# A Trajectory Refinement Scheme for Arbitrary Triangularly Tessellated Boundaries

Christopher E. Porter*
*NASA Glenn Research Center, Cleveland, OH, 44135*

Eric T. Galloway†, David L. Rigby‡, and William B. Wright§
*HX5 LLC, Brook Park, OH 44142*

**GlennICE relies on a Delaunay triangulation step to generate a two dimensional mesh of the point cloud of trajectory release locations in its adaptive refinement process. Previously GlennICE performed a 2.5D Delaunay triangulation to generate a two dimensional mesh from a point cloud in three-dimensional space. This process is not a general scheme and contains restrictions that prohibit GlennICE from being extended to tackle new challenges. This paper introduces a new method that does not rely on a 2.5D Delaunay triangulation, and instead performs a triangulation directly on the tessellated inlet surface. It accomplishes this by performing a triangulation on the local point cloud associated with each triangular face of the tessellated inlet surface. Each of these local triangulations are stitched together to form a single contiguous mesh. While the resulting mesh is not guaranteed to be a valid Delaunay triangulation, this formulation is significantly more general than its predecessor and enables the GlennICE code to be extended for a variety of different applications. This includes the ability to extend the software to handle reentraining splashed, bounced, or broken droplets and crystals.**

## I. Nomenclature

| | | |
|---|---|---|
| $CDT$ | = | Constrained Delaunay Triangulation |
| $CFD$ | = | Computational Fluid Dynamics |
| $CRM-HL$ | = | High Lift Common Research Model |
| $MWD$ | = | Minimum Wall Distance |
| $WD$ | = | Wall Distance |

## II. Introduction

Due to the increasing performance and decreasing cost of computational infrastructure, modeling tools have enabled the integration of many design considerations earlier in the design process. Icing is one such design consideration. Computational icing tools have been primarily applied to simulating ice accretion on a cantilever wing. For these configurations NASA's two dimensional and quasi-three dimensional tools, LEWICE [1] and LEWICE3D [2], are well respected having been validated against a large database of ice shape geometries [3] [4].

Current and future icing challenges are more general in nature. These challenges include icing on probes and radomes, engine icing, as well as unconventional configurations like truss braced wings and urban air mobility vehicles. NASA's Advanced Air Transport Technology and Transformational Tools and Technologies projects have an interest in tackling these more general icing challenges to enable the next generation of vehicle and engine designs. To meet these needs a shift from two and quasi-three dimensional approaches to a more generalized fully three dimensional approach is required. One such generalized module required within this framework is the ability to compute the three dimensional water impingement on an arbitrary geometry. The three dimensional water impingement computation

*Research Aerospace Engineer, Icing Branch, 21000 Brookpark Rd. MS 11-2
†Software Engineer, 2001 Aerospace Parkway
‡Aerospace Engineer V, 2001 Aerospace Parkway
§Aerospace Engineer V, 2001 Aerospace Parkway

is typically performed in an Eulerian or Lagrangian framework. The legacy NASA tools LEWICE and LEWICE3D leverage a Lagrangian framework while commercial tools such as ANSYS's FENSAP-ICE [5] and SIEMENS [6] utilize an Eulerian framework to compute the water impingement.

GlennICE [7] is NASA's next generational ice accretion solver. It leverages a Lagrangian based water impingement approach for two primary reasons. The first is that the Lagrangian approach provides a level of decoupling of the impingement accuracy from the refinement of the underlying CFD volume mesh. The second reason is the ability of the Lagrangian scheme to handle cases where the cloud can not be represented numerically as a continuum.

The Lagrangian methodology in GlennICE includes an algorithmic step that generates a two-dimensional triangular mesh. This mesh is constructed from the point cloud of initial trajectory release locations on the boundary surface of the CFD mesh. However, it is not trivial to generate a two dimensional triangulated surface from a point cloud in three dimensional space. Previously this was accomplished in GlennICE via a 2.5D Delaunay triangulation. While this method proved sufficient for simple configurations, it is unable to handle more complicated configurations of interest. This paper introduces a new method that is significantly more general in formulation, allowing it to operate on arbitrary underlying CFD boundary configurations.

## III. Methodology

GlennICE utilizes a refinement methodology that requires the formulation of a mesh from a point cloud that is coincident with the boundary of the CFD mesh. A Delaunay triangularization is utilized to facilitate the meshing of the point cloud. However, it is not trivial to mesh a point cloud coincident with an arbitrary tessellated surface in three space. The following sections describe the current approach in GlennICE, and the new approach introduced in this paper.

The user should note that prior to GlennICE V4.0, Delaunay triangulations were performed with the external utility Delaunator [8]. However, this package does not leverage robust predicates [9] and was susceptible to robustness issues. GlennICE v4.0 replaced the use of the Delaunator package with the use of the Constrained Delaunay Triangulation (CDT) [10] software package. The CDT utility does utilize robust predicates and has anecdotally been significantly more robust compared to the Delaunator utility.

### A. Current Approach

Figure 1 illustrates the current process for the triangulation of a point cloud in three dimensional space. It is effectively a 2.5 dimensional Delaunay triangulation, where one coordinate is discarded. This can be thought of as a projection of the three dimensional coordinates onto a fictitious two dimensional plane, a two dimensional Delaunay triangulation is performed, and the resulting connectivity is applied to the original three-dimensional point cloud. A visual representation of this can be seen in Fig. 1. The usage of a 2.5 dimensional triangulation is useful in the sense that it removes the restriction that the boundary is planar. However, there are still numerous restrictions with this approach. These restrictions include but are not limited to:

- Knowledge or assumption of the projection vector.
- Boundary's projected area must be completely convex.
- Boundary's projected area must be single value (i.e. it can't "fold" on itself).

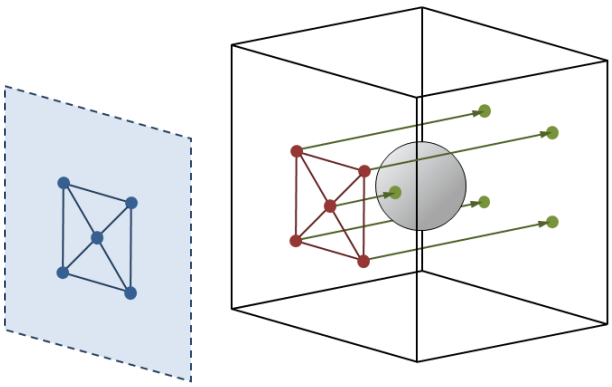

**Fig. 1 Schematic of the seed plane and the projection of the mesh onto the Inlet boundary.**

**B. New Approach**

The new approach differs from the current approach as the 2.5 dimensional triangulation is discarded. In it's place are a series of two dimensional triangulations that are stitched together to form a two dimensional mesh in three space. This methodology requires two restrictions. The first restriction is that the initial set of trajectories release locations, referred to in this paper as 'release points', are performed at the nodes of the underlying CFD inlet boundary. The second restriction is that refined, or new, release points are added at the mid-point between the release points of two previously computed trajectories.

Additionally, two new connectivity arrays are required for facilitating this methodology. The first connectivity array is a list of all of the faces of the inlet boundary that a given release point touches. For instance, it was mentioned that the initial set of release points are coincident with the nodes of the underlying inlet boundary. Thus for this initial condition, this connectivity array for a given release point lists all the inlet faces that contain the node that release point is coincident with. The second iteration of the algorithm will only add release points on the edges of the underlying inlet boundary mesh, and thus for these new release points, the two inlet faces that share that edge will be populated here. Eventually release points will be added that are on the inlet face, i.e. not coincident with a node or edge of the inlet boundary, and only a singular value will be stored in this connectivity array for those faces. The second connectivity array is a list of all of the release points that are coincident with a given face of the inlet boundary. This array is simply an inversion of the first connectivity array described above.

