

An Automated Refinement Process for Particle Trajectory Methods in GlennICE

Wright, William B.¹

HX5, LLC, Cleveland, OH, 44135, USA

Christopher. E. Porter²

NASA Glenn Research Center, Cleveland, OH, 44135, USA

Eric T. Galloway³ and David L. Rigby⁴

HX5, LLC, Cleveland, OH, 44135, USA

Computational methods for ice accretion can simulate the impact of water drops and ice crystals on an aircraft surface in a Lagrangian reference frame or in the Eulerian reference frame. In the Eulerian reference frame, particles are considered a continuous fluid while in the Lagrangian frame individual particle trajectories are calculated. Methods that use the Eulerian reference frame are typically easier to develop as established modules used for continuum mechanics can be leveraged. The Eulerian systems can also be faster since the user does not have to simulate millions of particles in order to achieve good results. It is imperative therefore that a Lagrangian method optimize the release points of trajectories such that accurate solutions can be obtained while minimizing as much as possible the number of trajectories computed. This paper will present a methodology for this refinement process and demonstrate its effectiveness on sample three dimensional test cases.

I. Nomenclature

A	node area, m ²
a	first ellipse coefficient, m
b	second ellipse coefficient, m
LWC	liquid water content, g/m ³
m	mass flow rate, kg/s
V	velocity, m/s
x	x-coordinate, m
y	y-coordinate, m
z	z-coordinate, m
α	angle of attack, radians
γ	side slip angle, radians
ψ	angle in triangle, radians
∞	freestream

II. Introduction

GlennICE (Glenn Icing Computational Environment) is a computational tool designed to calculate ice growth on complex three-dimensional geometries using the input from a user-supplied computational fluid dynamics (CFD) solution for the geometry of interest. The NASA John H. Glenn Research Center at Lewis Field is developing this tool to aid those evaluating, designing and certifying aircraft, engines, and aircraft components for flight in icing conditions. This domestically available software is being developed to enable the introduction of new icing physics

¹ Research Engineer, HX5 LLC, Senior Member AIAA.

² Research Engineer, NASA Glenn Member AIAA

³ Research Engineer, HX5 LLC

⁴ Research Engineer, NASA Glenn Member AIAA

into a computational environment in a manner that is open for evaluation and eventual use by industry, academia, and other government organizations.

For those readers familiar with NASA's other computational icing tools, LEWICE 0 and LEWICE3D [2], they should note that GlennICE is being designed to address a generalized three dimensional icing problem. This differs from LEWICE and LEWICE3D, both of which relied on assumptions that are well suited for performing analysis on special cases, such as cantilevered wings, but not necessarily the optimal approach for the general case. As such, GlennICE is at a lower Technology Readiness Level (TRL) [3] than LEWICE and LEWICE3D for these special cases. Future versions of the software are planned to be completed on a regular basis and will include the improvements and validation necessary to mature the software to a comparable TRL. As such, this version of GlennICE is currently a research tool rather than a design or certification tool.

GlennICE is also a completely new software that is being developed with modern software practices. Because of this, legacy routines from LEWICE3D are not being used. However, it calculates water collection on a surface using the Lagrangian reference frame as did LEWICE3D. Recent research in icing has seen the development of several tools that perform these calculations in the Eulerian reference frame [4-6]. Each method has advantages in its favor. Eulerian algorithms are easier to develop as many of the routines can be leveraged from the flow solver and can be solved as a coupled system in order to compute evaporation and sublimation effects which are important inside the engine. They can solve time accurate solutions more simply than Lagrangian systems. Eulerian calculations are typically computationally faster [7] than Lagrangian methods and provide particle concentrations throughout the domain. However, the Lagrangian reference frame is more physically accurate, since the particle flow may not represent a continuum. This is especially true for secondary particle flows such as splashing and breakup. A Lagrangian routine can more easily use flow solutions from any CFD solver. They are also better at capturing gradients in the particle flow. Eulerian systems typically use the same grid for the flow solution and the particle flow. A grid that is designed to capture gradients in the air flow cannot also capture large gradients in the water flow unless a more refined grid is used, which reduces the computational advantage of the Eulerian system.

LEWICE3D and other Lagrangian methods in icing do not track the trajectory of every particle but instead track a statistical subset of those particles in order to determine the mass flow impinging an icing surface. Additional trajectories can be added to estimate off-body concentrations of particles. It is imperative therefore that a Lagrangian particle tracking method for icing should have an efficient means to select starting locations of particles in order to reduce the computation time. This paper will present the methodology for the particle refinement method implemented in GlennICE. Examples will be presented on three-dimensional geometries that require this methodology and comparisons will be made to LEWICE3D both in terms of the water impingement, but also in terms of the computation time necessary to obtain the solutions.

III. Methodology

The starting point for the refinement system is an initial point cloud of points on a 2D plane. Currently, GlennICE uses a rectangular bounding box of regularly spaced seed points to represent this point cloud as shown in Fig. 1. The points are then projected onto the inlet using a horizontal line from the seed point to the surface. For version 1.1, the inlet must be flat and all of the seed points must project onto the surface. For future releases, these restrictions could be lifted. This box can be smaller than the inlet shown in the figure. Points that lie on the surface are used as starting points for particle trajectories. Since the bounding box is smaller than the inlet, the parts of the domain that were not covered will not have particle trajectories pass through those regions. While this can increase computational efficiency, care must be taken by the user to ensure that all geometries of interest are covered. The seed points are then triangulated by using Delaunay triangulation[8] to create connectivity between the points. GlennICE uses a library routine called delaunator [9] for the triangulation. An alternative to the bounding box approach would be to use the triangular faces of the inlet directly as the initial seed points. Not only would this ensure full coverage of the inlet but would provide reasonable coverage for off-body concentration factors.

