Automated Wingbox Structure Generation Through MATLAB

Jason Qian[*]

Stanford University, Stanford, CA 94305, USA

This paper describes a one-step procedure in generating a finite element model of an aircraft wingbox through a MATLAB-based environment, with the aim of significantly reducing the time to create a detailed preliminary structure. By parametrically defining the geometry and components, the finite element model can be built completely and quickly through MATLAB (within a minute). All major structural components can be defined (spar webs and caps, ribs, stringers and skin) to create a high-fidelity model, with wingbox of different layouts (transport, unconventional and fighter aircrafts) constructed to demonstrate the viability of the method. A final design optimization is then conducted through Nastran SOL 200 to complete the preliminary wingbox structure for an example Boeing 747-400 case.

I. Introduction

Conventionally finite element models are built through a CAD/CAE program or a pre-processing software. While they can provide highly detailed models, it is a time-consuming process and slows down the overall design process. For a conventional wing, this process can be significantly sped up through parametric modeling and automation due to the fact that the structure is composed of an orderly layout of spars, ribs, stiffeners and skin.

While CAD or pre-processing software do allow a certain degree of parametric modeling (as that shown in the case with MSC Patran[1] and CATIA[2]), an aircraft's wingbox is sufficiently complex with a large number of design variables to require the use of a specifically-written code. Sensmeier *et al.* developed a GUI-based tool OptWing, which allowed the user to enter the wing geometry and components before the wingbox was meshed.[3] Through the rapid generation, they demonstrated the capability to perform optimization and sizing routines by submitting models into the finite element solver. A similar GUI-based system was developed by Jiapeng *et al.* to parametrically define the wing geometry and components.[4] Hwang *et al.*[5] showed the ability to create a mesh of an entire aircraft by parametrically defining the geometry and creating splines of the outline in order to mesh the surface.

This paper aims to introduce a highly-parametrized and automated method of generating a finite element wingbox through MATLAB. The process should have a fast turnaround time (computationally quick) and the model being of a high fidelity, with all major structural components explicitly defined through shell elements. The resulting model can then be coupled with external software to provide design analysis and optimization.

MATLAB is suited for the purposes for several reasons. MATLAB's main advantage lies in its high-level language and vast amount of in-built libraries which allows the code to be easily prototyped and adapted to different situations, as shown by Skillen and Crossley's ability to rapidly generate an unconventional wingbox though MATLAB.[6] MATLAB also has excellent 2D and 3D visualization tools to verify the input geometry and the corresponding mesh. Furthermore, as the entire process is contained within the MATLAB environment, file input/output (IO) to external software (Gmsh, Nastran, AVL) can be integrated seamlessly

[*]Masters Student, Department of Aeronautics and Astronautics, Student Member AIAA.

within the generation and post-processing stages. A disadvantage of MATLAB is its slower speed compared to lower-level languages. These were offset through the use of optimized functions and resolving bottle-necks in the code by calling lower-level C++ MEX functions. As such, the entire generation process is very quick, taking approximately 60 seconds for 40,000 elements.

As the code is in its preliminary stages, all code is written and run through a series of interrelated MATLAB script files/functions, with aircraft data inputed manually as variables. Methods to make the process more user friendly have not yet been attempted, though possible through a GUI.

II. Methodology

II.A. Code Structure

The code structure is shown in Figure 1 (specific MATLAB functions denoted with the extension `.m` while those in italics indicate a block of functions). The initial function `WingSpecs.m` requires the geometry, structure and aircraft specifications to be entered manually with the data inputs described in Sections II.B and II.C. The following two functions (`MakeLines.m` and `MakeFEM.m`) are autonomous in generating the mesh and are described in Sections II.D. The *RunAVL* block consists of a series of functions which run Athena Vortex Lattice to provide a simplistic aerodynamic loading for optimization. The last block *NastranIO* consists a series of functions which can interface (write and read) with Nastran files for structural optimization. Auxiliary plotting functions are attached or in-built into each of the processes to display and verify the results.

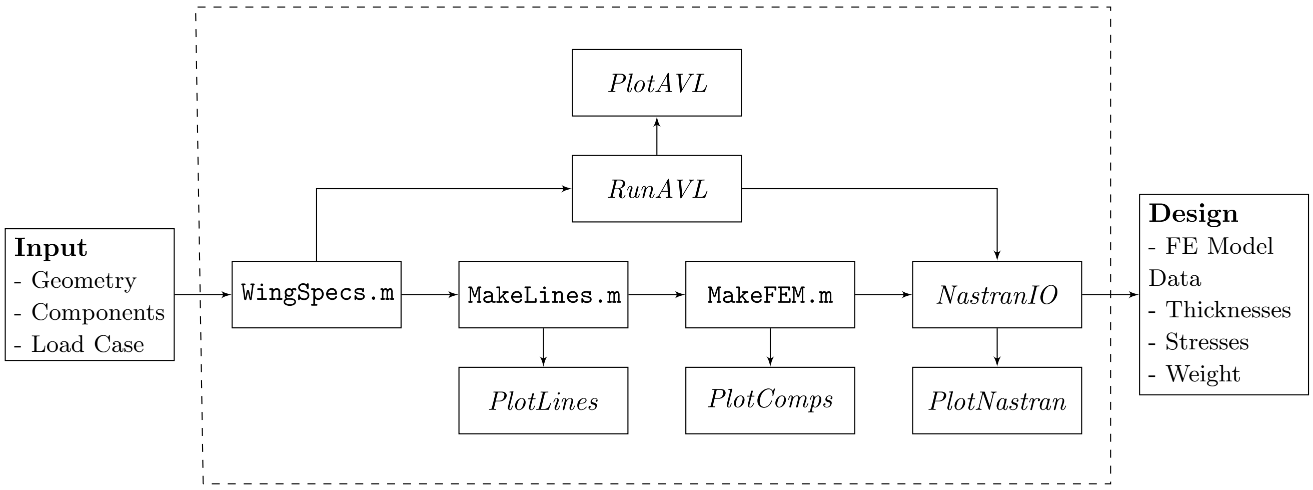

Figure 1: MATLAB Code Flowchart

The data variables generated from the `MakeFEM.m` function is stored in a MATLAB `.mat` data file so to allow further use in the process. Due to the speed of the other functions no data files are created after the run.

