VIII International Conference on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2019
E. Oñate, M. Papadrakakis and B. Schrefler (Eds)

A ROBUST 3D PARTICLE TRACKING SOLVER FOR IN-FLIGHT ICE ACCRETION USING ARBITRARY PRECISION ARITHMETIC

Tommaso Bellosta*†, Gianluca Parma† and Alberto Guardone†

† Department of Aerospace Science and Technology
Politecnico di Milano
Via La Masa 34, 20156 Milano, Italy
e-mail: tommaso.bellosta@mail.polimi.it

Key words: Particle laden flows; Particle tracking; Arbitrary Precision Arithmetic; in-flight icing

Abstract. A particle tracking code is presented to compute droplet trajectories within a known Eulerian flow field for in-flight ice accretion simulations. The implementation allows for hybrid or unstructured meshes used by common CFD solvers. A known vicinity algorithm was devised to identify particles inside the mesh by computing the intersection between the particle trajectory and the faces of the mesh elements. Arbitrary precision arithmetic is used in the intersection evaluation in order to avoid errors when selecting the exit face if the intersection point is close to or coincident with a vertex or an edge. State-of-the-art wall interaction models are used to take into account droplet rebound, splash and spread at the walls. Non planar surface elements are assumed to improve the accuracy in evaluating the trajectory of secondary re-emitted particles. The software exhibits almost linear scaling when running in parallel on a distributed memory system. The particle tracking code is assessed against the experimental results regarding the impingement of Supercooled Large Droplets over a wing.

1 Introduction

In-flight ice accretion is a relevant issue in aircraft design and aviation safety [1]. If an aircraft flies through cold wet air, the water particles from clouds can freeze after impacting upon its surfaces, possibly creating ice build-ups in the most exposed zones. Freezing occurs because impinging droplets are in a metastable state called supercooled state. Supercooled droplets exist in water form well below 0° C and can freeze if their equilibrium is perturbed, as it happens when they impinge against the surface of an aircraft. Ice formations have a detrimental effect on aircraft performance as they modify the aerodynamic shape of wings and blades, and can also affect aircraft stability as they

modify the weight distribution. Moreover ice build up on control surfaces and sensors can prevent the pilot or the control system from correctly manoeuvring the plane, which has led in the past to many accidents, some of which led to casualties. Ice accretion concerns also aircraft engines, as ice accumulation on the fan and spinner may eventually lead to ice shedding which can cause mechanical damage. Also, ice accretion on turbofan splitters and fan and booster vanes can result in slow acceleration and lead to compressor stall [4].

Ice accretion software are nowadays a fundamental tool in aircraft design. They are used to predict ice shapes, to investigate performances degradation and to aid the design of anti or de-icing systems. Diverse ice-accretion software can be found both in academy and industry [5, 6, 7]. The simulation of in-flight ice accretion requires the computation of the multiphase flow surrounding the aircraft. The flow is made up by a gaseous phase, namely, air, carrying the water particles, or droplets, in the liquid phase. From the solution of the flow field and of particle trajectories, the rate of mass of water impinging on the surfaces is extracted. Ice accretion is then computed by solving the energy and mass balance on the surface of the plane. Ice accretion is a time dependent problem: as ice starts to form, the shape of the surface changes and therefore the aerodynamic flow field around the body is altered. Since droplet trajectories strongly depend on the local value of the flow velocity and density, each trajectory is modified and the impact point is displaced, thus eventually altering the ice accretion rate. The ice accretion characteristic time is much larger than the aerodynamic one and therefore the ice accretion problem can be solved alternating the aerodynamic flow computation and the ice accretion computation. The output of the aerodynamic solver, including e.g. rate of mass impinging on the boundaries, heat transfer coefficient, shear stress, is fed into the accretion tool which computes the ice accretion modifying the geometry of the boundary. The (deformed) domain mesh is then regularized or the domain is re-meshed in order to comply with the deformed surface and the process is repeated until the total exposure time is reached.

The goal of the present contribution is to improve the Lagrangian particle tracking capabilities of PoliMIce by adding a robust tracking solver by including a known vicinity algorithm and by taking advantage of arbitrary precision arithmetic techniques to compute the intersection points of the droplet trajectories and the mesh edges. PoliMIce is an in-flight ice accretion framework developed at Politecnico di Milano capable of predicting rime and glaze ice formation on both 2D and 3D geometries [7].

The present paper is structured as follows. In Section 2, particle-laden flow equations are recalled and the particle solver is outlined. In Section 3, the tracking algorithm is described. To conclude, numerical results are reported in Section 4 and final comments are given in Section 5.