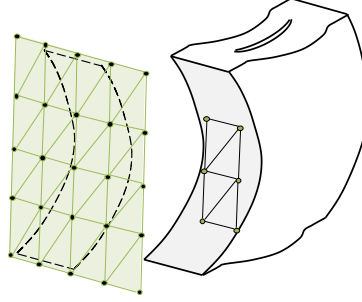


Fig. 1: Seed Point Bounding Box and Projection onto Inlet Face

As the trajectories are calculated, information is stored that will be used to determine future refinement. The first piece of information is the minimum distance to the wall. Turbulence models will contain an estimate of the distance to the wall in their models if a viscous wall exists. Wall distance can be computed internally by GlennICE, but this is not currently performed. As the trajectory path is determined, the closest location to the wall is determined and stored. Trajectories that hit the surface will have a minimum distance of zero. This is diagrammed in Fig. 2.

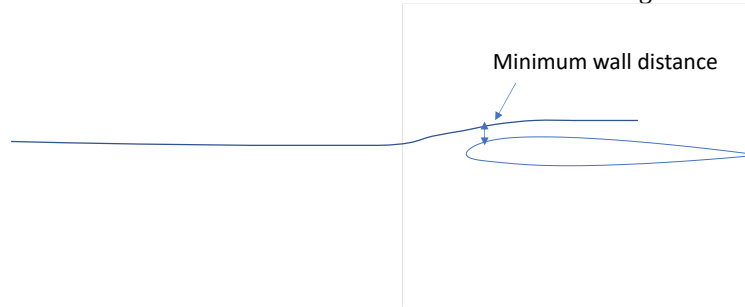


Fig. 2 Minimum Wall Distance for a Sample Trajectory

The minimum wall distance is then used to find trajectories that are closer to the surface by comparing the values obtained with each other. Since all of the seed points are connected by the triangulation, this is achieved by comparing all of the minimum distances using the node-to-face connectivity. The node-to-face connectivity is a list of all seed point indexes that are part of the same face. This is shown in Fig. 3. Seed points that have the smallest minimum distance compared to its neighbors are refined. Trajectories that hit the surface are also refined. As will be shown in the next section, this is a highly efficient means of finding trajectories that hit the surface. This occurs because the viscous walls act like a “sink” since the wall distance variable will decrease as the particle approaches the surface.

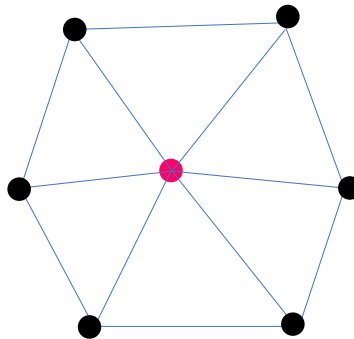


Fig. 3 Sample Node-to-Face Connectivity

The trajectory information gathered is used to identify seed points for refinement. The next step is to determine how to do that refinement. For trajectories that have been selected for refinement, all of the seed point triangles that contain that point are refined. The refinement adds a node at the midpoint of each edge of the triangle as shown in Fig. 4. This process can be expanded to divide a single edge or even to place new points inside existing triangles.

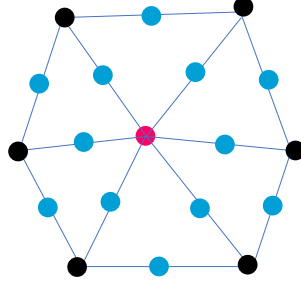


Fig. 4 New Seed Points Generated After Refinement

However, refining only the faces that contain the seed points was found to be insufficient for complex geometries. A second criteria was used that allows faces that neighbor those faces to be refined as well. A user input allows for multiple layers of neighbors to be refined. This means that not only are the neighboring faces refined, but the neighbors of those neighbors can be refined as well. The neighbor refinement process is shown in Fig. 5. Refining multiple layers of face neighbors can lead to a large number of trajectories. As stated earlier, there is a trade-off between robustness and computational efficiency that is case dependent. Because of this factor, the user can choose to turn off neighbor refinement when there are no more seed points that refine due to the minimum wall distance criteria. This will be demonstrated in the results.

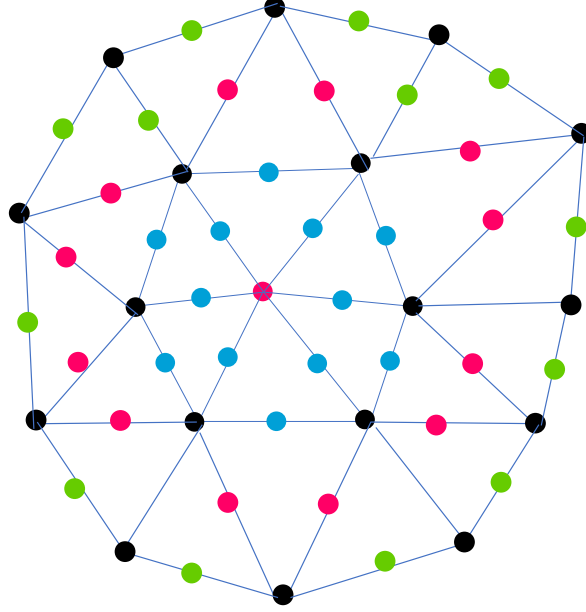


Fig. 5 New Seed Points Generated After Refinement of Three Neighbor Faces

Once the refinement has been applied to all of the selected seed points, a new triangulation is produced for the next iteration. The new seed points are projected to the surface and if those new points lie on the inlet boundary, a trajectory will be calculated from that location. The refinement will continue for the number of refinement iterations provided by the user or until a convergence criteria input by the user is reached. For the cases ran thus far, the freestream mass flux is constant. This makes the mass flow assigned to each trajectory a function of the area controlled by the release point. This is determined by