II.B. Defining the Wing Geometry

The first step involves defining the geometry and aerofoil sections of the wing planform. A parametric coordinate system is used in place of a Cartesian system, similar to that done by Sensmeier and Samareh.[7] The following geometric parameters can be defined:

- *Wingspan* (b): the overall wingspan, with the span of FE model equal to $b/2$.
- *Leading edge sweep* (x_{LE}): the leading edge sweep location as a function of the span.
- *Root chord*: the chord length at the root (this includes any lengths provided by the leading or trailing edge devices).

- *Chord length at span location*: the chord length as a percentage of the root chord as a function of the span location.
- *Aerofoil section*: the aerofoil section at each span location can be defined as NACA 4- or 5-series or through an aerofoil file (Selig format).

Other aircraft specifications such as the total weight, engine location/thrust and loading cases can also be inputed to determine the loads on the wing.

II.C. Defining the Wing Structure

The definable structural components of the wingbox are the spars (both its web and cap), ribs, stringers and skin. Figure 2 shows an example layout (using the Boeing 747-400 wing) as printed from the MATLAB visualization figure window.

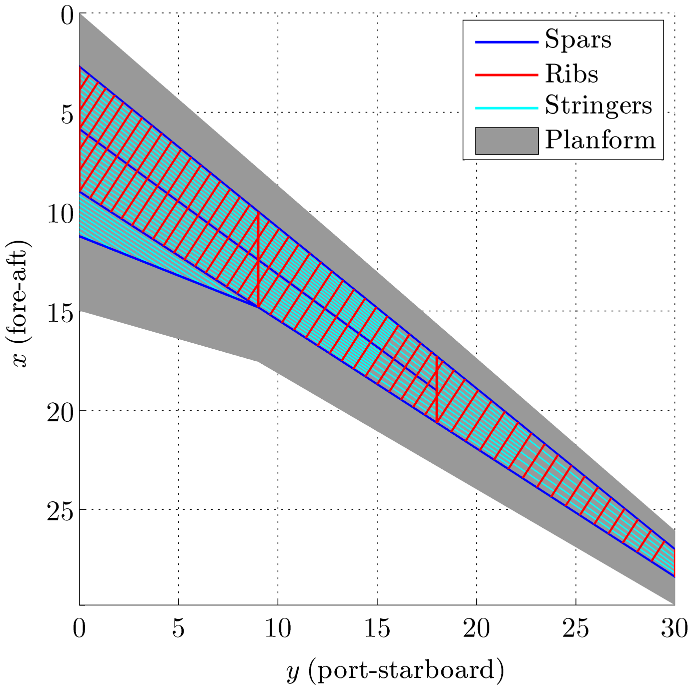

Figure 2: Components lines in an example wing structure (print from the figure window)

II.C.1. Spars

All spars within the code are defined individually (i.e. N number of entries for N spars). A spar line can be defined in three ways:

1. *Proportional*: this defines a spar which always lie in a certain proportion between the leading and trailing edge (i.e. c is constant). The start and end span values (s_1 and s_2) must be defined.
2. *Straight*: this defines a straight spar between any two points: (s_1, c_1) and (s_2, c_2)
3. *Joint*: this defines a straight spar between a point and another spar or between two spars. The span location of the intersection point(s) at the spars must be defined.

II.C.2. Ribs

Ribs are usually defined in groups as they are more numerous and generally have the same orientation. A rib line can be defined in three ways:

1. *Straight*: this defines a group of ribs which are parallel to the x-axis (centre-line of the fuselage) and originates from one spar line and inserts to another. The number of ribs (includes those at the start and end) must be defined.
2. *Perpendicular*: this defines a group of ribs which are perpendicular to an origin spar line and inserts into another spar line. The number of ribs (includes those at the start and end) must be defined.
3. *Perpendicular at XY*: this defines a singular rib which originates at an arbitrary position but is also perpendicular to a particular spar line at the same span location. This particular code allows the ribs at the root to be easily generated.

As overlapping groups of ribs have a common start and end rib, an option has been included to remove specific ribs within a group. This removes the need to calculate a starting and end span value for the two groups. Options have also been added to rotate a rib angle in the z-axis.

II.C.3. Lightening Holes

A lightening hole and associated flange can be added to a spar or rib section (defined as a section which is bounded by another spar or rib and not a stringer). For simplicity, only one hole can be added at the center of the plate. Each hole outline follows the equation shown below (an exponent of 2 describes an ellipse while a higher value creates gradually sharper corners until a rectangle at $n = \infty$), with the normal of the hole coinciding with the normal of the rib plane:

$$\left(\frac{x}{a}\right)^n + \left(\frac{y}{b}\right)^n = 1 \tag{1}$$

The values of a and b are based on the dimensions of the rib itself: that they are proportional to the maximum length and height respectively in order to ensure the hole lies inside the rib. The center of the lightening hole is located at the center of gravity of the rib. An example of the lightening holes is shown in Figure 3. In that case, the rib is divided into two sections from the spars and both holes have an a value of 80% of the rib length and b value of 60% of the height.

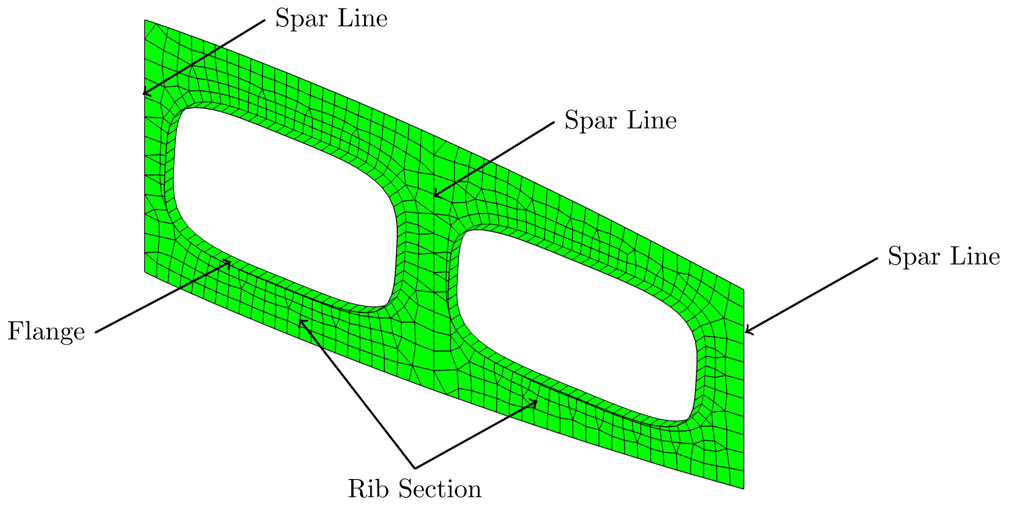

Figure 3: Example of two lightening holes in a rib

II.C.4. Stringers

Similar to the ribs, stringers are usually defined in groups. A stringer line can be defined in three ways:

1. *Between*: this defines a group of stringer lines which are equally spaced between two spar lines and terminate at a rib. Options are added to terminate individual stringers at different span locations.

