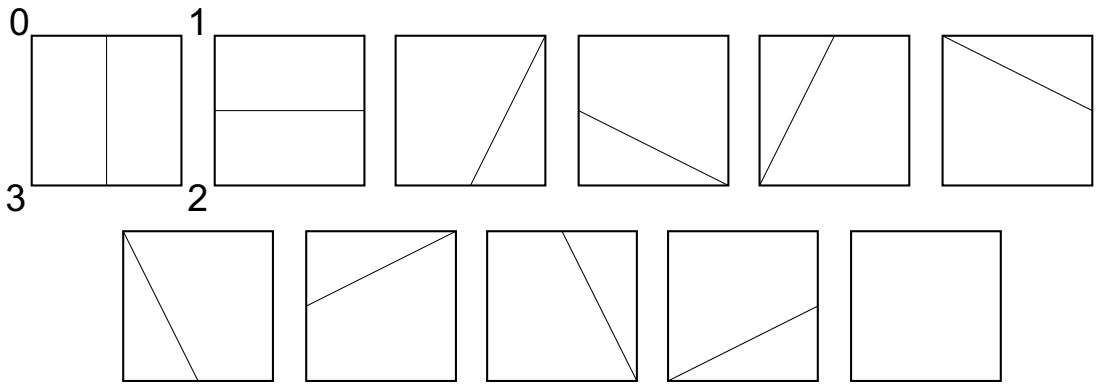


Figure 3.4. The eleven options to split a quadrilateral can be divided into three types: splitting by a line connecting the middle points of two opposite edges; splitting by a line connecting a vertex and the middle point of one opposite edge; and not splitting at all. The number surrounding the top left quad (option 0) is the local indexing of the quad. From the top left to the bottom right, the options are indexed from 0 to 10 [1].

or quadrilateral, i.e., 60 degrees for triangles and 90 degrees for quadrilaterals. Specifically, indicator i_s is represented as:

$$i_s = w_1 \sum_{i=0}^{n} \left(\frac{l_i}{l_{avg}} \right)^2 + w_2 \sum_{i=0}^{n} \left(\frac{\theta_i}{\theta} \right)^2 \tag{3.2}$$

where n is the number of edges in a polygon and equals 3 for a triangle or 4 for a quadrilateral. l_i is the length of each edge in the polygon, and l_{avg} is the average length of all the three or four edges in this polygon. θ_i is the angle of each corner in a polygon, w_1 and w_2 are the weighting coefficients regarding edge lengths and angles, and θ is the aforementioned standard angle. The coefficient vector c_s, with the same length as the design variable x_s, informs how much we prefer an splitting option of each polygon according to the result. So each component in c_s can be calculated as the total value of the indicator i_s for each polygon resulting from each splitting option. For instance, if the option is to split a quadrilateral into two new quadrilaterals, the component will be calculated as the sum of two indicators i_s, wihch is of two new quadrilaterals.

The equality constraints are based on two criteria: only one splitting option being chosen for each polygon and consensus regarding the splitting of shared edges. For the former, since every option contrasts with the other options and the polygon is either split (1) or not split (0),

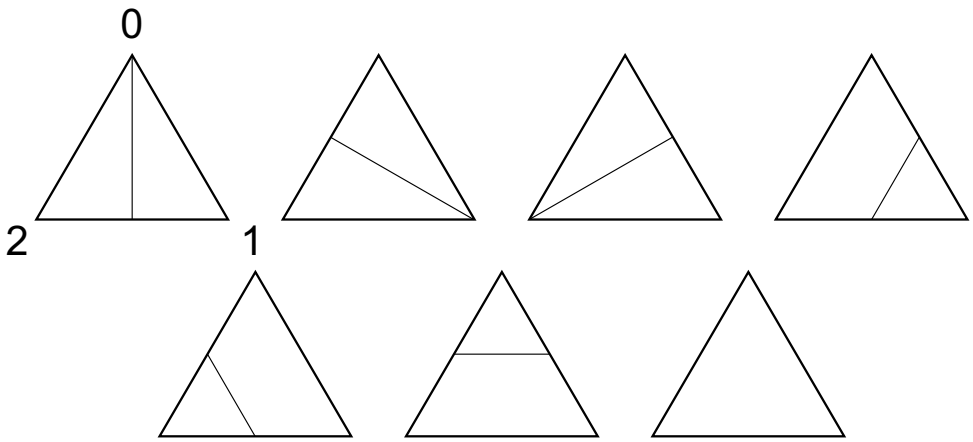


Figure 3.5. The seven options to split a triangle can be divided into three types: splitting by a line connecting the middle points of two opposite edges; splitting by a line connecting a vertex and the middle point of one opposite edge; and not splitting at all. The number surrounding the upper left triangle (option 0) is the local indexing of the triangle. The options are indexed from 0 to 6 [1].