$$\dot{m} = LWC * V_{\infty} * \cos \alpha * \cos \gamma * A_{node}$$

$$A_{node} = \frac{\sum_{i=1}^{N_{faces}} A_i \varphi_i}{\sum_{i=1}^{N_{faces}} \varphi_i}$$

Where φ_i is the angle associated with the release point for face i . This is diagrammed in Fig. 6. Angle weighting was considered more accurate than a simple average. Thürrner, and Wüthrich [10] contains more detail on the advantages of angle weighting.

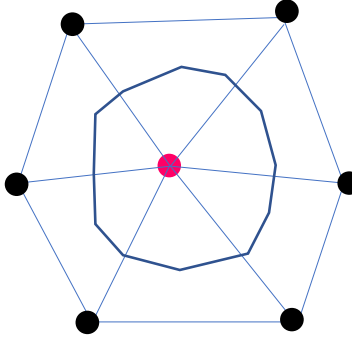


Fig. 6 Area Controlled by Trajectory

Finally, the collection efficiency is determined by taking the sum of the individual masses represented by each trajectory hitting a face and dividing by $LWC \cdot V_\infty \cdot A_{\text{face}}$. The face area is simply the triangle area whereas the node area is shown in Fig. 6. The next section will demonstrate the refinement capabilities on sample geometries.

IV. Results

The first example will present results from a unit test of the module. In this test, trajectories are not calculated but the module is supplied the distance information it needs in order to perform the refinement of the inlet plane. The test starts with the four corners of the bounding box which represents a rectangular inlet to the domain. The test then defines a rectangle and a circle that represent the targets for the refinement routine. The rectangle is slightly below the center line and offset slightly to the right. Its center is at (34,31.125) and is 14 units wide and 0.25 units high. The circle has a radius of 1 and a center at (14.5,14.5). The bounding box is 64x64 units. The bounding box and reference points are shown in Fig. 7. Each iteration, the minimum distance from each seed point to each of the reference geometries is calculated and stored as the minimal wall distance of a hypothetical trajectory starting at that location such that the refinement routine has the information it needs to proceed. Then the seed point indexes that meet the minimum wall distance criteria presented earlier are identified. Refinement around those locations is produced as described earlier as well as any seed point that hits within the defined geometry. After all new points have been created, a new triangulation is produced by the Delaunay routine. Fig. 8 shows the triangulation that is produced after 15 refinement iterations with contours showing the distance. This process creates 726102 triangles and 363058 seed points. This example shows that the process can successfully find features and refine around them.

