23rd AIAA Computational Fluid Dynamics Conference, Denver, Colorado, June, 2017.

Strategies Toward Automation of Overset Structured Surface Grid Generation

William M. Chan,* *NASA Ames Research Center, Moffett Field, CA 94035*

An outline of a strategy for automation of overset structured surface grid generation on complex geometries is described. The starting point of the process consists of an unstructured surface triangulation representation of the geometry derived from a native CAD, STEP, or IGES definition, and a set of discretized surface curves that captures all geometric features of interest. The procedure for surface grid generation is decomposed into an algebraic meshing step, a hyperbolic meshing step, and a gap-filling step. This paper will focus primarily on the high-level plan with details on the algebraic step. The algorithmic procedure for the algebraic step involves analyzing the topology of the network of surface curves, distributing grid points appropriately on these curves, identifying domains bounded by four curves that can be meshed algebraically, concatenating the resulting grids into fewer patches, and extending appropriate boundaries of the concatenated grids to provide proper overlap. Results are presented for grids created on various aerospace vehicle components.

I. Introduction

In recent years, structured overset grid technology has been successfully applied to computational fluid dynamics analysis on a wide variety of complex aerospace applications.[1–11] While structured overset grid flow solvers such as OVERFLOW[12] and LAVA[13] are highly efficient relative to typical flow solvers utilizing unstructured grids, structured overset grid generation remains a labor intensive and time consuming step relative to unstructured methods. Grid generation for overset structured grids usually consists of three steps: surface grid generation, volume grid generation, and domain connectivity.

The surface grid generation step consists of three main tasks. The first involves decomposition of the surface geometry into four-sided overlapping domains while capturing the surface features. The second task is to determine the grid point distribution on the domain bounding curves such that both geometry and flow features are appropriately resolved, and the third task is to select a combination of algebraic and hyperbolic methods to fill the domain interiors with grid points. For algebraic meshes, grid generation input involves specification of the four appropriately redistributed bounding curves. For the hyperbolic meshes, only one initial boundary curve needs to be prescribed, but further specifications are needed for an appropriate marching distance (constant or spatially variable) from the initial curve, the grid point distribution in the marching direction, and boundary conditions at each end of the initial curve. For complex configurations, the surface grid generation step requires significant user expertise, is highly labor intensive, and typically consumes at least 80% of the total grid generation time.[14]

Since the beginning of computational analysis of complex configurations using overset grids, two main advances have contributed to increasing the level of geometric complexity that can be handled. The first is the introduction of a graphical user interface[15,16] that is connected to standalone overset grid generation software modules. This allows the user to visualize the results while interactively specifying and iterating on the inputs for the grid generation process. The grid generation modules provide the ability to generate both hyperbolic and algebraic grids from one, two, three or four domain bounding curves, and are thus highly suited for the overset mesh approach. The second advance was the introduction of a best practice philosophy to write preprocessing scripts as the grids are generated. Such scripts, typically written in a scripting language such as Tcl, contain all instructions for generating the surface and volume meshes for each geometric component, as well as inputs for domain connectivity, aerodynamic loads integration, and flow solver boundary conditions.

*Computer Scientist, Computational Aerosciences Branch, M/S 258-2, AIAA Senior Member

These scripts enable rapid replay of the grid generation process and parameterization of geometry and grid inputs such as component dimensions and positions, as well as maximum grid stretching ratio and spacings. Scripts also permit rapid addition or removal of various components in a configuration (e.g., inclusion or exclusion of brackets and flap track fairings under a wing). Development of a script library[17] for many of these operations highly enhanced the script creation process. By breaking up a complex surface domain into components, the grids can be efficiently generated by a team of engineers developing independent component scripts in parallel.

While the scripting approach has enabled grid generation over highly complex configurations,[7,18] the limitations of the approach are beginning to impact the efficiency of the computational analysis process more and more frequently. First, construction of the pre-processing script for the first time on a new geometry still requires significant manual effort. The script is reusable on another instance of the geometry provided the new geometry is topologically identical to the first, including the ordering and direction of the initial feature curves. Small changes to the geometric topology of any component typically require modification to the script. In a design environment, changes of various magnitudes occur frequently, sometimes as often as on a weekly basis. Such updates then lead to significant effort in re-engineering the scripts. A more automated approach that can construct surface grids over new geometries would be highly beneficial to the analysis process.

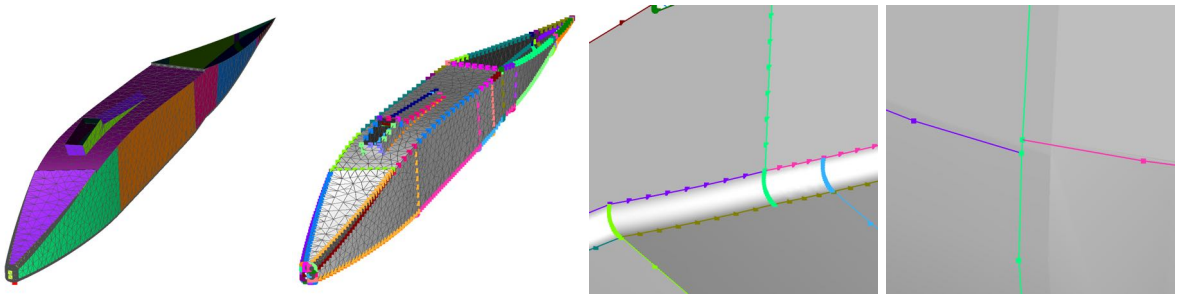
Surprisingly, very little attempt has been made to automate the overset structured grid generation process. A scheme for grid point distribution automation was presented in Ref. 19 but the method was mainly targeted for rocket geometries. Since overset surface meshes are allowed to overlap arbitrarily, the constraints on the grid boundaries are less severe as those required for abutting structured meshes. An early algorithm[20] to construct surface grids around surface features (such as sharp edges and maximum curvature lines) and junctions of such features utilized hyperbolic and polar meshes, respectively. There are two deficiencies to this method. The marching distances for the hyperbolic meshes have to be manually specified. Also, the scheme produces a polar mesh for every junction point (a point where multiple surface features meet), resulting in singularities and excessively small cell sizes in multiple regions of the flow. This in turn limits the stability of the flow solver in some situations. More recently, an algorithm was developed for cases where the Computer Aided Design (CAD) model feature trees are available.[21] Meshes are then constructed from basic shape grids using union, intersection, and difference Boolean operations. Unfortunately, such feature trees are not always available in many practical applications. For both methods described above, no attempt was made on automatic grid point distribution for proper resolution of geometric and flow features.

This paper presents a plan to significantly reduce the manual effort and time required to generate structured overset surface grids, with a long term goal of automating the entire process. Since most CAD and grid generation software packages are able to write an unstructured surface triangulation representation of the geometry relatively easily from native CAD models, or from vendor-neutral standard geometry exchange formats such as STEP and IGES files, the starting point for the current work is chosen to be an unstructured surface triangulation that describes the geometry accurately. In addition, a set of discretized surface curves that captures at least the surface features of interest is needed. The plan consists of three main steps: algebraic, hyperbolic and gap-filling. The first step utilizes algebraic grids to fill four-sided domains formed by the surface curves. Using unused surface curves and boundary curves from the first step, the second step employs hyperbolic grids to march away from these curves. The third step involves filling the remaining gaps on the surface geometry to provide proper overlap over all surface grids. This paper will focus on presenting details of the algebraic step, while details of the hyperbolic and gap-filling steps will be presented in future papers. Section II below describes various parts of the algorithm for the algebraic step. Results on several test cases are presented in Section III. Future plans on development for the hyperbolic and gap-filling steps are given in Section IV. A summary and conclusions are presented in Section V.