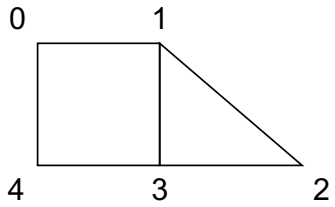


Figure 3.6. An example showing the formulation of the equality constraint: The numbers are the indices for each vertex in the mesh. The quad can be presented as $(0,1,3,4)$ and the triangle can be expressed as $(1,2,3)$. They have one shared edge [1].

the sum of all option decisions for one original polygon must equal one. The latter constraint considers the adjacency of each polygon and constructs the agreement on whether to split the shared edges of adjacent polygons or not. Using the simple mesh in Fig.3.6 as an example, the quadrilateral (0, 1, 3, 4) and the triangle (1, 2, 3) share one common edge (1, 3) and must agree on whether to split this edge. For the triangle, the local indices of the shared edge are (2, 0) according to Fig.3.5. For the quadrilateral, the local indices of the shared edge are (1, 2) and options 0, 2, 6 tend to split the shared edge, but options 1, 3, 4, 5, 7, 8, 9, 10 do not split the shared edge. Since the dividing decisions for the quad and triangle are the same, we may

formulate a linear equality constraint:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_q \end{bmatrix} \tag{3.3}$$

where x_t represents the splitting decision of the triangle, while x_q represents the splitting decision of the nearby quadrangle. The first row of the first matrix indicates that the triangle and quadrangle agree to divide the shared edge, while the second row indicates that they agree to retain the shared edge. This matrix can be transferred to the following format for the sake of the formulation of the equality constraint:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} x_t \\ x_q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.4}$$

Merging optimization

Due to the irregular triangles distributed around the intersection curves, merging optimization is developed to merge two adjacent triangles into a high-quality quadrilateral that resembles a square. Merging optimization will result in a quad-dominant mesh, and it will also help to reduce the number of edges and polygons in the mesh. This reduction will moderate the resolution of the final mesh and decrease the computational cost for the following optimization or potential analysis steps. Linear programming problem is also used for merging optimization just like splitting optimization. The detailed formulation is shown as:

$$\begin{aligned} \textbf{minimize} \quad & f_m = c_m^\top x_m \\ \textbf{with respect to} \quad & lb_m \leq x_m \leq ub_m \\ \textbf{subject to} \quad & A_m x_m \leq b_m \end{aligned} \tag{3.5}$$

The design variable x_m is a vector that indicates whether each edge is merged or not, thus its length is the number of total edges in the mesh. The binary component in vector x_m would be 1 for merging the edge while 0 for not merging it. Therefore, the lower bounds lb_m and the upper bounds ub_m are usually 0 and 1, except for the edges that are not shared by two triangles and the edges consisting of intersection curves between different geometry components. Since the former would be the edges that shared by two quadrilaterals or a quadrilateral and a triangle, and the merging result will not be an quadrilateral and cannot happen. The latter are edges that are supposed to be fixed to keep the internal structure. Therefore, the upper bounds of these edges are set to 0.

We also define indicator i_m to describe the regularity achieved from merging result. It has the exact same first two items as indicator i_s defined for the splitting algorithm, i.e. one for aspect ratio and the other one for interior angles. But there is also a third item that comes from the penalty of the dihedral angle of the polygon. The indicator i_m is shown as:

$$i_m = w_1 \sum_{i=0}^{n} \left(\frac{l_i}{l_{avg}} \right)^2 + w_2 \sum_{i=0}^{n} \left(\frac{\theta_i}{\theta} \right)^2 + w_3 \sum_{i=0}^{n} e_i^{\top} n_q \tag{3.6}$$

w_3 is the weighting coefficient for dihedral angle. e_i is the unit vector pointing along each polygon edge, and n_q is the normal vector determined as the cross product of the unit vectors pointing in the two diagonal directions. Each component in c_m is calculated by subtracting the indicator of each polygon before and after the merging. The shared edge is more likely to be merged when the indicator after merging is lower and the energy before merging is higher.

The inequality restriction arises from the criterion that only one edge per triangle may be merged, as allowing several edges to be merged in a triangle could result in the formation of a polygon with more than four edges. Due to the fact that our algorithm only expects triangle, quad-dominant, and totally quadrilateral meshes, we do not want this to occur. To construct the matrix A_m and b_m in the inequality constraint, we first loop through all of the edges to obtain the indices for each polygon's edges. This will result in a matrix whose size is the number of

polygons times three. The inequality constraint is formed by designating corresponding elements in A_m as one with respect to the generated matrix and setting all of the items in b_m.

Smoothing optimization

Due to the element-wise computation of the regularity of the mesh elements, the splitting and merging optimization tends to optimize the mesh quality locally by focusing on the quality of each element. Consequently, although there exists the constraint that two adjacent polygons must have the same splitting or merging operation on their common shared edge, the final mesh may have different element sizes around different regions. And the global aspect ratio could remain large after splitting and merging optimization. To address this flaw, the smoothing optimization attempts to modify the location of each nodes in the mesh to reduce the aspect ratio globally and enhance the overall mesh quality. All nodes are allowed to move within a small range in their tangential plane, and the sum of all edge lengths is minimized with respect to the displacement vector. The smoothing optimization method is formulated as a constrained quadratic programming (QP) problem:

$$\begin{aligned} \textbf{minimize} \quad & f_{sm} = \frac{1}{2}[(v_0^T + d^T[I_0]) - (v_1^T + d^T[I_1])] \cdot [(v_0 + d[I_0]) - (v_1 + d[I_1])] \\ \textbf{with respect to} \quad & d \\ \textbf{subject to} \quad & Md \leq r \\ & Nd = 0, \end{aligned} \tag{3.7}$$

where D is the vector that represents the displacement for all the nodes along three dimensional directions, so its length is three times the number of nodes in the mesh. V_0 and V_1 are the initial positions of starting points and end points for all edges, I_0 and I_1 are the indices of them, and $V_0 + D[I_0]$ and $V_1 + D[I_1]$ are the updated positions of the starting points and end points for all edges. The variable $(V_0 + D[I_0]) - (V_1 + D[I_1])$ represents the vector along the edge by subtracting the updated end points from the updated starting points. Then, the objective function

is formulated to indicate the lengths of every updated edge in the mesh.