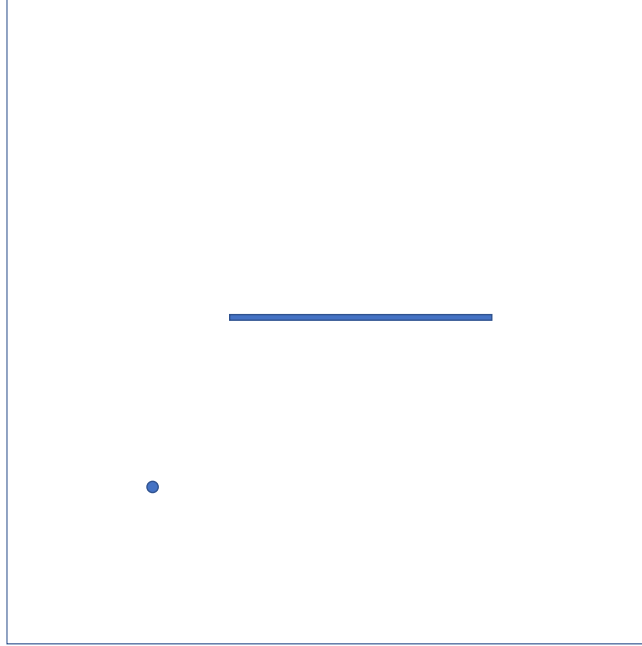


Fig. 4 Bounding Box with Reference Geometry

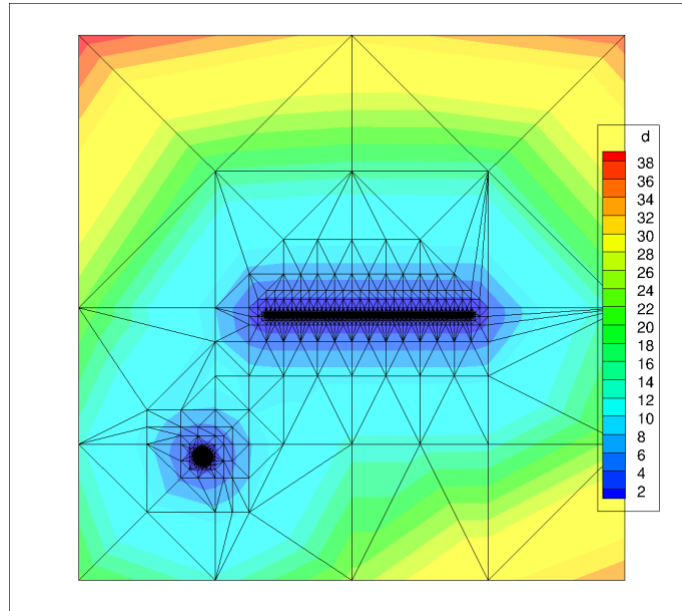


Fig. 5 Refinement Results for Unit Test Example

The second example will demonstrate the adaptive refinement process on a NACA0012. The geometry simulated is a straight wing NACA0012 with a chord length of 0.5334 m (21") with a far-field boundary approximately 20 chord lengths upstream and downstream of the wing. The outer block of the domain is a hexahedron. The top and bottom boundaries were also placed 20 chord lengths from the wing. The span of the wing was 0.9 m. Slip wall boundaries were used on the sides, top and bottom of the domain. The minimum x boundary was designated the inlet while the maximum x-boundary was the outlet. The grid was generated in Pointwise [11] by creating an unstructured domain in 2D and extruding spanwise. The 2D unstructured grid used structured quads near the surface using the T-Rex function with a δs at the wall of $1.7e^{-6}$ to achieve a $y^+ \leq 1$ with a boundary decay of 0.9 m. The wing section has a no slip wall boundary. The final 3D grid consists of 35,194 tetrahedrons, 290,601 hexahedrons, 891 pyramids and 53,829 prisms for a total of 380,515 cells in the mixed element domain. The flow solution was performed using ANSYS-CFX [12] with a Shear Stress Transport turbulence model [13]. Two flow solutions, using different wall temperatures, are used

in GlennICE to compute the convective heat transfer coefficient. Both flow solutions were generated using the following flow conditions: $AOA = 0^\circ$, $V_\infty = 66.88 \text{ m/s}$ (130 knots), $T_o = -2.2^\circ\text{C}$, $P_o = 101325 \text{ Pa}$ (1 atm). The first flow solution used a constant wall temperature of 20°C and the second flow solution used a constant wall temperature of 30°C . The two different wall temperatures are used in GlennICE to compute the heat transfer coefficient. For each trajectory the minimum distance the trajectory comes to the nearest body is identified and stored. This data is projected onto the two dimensional computational plane where the refinement methodology occurs. The desire is to only release trajectories that will result in impingement, i.e. minimum wall distances of exactly zero. These regions are the global minimum of the two dimensional data set described earlier. Fig. 9 shows the wing section with the inlet plane. The inlet plane with the seed point release plane is shown in Fig. 10. Trajectories are computed starting at the grid locations shown in this figure. A close up of the refinement is shown in Fig. 11 and collection efficiencies for this case are shown in Fig. 12. The collection efficiencies are the same as those produced by a uniform release grid and those computed with LEWICE3D.