II. Algebraic Step

II.A. Starting Point and Control Parameters

The starting point for any surface grid generation process requires a geometry description. In addition, a representation of the surface features that should be retained in the final grid system is needed. It is highly preferable that little effort is required to obtain the starting point files needed to begin the automation process. For the geometry representation, an unstructured surface triangulation fits this requirement well since most geometry processing software such as ANSA[22] and Engineering Sketch Pad[23] can output such



(a) (b) (c) (d)

Figure 1. Starting point for automation process. (a) Unstructured reference triangulation where triangles are tagged by region number. (b) Surface curves representing CAD edge curves that include all surface features of interest. (c) A set of surface curves that appear to meet at a point from a zoomed-out view. (d) A zoomed-in view at a junction where end points of CAD edge curves do not exactly match due to differences in geometry translation tolerances.

files irrespective of whether the original geometry definition is in the form of native CAD, STEP, or IGES format. Since the ultimate surface grids are to be created based on this triangulation, care should be taken to provide sufficient resolution of high curvature regions in the triangulation. A good guideline for this is that the mesh resolution on curved regions of the triangulation needs to be finer than that of the surface mesh to be created. Note that this reference triangulation only needs to provide sufficient resolution to represent the geometry accurately but is not a grid used for resolving flow features. Hence, large cells are allowed in flat regions of the geometry. The triangulation quality is typically adjusted by control parameters for maximum dihedral angle between adjacent triangles, maximum cell size, and maximum cell face deviation from the geometry definition. At the end of the surface mesh generation process, this triangulation can be utilized as a guide to project the grid points onto the original geometry definition in native CAD, STEP or IGES entities. For the purpose of demonstrating the automation strategy in this paper, this last step is omitted.

The surface features to be retained in the final grid system may include geometric sharp (hard) edges, maximum curvature lines such as wing leading edges, and open boundaries of the surface domain. Such surface features are usually modeled by some of the CAD edge curves. The set of all CAD edge curves tesselates the surface geometry into regions bounded by three or more such curves. By tagging all triangles within each region with a unique integer tag, all triangle edges that separate two triangles with different tags can be easily identified (Fig. 1a). The set of all such edges is then processed to recover a discrete version of the CAD edge curves where the vertices are shared with the vertices of the reference surface triangulation (Fig. 1b). For the current work, it is this complete set of discrete surface curves, together with the reference surface triangulation, that are chosen as the starting point for surface mesh generation automation. Care must be taken in analyzing the set of discrete surface curves obtained in this manner. In the translation of a solid model from native CAD to a STEP format for example, mis-matches in translation tolerances between neighboring faces of the solid could result in small mis-matches in the end points of CAD edge curves as shown in Fig. 1c,d. Such mis-matches are currently manually fixed by collapsing the end points of the mismatched curves into a single point.

When the original geometry definition is in the form of Boundary Representations (BRep) with both geometric and topological entities,[24] creation of the triangulation and surface curves described above is a simple process. The tagged regions discussed in the previous paragraph are the BRep Faces, while the surface curves are the BRep Edges. In the absence of a BRep model, some work is needed to construct such a representation by specifying connectivity between a set of surfaces that describes the geometry.

II.B. Determination of Curve Network Connectivity

The initial surface curve set contains curves that are point matched at the end points while some curves may also have their end points matching interior vertices from other curves. The first operation here is to sweep through all curves with an interior vertex matching the end point of another curve and splitting the original curve at those interior vertices. The end result is a new set of curves that only meet at their end points. From here, a junction point is defined to be a vertex where three or more curve end points meet. The degree of a junction is the number of curves that meet at the junction (see Fig. 2). Connectivity between the new curves is established by storing the list of curve end points that meet at each junction. Given this information, it is

straight-forward to identify four-sided domains that are bounded by four curves using junction information at the four corners of the domain (see Section II.E). Such domains will be referred to as four-curve domains in the remainder of this paper.

II.C. Determination of Curve Attributes

For the purpose of grid point distribution and adjacent domain concatenation described in later sections, it is convenient to determine various attributes of each curve. There are three attributes that are automatically detected (Fig. 3): short or long curve, turning or non-turning curve, resident geometric feature (i.e., residing on sharp edge, open boundary, or neither). Several control parameters are introduced to define these attributes while the user is free to specify different values from the defaults depending on the application. Let Np_{min} be the minimum number of grid points to be placed on a curve, and let Δs_{max} be the maximum grid spacing allowed. Then a curve is classified as 'short' if the arc length of the curve $A_c < (Np_{min} - 1)\Delta s_{max}$. Also, a curve is labelled as a turning curve if the total turning angle over the entire curve exceeds $\Delta\theta_{tot}$, typically chosen to be 30 degrees. Edges on the surface triangulation are marked as 'sharp' or 'not sharp' based on the angle between the normals of the two triangles that are adjacent to each side of the edge. If the angle is larger than θ_{sharp}, typically chosen to be 20 degrees, then the edge is classified as sharp. A curve is labelled as a sharp edge curve if most of its vertices fall on a sharp edge of the triangulation.

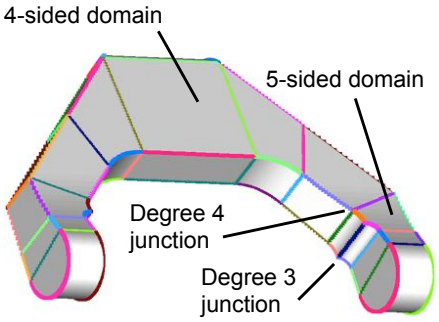


Figure 2. Network of surface curves showing junctions and domains bounded by surface curves.