2. *Parallel*: this defines a group of stringer lines which are equally spaced between two points and are parallel to one spar line and terminate at another spar.
3. *Extension*: this defines a singular stringer line which starts from the end of a spar line and runs parallel to it until a defined span value. This option allows a stringer line to easily extended from a spar which is terminated in the middle of the wingbox.

Both a spar and rib must terminate at another spar or rib (i.e. it cannot be only attached to the skin). A stringer will also automatically terminate at a rib or spar. This ensures that the boundary of every skin patch is encased properly for meshing.

II.D. Component Meshing

II.D.1. Meshing Strategy

The mesh was constructed inside the boundary of each component. A global mesh characteristic length d is set to ensure equal mesh sizes across the wingbox. Each edge of the boundary (from corner i to corner j) is divided into n segments in accordance with Equation 2:

$$n_{ij} = \text{ceil}(d_{ij}/d) \tag{2}$$

For quadrilateral patches, if the number of divisions on *both* opposite sides are equal, a structured mesh is formed as represented in Figure 4a. For triangular patches which have equal number of divisions on two sides or quadrilaterals which have equal number divisions on *one* pair of opposite sides, a semi-structured mesh is used. This involves breaking the overall shape into a series of structured meshes and connecting adjacent edges with triangular and quadrilateral elements as shown in Figure 4b. Both meshing processes are performed solely through MATLAB for the fastest speed.

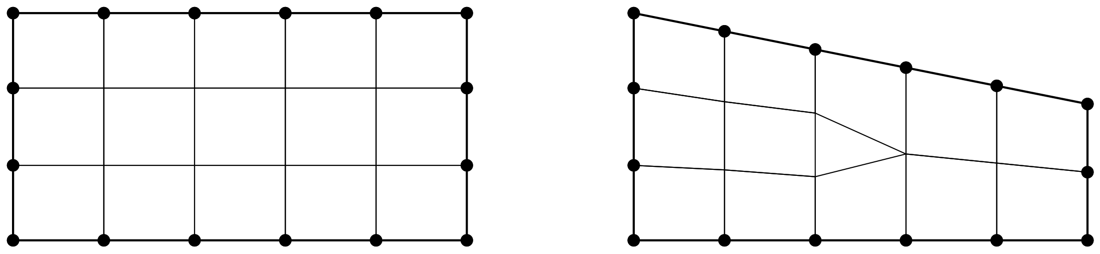
(a) Structured (b) Semi-structured

Figure 4: Mesh types generated through MATLAB

For other patch types, a more powerful unstructured meshing software is required, with the software Gmsh used. Gmsh is a 2D and 3D finite element mesh generator with geometry, meshing and solver modules.[8] Within this code, only the 2D meshing module was used to convert an arbitrary 2D boundary into an unstructured mesh. While Gmsh can also produce structured meshes, linking it through MATLAB (creating the input file, running it through the command line and reading the output) is a much slower process than the dedicated structured and semi-structured mesh generators written specifically in MATLAB. Therefore, it is more advantageous to run Gmsh for unstructured meshes only.

II.D.2. Skin Mesh

The skin mesh is constructed first as all the internal structural components in the wingbox impose a line constraint on the surrounding skin. Every patch, bounded with the constraints, is found and meshed independently of the other patches. Although the capability to directly mesh with line constraints has been shown,[9] meshing each patch individually allows the most efficient meshing strategy to be applied and is the computationally most efficient method. This is process is similar to that described by Jang *et al.*[10] but also allows patches with more than four sides to be meshed without breaking it into multiple triangles/quadrilaterals to avoid patches with potentially high aspect ratios or small internal angles. Furthermore, by considering each patch individually, a separate shell property identifier (`PSHELL` card in Nastran) can assigned. Physically, this means each skin panel can be given a separate thickness in the optimization stage.

To start, the intersection points of all the lines constraints were found through the MATLAB function **`polyxpoly`**. From the intersection points, the boundary of each skin patch can be determined by finding the smallest bounding polygon. This was done by selecting a line and finding the branch at its end point which gives the largest counter-clockwise angle. This is iterated until the branch returns to its initial starting point. By applying the process to all possible lines, all skin patches can be generated. An illustration of the process is shown in Figure 5, where the blue arrows indicate the traversed path and the red arrows indicate other branch possibilities at the end point of each line. Elements which are degenerate (such as the upper-right 5-sided patch in the figure) are not reduced to simpler polygons to ensure that the edges of different patches can be meshed correctly.

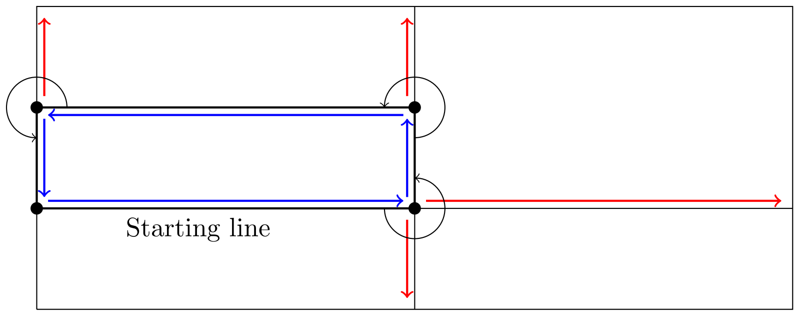

Figure 5: Bounding box example (with five different patches)

Figure 6 shows the different skin patches on the wingbox. For the majority of the main wingbox, 4-sided patches are formed (bounded as expected with ribs and spars/stringers) and allow the use of structured meshing. Quadrilaterals at the root trailing edge extension (due to their large aspect ratios) and other patches with higher number of sides however must be meshed using unstructured methods.