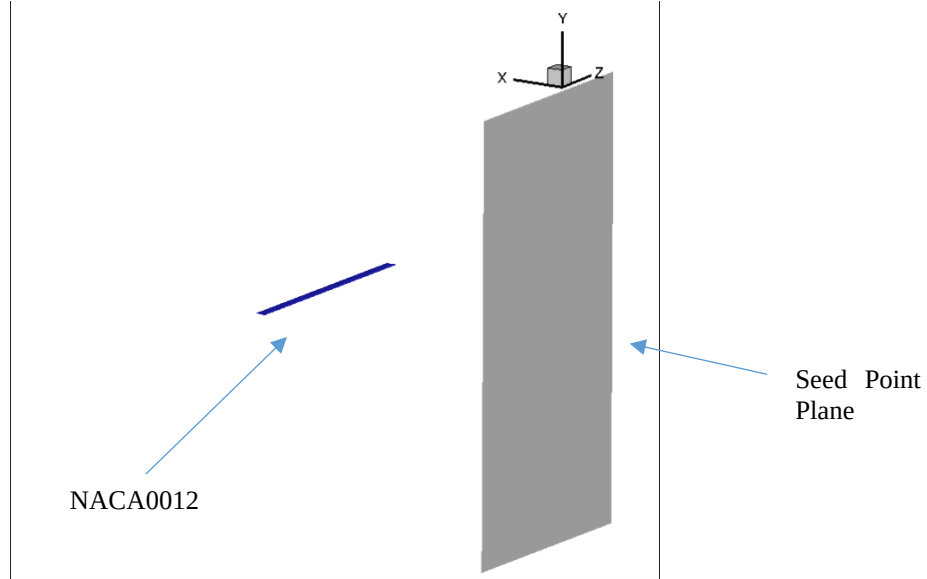


Fig. 9 NACA0012 Wing with Seed Point Plane

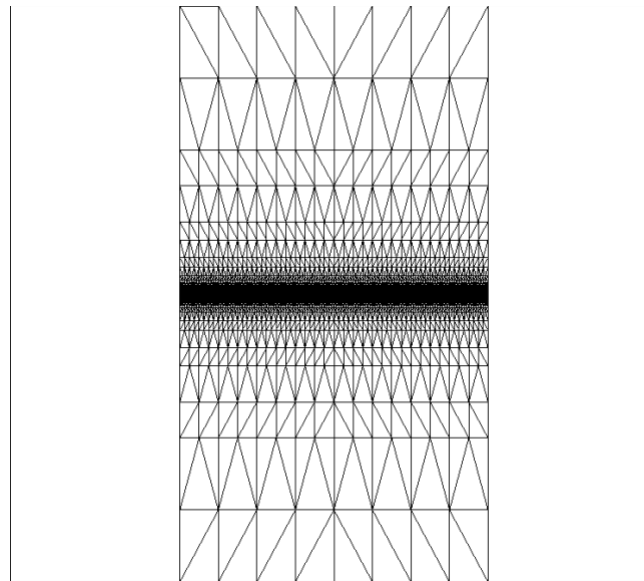


Fig. 10 Seed Point Refinement on NACA0012 Case

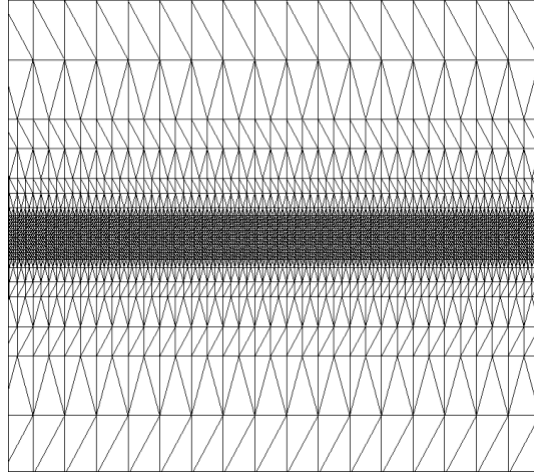


Fig. 11 Close Up of Seed Point Refinement Grid for NACA0012 Case

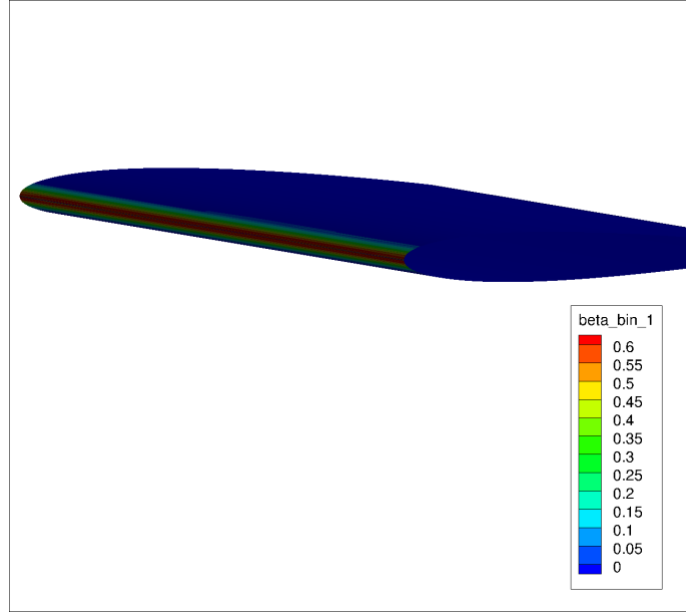


Fig. 12 Collection Efficiency on NACA0012 Wing

The next example shows adaptive refinement and collection efficiencies for an annulus. The annulus is defined by an ellipse in the xz -plane rotated a full revolution about the x axis. The ellipse has coefficients a and b of 0.1 and 0.5 respectively with a center located at $(0,0,1)$ as shown in the equation below. The semi-major axis is in the x direction and the semi-minor axis is in the z direction. Flow solutions were computed with ANSYS-CFX as described before.

$$\frac{x^2}{a} + \frac{z^2}{b} = 1$$

The local and global minimums are areas of interest and additional trajectories are released near these areas of interest. The local minimums are found and refined in the feature refinement portion of the adaptive algorithm, and the global minimums are found and refined in the hit refinement portion of the adaptive algorithm.

It should be noted that due to the discretization/sparseness of the available dataset, regions of this two dimensional problem with an analytic global minimum may not contain a local or global minimum on the discretized mesh the problem is being solved on. This can lead to undesirable behavior when refining only the local and global minimums. Fig. 13 illustrates how some regions of the annulus are ignored due to the errors introduced by the discretization, and Fig. 14 illustrates the impact this has on the predicted collection efficiency.

A neighbor refinement methodology is included to aid in robustness. The neighbor refinement expands the refinement region to refine not just local and global minimums, but also the regions in the vicinity of the local and global minimums. The results in Fig. 15 and 16 utilized five layers of neighbor refinement around both the local and global minimums. In comparison to the results presented in Fig. 13 and 14 it can be easily seen how the use of neighbor refinement increases the robustness of the adaptive refinement algorithm so that features of interest are not ignored when adaptively refining. The penalty of this robustness increase is additional computational cost due to the increase in trajectories being computed that do not impinge.