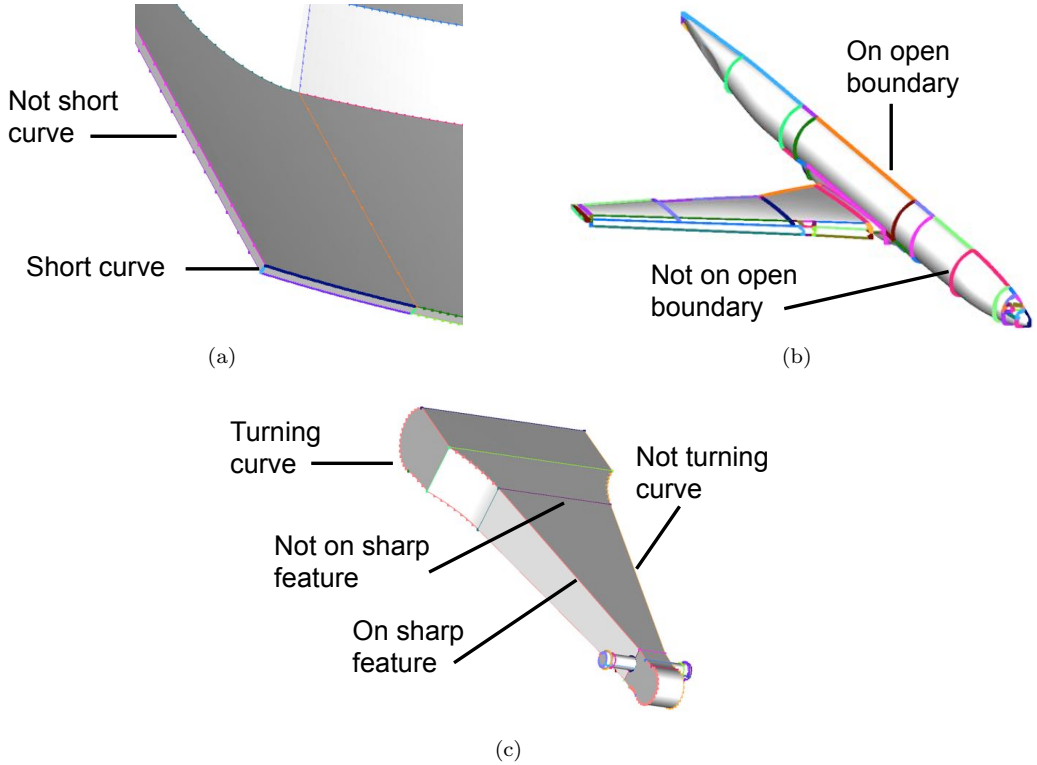


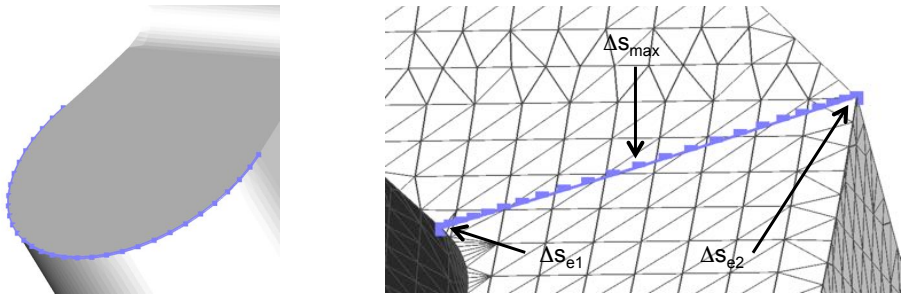
Figure 3. Examples of curve attributes. (a) Short curve. (b) Open boundary. (c) Sharp edge and turning curve.

II.D. Determination of Grid Point Distribution

A uniform grid point distribution is applied to turning curves and short curves. Let θ_{ml} be the maximum local turning angle allowed in a redistributed curve. Then the uniform spacing Δs assigned to a turning curve is determined by the minimum number of grid points that ensures the local turning angle at any point is no larger than θ_{ml} (Fig. 4a). For short curves, the number of grid points is given by Np_{min}. If the short curve is also a turning curve, then the number of grid points is determined by the maximum predicted by Np_{min} and θ_{ml}.

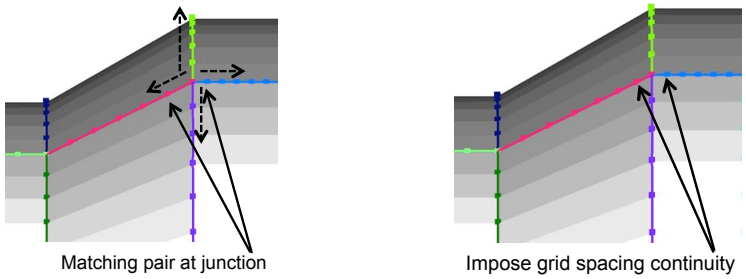
For a non-turning, non-short curve, a non-uniform grid point distribution is applied using the hyperbolic tangent stretching function,[25] which allows the specification of end point spacings Δse_1 and Δse_2, and the number of grid points. The number of grid points is increased by one starting at Np_{min} until the additional constraints on maximum stretching ratio SR_{max}, and maximum interior spacing Δs_{max} are satisfied (Fig. 4b).

The end point spacing Δse_i (where $i = 1, 2$) is determined by the location of the curve end point. If the end point does not lie on any surface feature or open boundary, the end spacing is set to Δs_{max}. If the end point lies on an open boundary of the geometry, a slight tightening of grid spacing is applied by setting the end point spacing to $\Delta s_{max}/2$. For an end point falling on a sharp surface feature, the end point spacing is dependent on whether the sharp feature is concave or convex. For a concave corner, the prescribed grid spacing Δse_i increases when the acute concave corner angle gets smaller, up to a specifiable upper limit which is set to $2 \times \Delta s_{max}$ by default. For a convex corner, the prescribed grid spacing Δse_i decreases when the convex corner angle gets smaller, down to a specifiable lower limit which is set to $0.01 \times \Delta s_{max}$ by default.



(a) (b)

Figure 4. Grid point distribution options on curves. (a) Uniform spacing on turning curves. (b) Non-uniform grid spacing distribution with end spacings $\Delta s_{e1}, \Delta s_{e2}$, maximum interior spacing Δs_{max}, and maximum stretching ratio SR_{max}.



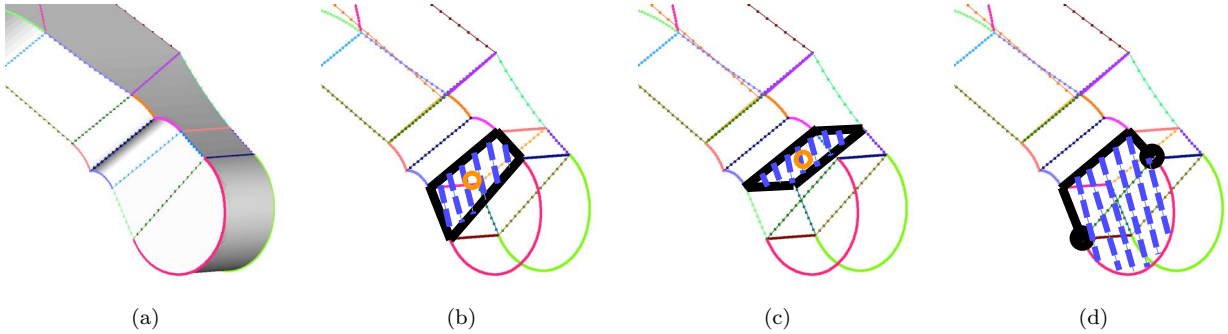
(a) (b)

Figure 5. Grid spacing continuity at junctions. (a) Identify matching pairs of curves at junctions using the curve end point unit vectors (dashed arrows). (b) Impose grid spacing continuity for matching pair end point spacings by taking the minimum of the two.

After assigning the grid point distribution on all curves based on the above criteria, an additional sweep is performed to ensure grid spacing continuity at junction points. At each junction point, two (and sometimes three) curves with end point unit vector directions that best match each other are formed into groups (Fig. 5a). The minimum end point grid spacing over all curves in the group is applied to the end points of other members of the group at the junction (Fig. 5b). For example, a junction of degree four typically

contains two matching groups of two curves each, while a junction of degree three may contain one matching group with two curves, or a matching group with three curves.