In the inequality constraint, M is the sparse matrix form of d with the shape of number of node times triple number of nodes. The three-dimension coordinates of each point are iteratively assigned to their corresponding positions in the column for each row, while all other positions in the column remain zero. Multiplying M and d, we can measure the squared distance that each vertex moves. This distance is smaller than r, the vector of maximum radius each vertex allowed to move, which is computed by the distance between the closest neighboring vertex and itself times a coefficient.

In the equality constraint, N represents the normal matrix produced by assigning the normal vector of each vertex to the associated places in each row. By assigning the displacement vector to the normal vector, which is determined as the area-weighted normal of surrounding triangles for each vertex, the equality constraint ensures that each vertex remains in its own tangential plane.

To simplify the QP problem, the inequality constraint is added to the objective function as a regularization term:

$$\begin{aligned} \textbf{minimize}\ \ & f_{sm} = \frac{1}{2}[(V_0^T + D^T[I_0]) - (V_1^T + D^T[I_1])] \cdot [(V_0 + D[I_0]) - (V_1 + D[I_1])] + \frac{1}{2} D^T w D \\ \textbf{with respect to}\ \ & D \\ \textbf{subject to}\ \ & nD = \mathbf{0} \end{aligned} \tag{3.8}$$

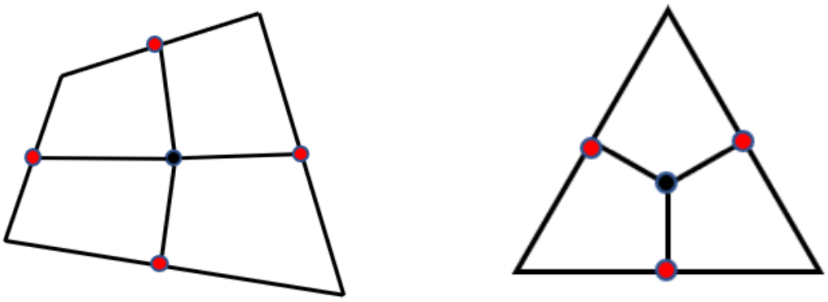
where w is the vector of regularization parameter decided by the local curvature for each vertex. If some points are located in the area where the curvature is found to be very high or the constrained area to keep the intersection connectivity, the regularization parameters are set to an extremely large number to prevent any movement. Eqn. 3.8 can be solved by applying the Karush–Kuhn–Tucker (KKT) conditions:

$$\begin{bmatrix} B & N^\top \\ N & \mathbf{0} \end{bmatrix} \begin{bmatrix} D^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix} \tag{3.9}$$

where B is the partial derivative of the first term in the objective function with respect to D, D^* and λ^* are the optimal displacements and associated Lagrange multiplier, b is the coefficient of the linear term and all the components in c equals zero. Since we presume that vertices that move along the tangential plane within a restricted range can still be deemed "near" to the actual geometry, the regularization term w is set to a large value, and the vertices are not permitted to move very far during each position iteration. Therefore, several smoothing optimization steps are done sequentially, which could allow the geometry to be maintained while moving the vertices a reasonable amount.

Fully quadrilateral transformation

Following the aforementioned optimization step, the initial triangular meshes have been transformed into quadrilateral-dominant meshes. To convert them into fully quadrilateral meshes, one step of the Catmull-Clark subdivision is needed. Firstly, an average centering node is added to each polygon. Then, the middle points between the two endpoints of each edge are added to each edge. Next, edges between the average nodes and all of the middle points are added for each polygon. Thus, the meshes of structural members are subdivided into full quadrilateral meshes (Figure 3.7).



(a) Division of quadrilateral **(b)** Division of triangle

Figure 3.7. The black nodes are the average nodes. The red points are the middle points of each edge. Connecting the average points to the middle points, a polygon is divided into four quadrilaterals. [2].

3.4 Interface with large-scale MDAO framework

Our mesh generation algorithm is expected to interface with an under-development large-scale MDAO framework, which will be presented in a series of papers. We use the aircraft geometry parametrization part of this framework as an auxiliary geometry tool to generate B-spline surfaces of the internal structure. This auxiliary tool creates components by defining the number and location of internal members to obtain a structural model driven by the initial central geometry parametrically. Taking the B-spline surfaces generated by this tool, the generated structural mesh can also be easily updated as the central geometry changes.