2 Particle-laden flow solver

Particle laden flows are multi-phase flows in which a carrier phase, liquid or gas, is mixed with a particulate of another phase. The carrier phase is continuously connected while the other is made of small, immiscible particles. An aircraft flying through a cloud

of supercooled water droplets is indeed a typical example of particle laden flows.

The concentration of droplets in the cloud is expressed via the Liquid Water Content (LWC), namely the measure of the mass of water in a cloud in a specified amount of dry air. It is usually measured in grams per volume of air. The LWC of a cloud varies significantly depending on the type of clouds present in the atmosphere: typical values range from 0.002 to 3 g/m^3. A representative value for the size of droplets in a cloud is the Median Volume Diameter (MVD). It is defined as the median value of the drop diameter in the drop size distribution, averaged by the liquid water content. Particle concentration is a key parameter in determining coupling between phases in particle laden flows. If the dispersed phase does not alter the dynamics of the continuous fluid the flow is said to be one-way coupled; otherwise it is two-way coupled.

Given the low volume fractions of droplets in clouds, the particle laden flow encountered in the ice-accretion problem can be described as a one-way coupled flow. Therefore, the carrier flow equations can be solved independently from the particles, using an e.g. Eulerian formulation of the flow equations, and the particle trajectories can be later solved as a post processing of the continuous flow solution, using either an Eulerian or a Lagrangian description of the particle dynamics. In this work, the compressible RANS equations governing the carrier fluid are solved in the Eulerian frame using the open-source solver SU2 [8].

The Discrete Parcel Method (DPM) is used to compute the particulate phase. Each computational particle, or parcel, represents a set of N neighbouring droplets having an average velocity $\mathbf{u}_p$ and average diameter d_p. Each parcel trajectory is solved in a Lagrangian frame by integrating the following

$$\begin{cases} \frac{d\mathbf{x}_p}{dt} = \mathbf{u}_p \\ m_p \frac{d\mathbf{u}_p}{dt} = \frac{\pi}{8} \mu d_p \mathrm{Re_p} \left(\mathbf{u} - \mathbf{u}_p\right) C_D + V_p \mathbf{g} \left(\rho_p - \rho\right) \end{cases} \tag{1}$$

where $\mu\, \rho\, \mathbf{u}$ are the gas viscosity, density and velocity at the particle position, V_p is the volume of the average particle in the parcel and C_D its drag coefficient and where only forces acting on the particle are gravity and drag forces since the droplet density is much larger than the gas density ($\rho_p >> \rho$) [9] . $\mathrm{Re_p}$ is the Reynolds number computed using the relative droplet-flow velocity,

$$\mathrm{Re_p} = \frac{\rho \left(\mathbf{u} - \mathbf{u}_p\right) d_p}{\mu} \tag{2}$$

The model used for the drag coefficient takes into account the deformations that can occur for lager droplets. For a spherical particle the model by Morrison [10] is linked to that reported by Clift, Grace, and Weber [11] at $\mathrm{Re_p} = 10^6$ to best fit experimental data:

$$C_D = \begin{cases} \frac{24}{\mathrm{Re_p}} + 2.6 \frac{\frac{\mathrm{Re_p}}{5}}{1+\left(\frac{\mathrm{Re_p}}{5}\right)^{1.52}} + 0.411 \frac{\left(\frac{\mathrm{Re_p}}{263000}\right)^{-7.94}}{1+\left(\frac{\mathrm{Re_p}}{263000}\right)^{-8}} + 0.25 \frac{\frac{\mathrm{Re_p}}{10^6}}{1+\frac{\mathrm{Re_p}}{10^6}} & \mathrm{Re_p} \leq 10^6 \\ 0.19 - \frac{8 \cdot 10^4}{\mathrm{Re_p}} + \delta & \mathrm{Re_p} > 10^6 \end{cases} \tag{3}$$

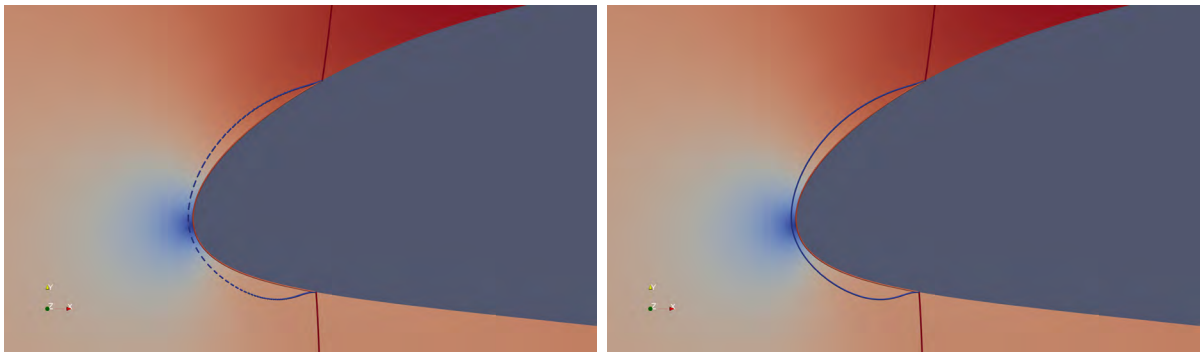

Figure 1: Behaviour of droplet splashing from a discontinuous boundary (left) and from locally splined boundary (right).