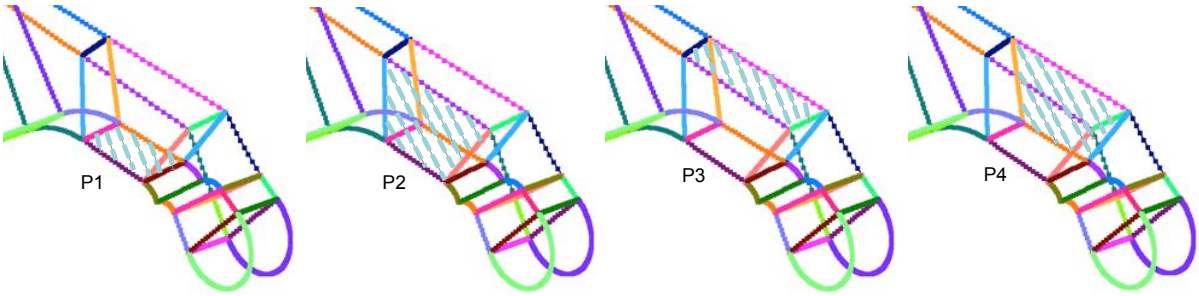
II.E. Construction and Concatenation of Algebraic Grids



(a) (b) (c) (d)

Figure 6. Identification of valid four-curve domains (domain validity test point shown in open orange circle). (a) Geometry and initial surface curves. (b) Valid and preferred four-curve domain that lies on the surface geometry shown by thick black lines. (c) Invalid four-curve domain that does not lie on the surface geometry shown by thick black lines. (d) Potentially valid but not preferred four-curve domain formed by thick black lines and red semi-circular curve.

The next step involves the identification of domains that are bounded by four curves. This is simply accomplished using the curves connectivity information at the junction points as described in Section II.B. Some domains may visually appear to be four-sided but are bounded by more than four curves. Treatment of such domains will be handled in the hyperbolic part of the process to be addressed in future work. The scheme presented below works for domains bounded by four curves that form a closed loop. However, not all such loops result in a valid surface domain. A valid four-curve domain must lie on the surface geometry. Fig. 6a shows the initial surface curves for a component and Fig. 6b shows a valid four-curve domain that lies on the surface geometry. Fig. 6c shows four curves that form a closed loop but the resulting domain is invalid since it does not lie on the surface geometry. The determination of whether a four-curve domain lies on the surface geometry is performed by computing the coordinates of a test point for the domain using transfinite interpolation from the mid points (by arc lengths) of the four bounding curves. If the distance of the test point from the geometry surface is less than a specified fraction of the bounding box of the boundary curves, the four-curve domain is considered a valid domain. Fig. 6d shows a potentially valid four-curve domain formed by the three black lines and the red semi-circular curve. However, the option illustrated in Fig. 6b is preferred since the resulting grid corner points subtend an angle close to 90 degrees whereas the grid corner points from the option shown in Fig. 6d subtend a much larger angle (close to 180 degrees).



(a) (b) (c) (d)

Figure 7. Concatenatable four-curve domains. (a) Patch 1. (b) Patch 2. (c) Patch 3. (d) Patch 4.

Starting with any four-curve domain that contains a curve on an open boundary or on a sharp edge of the geometry, a search is performed to find an adjacent four-curve domain that shares the given boundary curve (P1 and P2 in Fig. 7a,b). If such a domain is found, this establishes a sweep direction for adjacent four-curve domains. Further searches are then performed in the same sweep direction (forwards and backwards) to look for more adjacent four-curve domains that can be concatenated to the starting two (P3 and P4 in Fig. 7c,d).

When no further adjacent four-curve domains are found, the search terminates. The four-curve domains found could potentially form a group that can be concatenated into a single grid. In order to avoid large discrepancies in grid spacings in the concatenated grid, a constraint is imposed such that all curves that must share the same number of grid points in the concatenation group cannot differ in total arc length by a factor of R_{max} where R_{max} is a parameter usually set to 3.

It should be mentioned here that the gathering of four-curve domains into concatenation groups is not a unique procedure. A different starting domain and sweeping in different directions would result in a different final set of concatenated meshes. For example, applying a different sweeping direction as that shown in Fig. 7, five four-curve domains (P1, P2, P3, P4, and P5) could be concatenated into a single patch (Fig. 8). Provided best practice meshing guidelines are followed (such as grid point distribution), all possibilities of the final set are valid. Some outcomes of the final set may be preferred over others based on the total number of grids and the aspect ratio of each concatenated grid which may affect the convergence rate of the flow solution. Such a study is beyond the scope of the current paper.

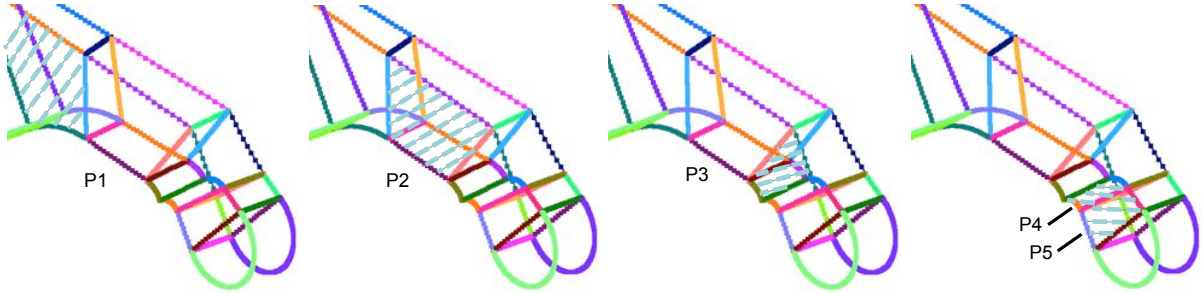


Figure 8. Alternative concatenatable four-curve domains. (a) Patch 1. (b) Patch 2. (c) Patch 3. (d) Patches 4 and 5.

From the set of curves that must share the same number of grid points in the concatenation group, the curve with the longest arc length is used to determine the number of grid points to be imposed on other curves in the group. The surface mesh for each four-curve domain is then created by transfinite interpolation, followed by projection to the surface triangulation geometry definition (Fig. 9a). Finally, the surface meshes in each group are concatenated along the sweep direction (Fig. 9b).

After the concatenation sweep in one surface direction, one might consider an additional sweep to concatenate adjacent surface meshes in the other surface direction. Since most structured flow solvers utilizing MPI would prefer to split large grids into smaller chunks for load balancing needs, the additional concatenation sweep is omitted in the current procedure in order to maintain smaller size grids for load balancing.