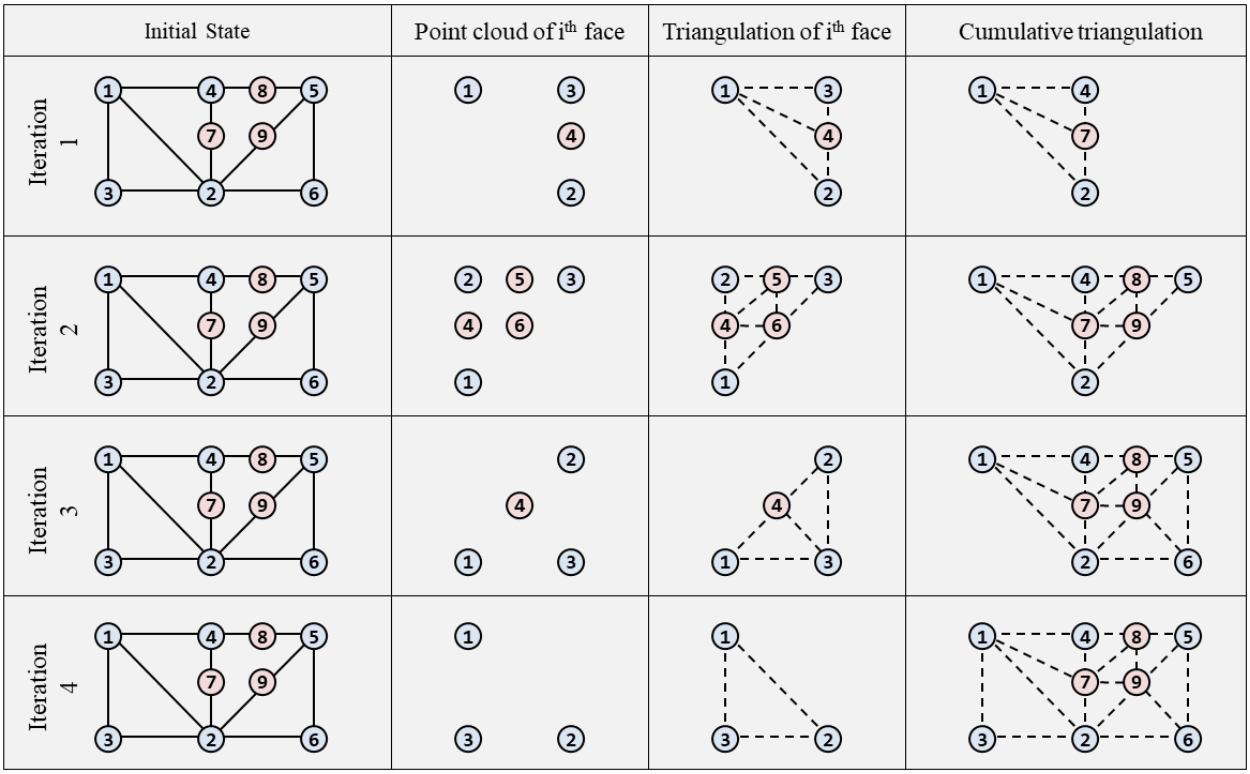


**Fig. 2 Tabular schematic illustrating the meshing algorithm. Nodal indexes in the first and fourth columns are globally indexed, and the second and third columns are locally indexed. Blue colored nodes denote the initial release points, while red colored nodes denote new release points added by GlennICE's refinement algorithms.**

Figure 2 illustrates the new meshing refinement process. In the left most column is the initial condition to the algorithm. The solid black lines are the underlying inlet CFD boundary. The colored circles denote the release points. The blue colored circles are the initial set of release points, note that these release points are coincident with the nodes of the underlying inlet CFD boundary. The red colored nodes denote new release points added by GlennICE's adaptive refinement scheme [11], note how the new release points are located at the mid-point between two pairs of existing release points.

For each face of the inlet boundary a two dimensional Delaunay triangulation is performed. Thus the number

of iterations of this meshing process is equivalent to the number of faces of the underlying inlet boundary. On each iteration the following set of steps is performed:

1) The release points coincident with face 'i' of the inlet boundary are extracted and the indexing of these release points is transformed from a global indexing to a local indexing for these subset of points.
2) These points in three dimensional space are rotated to the yz plane.
3) A two dimensional Delaunay triangulation is performed.
4) The local indexing of the subset of release points are transformed back to the global indexing of the total set.
5) The resulting triangulated release points are added to a cumulative list of triangulated release points.

Previously it was described that the triangles from each two dimensional triangulation were stitched together to form a single mesh. However, no additional stitching step is required. This stitching is handled implicitly by the two connectivity arrays described above allowing for the indexing transformations described in steps one and four of the process. The result from steps one to five above returns a list of triangles defined by the release points, where the release points are all unique, i.e. there are no duplicated indexes referencing a singular release point at a specific x, y, z coordinate. This is the same format that all tessellated surfaces are provided to GlennICE, and therefore no additional work is needed in this algorithm to stitch the individual triangulations together.

### C. Robust Triangulation

The previous section discussed the algorithmic process for grabbing a subset of points and appending the traingulation results of the subset to the running total. However, the details of the triangulation process are left out of that discussion. Ideally this triangulation could be accomplished with an arbitrary triangulation library, especially one that utilizes robust predicates, like the CDT library used in GlennICE [10]. In practice, however, a variety of point cloud preprocessing steps are required prior to triangularization.

The first preprocessing step is to transform the points from an XYZ coordinate system to a Y'Z' coordinate system such that a two dimensional Delaunay triangularization can be performed. GlennICE accomplishes this with the steps in the list below. Note that GlennICE uses the three nodes of the underlying CFD mesh for ($p_1$, $p_2$, $p_3$) as this guarantees that these points will be non-collinear. These are the blue colored nodes in both Figs. 2 and 3.

1) Identify three non-collinear points ($p_1$, $p_2$, $p_3$)
2) Translate all points such that $p_1$ is at the origin.
3) Rotate all points about the Z axis and subsequently around the X axis such that $p_2$ lies on the Z axis.
4) Rotate all points about the Z axis such that $p_3$ lies on the YZ plane.

A depiction of the rotated point cloud in the Y'Z' coordinate system can be seen in subfigure (a) of Fig. 3. In this figure the blue nodes represent the three nodes of the underlying CFD mesh, which is also the initial refinement for the first refinement iteration. The red nodes in this figure represent nodes that were added in subsequent refinement iterations. The axes are labeled in meters as this transformation is made up of a series of translations and rotations and these transformations do not distort the point cloud.