Droplet deformation can be modelled as a shape change from a sphere to an oblate disk; the parameter governing this transformation is the Weber number,

$$\mathrm{We} = \frac{\rho_p d_p \left(\mathbf{u}_p \cdot \mathbf{n}\right)^2}{\sigma} \tag{4}$$

where $\mathbf{u}_p$ is the droplet velocity at the moment of impact and $\hat{\mathbf{n}}$ is the surface normal at the impingement point, which is a measure of the external forces acting on a droplet relative to its surface tension. The value for the droplet eccentricity is given as a function of the Weber number as $f = \left(1 + 0.07\sqrt{\mathrm{We_b}}\right)^{-6}$ were f is the spheroid eccentricity [12]. The deforming droplet drag coefficient can be obtained by a weighted average of that of a rigid sphere and that of a disk:

$$C_D = \begin{cases} (1-f)C_{D_{\mathrm{Sphere}}} + fC_{D_{\mathrm{Disk}}} & We_b \leq 12 \\ C_{D_{\mathrm{Disk}}} & We_b > 12 \end{cases} \tag{5}$$

where the disk drag coefficient is reported in [11].

Modelling of droplet-wall interactions is of paramount importance to compute the amount of water collected at the wall boundaries. The impingement of a droplet against a solid surface can result in different behaviours, whose occurrence depends on a large set of parameters, such as the properties of the droplet (size, density, viscosity, surface tension), its impact velocity and direction relative to the surface, in addition to the properties of the surface, including the temperature. Bai and Gosman [13] identify four different mechanisms of droplet-wall interactions that can happen in the in-flight icing framework: stick (complete adhesion and deposition), rebound (complete bouncing), spread (on the surface) and splash (partial deposition with formation of secondary droplet). In order to simulate all possible interactions, semi-empirical models are used to discriminate between the different regimes. In our implementation, the rebound mechanism is not included and bouncing is considered as a special case of splashing, when only one secondary droplet

is formed and no water is left on the wall. This choice is consistent with the scientific literature concerning the wall interaction problem, as many authors [14] identify only two results of the impingement: deposition and splashing. The model adopted is based on the observations made by Cossali [15] and Trontin and Villedieu [16] and defines two different conditions for which a splash/rebound can happen. The first is based on the amount of normal impact energy and is expressed as a lower limit for what is called the Cossali parameter. A second splashing regime is observed at low incidence angles, defined as the complimentary of the angle between the droplet velocity and the surface normal. At low incidence, the normal impact velocity is not high enough to activate the splashing condition; nonetheless splashing can still be observed. Indeed, in this case the droplet does not spread much on the surface and a part of the particle results not to be attached to the wall during the impact. Hence, the tangential kinetic energy is not efficiently dissipated by the viscous forces and the liquid droplet may partially or completely bounce off the wall.

The splashing mass is divided in a number N_s of secondary droplet fragments proportional to the square of the impact velocity [17]. The splashing droplets are re-emitted inside the computational domain by computing the trajectory of n_s new parcels, each representing a total of NN_s/n_s particles, where N is the number of droplets in the impinging parcel. A value for n_p of at least 10 was found to give a good compromise between accuracy and computational cost. The splashing parcel velocities are sampled from a normal probability density function.[17]. As the wall interaction model presented depends on the boundary normal at the impingement point, some unwanted behaviour was noticed for the particles splashing against a wall. Due to the discrete description of the boundary, the front of the splashing droplets presented a dashed-like distribution (Figure 1). A solution to this problem was obtained by using a spline to locally represent the boundary. When computing the interaction model, the normal used is that of the splined boundary, which allows to obtain a more realistic behaviour for the splashing droplets.