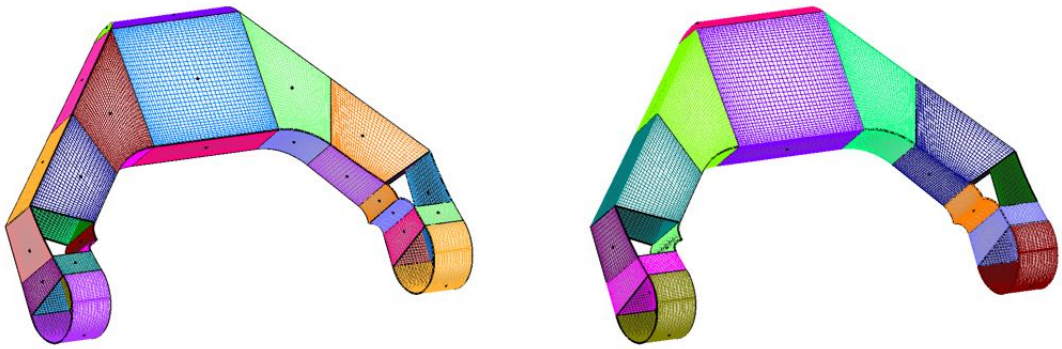
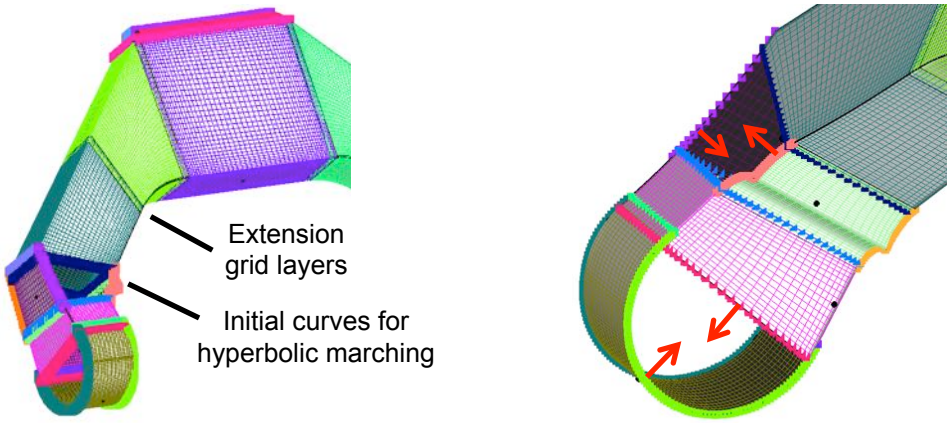


Figure 9. Creation of algebraic surface grids by transfinite interpolation. (a) Un-concatenated four-curve domains. (b) Concatenated four-curve domains.

II.F. Construction of Extension Grid Layers

After creating the surface grids over each four-curve domain followed by concatenation where appropriate, neighboring grids do not contain any overlap. For a surface grid boundary that lies adjacent to other surface grid boundaries, an extension can be built tangential to and outwards from the boundary into the adjacent surface space while matching neighboring surface curves where needed. This extension provides proper overlap between adjacent grids by introducing N_f layers of grid points where N_f is the number of fringe layers needed, and is dependent on the flow solver's differencing stencil. For example, a flow solver with a 5-point stencil would require two layers of fringe points ($N_f = 2$) so that the first solved point adjacent to a fringe point is supported by a full stencil. An example of this automatic extension using transfinite interpolation is shown in Fig. 10a. In future work, the extension layers of grid points will be built using a hyperbolic marching method which should provide a more robust scheme for following the surface geometry. At a surface grid boundary that is not adjacent to other surface grid boundaries, the boundary curve is used as an initial curve for hyperbolic marching (Fig. 10a). The marching direction for each initial curve is shown by a red arrow in Fig. 10b (see Section IV for future plans on the hyperbolic marching part of the process).



(a) (b)

Figure 10. Processing of concatenated algebraic grids. (a) Addition of extension layers to provide grid overlap, and identification of initial curves adjacent to uncovered surface geometry for hyperbolic marching. (b) Zoomed in view showing initial curves for hyperbolic marching with direction shown by red arrows.

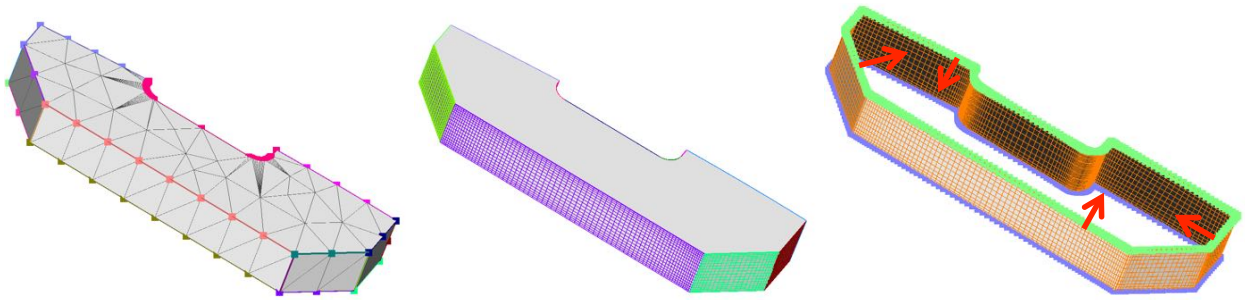
III. Results

The algebraic step described in the previous section is applied to several test cases. The only input information required were the unstructured surface triangulation geometry file, the discrete initial curves file, and an ASCII input file containing the names of the previously mentioned files, along with the name of the output structured surface mesh file. Optional inputs include parameters that control various grid attributes in the output surface meshes such as maximum stretching ratio, maximum grid spacing, maximum turning angle, and others. In all test cases, default control parameters were employed to run the procedure up to the algebraic grid concatenation step presented in Section II.E using one Intel Xeon E5 v2 processor (2.7 GHz) on a Linux workstation.

The first test case is a bracket attached to a high-lift geometry. Fig. 11a shows the surface geometry and 33 surface curves. Fig. 11b shows 11 surface grids created before the concatenation step with auto-reduction of grid spacing at convex corners and high curvature regions. All 11 grids are auto-concatenated into one surface grid shown in Fig. 11c. Automatically identified initial curves for hyperbolic marching are illustrated in the same figure with the marching direction marked by red arrows.

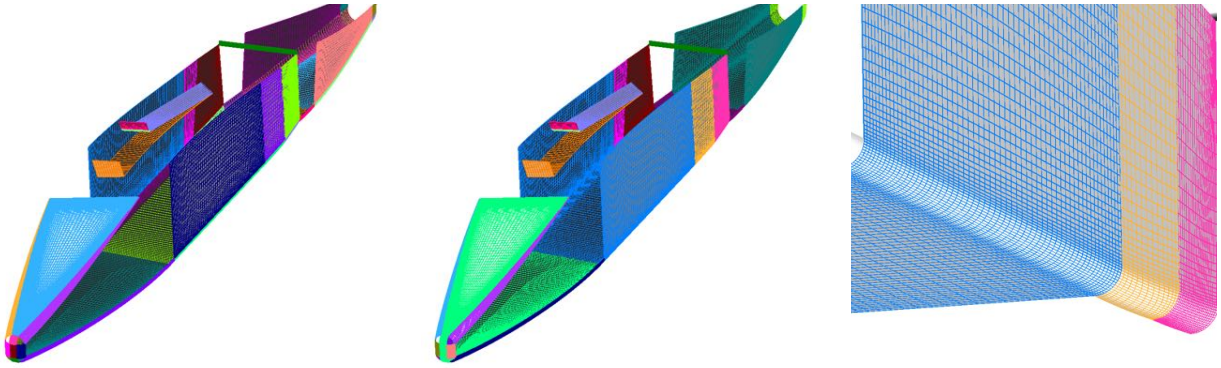
The second test case is a flap track fairing from a high-lift configuration where the geometry and 43 surface curves are shown in Fig. 1b. Surface grids before (43 counts) and after the concatenation step (21 count) are shown in Fig. 12a and Fig. 12b, respectively. Automatic grid spacing reduction around high curvature regions is demonstrated in Fig. 12c.