After the transformation to the two dimensional Y'Z' plane, the points could be passed to the triangulation library. However, GlennICE ran into robustness issues with maintaining the desired convex hull. The inability to maintain the desired convex hull was tracked to two primary sources. The first source arises from the large number of collinear points on the convex hull. Small floating point discrepancies can lead to these points not being exactly collinear. The second source arises for high aspect ratio point clouds, e.g. inlet cells in the viscous wall region of the mesh for in tunnel icing simulations.

$$\begin{bmatrix} 1 \\ p_{iY'} \\ p_{iZ'} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ p_{1Y'} & p_{2Y'} & p_{3Y'} \\ p_{1Z'} & p_{2Z'} & p_{3Z'} \end{bmatrix} \cdot \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ p_{1Y'} & p_{2Y'} & p_{3Y'} \\ p_{1Z'} & p_{2Z'} & p_{3Z'} \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 \\ p_{iY'} \\ p_{iZ'} \end{bmatrix} \tag{2}$$

In order to maintain the desired convex hull, the point cloud undergoes two distortion transformations that greatly increases the ability of the triangularization to maintain the desired convex hull. Both distortion transformations make use of the barycentric coordinate system. Equation 1 shows the linear system of equations governing the transformation

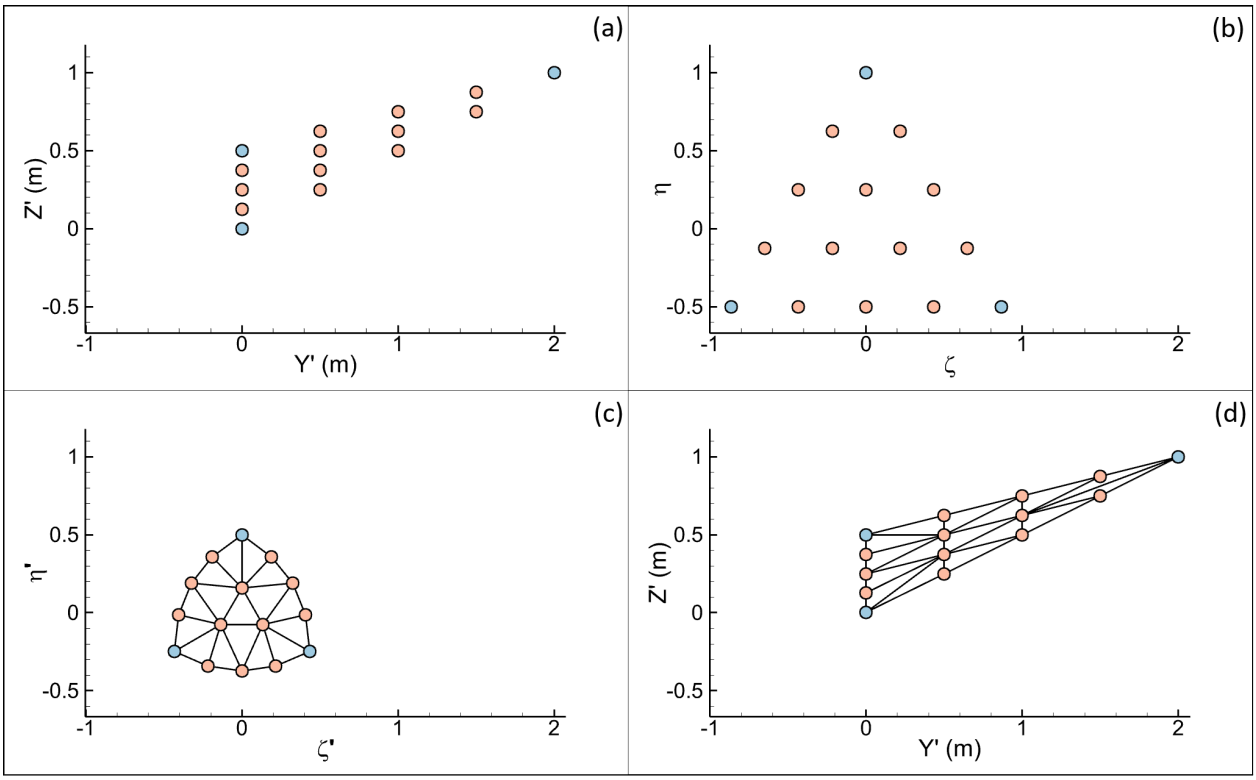


**Fig. 3 Depiction of the triangulation workflow in GlennICE's refinement algorithm. Blue colored nodes denote the initial release points, while red colored nodes denote new release points added by GlennICE's refinement algorithms.**

from cartesian coordinates to barycentric coordinates. The transformation matrix is defined by the vertices of the triangle, ($p_1$, $p_2$, $p_3$). Knowing the coordinates of a point on the triangle, $p_i$, the barycentric coordinates can be computed from solving Eq. 2.

The first distortion performed transforms the point cloud from an arbitrary aspect ratio to an equilateral triangle where $p_1$, $p_2$, and $p_3$ lie on the unit circle. This distortion is achieved by modifying the transformation matrix in Eq. 1 to that shown in Eq. 3. For a given point, $p_i$, Eq. 2 is solved for the barycentric coordinates $u_i$, $v_i$, and $w_i$. These values are then substituted into Eq. 3 to compute $p_{i\eta}$ and $p_{i\zeta}$. This distortion is intended to eliminate robustness issues due to the aspect ratio of the point cloud. A visual depiction of this transformation can be seen in subfigure (b) of Fig. 3.

$$\begin{bmatrix} p_{i\eta} \\ p_{i\zeta} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} \tag{3}$$

The second distortion performed translates all the points towards the center of the equilateral triangle. However the magnitude of the translation to the center is nonlinear. This introduces curvature to the convex hull of the point cloud. This distortion is achieved by distorting the barycentric coordinates computed from Eq. 2. These distorted barycentric coordinates are then passed through the transformation matrix from Eq. 3 to obtain the distorted cartesian coordinates $p_{i\eta'}$ and $p_{i\zeta'}$. The functional form of this distortion can be seen in Eq. 4. The value of $\gamma$ controls the level of distortion applied. For the case where $\gamma$ is equal to zero, the quadratic term in Eq. 4 vanishes and Eq. 3 is recovered.

This distortion is intended to eliminate the robustness issues due to collinear points on the convex hull. A graphical depiction of this transformation can be seen in subfigure (c) of Fig. 3. Note that for this figure a value of 0.5 is used for $\gamma$. This is the value of $\gamma$ utilized in v5.1.0 of the GlennICE software [7]. This parameter is hardcoded and is not a parameter available to the user via namelist input. After these two distortions, the point cloud is triangularized using the CDT library [10]. This triangularization can also be seen in the subfigure (c) of Fig. 3.

$$\begin{bmatrix} p_{i\eta'} \\ p_{i\zeta'} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} u_i - \gamma u_i^2 \\ v_i - \gamma v_i^2 \\ w_i - \gamma w_i^2 \end{bmatrix} \tag{4}$$

This connectivity is then applied to the original point cloud in the XYZ coordinate system. This application of the connectivity on the undistorted point cloud can be seen in subfigure (d) of Fig. 3. Note that this figure shows the point cloud in the Y'Z' coordinate system, and not the XYZ coordinate system. The reader is reminded that the transformation from the XYZ to Y'Z' coordinate system does not distort the point cloud. So the depiction of the connectivity on the point cloud in Y'Z' coordinate system is sufficient for this paper.

The reader can see in subfigure (d) of Fig. 3 that a valid triangularization with the desired convex hull has been computed for the point cloud. However, the user is not guaranteed that the resulting triangulation on the undistorted point cloud will be a Delauny triangularization. This is not deemed to be an issue as a Delauny conforming mesh is not required, however algorithmic robustness is required.