Exportation of initial b-spline surface

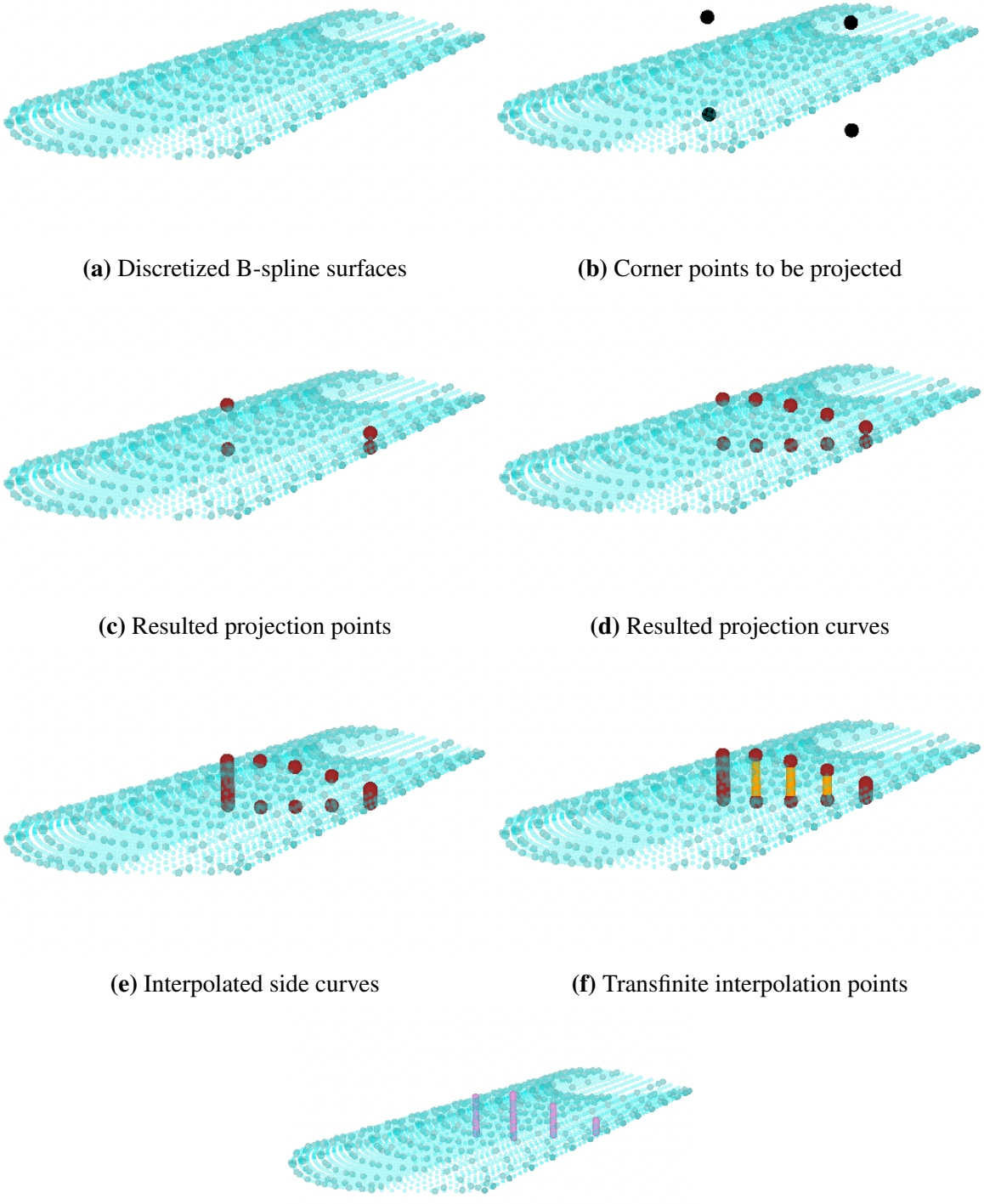
The general class of non-uniform rational B-splines (NURBS) is the most widely considered spline-based mathematical function in CAD and IGA. Given a set of $m+1$ rows and $n+1$ control points $\mathbf{p}_{i,j}$, where $0 <= i <= m$ and $0 <= j <= n$, and two knot vectors of $h+1$ and $k+1$ knots respectively in the u-direction and v-direction, $U = u_0, u_1, ..., u_h$, $V = v_0, v_1, ..., v_k$. The B-spline surface defined by these information is the following [33]:

$$\mathbf{P}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) \mathbf{p}_{i,j} \tag{3.10}$$

where $N_{i,p(u)}$ and $N_{j,q(v)}$ are B-spline basis functions of degree p and q, respectively. Note that the fundamental identities for each direction must hold: $h = m + p + 1$ and $k = n + q + 1$.

Using the model of eVTOL as an example, the model from OpenVSP as Standard for the Exchange of Product Model Data (STEP) format is exported as a bunch of b-spline surfaces (Figure 3.8a). To establish a rib in an aircraft wing, for instance, we first identify two points straight above the wing and two points straight below the wing, as shown in Figure 3.8b, which are later projected to the upper and lower wing B-spline surfaces, respectively (Figure 3.8c). Then we discretize the line segment connecting two points on the same side of the wing and discretize the line again to project the whole curve on the wing surfaces as in Figure 3.8d. Next,

we perform linear interpolation to create two side curves (Figure 3.8e). With four boundary curves, the four sides of the member surface are defined. Therefore, transfinite interpolation is applied to get a continuous parametric surface based on the four boundary curves as presented in Figure 3.8f. Finally, in Figure 3.8g, a B-spline surface is fitted using the points generated by transfinite interpolation.