The third test case is a nacelle from a high-lift configuration where the geometry and 48 surface curves are shown in Fig. 13a. Surface grids before (16 count) and after the concatenation step (8 count) are shown



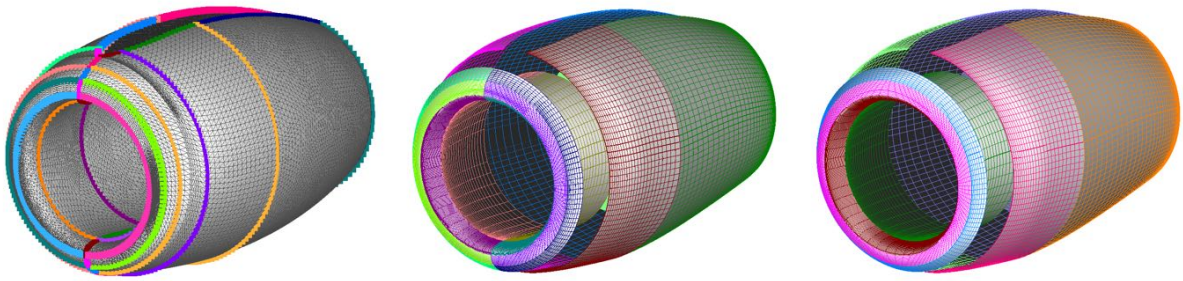
(a) (b) (c)

Figure 11. Bracket in high-lift geometry. (a) Surface geometry with 33 surface curves. (b) Surface grids on 11 four-curve domains before concatenation sweep. (c) Surface grids (1 count) after concatenation sweep with initial curves for hyperbolic marching (direction of marching marked by red arrows).



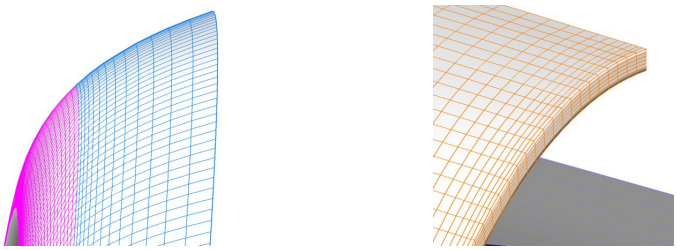
(a) (b) (c)

Figure 12. Flap track fairing in high-lift geometry. (a) Surface grids on 43 four-curve domains before concatenation sweep. (b) Surface grids (21 count) after concatenation sweep. (c) Zoomed-in view showing automatic grid spacing reduction in high curvature regions.



(a) (b) (c)

Figure 13. Nacelle in high-lift geometry. (a) Surface geometry with 48 surface curves. (b) Surface grids on 16 four-curve domains before concatenation sweep. (c) Surface grids (8 count) after concatenation sweep.



(a) (b)

Figure 14. Automatic grid spacing specification for nacelle in high-lift geometry. (a) Small grid spacing in high curvature region at leading edge. (b) Stretched grid spacing near finite thickness trailing edge.

in Fig. 13b and Fig. 13c, respectively. Automatic grid spacing reduction in the high curvature leading edge region is illustrated in Fig.14a. The blunt trailing edge grid spacing is determined by Np_{min} since it is a short curve. This small spacing is propagated to the outer and inner sides of the nacelle via the grid spacing continuity constraint (Fig. 14b). Automatic grid spacing stretching is performed moving away from the trailing edge.

The fourth test case is a landing gear strut from the 3rd AIAA Benchmark Airframe Noise Computations (BANC) Workshop with 144 initial curves (Fig.15a). There are 40 algebraically generated meshes before the concatenation step which are then concatenated into 7 meshes (Fig. 15b,c). The entire process took 7.3 seconds of computational wall clock time.

(a) (b) (c)

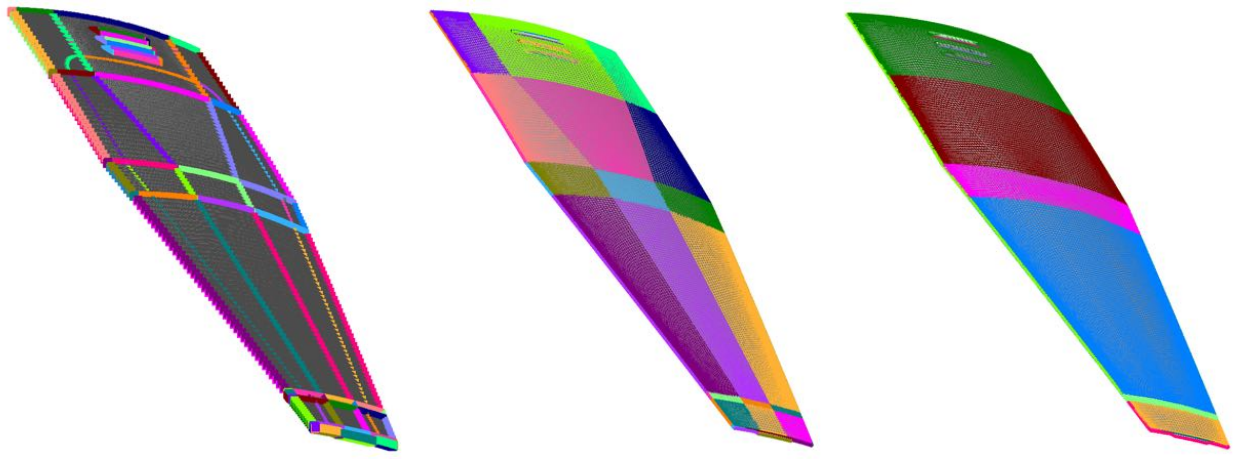
Figure 15. Landing gear strut. (a) Surface geometry with 144 surface curves. (b) Surface grids on 40 four-curve domains before concatenation sweep. (c) Surface grids (7 count) after concatenation sweep.

The fifth test case is the landing gear door from the 3rd AIAA Benchmark Airframe Noise Computations (BANC) Workshop with 188 initial curves (Fig.16a). There are 81 algebraically generated meshes before the concatenation step which are then concatenated into 28 meshes (Fig. 15b,c). The entire process took 12.5 seconds of computational wall clock time. Fig. 17a shows automatic grid spacing reduction around the convex corner at the edge of the door. The direction of the hyperbolic marching step to be performed is illustrated first in Fig. 17a around the boundary of the cap grid that wraps around the edge of the door, and then in Fig. 17b growing outwards from the oblong holes on the door.