### D. Refinement Algorithms

There are two distinct trajectory refinement algorithms in GlennICE that work in tandem to adaptively generate the trajectory release locations necessary for computing the local water collection to an engineering level of accuracy. The first refinement algorithm is tasked with finding the feature of interest starting from a sparse initial set of trajectories. The second refinement algorithm is focused on refining the trajectories that hit the surface of interest to improve the prediction of local water collection to an engineering accuracy.

A variety of papers have been published detailing GlennICE's refinement algorithms [11] [12] [13] [14]. However, only the initial paper by Wright [11] discusses the feature finding algorithm in any level of detail. The feature finding algorithm in that paper has been replaced by a simpler, but significantly more robust, implementation. The detail of GlennICE's current feature finding algorithm have not yet been published and will be discussed here. The reader is referred to the references listed above for details on the second refinement algorithm that controls refinement after the surface of interest has been found.

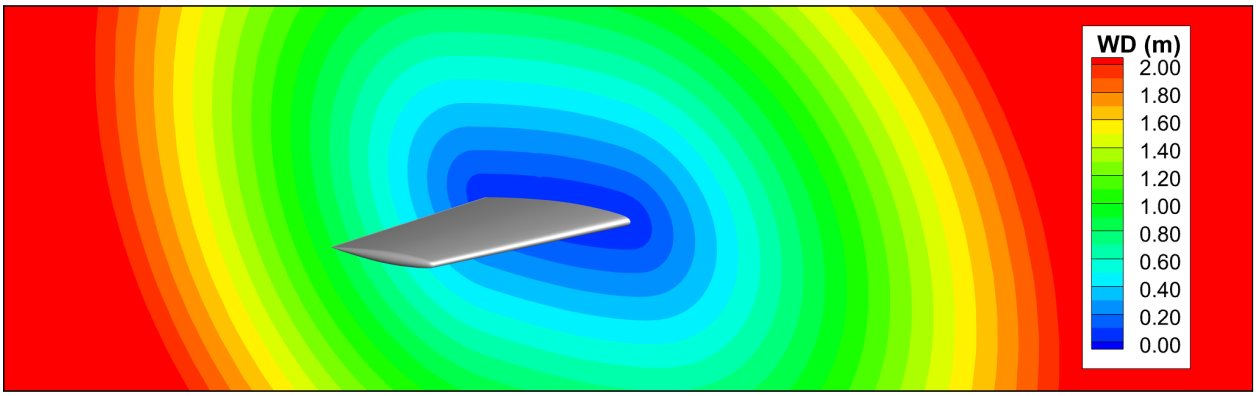


**Fig. 4 The wall distance scalar field at the 0.75m spanwise location on the ONERA M6 wing.**

The feature finding algorithm requires knowing the scalar field of the nearest distance to the feature of interest. This variable is equivalent to the 'wall distance' variable used in RANS based turbulence modeling, where the features of interest are the viscous surfaces of the simulation. GlennICE computes this variable internally using the 'Fast Distance Calculation' method of Wigton [15]. By computing this variable internally, GlennICE can compute the wall distance to any CFD surface(s) the user desires to analyze. For instance, the user can specify a subset of the viscous wall surfaces in a CFD simulation allowing the user to target specific components of the vehicle for impingement. A contour of the wall distance scalar field on a volume slice of the ONERA M6 wing can be seen in Fig. 4

In order to integrate trajectories, GlennICE is required to access the volume solution to grab relevant data such as the local air velocity. This capability was extended to allow GlennICE to also grab the wall distance variable. Thus for every time step along the trajectory, GlennICE queries and stores the value of wall distance. This can be seen in Fig. 5 for a rake of trajectories released upstream in the y = 0.75m plane.

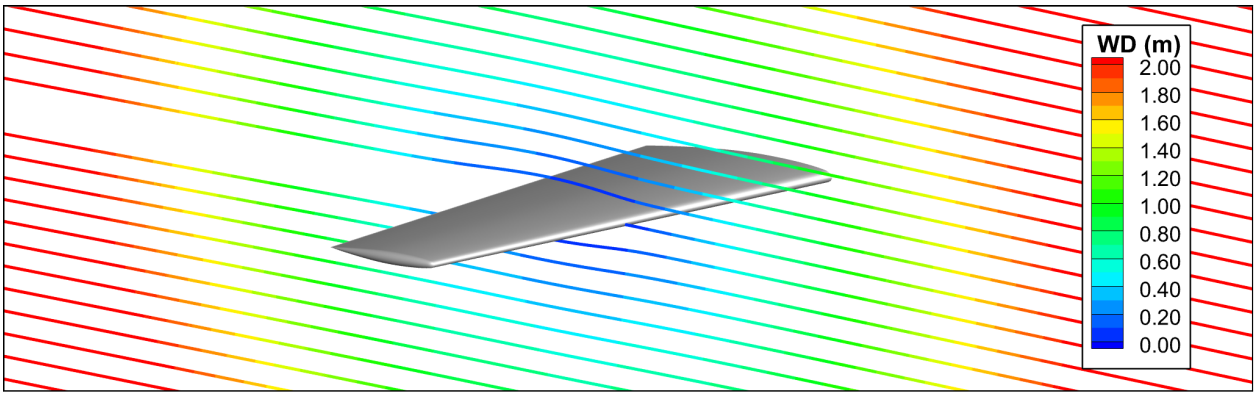


**Fig. 5 Values of wall distance along a rake of trajectories.**

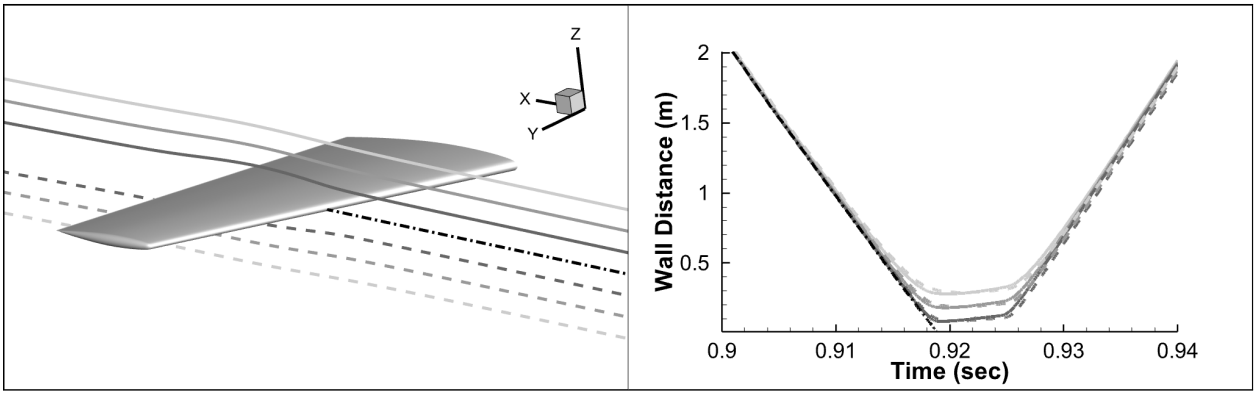


**Fig. 6 The wall distance as a function of time (right) for a subset of trajectories denoted graphically (left).**

After the trajectory integration is completed, each trajectory is post processed to identify the minimum value of wall distance along the entire trajectory. This gives a single scalar measure of how close each trajectory came to a feature of interest. Examining Fig. 6, the wall distance decreases with time as the trajectory approaches the feature of interest. The trajectory can hit the feature of interest and terminate (black dash-dot trajectory). Alternatively the trajectory can miss the feature of interest (grey solid and dashed trajectories) in which case the wall distance increases after the trajectory has passed. The minimum wall distance extracted from these trajectories from bottom to top are: 0.289m, 0.188m, 0.088m, 0.000m, 0.083m, 0.182m, 0.280m.