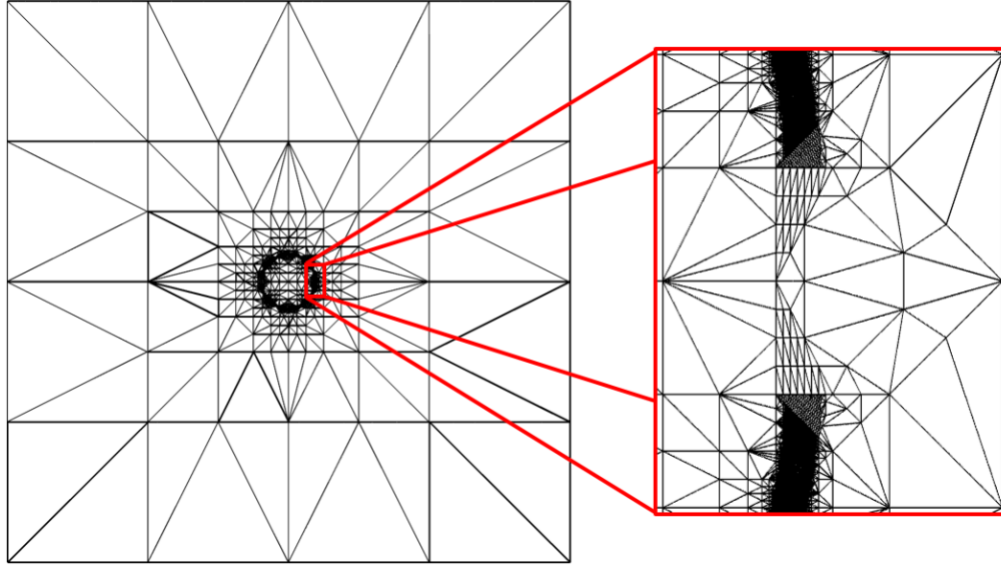


Fig. 13 Adaptive refinement of an annular test case where $\text{local_min_neighbors}$ and $\text{global_min_neighbors}$ are both set to values of zero.

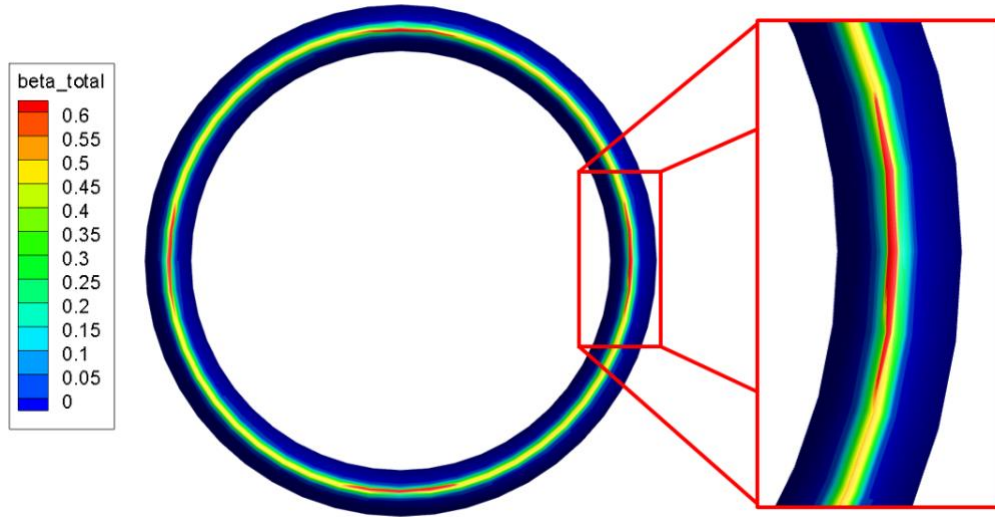


Fig. 14 Resulting predicted collection efficiency where $\text{local_min_neighbors}$ and $\text{global_min_neighbors}$ are both set to values of zero.

There is always a balance between robustness and efficiency. There exist two additional parameters available to the user to increase efficiency at the potential expense of robustness. The first parameter is max_face_count . This is an integer that sets a local cutoff for the adaptive refinement algorithm. Once a given surface face has exceeded the value of max_face_count , there will be no further refinement around the seed points that impinged this face. The

effect of this can be seen in Fig. 15. Here the impingement limits continue to refine, but the rest of the annulus has reached the cut off criteria specified by `max_face_count`. There is no particular guidance available at the moment for a proper value of `max_face_count`. It is possible that a proper value is dependent on case specific parameters such as the local gradient of collection efficiency, as well as the aspect ratio of the face being impinged. As such the default value of this parameter is set to `huge`, which means that the regions around seed points that result in impingement will always be refined. Fig. 16 shows the collection efficiency result for this case while Fig. 17 shows how the adaptive refinement scheme changes with particle size for this case.