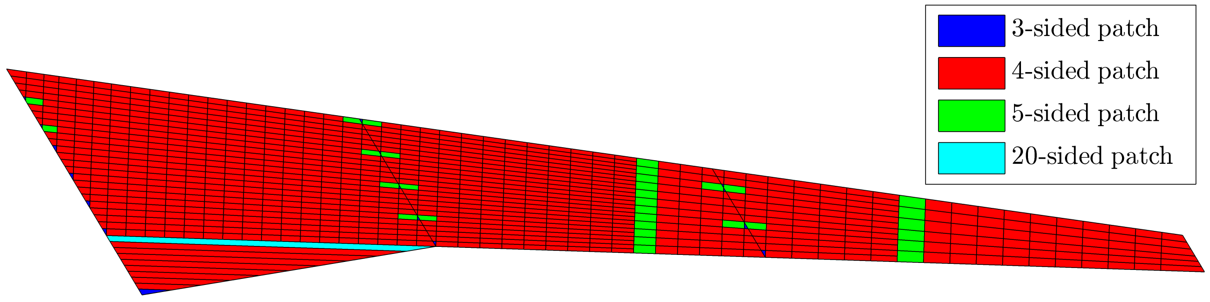

Figure 6: Patches of a wingbox

II.D.3. Spar Web and Ribs

Each spar web or rib without a lightening hole is bound by the upper and lower skins, making them much simpler to mesh. As an equal number of nodes lie on the two edges, a semi-structured meshing approach is taken. However when a lightening hole is present, Gmsh is used to construct an unstructured mesh.

II.D.4. Spar Cap Mesh

The spar cap is meshed structurally. The width of the spar cap is defined at the start and end of the spar line. All widths are calculated perpendicular to the line. At the root and tip, the caps are parallel to the x-axis. Options are added to allow two adjacent spar caps to be connected to each other and ensure the load path is not severed, as shown in Figure 7.

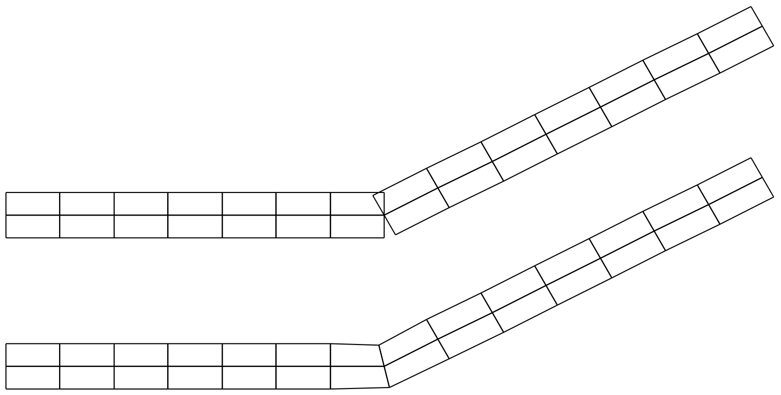
Figure 7: Spar cap connections

II.D.5. Stringers

Stringers can be defined implicitly through `CBAR` elements or explicitly through shell elements. In the case of the bar elements, the cross-sectional area and second moment of area are only two variables required. The offset vectors at the two nodes are not simulated. For the explicitly-defined shell elements, simple quadrilateral shells are used in place of more complex stringer geometry. The height and thickness of the stringer is calculated by solving for the area and second moment of area.

II.E. Writing MATLAB Data into a Nastran File

With the completion of the individual meshes, shared nodes along the boundary of each component are removed. This process is similar to the *equivalence* function in MSC Patran and ensures connectivity between the elements. To do so, every pair of nodes are compared. Although the procedure can be completely vectorized in MATLAB, the large number of combinations (e.g. 450 million exist for 30,000 nodes) means that memory limits could be easily reached. Instead, a C++/MEX file was written to make use of the faster lower-level code. Fourfold speed increases were achieved with the MEX function compared to a similar loop-intensive MATLAB code.

The nodes (GRID cards) and elements (`CQUAD4`, `CTRIA3` and/or `CBAR` cards) can be written into the Nastran input file using the standard 8-field format. Specially-written I/O functions `readNastranInput.m` and `readNastranF06.m` were constructed to allow MATLAB to read the input and output (text-based .F06) files. To achieve reasonable speeds, heavy use of the `textscan` function were used, relying on the systematic format of the files to read blocks of text quickly. The output-read function itself is capable of reading 500,000 lines of text in approximately 12 seconds.

III. Finite Element Models

Finite element meshes of existing aircraft wings were generated to illustrate the viability of the program. The first two examples are large transport airliners, the third being a fighter aircraft and the last being a unconventional design. These examples were chosen as they represent a spectrum of design geometries and layouts (large number of ribs with few spars and vice versa).

III.A. Boeing 747-400

The mesh for the B747 wingbox is shown in Figure 8 (the image itself is a print of the visualization shown in the MATLAB figure window). The components geometry were estimated from cutaway drawings and are representative only. As the exact aerofoil sections of the wing are not known, the aerofoil coordinates for the B737 (taken from the UIUC Airfoil Coordinates Database) were used to provide a reasonable structural shape. All stringers have defined explicitly as shell elements with an arbitrary height. The components have been color-coded, though each patch has separate shell property identifiers in the file. The contour of wing is shown while the upper skin/stringers are removed to show the internal components.

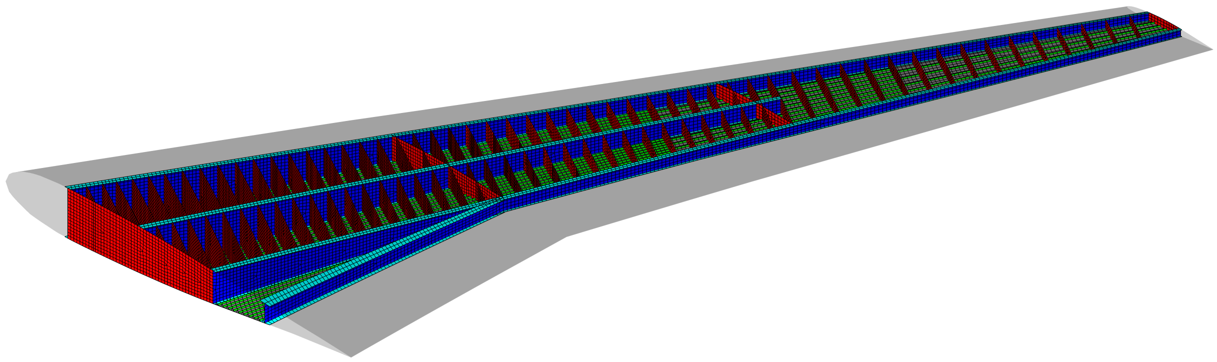
Figure 8: B747 wingbox mesh

III.B. Boeing B787

The second example mesh is of the B787 wingbox, shown in Figure 9. Compared to the B747, a more complex planform is seen with a raked wingtip. Multiple kinks are also modeled in the spars at the midspan as well as the wingtip in accordance with cutaway diagrams.