The feature finding criteria in GlennICE uses the minimum wall distance to determine if the edge between neighboring trajectories should be split. If the distance between the release locations of neighboring trajectories is larger than `mwd_scale_factor * min_wall_distance`, a new trajectory is computed between these two neighboring trajectories, cutting the distance between neighboring trajectories in half. The variable `mwd_scale_factor` is a variable in the `&adaptive_refinement` namelist that allows the user control over the aggressiveness of the feature finding algorithm. This value is defaulted to 1.0, and has been sufficient for virtually all cases of engineering interest. However, the default value may need to be adjusted for cases of abnormally low modified inertial parameter as tightly spaced trajectories may be required in those simulations.

This methodology does not naturally self terminate. This feature finding algorithm will continuously refine the impingement limit unless another metric shuts off the algorithm. In GlennICE this is accomplished by examining the ratio of trajectories that hit the surface with the number of total trajectories. Once a certain threshold is reached, GlennICE assumes that the refinement is sufficient enough for the feature finding algorithm to be terminated. In GlennICE this metric, called `hit_percent_limit`, is located in the `&adaptive_refinement` namelist. This value is defaulted to 25.0%, and has been sufficient for virtually all cases of engineering interest.

After the feature finding algorithm terminates, the second refinement algorithm then takes over complete control of the refinement algorithm. As mentioned previously, this second algorithm is focused on refining the hit trajectories to

improve the prediction of local water collection. Again, the reader is referred to Refs. [11] [12] [13] [14] for additional details on the second refinement algorithm.

## IV. Results

The results presented here focused on accomplishing two distinct tasks. The first task is to illustrate that the modification in the release and refinement methodology has no impact on the predicted impingement. This first task is accomplished by investigating the predicted collection efficiency on the ONERA M6 wing, a geometry that is used as an example case in 'Section 8 Example Cases' of the GlennICE Manual 5.1.0 [7].

The second task is to demo the new refinement methodology performing capabilities that the previous methodology was incapable of simulating. This is accomplished by illustrating the ability of this methodology to release and refine droplets emanating from the nose of the High Lift Common Research Model (CRM-HL) and impinging on the downstream wing. This demonstration is meant to illustrate a possible future workflow in GlennICE that would allow users to simulate the water droplets that splash or bounce off of the surface of an aircraft and impinge another aircraft component downstream.

### A. Freestream Inlet

The classical applications for inlet trajectory refinement is the refinement of a freestream inlet boundary to compute the collection on a fixed wing geometry. The test case used to illustrate this capability is the ONERA M6 wing. Dry air simulations were performed using NASA's FUN3D v13.7 [16] with the dry air operating conditions specified in Table 1. The cloud simulated in GlennICE is a monodispersed 20 micron cloud. A subset of the GlennICE refinement namelists for both the current and new refinement methodoliges are shown in the grey boxes below. Note how the new methodology removes the need for the user to specify a release box on the hemispherical freestream inlet.

**Table 1 ONERA M6 CFD Run Conditions**

| AOA (deg) | $U_\infty (m/s)$ | $M_\infty$ | $T_{s,\infty}$ $(K)$ | $\rho_\infty$ $(kg/m^3)$ | $a_\infty$ $(m/s)$ | $Re/C \times 10^6$ |
|---|---|---|---|---|---|---|
| 2 | 128.9 | 0.40 | 267.15 | 0.771 | 322.26 | 5.3087 |

```
! Current Refinement Namelist
&release
  inlet_coverage =  "rectangular"
  n_ytraj        =    9
  n_ztraj        =   16
  y_min          =    0.00
  y_max          =   83.46
  z_min          =  -83.46
  z_max          =   83.46
/
&surface_nml
  category(1) = "Inlet"
  category(3) = "Icing"
/
```

```
! New Refinement Namelist
&release
  inlet_coverage =  "full"
/
&surface_nml
  category(1) = "Inlet"
  category(3) = "Icing"
/
```

Figure 7 illustrates the existing 2.5 dimensional Delaunay trajectory refinement algorithm. The values of `n_ytraj` and `n_ztraj` were chosen to have similar edge spacing as the underlying inlet CFD mesh for a more direct comparison to the new refinement algorithm. Notice how the refinement occurs on a two dimensional plane upstream of the hemispherical inlet. Additionally, due to the hemispherical inlet, the user needs to specify a release box using the `y_min`, `y_max`, `z_min`, and `z_max` namelist parameters such that the projection of the box in the x direction is interior of the inlet boundary.

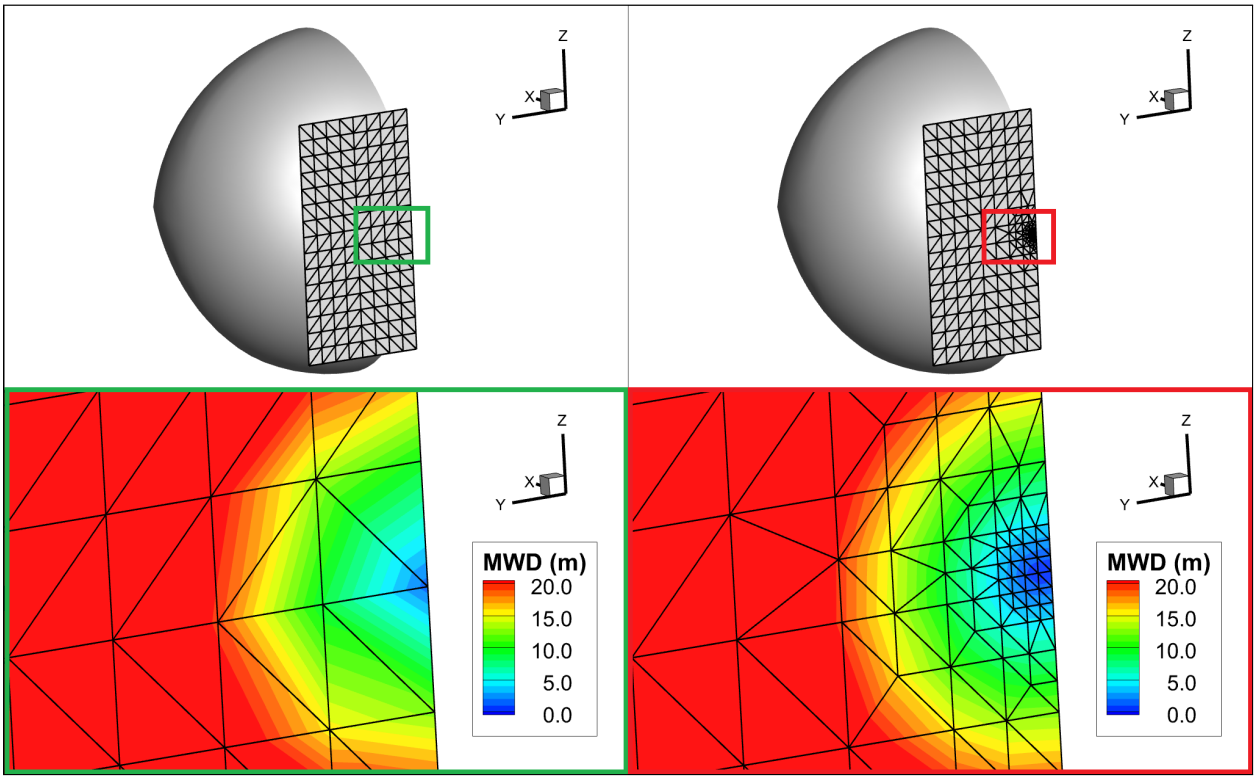


**Fig. 7 Seed mesh for the freestream inlet of the ONERA M6 simulation after one (left) and four (right) refinement iterations for `inlet_coverage = "rectangular"`.**

Figure 8 illustrates how the new algorithm progresses. Note that mesh generated in Iteration 1 is equivalent to the mesh of the underlying CFD boundary mesh. The mesh generated in Iteration 4 adds local refinement directly coincident with the underlying CFD boundary mesh. Since the refinement is generically computed directly on the tesselated surface of the CFD inlet, there is no need for the user to manually specify a release box. However, it should be noted that the triangulation in the bottom right subfigure contains regions where the triangulation is not delauny conforming due to the barycentric distortions applied to the point cloud for robustness.