The sixth test case is a rocket feedline with 126 initial curves and 62 four-curve domains. In this case, a manual approach using scripting best practices to create overset surface grids was attempted and took about 45 minutes of manual input time to identify the four-curve domains, distribute grid points on the bounding curves appropriately in these domains, specify the identification tags of the bounding curves of each domain, perform surface mesh generation, and concatenation of adjacent grids in each surface tangent direction. The computational wall clock time required was about 2 seconds. Under the automatic grid generation mode, at the end of the algebraic mesh concatenation step, two algebraic grids were generated in 2.5 seconds of computational wall clock time (Fig. 18). Since no attempt was made to perform concatenation sweeps in the other surface direction, two final grids remained after the one-direction concatenation sweep. While the computational wall clock time between the manual and automatic modes are approximately equal, significant savings in manual effort and time was achieved under the automatic mode.

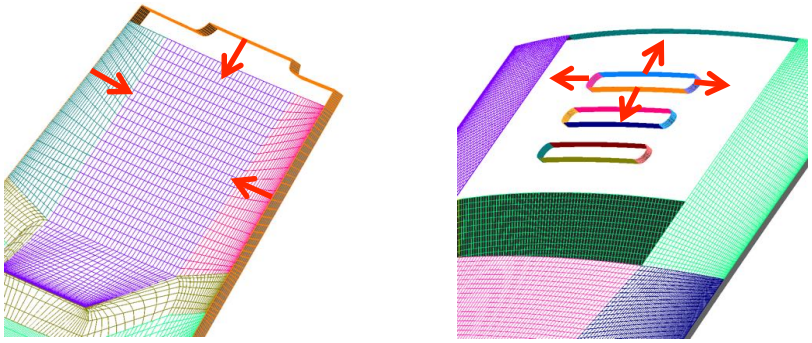
IV. Future Plans

For future work, the automation scheme for the algebraic step described in Section II could be further enhanced. The auto-generated grid point distribution is usually appropriate but there are some situations where a manual override is preferred. Further investigation into more sophisticated grid point distribution



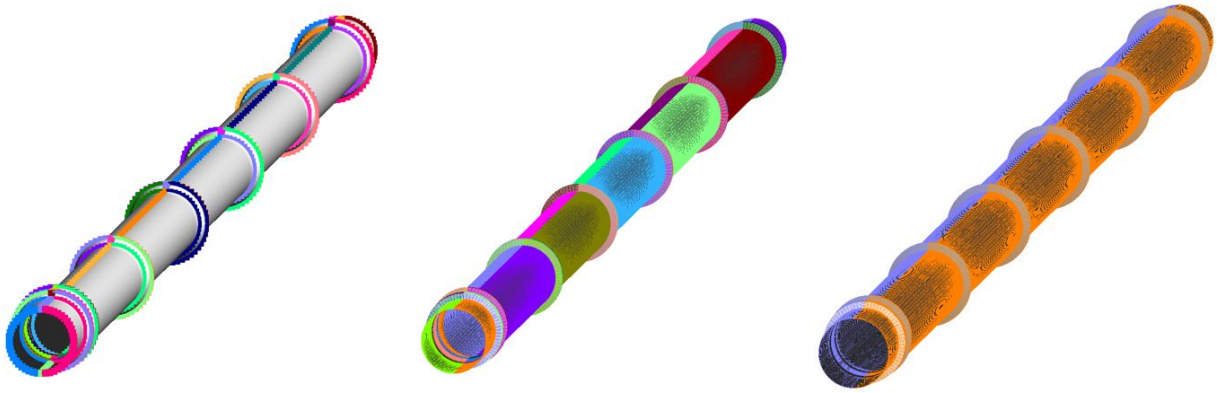
(a) (b) (c)

Figure 16. Landing gear door. (a) Surface geometry with 188 surface curves. (b) Surface grids on 81 four-curve domains before concatenation sweep. (c) Surface grids (28 count) after concatenation sweep.



(a) (b)

Figure 17. Landing gear door zoomed-in view (red arrows indicate direction of hyperbolic marching). (a) Automatic grid spacing reduction at sharp convex corner on door edge. (b) Surface grids near holes on door.



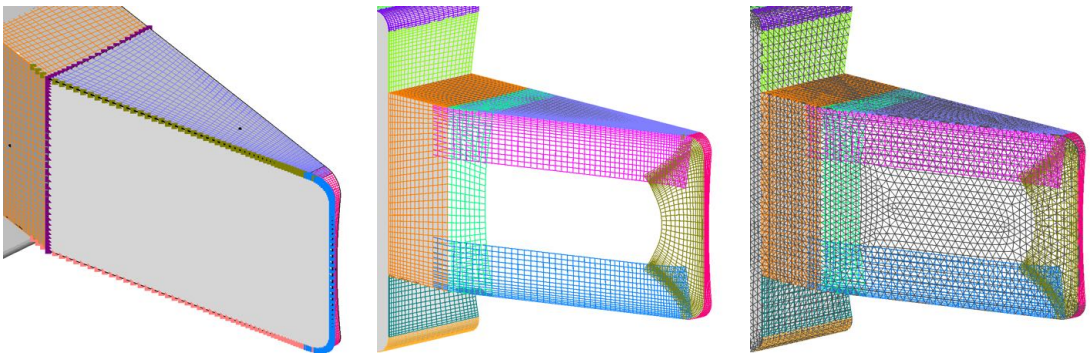
(a) (b) (c)

Figure 18. Long feedline. (a) Surface geometry with 126 surface curves. (b) Surface grids on 62 four-curve domains before concatenation sweep. (c) Surface grids (2 count) after concatenation sweep.

rules could reduce the need for such overrides. Depending on which four-curve domain patch is selected as the starting patch, many possible outcomes could arise from the patch concatenation scheme described in Section II.E. Further study could separate the choices that would lead to better solution convergence and robustness.

At the end of the algebraic step, four-curve domains have been filled with algebraic surface grids. Extensions have been added to grid boundaries that are adjacent to other algebraic surface grids to provide appropriate overlap between neighboring grids. Grid bounding curves that do not lie adjacent to other surface grids and curves from the initial set that have not been employed in the algebraic step are then utilized as initial curves for hyperbolic grid generation (Fig. 19a). A manually created set of hyperbolic grids is shown in Fig. 19b. Automation of the determination of marching distance and direction from each initial curve for each hyperbolic mesh, as well as grid spacing distribution in the marching direction are beyond the scope of the current paper, but will be presented in a future paper.

At the end of the algebraic and hyperbolic steps, the surface domain may or may not be completely covered. However, since all surface features of interest are modeled by the initial set of curves, and since all curves have been utilized by either the algebraic or hyperbolic steps, the surface geometry around all surface features has been covered. Any remaining gaps not covered by the existing surface grids lie on the smooth regions of the surface. Such regions can be automatically identified by projecting vertices of the initial surface triangulation onto the current set of algebraic and hyperbolic surface grids (Fig. 19c). Vertices from the triangulation that do not lie inside a structured grid cell are marked as orphan vertices. Cells on the triangulation that contain only orphan vertices are labeled as orphan cells. An automated scheme to cover such orphan cells with algebraic or hyperbolic structured grids is also beyond the scope of the current paper and will be presented in a future paper.