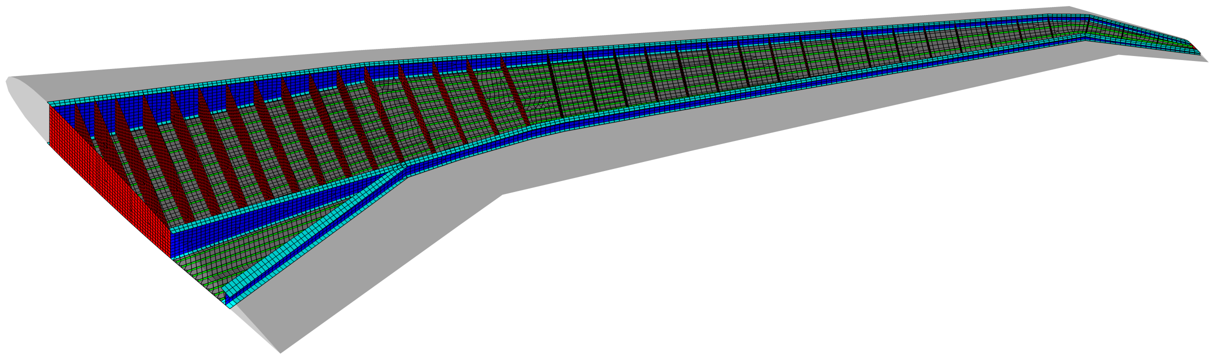
Figure 9: B787 wingbox mesh

III.C. Boeing F-15 Eagle

The third example mesh is for the F-15, shown in Figure 10. Unlike the two airliners, the F-15 has more spars and a fixed leading edge, stiffened by multiple rib sections. The wingtip fairing was not generated due to the inability of the code to generate the internal honeycomb structure.

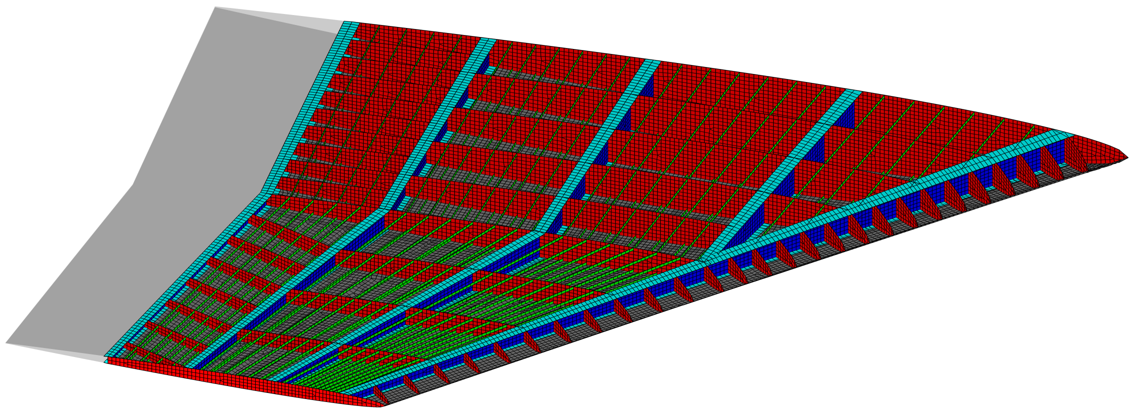
Figure 10: F-15 wingbox mesh

III.D. Blended Wing Body

The last example is for the primary load-carrying structure of a blended body wing (BWB). A BWB uses a smoothly-contouring wing for aerodynamic efficiency with the fuselage built in the root of the wing. Due to the similarities with a normal wing, a simple structure was built using the code, with the design shape based on Liebecks'.[11] The leading and trailing edges were read through an image and processed into vectors for use in the generation. The outer wing was modeled in a similar manner to a conventional wing. The fuselage is partitioned in five sections, which were modelled simply with ribs with stringer-stiffener upper and lower skins.

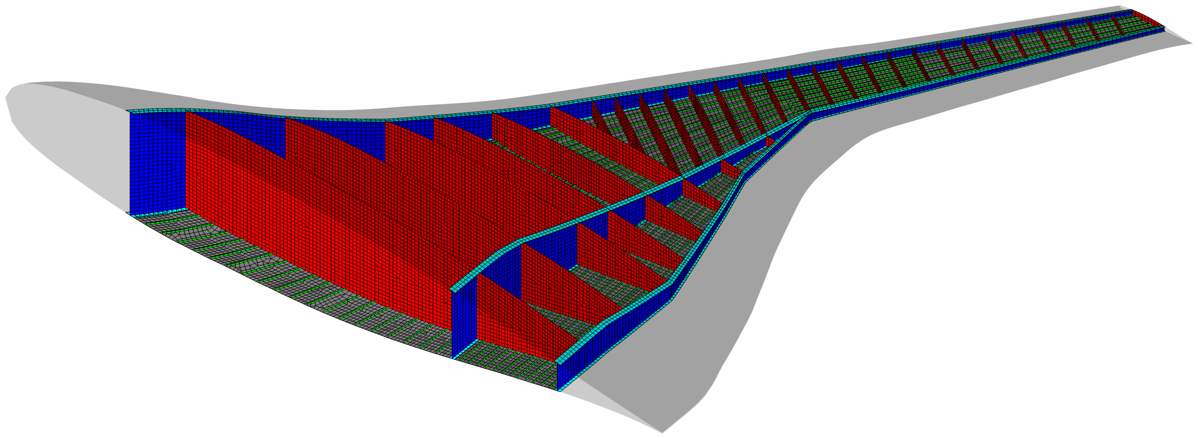

Figure 11: BWB wingbox mesh

III.E. Generation Metrics

The model summary and quality metrics for the FE models are listed in Table 1. The first parameter listed is the time for MATLAB to generate the model, run the AVL of the geometry and plot all intermediate figure windows (using an i7 processor with a CPU speed of 2.40 GHz). All times are under or just at one minute, satisfying the computationally quick criterion. The discrepancy between the models with a similar number of nodes/elements is due to the number of times Gmsh needed to be run to generate an unstructured mesh. In the B747, the placement of the engine pylon ribs meant that highly-skewed and patches with more than four sides were formed and this necessitated more Gmsh runs (most of the time taken to link the executable to MATLAB and delete post-production files). In contrast, the B787 and BWB have more nice quadrilateral patches which can be meshed structurally, making them faster to generate.