(a) Discretized B-spline surfaces

(b) Corner points to be projected

(c) Resulted projection points

(d) Resulted projection curves

(e) Interpolated side curves

(f) Transfinite interpolation points

(g) Control points the created members

Figure 3.8. Workflow of the creation of internal members.

Acknowledgements

Chapter 3 section 3.2 and section 3.4, in full is currently being prepared for submission for publication of the material. Liu, Xiangbei; Xiang, Ru; Pasetto, Marco; Li, Ning; Kamensky, David M; Hwang, John T. The thesis author was the primary investigator and author of this material.

Chapter 3 section 3.3, in full is currently being prepared for submission for publication of the material. Liu, Xiangbei; Xiang, Ru; Pasetto, Marco; Li, Ning; Kamensky, David M; Hwang, John T. The thesis author was not the primary investigator of this material.

Chapter 3 section 3.3, in full, is also a rephrase of the material as it appears in Automatic generation of global shell-element meshes for large-scale structural design optimization in AIAA AVIATION 2020 FORUM 2020 . Li, Ning; John T. Hwang., 2020. The thesis author was not the primary investigator and author of this paper.

Chapter 4

Results

4.1 Algorithm versatility and mesh quality validation

To test the versatility and robustness of our mesh generation algorithm, we perform several tests on the Uber's eVTOL eCRM-002 model [1], a common reference model (CRM) of an electric air taxi, and the undeflected common research model (uCRM-9) [2].

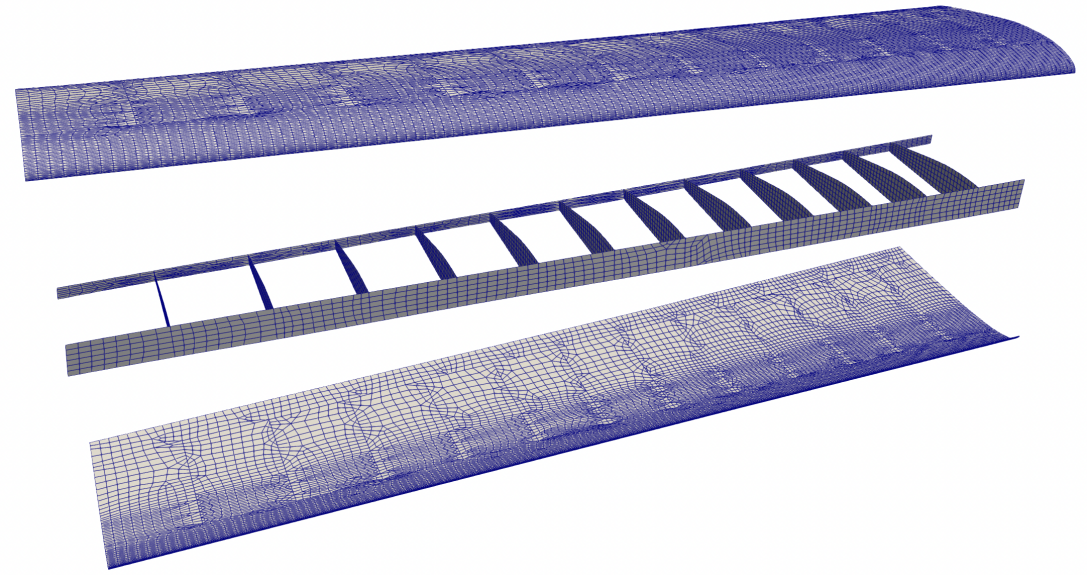


Figure 4.1. Exploded view of the mesh for the eVTOL wing with 12 ribs and 2 spars. The total number of nodes is 26486. The total number of elements is 26936.

For eVTOL model, the initial geometry comes from NASA's conceptual aircraft design

[1] http://hangar.openvsp.org/vspfiles/524

[2] https://data.mendeley.com/datasets/gpk4zn73xn/1

tool, OpenVSP. The skin part of the wing structure is kept, while landing gears, inner lift nacelles, lift propellers, and cruise propellers are removed to make the illustration more clear. Then we use the auxiliary geometry to create the B-spline surfaces of the 12 ribs and 2 spars on the wing. As depicted in Fig.4.1, the junction curves between the internal members and the aircraft wing skin appear to align properly, and all internal member intersections are identified correctly. For the surface of the wing, the mesh quality improvement algorithms construct elements with more regular shapes in the area between the intersection curves of the internal members. In the area around the intersection curves, the elements are less regular with a large aspect ratio since the boundary points are fixed throughout the optimization process, preventing them from moving to increase the quality of the meshes. However, the overall quality is regarded to be good considering 82% of the mesh elements have internal angles in the range of 70 degrees to 110 degrees, deviating 20 degrees from the ideal angle of 90 degrees.