(a) (b) (c)

Figure 19. Future plans. (a) Surface meshes after algebraic step and initial curves to be used in hyperbolic step. (b) Surface meshes after hyperbolic step. (c) Reference surface triangulation plotted over algebraic and hyperbolic surface meshes.

V. Summary and Conclusions

Starting with an unstructured surface triangulation derived from a native CAD, STEP, or IGES geometry definition, and a set of surface curves that tesselates the surface geometry, a three-step procedure has been outlined for the automation of overset structured grid generation. The process includes an algebraic step, a hyperbolic step, and a gap-filling step. This paper focuses on the details of the algebraic step only, while work on the subsequent steps will be presented in future papers. Connectivity between the initial curves is first established by identifying curve end points meeting at junction points. Grid point distribution on the curves involves two types. Uniform spacing is applied to turning or short curves. Stretched non-uniform spacing is applied to the remaining curves with consideration of the geometry feature at the curve end points. Grid spacing adjustment is performed to ensure continuity at junction points between matching curve groups. Groups of four-curve domains that can be concatenated together are then identified. Grid point count adjustment is subsequently performed on the curves in each group in the concatenation direction. Next, algebraic meshes are generated inside each four-curve domain followed by concatenation within each group. Extension layers from the outer boundaries of the concatenated meshes are created algebraically to provide mesh overlap by following neighboring surface curves. Initial curves to be used in the next hyperbolic

marching step are then identified. The scheme has been demonstrated on a variety of geometric components in aerospace applications. Significant savings in manual effort and time are achieved with just the algebraic step of the automation scheme.

Acknowledgements

This work is partially funded by the Transformational Tools, and Technology (TTT) Project under NASA's ARMD Transformative Aeronautics Concepts Program (TACP). The author is grateful to Dr. Robert Haimes from MIT for insightful discussions on BRep geometry definitions, and to Dr. Shishir Pandya from NASA Ames Research Center for discussions on various grid generation issues.

References

[1]Gomez, R. J., Vicker, D., Rogers, S. E., Aftosmis, M. J., Chan, W. M., Meakin, R. L. and Murman, S., "STS-107 Investigation Ascent CFD Support," AIAA Paper 2004–2226, 2004.

[2]Ahmad, J. U., Pandya, S. A., Chan, W. M. and Chaderjian, N. M., "Navier-Stokes Simulation of Air-Conditioning Facility of a Large Modern Computer Room," FEDSM 2005-77225, Proceedings of the 2005 ASME Fluids Engineering Division Summer Meeting and Exhibition, Houston, Texas, 2005.

[3]Pandya, S., Onufer, J., Chan, W. and Klopfer, G., "Capsule Abort Recontact Simulation," AIAA Paper 2006–3324, 2006.

[4]Kiris, C. C., Kwak, D., Chan, W. M., Housman, J. A., "High-Fidelity Simulations of Unsteady Flow Through Turbopumps and Flowliners," *Computers & Fluids*, Vol. 37, pp. 536-546, 2008.

[5]Bhagwat, M., Dimanlig, A., Saberi H., Meadowcroft, E., Panda, B. and Strawn, R., "CFD/CSD Coupled Trim Solution of the Dual-Rotor CH-47 Helicopter Including Fuselage Modeling," Proceedings of the American Helicopter Society Aeromechanics Specialist's Conference, San Francisco, 2008.

[6]Kiris, C., Housman, J., Gusman, M., Chan, W. and Kwak, D. , "Time-Accurate Computational Analysis of Ignition Overpressure in the Flame Trench," *Computational Fluid Dynamics Review*, Eds. Hafez, Oshima, Kwak, Publisher: World Scientific, 2010.

[7]Rogers, S. E., Dalle, D. J. and Chan, W. M., "CFD Simulations of the Space Launch System Ascent Aerodynamics and Booster Separation," AIAA Paper 2015–0778, 2015.

[8]Housman, J. A., Sozer, E., Moini-Yekta, S. and Kiris, C. C., "LAVA Simulations for the AIAA Sonic Boom Prediction Workshop," AIAA Paper 2014-2008, 2014.

[9]Pandya, S. A., Huang, A., Espitia, A., Uranga, A., "Computational Assessment of the Boundary Layer Ingesting Nacelle Design of the D8 Aircraft," AIAA Paper 2014-0907, 2014.

[10]Housman, J. A. and Kiris, C. C., "Numerical Simulations of Shock/Plume Interaction Using Structured Overset Grids," AIAA Paper 2015-2262, 2015.

[11]Housman, J. A. and Kiris, C. C., "Structured Overset Grid Simulations of Contra-Rotating Open Rotor Noise," AIAA Paper 2016-0814, 2016.

[12]Nichols, R. H., Tramel, R. W. and Buning, P. G., "Solver and Turbulence Model Upgrades to OVERFLOW 2 for Unsteady and High-Speed Applications," AIAA Paper 2006–2824, 2006.

[13]Kiris, C. C., Housman, J. A., Barad, M. F., Brehm, C., Sozer, E., Moini-Yekta, S. , "Computational Framework for Launch, Ascent, and Vehicle Aerodynamics (LAVA)," *Aerospace Science and Technology*, Vol. 55, pp. 189-219, 2016.

[14]Chan, W. M., "Best Practices on Overset Structured Mesh Generation for the High-Lift CRM Geometry," AIAA Paper 2017-0362, 2017.

[15]Chan, W. M., "The OVERGRID Interface for Computational Simulations on Overset Grids," AIAA Paper 2002–3188, 2002.

[16]Steinbrenner, J., Wyman, N. and Chawner, J., "Development and Implementation of GRIDGEN's Hyperbolic PDE and Extrusion Methods," AIAA Paper 2000-0679, 2000.

[17]Chan, W. M., "Developments in Strategies and Software Tools for Overset Structured Grid Generation and Connectivity," AIAA Paper 2011-3051, 2011.

[18]Rogers, S. E., Roth, K., Cao, H. V., Slotnick, J. P., Whitlock, M., Nash, S. M. and Baker, M. D., "Computation of Viscous Flow for a Boeing 777 Aircraft in Landing Configuration," *J. of Aircraft*, Vol. 38, No. 6, pp. 1060-1068, Dec. 2001.

[19]Pandya, S. A. and Chan, W. M., "Automation of Structured Overset Mesh Generation for Rocket Geometries," AIAA Paper 2009–3993, 2009.

[20]Chan, W. M. and Gomez, R. J., "Advances in Automatic Overset Grid Generation Around Surface Discontinuities," AIAA Paper 1999–3303, 1999.

[21]Dannenhoffer, J. and Haimes, R., "Automated Creation of 3-D Overset Grids Directly from Solid Models," AIAA Paper 2011-3540, 2011.

[22], "ANSA: The Advanced CAE Pre-processing Software for Complete Model Build-up," BETA CEA Systems (https://www.beta-cae.com/ansa.htm), 2016.

[23]Haimes, R. and Dannenhoffer, J. F., "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry," AIAA Paper 2013–3073, 2013.

[24]Haimes, R. and Drela, M., "On the Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design," AIAA Paper 2012–0683, 2012.

[25]Vinokur, M., "On One-dimensional Stretching Functions for Finite-Difference Calculations," *J. Comp. Phys.*, Vol. 50, Issue 2, pp. 215–234, 1983.