Table 1: Model summary and quality metrics

Parameter	**B747**	**B787**	**F-15**	**BWB**
Time to Generate (sec)	60.2	44.3	24.8	35.1
No. GRID nodes	46,558	42,509	30,818	40,807
No. CQUAD4 elements	48,752	43,873	32,428	41,466
No. CTRIA3 elements	587	611	669	722
Element quality (μ, σ)	0.894, 0.0750	0.941, 0.0540	0.890, 0.115	0.852, 0.124
Corner angle (μ, min, max) (°)	93.8, 61.7, 147	94.4, 60.3, 148	97.9, 60.4, 160	99.1, 60.1, 160
Aspect ratio (μ, σ)	1.45, 0.335	1.06, 0.0540	1.23, 0.286	1.40, 0.436
Skewness (μ)	0.0461	0.0543	0.0952	0.107

In terms in of element quality, all models are largely made of favourable quadrilateral elements, though a high number of triangular elements exist near the root (not favourable as it is generally a higher stress region). The element quality for the meshes is high with good aspect ratios and low skewness. Some of the poorer-quality elements occur at regions where there are a large number of line constraints: near the engine pylons on the B747 and between the first two spars for the F-15. A reason for the overall lower element quality for the BWB is a that a large number of skewed parallelogram-shaped elements are generated due to the large sweep angles and straight ribs in the fuselage (in the other two transport aircraft, the ribs are roughly perpendicular to the spar and hence the interior elements are less skewed). Elements in the outer wing are much more rectangular and similar to quality to the standalone wings.

IV. Structural Optimization

In the final step, the structural optimization is conducted through the Nastran SOL 200 module (Design Sensitivity and Optimization) to complete a preliminary wingbox structure. The resulting model, while not at the level of a detailed design, aims to provide platform in which structural weights and stresses can be determined with a higher degree of precision.

For the optimization, the `STATICS` analysis module was used in place of the aeroelastic (`SAERO`) solver due to the much faster computational time. To apply the aerodynamic loading, Athena Vortex Lattice (AVL) was used to generate the aerodynamic pressure. The pressure across the skins are applied directly as `FORCE` cards to the appropriate nodes. As the leading and trailing edge surfaces are not modelled for this case, the total lift of surfaces are connected through `RBE3` elements to the spars. The FE model of the B747 for optimization is shown in Figure 12 (upper skin and stringers not shown). Each patch color indicates a different shell PID for thickness optimization (each spar web and cap broken into multiple patches). The magenta circles connected to the spar webs are centroid of the total lift from the spar to the leading or trailing edge, with `RBE3` elements represented by the gray lines. Similarly, the nodes below the wingbox which are connected to the spar caps are for the engine thrust and weight.

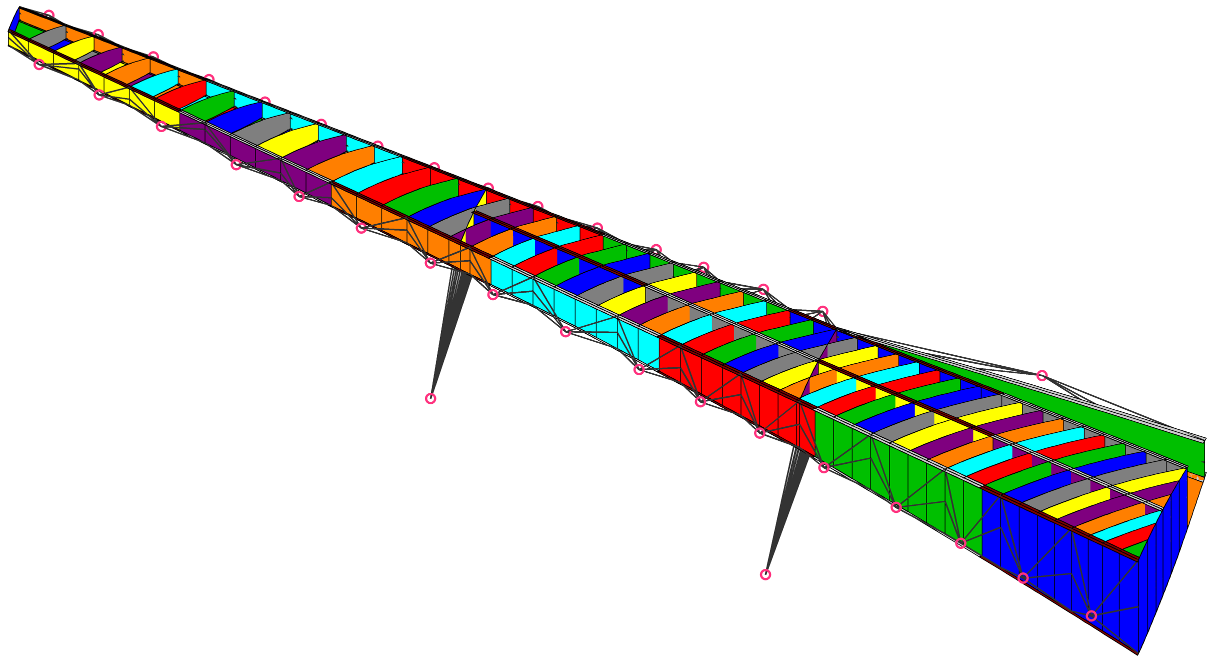

Figure 12: B747 Design Optimization FE Model

IV.A. Design Case

The design case used for the optimization is the limit load case ($n_z = 2.5$). The limit load case as opposed to ultimate load will ensure that the linear static optimization is valid. The load case is applied at the maximum zero fuel weight (MZFW) of 256,200 kg, which is assumed to give the maximum bending loads in the wing.