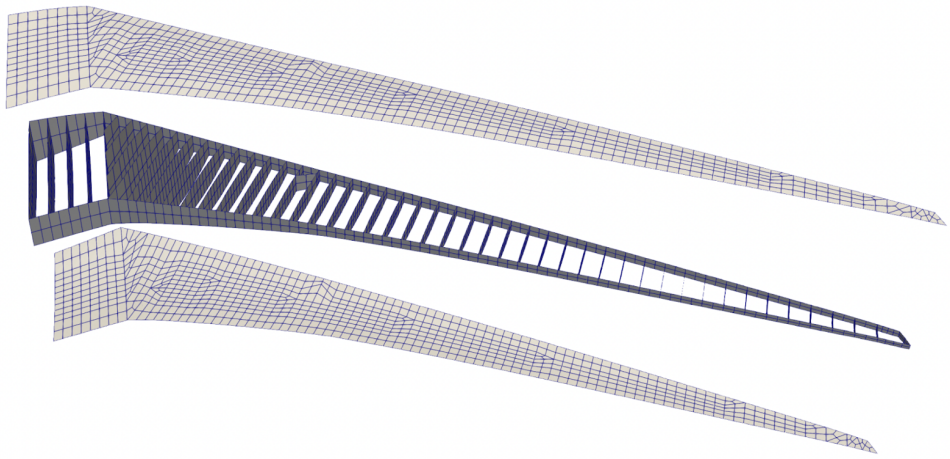


Figure 4.2. An exploded view of the mesh for the uCRM-9 wingbox. The total number of nodes is 2368. The total number of elements is 3802.

We use the structural geometry of the uCRM-9 wingbox from Brooks's paper [34] to produce a quadrilateral mesh as shown in Fig.4.2. Then, the B-spline surfaces are discretized and the connectivity between different members is constructed. The optimization algorithm is also executed after the initial triangulation of each member. In Brooks's paper, they generated the

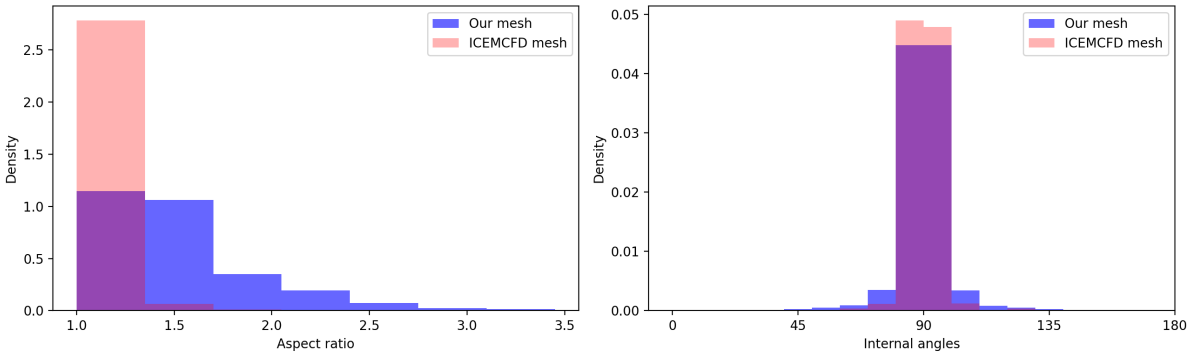


Figure 4.3. The comparison histogram of the aspect ratio and the internal angle between our meshes and ICEMCFD meshes.

meshes for the geometry of the wingbox using the commercial grid-generation tool ICEMCFD. To verify the quality of our generated meshes, we plot the comparison histogram between our meshes and ICEMCFD meshes with respect to the aspect ratio, which is the length of the longest edge divided by the shortest one, and the four internal angles for every polygon in the quadrilateral mesh. As shown in Fig.4.3, the meshes generated by our algorithm have similar quality as the ICEMCFD mesh, with a small aspect ratio and internal angles around 90 degrees.

4.2 Mesh analysis and convergence result

We also perform a demonstrative stress shell-analysis of the uCRM-9 wingbox, specifically, the FEniCS implementation of the Reissner–Mindlin shell element presented in [35]. We clamped the wing root as the boundary condition, and a distributed upward load of body force equal to the weight of the wing is applied. We assume the thickness of the shell is 3mm and the material of wingbox is the aluminum with Young's modulus of 73.1 GPa, Poisson's ratio of 0.3, and density of 2780 kg/m^3. The quantity of interest for this analysis is the vertical displacement at the wingbox tip. To validate the results, we performed the same analysis on an ICEMCFD mesh. It is clear that the displacement has converged to the maximum value of 0.1776 m (Fig. 4.4). The displacement field of the wingbox is also shown in Fig. 4.5.