Both refinement algorithms were run to completion using GlennICE's default convergence parameters. Comparisons of the collection efficiency surface contours for both refinement algorithms can be seen in Fig. 9. Visually there is no discernable difference between the two contours. Data was extracted from the y = 0.75m plane, denoted by the black line in Fig. 9, and plotted in Fig. 10. Similarly to the contour plot, there is visually no discernable difference between the resulting collection efficiency prediction for the two refinement methods. These qualitative comparisons are deemed to be sufficient in illustrating that the new refinement methodology is a suitable replacement for the current refinement method. As such, a detailed quantitative comparison was not performed. Instead, the quantitive analysis of the accuracy of the new refinement scheme is shown in GlennICE's Verification and Validation report [17].

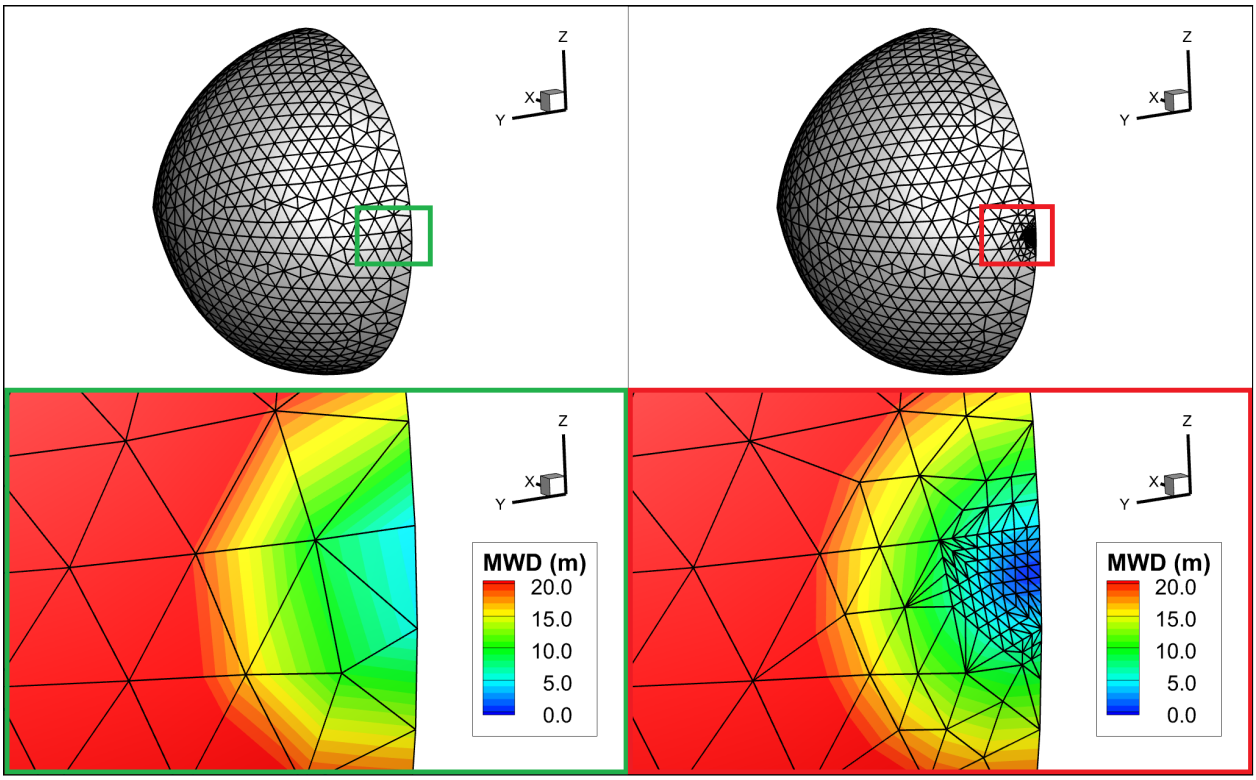


**Fig. 8 Release mesh for the freestream inlet of the ONERA M6 simulation after one (left) and four (right) refinement iterations for `inlet_coverage = "full"`.**

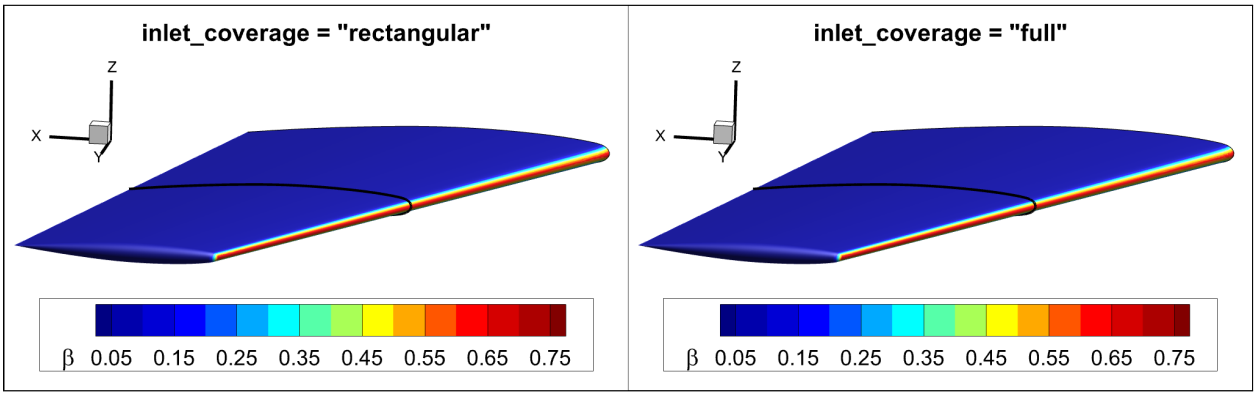


**Fig. 9 Surface contours of collection efficiency on the ONERA M6 wing for the legacy (left) and new (right) refinement schemes.**