Figure 13 shows the pressure coefficient of the planform for the loading (winglets ignored in the analysis). Although the C_p distribution is not correct in a transonic airliner (both in the span and chord directions), the total lift is correct and the simplistic loading means can be easily applied to the skin and rigid element nodes for demonstration purposes. Higher fidelity aerodynamic methods such as transonic panel codes or CFD were not used due to their long computational times but can be incorporated and integrated into the code for higher accuracy. Bending relief provided by the mass of the wingbox and engines are modeled with `GRAV` and `FORCE` cards respectively.

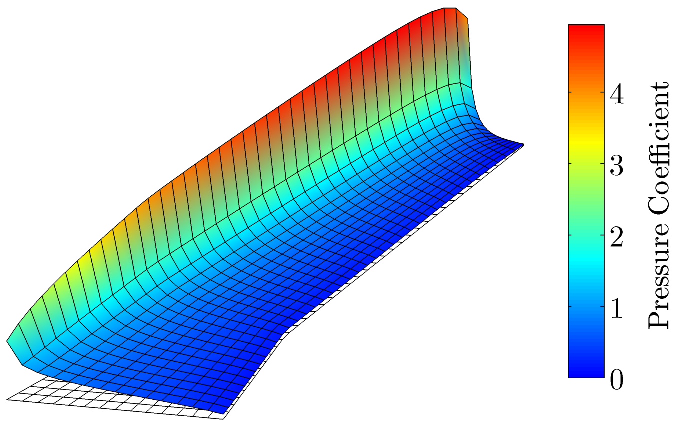

Figure 13: B747 AVL pressure coefficient distribution

The design variable and constraints are listed in Table 2. A fully-stressed design is used to size the components (no stiffness or buckling constraints applied yet). A maximum allowable von Mises stress of 275 MPa is placed on all components. This was chosen to ensure a stress of 110 MPa for nominal 1-g cruise (fatigue limit as advised by Torenbeek[12]). However, as Nastran groups both the Tresca (max shear failure) and von Mises criterion together in its Item Code, the von Mises stress limit was reduced to 250 MPa as it is less conservative.

Table 2: Design variables and constraints

Component	**Number of Design Variables**	
Spar webs	18	
Spar caps	18	
Ribs	57	
Stringers	32	
Skin	196	
Total	321	
Constraints	**Min**	**Max**
Thickness (mm)	0.5	100
σ_{vm} (MPa)	-250	250

IV.B. Results

The thickness of the component thickness are shown in Figure 14 and the resulting stress distribution is shown in Figure 15. The maximum stress of 270 MPa is seen over a significant portion of the skin indicating that the fully-stressed criteria is being met. The weight of a one-side wingbox as read from the Nastran Grid Point Weight Generator is 12,930 kg (total full-span wingbox weight of 57,000 lb).