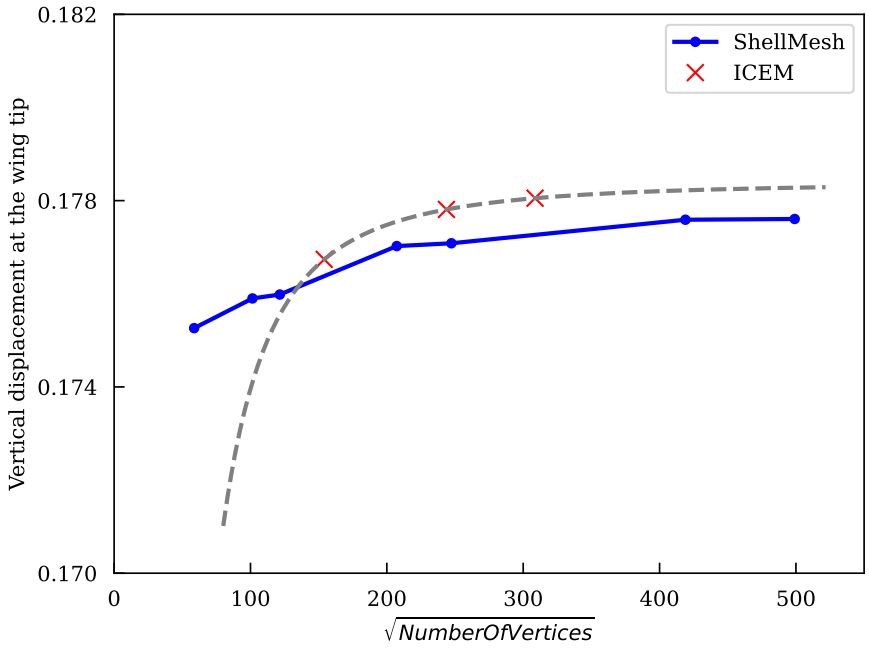


Figure 4.4. Convergence test comparing meshes generated by ShellMesh and ANSYS ICEMCFD of an uCRM wingbox model.

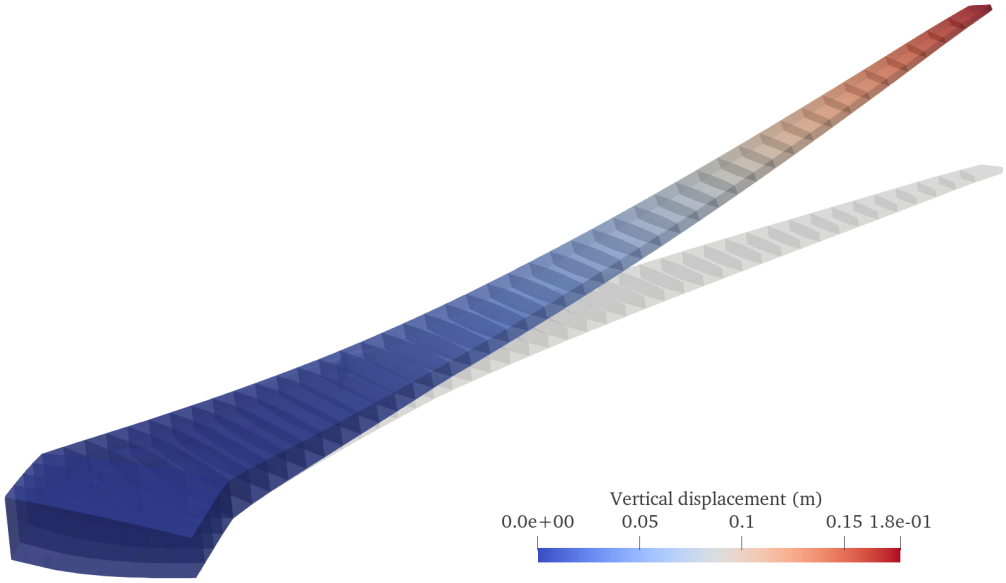


Figure 4.5. Displacement field of an uCRM wingbox with upward distributed loads.

Acknowledgements

Chapters 4, in full is currently being prepared for submission for publication of the material. Liu, Xiangbei; Xiang, Ru; Pasetto, Marco; Li, Ning; Kamensky, David M; Hwang, John T. The thesis author was the primary investigator and author of this material.

Chapter 5

Conclusion

In this research, I significantly improved a novel approach for generating shell element meshes for large-scale MDAO based on prior work. Starting with the geometry model consisting of B-spline surfaces, I implement the initial constrained triangulation of the discretized nodes. After a series of splitting, merging, and smoothing optimizations inherited from previous work, the mesh quality of the skin and the internal members increases.

The first contribution is to make the algorithm handle a versatile input with any given B-spline geometry model. This means the algorithm is also applicable to geometries besides aircraft. For instance, it can be applied to the generation of shell-element meshes for other engineering systems, such as cars, robots, medical devices, etc.

Another remarkable contribution is implementing the interface with a large-scale MDAO framework. As a result, the mesh is able to be computed as a function of design variables since the changes in the initial central geometry will cause the structural mesh to automatically transform. Additionally, the physical coordinates of the mesh nodes are differentiable so that the computation of the structural mesh coordinates' derivatives with respect to the form design variables will be efficient.

In conclusion, the entire algorithm is useful and efficient for generating shell-element meshes and adding internal components to structures with complex shapes. Using structural shell-element meshes integrated with the gradient information, exact and efficient analysis and

design optimization could be implemented. Furthermore, a shape design optimization problem will be performed using the aforementioned large-scale MDAO framework.

Bibliography

[1] Ning Li and John T Hwang. Automatic generation of global shell-element meshes for large-scale structural design optimization. In *AIAA AVIATION 2020 FORUM*, page 3135, 2020.

[2] Ning Li. Automatic generation of global shell-element meshes for large-scale structural design optimization. Master's thesis, University of California San Diego, 2020.

[3] Pedro D Bravo-Mosquera, Fernando M Catalano, and David W Zingg. Unconventional aircraft for civil aviation: A review of concepts and design methodologies. *Progress in Aerospace Sciences*, 131:100813, 2022.

[4] Manuel A Rendón, Carlos D Sánchez R, Josselyn Gallo M, and Alexandre H Anzai. Aircraft hybrid-electric propulsion: Development trends, challenges and opportunities. *Journal of Control, Automation and Electrical Systems*, 32(5):1244–1268, 2021.