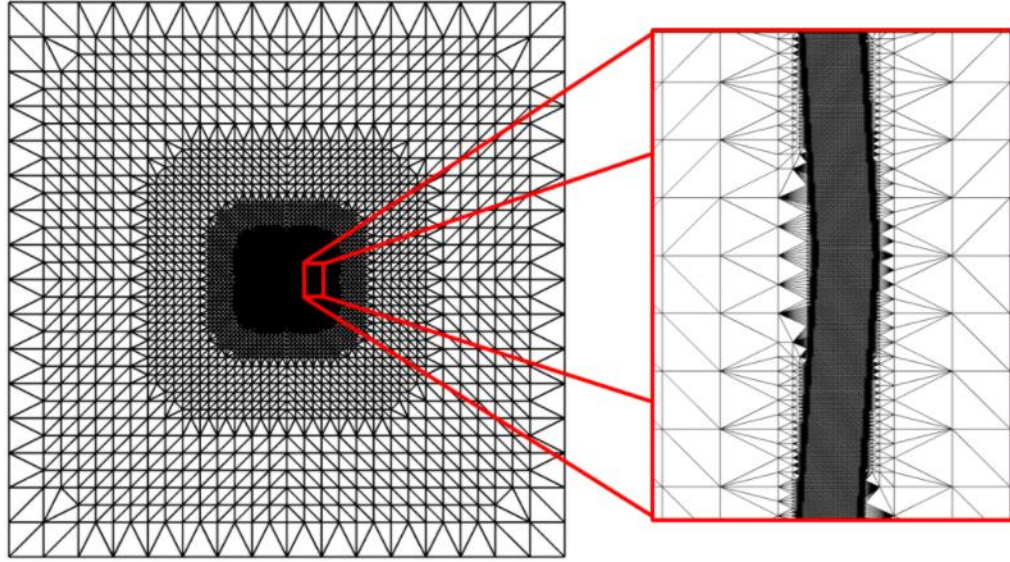


Fig. 15 Adaptive refinement of an annular test case where `local_min_neighbors` and `global_min_neighbors` are both set to values of five, `max_face_count` set to a value of 240 and `cease_neighbor_refinement` being set to `.true`.

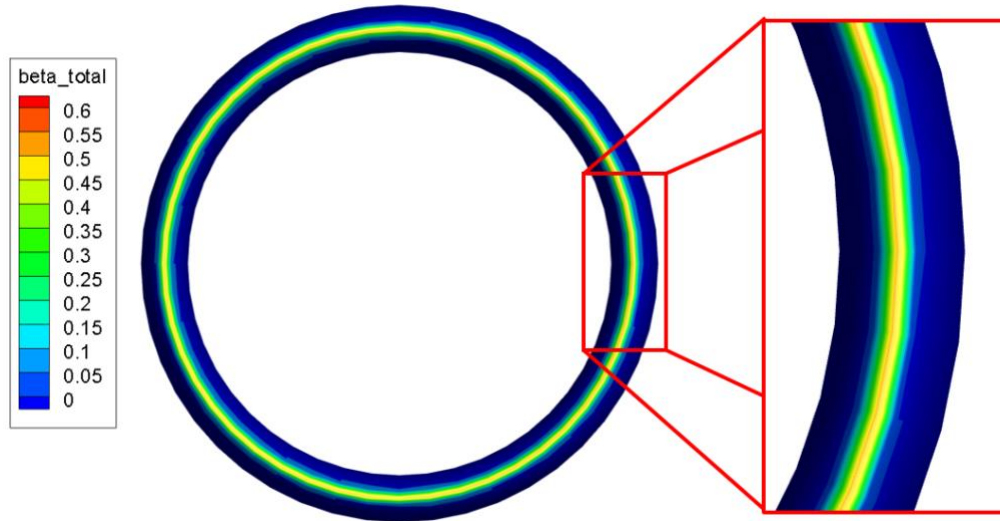


Fig. 16 Resulting predicted collection efficiency where `local_min_neighbors` and `global_min_neighbors` are both set to values of five.

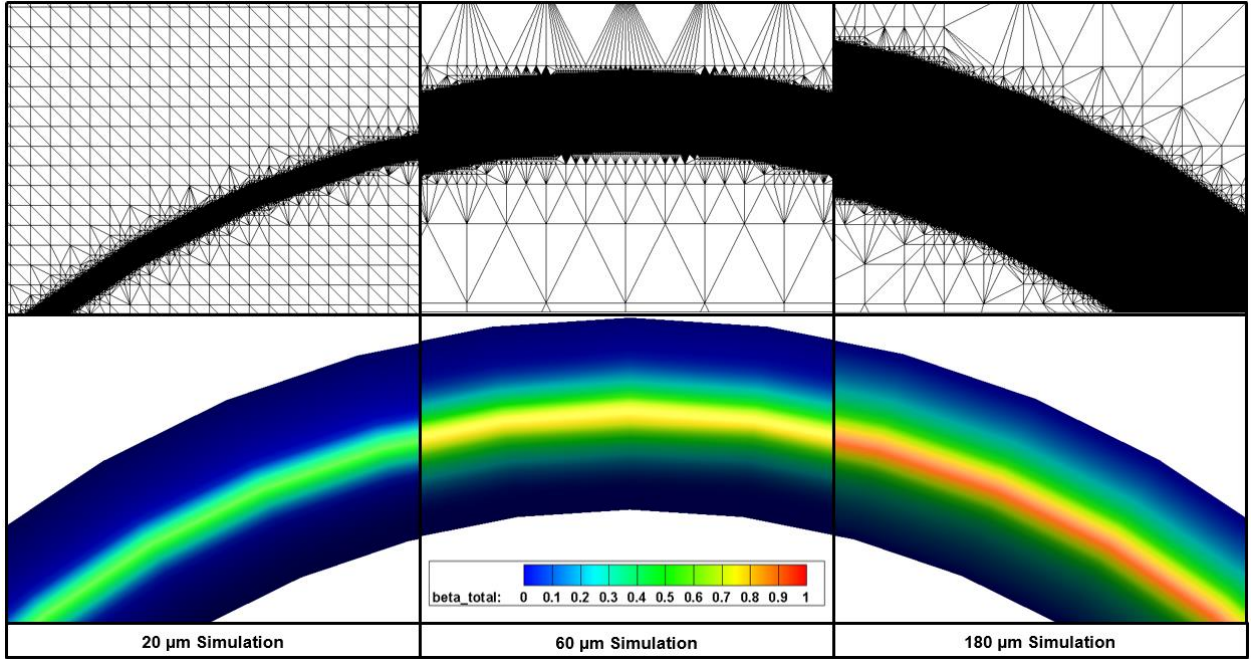


Fig. 17 Variation in Adaptive Refinement Grid and Collection Efficiencies for Three Sample Drop Sizes

The other parameter that can be varied by the user is `cease_neighbor_refinement`. Setting this value to `.true.` disables the conservative neighbor refinement when no local minimums are found. A local minimum occurs when neighboring trajectories pass on opposite sides of a geometry, with none of the trajectories impinging. When no more local minimums are found, it is assumed that there are no more regions that exist where neighboring trajectories pass on either side of the geometry with no neighbor impinging. i.e. it is assumed that all the features have been “found”.