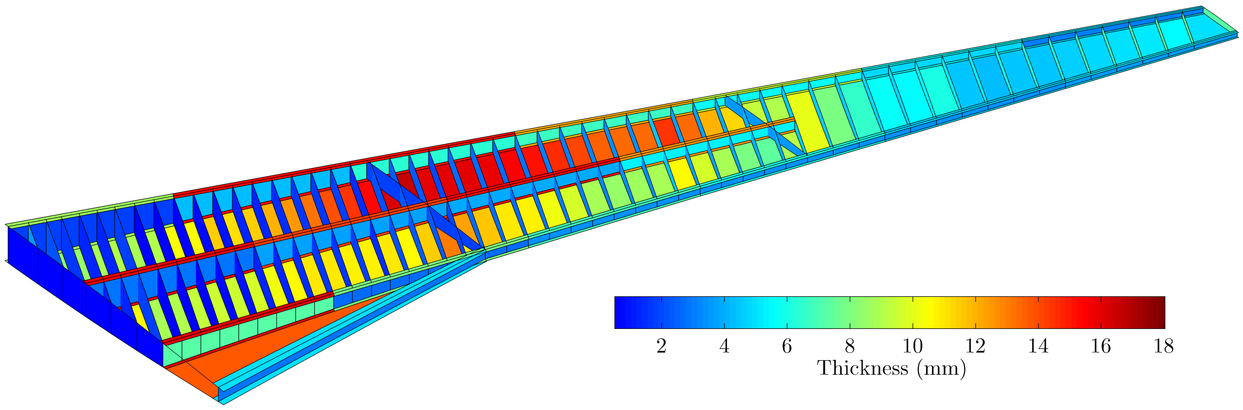

Figure 14: Thickness distribution

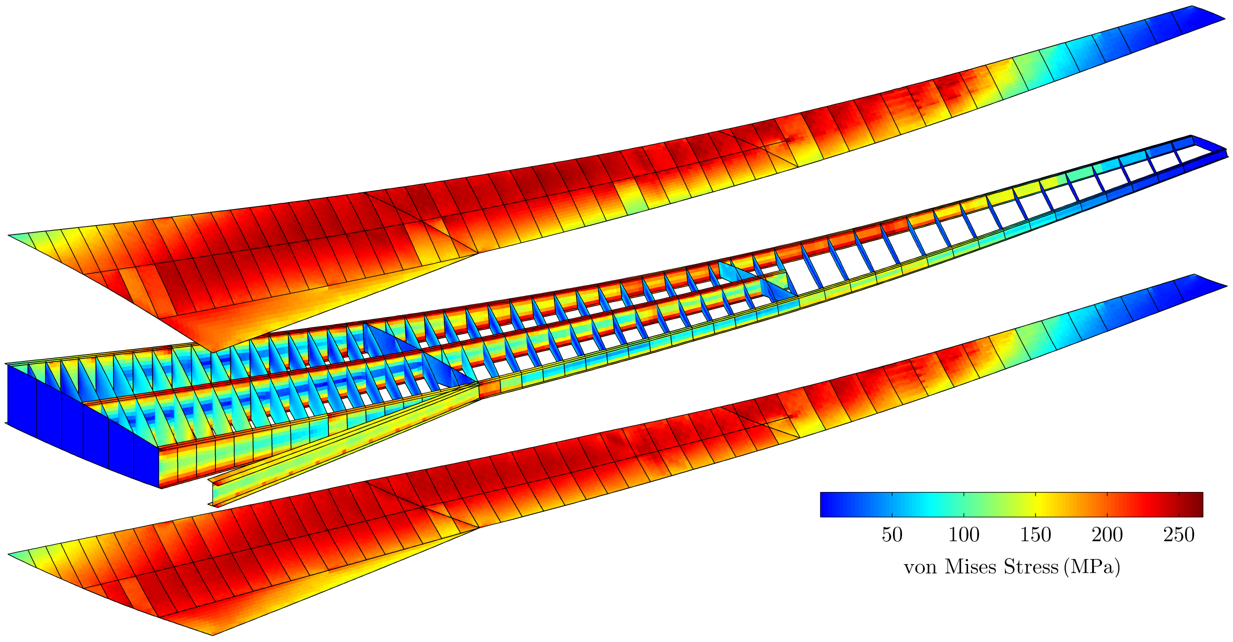

Figure 15: Von Mises stress distribution

IV.C. Statistical Validation

Validation of the preliminary design was conducted by comparing the structural weight to empirically-derived weights. These weights are based on statistical analysis and are given by Raymer[13] and Roskam[14] in Equations 3 and 4 respectively:

$$W_w = 0.0051\,(W_{TO}N_z)^{0.557}\,S_w^{0.649}\,AR^{0.5}\,(t/c)_{root}^{-0.4}\,(1+\lambda)^{0.1}\,\cos(\Lambda)^{-1}\,S_{cw}^{0.1} \tag{3}$$

$$W_w = \frac{0.00428\,S_w^{0.48}\,AR\,M_H^{0.43}\,(W_{TO}\,N_z)^{0.84}\,\lambda^{0.14}}{[100\,(t/c)_m]^{0.76}\,\cos(\Lambda)^{1.54}} \tag{4}$$

Table 3 shows the empirical weights for the entire wing. The average wing to MTOW percentage correlates very well to actual values seen in large jet transport aircraft (9 to 11%).[14] However, the optimization result is 40% lower than the empirical weight. This discrepancy is due to several factors:

- The empirical formula takes into account of all structural and non-structural weights in the wing (including control surfaces, landing gear/engine attachments etc.). These secondary components are significant and can take up to 35% of the total weight of the wing.[12]

- The carry-through box through the fuselage is not modeled. As that structure takes the largest loads, it would account for a significant portion of the total weight.

Table 3: Empirical weights

Method	Weight (lb)	Percent of MTOW
Raymar	108,000	11.9
Roskam	82,700	9.09
Average	95,400	10.5

V. Conclusion

This paper demonstrated the capability of a MATLAB-based procedure to autonomously generate a finite element model for a structural wingbox. The code allows the planform geometry and various components (spars, ribs, stringers and miscellaneous lightening holes) to be defined parametrically in order to create a high-fidelity model and give the user a great amount of freedom to update the model. Through optimized meshing strategies, the resulting models show high element quality across four different example wing layouts (two conventional, one fighter and an unconventional BWB design) with quick generation times in the order of a minute. Easy coupling between MATLAB and external software meant that load generation (vortex lattice through AVL) and structural analysis and optimization through Nastran can be attempted. In the example case, the optimized wing showed good correlation to empirical formulas. Thus, the procedure could be highly useful in the preliminary aircraft design stage, providing a high-fidelity model and analysis earlier in the process with a high level of parametric-based modeling to allow quick generation and changes.

References

[1]Crose, J. G., Marx, D. A., Kranz, M., Olson, P., and Ball, C., "Parametric Design/Analysis with MSC/PATRAN - A New Capability," .

[2]Ghionea, I. G., Devedžić, G., and Ćuković, S., "Parametric Modeling of Surfaces using CATIA v5 Environment," *Applied Mechanics and Materials*, Vol. 760, 2015, pp. 93–98.

[3]Sensmeier, M. D., Stewart, B. T., and Samareh, J., "Rapid generation and assessment of aircraft structural topologies for multidisciplinary optimization and weight estimation," *Collection of Technical Papers–AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Vol. 7, 2006, pp. 4722–4733.

[4]Jiapeng, T., Ping, X., Baoyuan, Z., and Bifu, H., "A finite element parametric modeling technique of aircraft wing structures," *Chinese Journal of Aeronautics*, Vol. 26, No. 5, 2013, pp. 1202–1210.

[5]Hwang, J. T., Kenway, G. K., and Martins, J., "Geometry and structural modeling for high-fidelity aircraft conceptual design optimization," *Proceedings of the 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Atlanta, GA*, 2014, pp. 2014–2041.

[6]Skillen, M. and Crossley, W., "A Matlab-Based Object-Oriented Process Architecture for Rapid Generation of Unconventional Wing Finite Element Models," *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 16th AIAA/ASME/AHS Adaptive Structures Conference, 10th AIAA Non-Deterministic Approaches Conference, 9th AIAA Gossamer Spacecraft Forum, 4th AIAA Multidisciplinary Design Optimization Specialists Conference*, 2008, p. 2160.

[7]Sensmeier, M. D. and Samareh, J. A., "Automatic aircraft structural topology generation for multidisciplinary optimization and weight estimation," *Proceedings of 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics\ & Materials Conference*, Vol. 7, 2005.

[8]Geuzaine, C. and Remacle, J.-F., "Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, Vol. 79, No. 11, 2009, pp. 1309–1331.

[9]Park, C., Noh, J.-S., Jang, I.-S., and Kang, J. M., "A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints," *Computer-Aided Design*, Vol. 39, No. 4, 2007, pp. 258–267.

[10]Jang, B.-S., Suh, Y.-S., Kim, E.-K., and Lee, T.-H., "Automatic FE modeler using stiffener-based mesh generation algorithm for ship structural analysis," *Marine Structures*, Vol. 21, No. 2, 2008, pp. 294–325.

[11]Liebeck, R. H., "Design of the blended wing body subsonic transport," *Journal of aircraft*, Vol. 41, No. 1, 2004, pp. 10–25.

[12]Torenbeek, E., *Advanced aircraft design: Conceptual design, technology and optimization of subsonic civil airplanes*, John Wiley & Sons, 2013.

[13]Raymer, D. P., *Aircraft Design: A Conceptual Approach*, AIAA (American Institute of Aeronautics & Astronautics), 2006.

[14]Roskam, J., *Airplane Design: Part 5 - Component Weight Estimation*, DARcorporation, 1985.