[5] Tae H Ha, Keunseok Lee, and John T Hwang. Large-scale design-economics optimization of evtol concepts for urban air mobility. In *AIAA Scitech 2019 Forum*, page 1218, 2019.

[6] Bernd Liebhardt, Volker Gollnick, and Klaus Luetjens. Estimation of the market potential for supersonic airliners via analysis of the global premium ticket market. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIAA Balloon Systems Conference and 19th AIAA Lighter-Than*, page 6806, 2011.

[7] John T Hwang, Arnav V Jain, and Tae H Ha. Large-scale multidisciplinary design optimization—review and recommendations. In *AIAA Aviation 2019 Forum*, page 3106, 2019.

[8] John T Hwang and Joaquim RRA Martins. An unstructured quadrilateral mesh generation algorithm for aircraft structures. *Aerospace Science and Technology*, 59:172–182, 2016.

[9] Tanja Führer, Christian Willberg, Sebastian Freund, and Falk Heinecke. Automated model generation and sizing of aircraft structures. *Aircraft Engineering and Aerospace Technology: An International Journal*, 88(2):268–276, 2016.

[10] John T. Hwang and Joaquim RRA Martins. GeoMACH: geometry-centric MDAO of aircraft configurations with high fidelity. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012.

[11] Gaetan Kenway, Graeme Kennedy, and Joaquim RRA Martins. A cad-free approach to high-fidelity aerostructural optimization. In *13th AIAA/ISSMO multidisciplinary analysis optimization conference*, page 9231, 2010.

[12] Robert A McDonald and James R Gloudemans. Open vehicle sketch pad: An open source parametric geometry and analysis tool for conceptual aircraft design. In *AIAA SCITECH 2022 Forum*, page 0004, 2022.

[13] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):189–1, 2015.

[14] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2), 2008.

[15] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schöder. Globally optimal direction fields. *ACM Trans. Graph.*, 32(4), 2013.

[16] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schöder. Stripe patterns on surfaces. *ACM Trans. Graph.*, 34(4), 2015.

[17] Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J. Guibas. Quadriflow: A scalable and robust method for quadrangulation. *Computer Graphics Forum*, 37(5):147–160, 2018.

[18] Carlos A. Recarey Morfa, Irvin Pablo Pérez Morales, Márcio Muniz de Farias, Eugenio Oñate Ibañez de Navarra, Roberto Roselló Valera, and Harold Díaz-Guzmán Casañas. General advancing front packing algorithm for the discrete element method. *Computational Particle Mechanics*, 5:13–33, 2018.

[19] Alexander Banks Costenoble, Bharath Govindarajan, Yong Su Jung, and James Baeder. Automatic mesh generation and solution analysis of arbitrary airfoil geometries. In *23rd AIAA Computational Fluid Dynamics Conference*, 2017.

[20] Jasmeet Singh and Carl F. Ollivier-Gooch. Advancing layer surface mesh generation. In *AIAA Scitech 2020 Forum*, 2020.

[21] Scott A. Canann, Senthilmurugan N. Muthukrishnan, and R. K. Phillips. Topological improvement procedures for quadrilateral finite element meshes. *Engineering with Computers*, 14(2):168–177, 1998.

[22] V.N. Parthasarathy and Srinivas Kodiyalam. A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design*, 9(4):309–320, 1991.

[23] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover-surface parameterization using branched coverings. *Computer Graphics Forum*, 26:375–384, 2007.

[24] Jianwei Guo, Fan Ding, Xiaohong Jia, and Dong-Ming Yan. Automatic and high-quality surface mesh generation for cad models. *Computer-aided design*, 109:49–59, 2019.

[25] Pierre-Alexandre Beaufort, Christophe Geuzaine, and Jean-François Remacle. Automatic surface mesh generation for discrete models – a complete and automatic pipeline based on reparametrization. *Journal of Computational Physics*, 417:109575, 2020.

[26] Y. K. Lee, Chin K. Lim, Hamid Ghazialam, Harsh Vardhan, and Erling Elkund. Surface mesh generation for dirty geometries by the cartesian shrink-wrapping technique. *Engineering with Computers*, 26(4):377–390, 2010.

[27] Benjamin Villard, Vicente Grau, and Ernesto Zacur. Surface mesh reconstruction from cardiac mri contours. *Journal of Imaging*, 41(1), 2018.

[28] Dawei Zhao, Jianjun Chen, Yao Zheng, Huang Zhengge, and Jianjing Zheng. Fine-grained parallel algorithm for unstructured surface mesh generation. *Computers & Structures*, 154:177–191, 2015.

[29] L Paul Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989.

[30] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.

[31] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Workshop on Applied Computational Geometry*, pages 203–222. Springer, 1996.

[32] David G Luenberger and Yinyu Ye. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[33] William J Gordon and Richard F Riesenfeld. B-spline curves and surfaces. In *Computer aided geometric design*, pages 95–126. Elsevier, 1974.

[34] Timothy R Brooks, Gaetan KW Kenway, and Joaquim RRA Martins. Benchmark aerostructural models for the study of transonic aircraft wings. *AIAA Journal*, 56(7):2840–2855, 2018.

[35] EMB Campello, PM Pimenta, and P Wriggers. A triangular finite shell element based on a fully nonlinear shell formulation. *Computational mechanics*, 31(6):505–518, 2003.