Table 1 presents a numerical comparison of the effect of this parameter. Prior to iteration 12 for this particular test case, the algorithm behaves identically. On iteration 12, the neighbor refinement is disabled due to no more local minimums being found. By iteration 17, the difference in total trajectories is quite significant, with no change in the total mass flow, i.e. the impingement limits are similarly resolved.

Table 1 Comparison of number of trajectories and total impinging mass flow for the annular test case with the only differing parameter being the `cease_neighbor_refinement` boolean.

Iteration	<code>cease_neighbor_refinement=.false.</code>		<code>cease_neighbor_refinement=.true.</code>	
	Trajectories	Total Mass Flow (kg/s)	Trajectories	Total Mass Flow (kg/s)
7	1657	0.000000000E+00	1657	0.000000000E+00
8	2605	1.367182903E-02	2605	1.367182903E-02
9	4249	1.025387171E-02	4249	1.025387171E-02
10	7593	7.797214877E-03	7593	7.797214877E-03
11	14593	9.345976642E-03	14593	9.345976642E-03
12	28971	9.192435516E-03	17211	9.192435516E-03
13	59851	9.115664910E-03	24494	9.115664910E-03
14	129685	9.149460544E-03	47168	9.149460544E-03
15	302368	9.154362925E-03	125531	9.154362925E-03
16	779585	9.154727918E-03	413988	9.154727918E-03
17	1741761	9.153417468E-03	620922	9.153417468E-03

V. Conclusion

A method was presented for efficiently finding starting locations of trajectories that will actually hit three dimensional geometries of interest in aircraft icing. The results shown thus far demonstrate that starting with a very sparse initial set of release points the process can successfully refine these points and find starting locations that hit the surface with over 80% of the release points hitting the geometry.

Acknowledgments

The authors would like to acknowledge the support of the Advanced Air Transport Technology program for support for this research effort.

References

- [1] Wright, W. B. "Users Manual for LEWICE Version 3.2, NASA CR-2008-214255, Aug. 2008
- [2] Bidwell, C. S., and Potapczuk, M. G., "Users Manual for the NASA Lewis Three-Dimensional Ice Accretion Code (LEWICE 3D), NASA TM-105974, Dec., 1993.
- [3] "Technology readiness level" Wikipedia, Wikimedia Foundation, 29 September 2020, https://en.wikipedia.org/wiki/Technology_readiness_level
- [4] Pueyo, A., Ozcer, I., and Baruzzi, G., "An Eulerian Approach with Mesh Adaptation for Highly Accurate 3D Droplet Dynamics Simulations," SAE Technical Paper 2019-01-2012, 2019, <https://doi.org/10.4271/2019-01-2012>.
- [5] Villedieu, P., Trontin, P., Aouizerate, G., Bansmer, S. et al., "MUSIC-haic: 3D Multidisciplinary Tools for the Simulation of In-Flight Icing due to High Altitude Ice Crystals," *SAE Int. J. Adv. & Curr. Prac. in Mobility* 2(1):78-89, 2020, <https://doi.org/10.4271/2019-01-1962>.
- [6] Al-Kebisi, A., Mose, R., and Hoarau, Y., "Multi-Step Ice Accretion Simulation Using the Level-Set Method," SAE Technical Paper 2019-01-1955, 2019, <https://doi.org/10.4271/2019-01-1955>.
- [7] Durst, F. Milojevic, D. Schonung, B., "Eulerian and Lagrangian Predictions of Particulate Two-Phase Flows: A Numerical Study," *Appl. Mathematical Modelling*, Vol. 8, Issue 2, April 1984, pp. 101-115
- [8] "Delaunay Triangulation" Wikipedia, *Wikimedia Foundation*, 2 February 2021, https://en.wikipedia.org/wiki/Delaunay_triangulation
- [9] Delaunator-cpp, Vladi Bilonenko, GitHub repository, 30 October 2020, <https://github.com/delfrrr/delaunator-cpp>
- [10] Thürrner, G. and Wüthrich, C. A. Computing Vertex Normals from Polygonal Facets, *Journal of Graphics Tools*, 3:1, 43-46, DOI: [10.1080/10867651.1998.10487487](https://doi.org/10.1080/10867651.1998.10487487), (1998)
- [11] Pointwise, *Pointwise Inc.*, Fort Worth, TX; software available at <https://www.pointwise.com>
- [12] Ansys-CFX, *Ansys Inc.*, Canonsburg, PA; software available at <https://www.ansys.com>
- [13] Menter, F. "Zonal Two Equation κ - ω Turbulence Models for Aerodynamic Flows." 24th AIAA Fluid Dynamics Conference, Orlando, FL (1993).