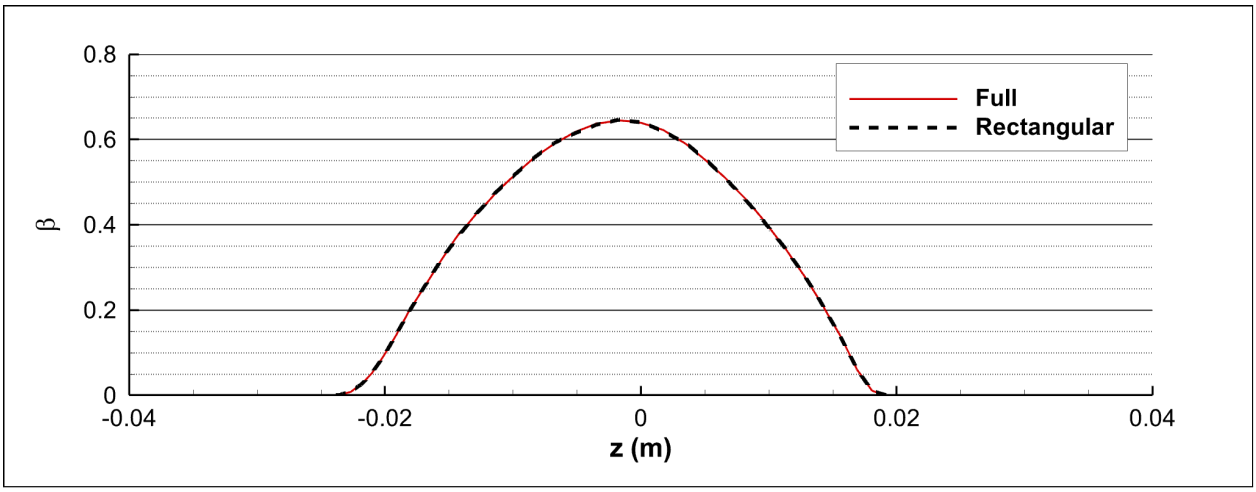


**Fig. 10 Line plot of collection efficiency on the ONERA M6 wing at the 0.75m spanwise location. The black line in Fig. 9 shows the spanwise location graphically.**

### B. On Surface Inlet

In order to illustrate the some of the additional capability that can be achieved with the new generic refinement methodology, a more complex case is demonstrated. This demonstration is performed on the cruise configuration of the High Lift Common Research Model (CRM-HL) seen in Fig. 11. Dry air simulations were performed using NASA's FUN3D v13.7 [16] with the dry air operating conditions specified in Table 2. The cloud simulated in GlennICE is a monodispersed 20 micron cloud. Note how this computational geometry is split up into multiple surfaces. For this demonstration, the nose of the aircraft is specified as an inlet, with the surface of interest being the wing. This demonstration is meant to emulate a possible workflow for computing the reinjection and reentrainment of splashed droplets into the flow. As seen in the sample namelist file, this is accomplished by setting the nose (surface 3) to an "Inlet" surface and the wing (surface 4) to an "Icing" surface.

```
&release
  inlet_coverage =  "full"
/
&surface_nml
  category(3) = "Inlet" ! CRM-HL Nose Surface
  category(4) = "Icing" ! CRM-HL Wing Surface
/
```

**Table 2 CRM-HL CFD Run Conditions**

| AOA (deg) | $U_\infty (m/s)$ | $M_\infty$ | $T_{s,\infty}$ $(K)$ | $\rho_\infty$ $(kg/m^3)$ | $a_\infty$ $(m/s)$ | $Re/C \times 10^6$ |
|---|---|---|---|---|---|---|
| 0 | 128.9 | 0.40 | 258.428 | 0.771 | 322.25 | 6.0505 |

There is currently no ability for GlennICE to read in generic contours of initial particle velocity, so for this demonstration the initial particle velocity had a direction of the surface normal of the aircraft, and a magnitude equal to the freestream velocity. Even with such a large initial particle velocity vector, the 20 micron droplets get quickly swept downstream as seen in Fig. 12. Some of these droplets impinge close to the wing.

Figure 13 illustrates the initial refinement on the nose, as well as the refinement after four refinement iterations. Note how the feature finding algorithm based on using the minimum wall distance behaves quite well even for a case as complex as the one demonstrated here. Ultimately, it would be desirable to extend GlennICE such that it can read in contours of trajectory intialization data, such as particle velocity and local liquid water content. With such a minor extension of the code base, GlennICE should be able to simulate the reinjection of splashed droplets in a similar workflow that GlennICE uses to compute the direct impingement.

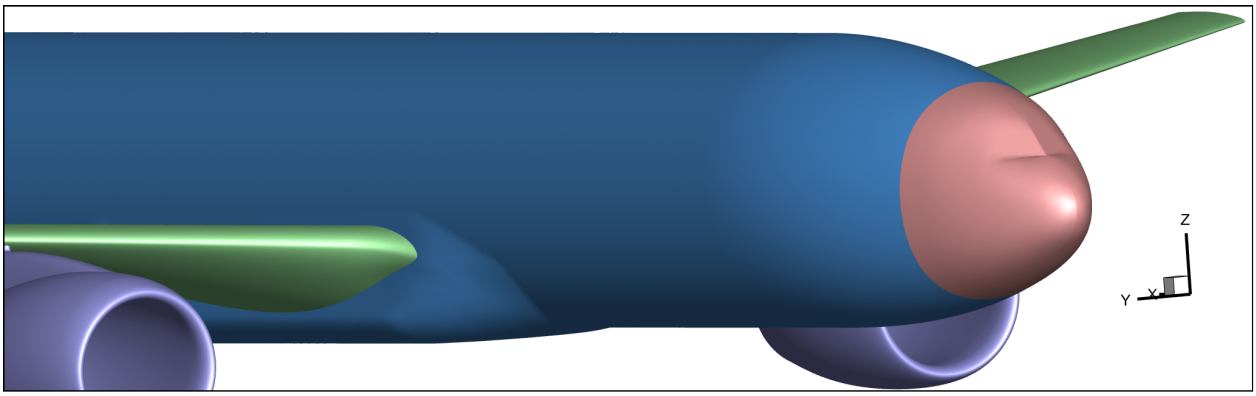


**Fig. 11 Computational surfaces of the CRM-HL simulation. (Pink - Nose, Blue - Fuselage, Green - Wing, Purple - Pylon/Nacelle)**

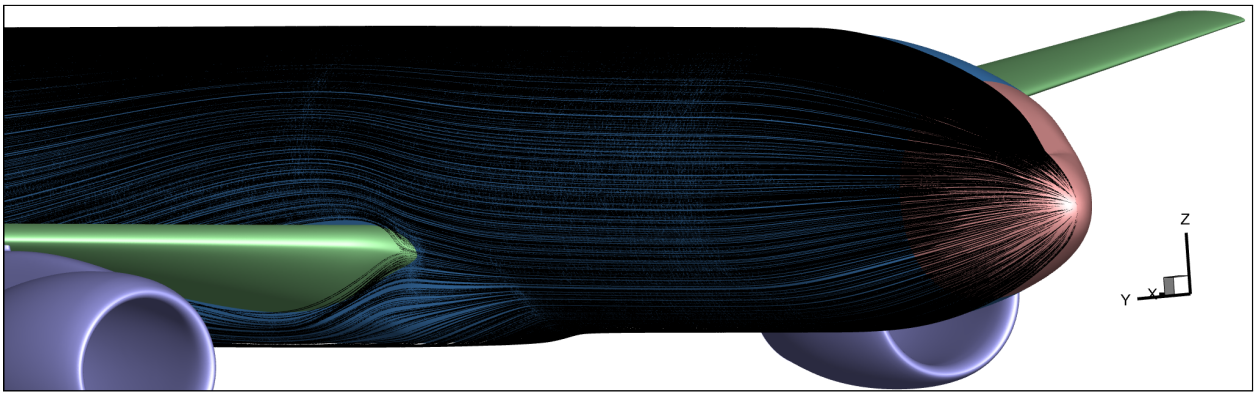


**Fig. 12 The first iteration of trajectories released from the nose surface of the CRM-HL.**

## V. Summary

Prior to GlennICE v4.1.0, the only trajectory refinement scheme available to users was a 2.5D Delaunay triangulation technique. This technique is sufficient for cases with simple inlet boundaries, such as a freestream farfield boundary. However, this technique is not capable of handling more complex boundaries such as the nose of an aircraft, or annular inlets commonly used in turbomachines.

A new general trajectory refinement process was introduced into GlennICE that refines an arbitrary triangularly tesselated surface. This is accomplished by triangulating multiple point clouds that are subsets of the total point cloud. These subsets are simply the point clouds that are coincident with each of the underlying CFD faces. The triangulations of these subsets are 'stiched' together to generate a singular traingulated mesh over the entire boundary surface.

This refinement method has two restrictions. The first restriction is that the initial trajectory locations must be the nodal locations of the underlying CFD mesh the trajectoreis are being released from. The second restriction is that the refinement must be performed by dividing the edges of the mesh in order to maintain consistency in the connectivity between the piecewise point cloud triangulations.

A comparison of the existing and new refinement methodlogies was performed on an ONERA M6 example case using the default refinement metrics in GlennICE. This comparison illustrated that the new refinement methodology generates the same predicted impingement as the existing refinement methodology. Additionally, a demonstration of the new refinement methodology was performed on a more complex test case where trajectories were introduced from the nose of the High Lift Common Research Model. This test case emulated the reinjection of droplets into the flow due to droplet splashing. This demonstration illustrated that the new refinement scheme is capable of appropriately releasing and refining trajectories on a complex surface.

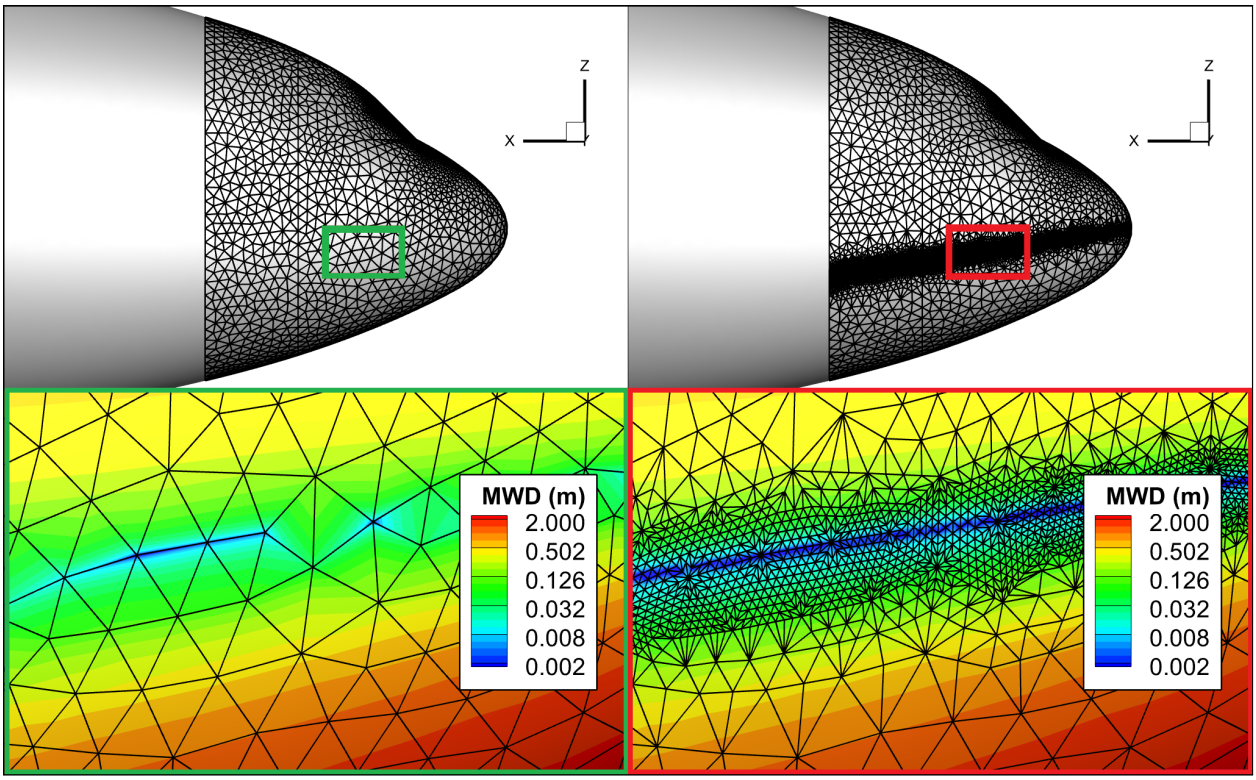


**Fig. 13 Release mesh for the nose of the CRM-HL after one (left) and four (right) refinement iterations.**

## Acknowledgments

The authors would like to thank NASA's Transformational Tools and Technologies (TTT) and Advanced Air Transport Technology (AATT) Projects for the financial support of this effort.

## References

[1] Wright, W. B., "User's Manual for LEWICE Version 3.2," *NASA/CR-20080048307*, 2008.

[2] Bidwell, C. S., "User's Manual for the NASA Glenn Three-Dimensional Grid Based Ice Accretion Code (LEWI3DGR Ver. 1.7)," *NASA*, 2005.

[3] Wright, W. B., "A summary of validation results for LEWICE 2.0," *NASA/CR-1998-208687*, 1998.

[4] Wright, W. B., "Validation results for LEWICE 3.0," *NASA/CR-2005-213561*, 2005.

[5] Morency, F., "FENSAP-ICE - A comprehensive 3D simulation system for in-flight icing," *AIAA 2001-2566*, 2001.

[6] Snyder, D., "Streamlining Aircraft Icing Simulations," *STAR Global Conference*, 2014.

[7] Porter, C., "GlennICE Manual 5.1.0," *NASA/TM-20250002839*, 2025.

[8] Bilonenko, V., "Delaunator-cpp," , 2018. URL `https://github.com/delfrrr/delaunator-cpp`.

[9] Shewchuk, J., "Robust Adaptive Floating-Point Geometric Predicates," *Proceedings of the twelfth annual symposium on computational geometry*, 1996.

[10] Amirkhanov, A., "CDT," , 2023. URL `https://github.com/artem-ogre/CDT`.

[11] Wright, W., "An Automated Refinement Process for Particle Trajectory Methods in GlennICE," *AIAA 2021-2631*, 2021.

[12] Porter, C., "A Comparison of Trajectory Refinement Schemes for GlennICE," *AIAA 2022-3692*, 2022.

[13] Rigby, D., "Convergence Criteria for Lagrangian Collection Efficiency Simulations," *AIAA 2022-3693*, 2022.

[14] Wright, W., "Improvements to GlennICE Collection Efficiency Algorithm," *AIAA 2024-4164*, 2024.

[15] Wigton, L., "Research in Computational Aeroscience Applications Implemented on Advanced Parallel Computing Systems," *NASA/CR-96-206062*, 1996.

[16] Biedron, R., "FUN3D Manual: 13.7," *NASA/TM-20205010139*, 2020.

[17] Porter, C., "GlennICE: Verification and Validation v5.1.0," *NASA/TM-20250002834*, 2025.