To compute the particle trajectories, a computer code was implemented to integrate the equation of motion (1) for each particle in the simulation. The equations (1) are written in terms of quantities defined in the gas phase and therefore the carrier flow solution needs to be interpolated at the particle position. To do so, at each particle time step, each droplet must be mapped to the cell of the mesh containing the particle position, as described in Section 3. The knowledge of the particle owner cells is then used to interpolate the Eulerian solution at the cell nodes to the Lagrangian position of the particle, which is used to compute the forces acting on the droplet. A fifth order explicit Runge-Kutta scheme [18] is then used to find the position and velocity at the next time step. The embedded fourth order formula is used to compute the integration truncation error and adjust the time step accordingly. The process is repeated until all particles have left the computational domain or the next Eulerian time step is reached. In the latter case, the new updated carrier fluid solution is loaded and the process is repeated until the final simulation time is reached. The unsteadiness of the gas phase is treated as step-wise

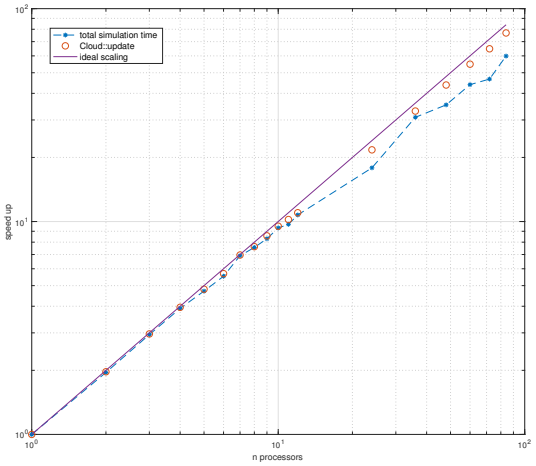

Figure 2: Parallel scaling for a simulation of one million droplets.

constant in time, so that inside every Eulerian time step the cloud of droplet is coupled to a steady carrier flow solution.

A parallel algorithm was implemented using the MPI standard. Each process simulates a sub-cloud of droplets. Due to the nature of one-way coupled particle laden flows, communication and synchronisation between processors is reduced to a minimum and each sub-cloud can evolve independently. Processes communicate only at the end of the simulation to compute the mass impinged on the walls, or to output the cloud to file. To assess code performance and scalability, a simulation on a cloud of one million particles was performed on an increasingly number of processor. A cluster of 16 nodes, each node containing two Intel Xeon X5650 2.67 GHz (6 core units, totalling 192 cores) was used. In Figure 2 the speed up factor for the total simulation time and for the mapping, interpolation, and integration blocks is shown. The latter group represent the part of the code that can be executed in parallel; as such it presents an almost linear scaling and at the maximum number of processors used (84) the speed up factor is just 8% short of the ideal scaling. The full simulation time instead includes operations that execute in serial and therefore cannot benefit from the parallel implementation. The relative weight of these overhead costs increases as the number of processors is increased, thus slowing down the scaling. Nevertheless the speed up factor kept increasing for the number of processor tested; the total simulation time on 84 processors resulted in a speed up factor 28% lower than that of the ideal scaling.

3 Tracking algorithm

The mapping of each particle position to a given cell is at the center of a particle tracking code. It's needed to interpolate the Eulerian solution at the particle position,

and to detect if a droplet has left the domain or impacted a boundary. A known vicinity algorithm [19] is used in the code to perform the mapping. Known vicinity algorithms make use of the knowledge of the particle owner cell before being displaced to find the new owner at the next position. Starting from the initial position in the mesh the particle is searched following a certain path that is defined by the algorithm. Each known vicinity algorithm must therefore define two different operations: the Particle In Cell test (PIC), which determines whether the particle is inside the current cell, and the definition of the search direction. The literature reports many examples of both the PIC and search direction being computed through the use of the cells shape functions [19] or via the definition of geometric tests [20]. This work follows the latter approach and defines an extended PIC that also accounts for the search direction. The algorithm follows the particle along its trajectory, computing the intersections with all faces of all cells it crosses during the time step. The extended PIC takes as input two positions $\mathbf{p}_0$ and $\mathbf{p}_1$ and the cell to check. One of the two positions in input ($\mathbf{p}_1$) is always the particle next position $\mathbf{x}_p^{n+1}$ that hast to be tracked, while the other can vary. A ray $\mathbf{r}$ stemming from $\mathbf{p}_0$ and directed towards $\mathbf{p}_1$ is created, and the distance d between the two points computed. Then the ray is intersected with the faces of the cell until a valid intersection is found and the intersection distance d_I is computed. If this distance is greater than or equal to d then the particle is in the current cell. This extended test is equivalent to the original PIC if point $\mathbf{p}_0$ is taken as the cell centroid. If instead it is set equal to the position of the particle at the earlier time step $\mathbf{x}_p^n$ then the ray represents the particle trajectory. When $d_I < d$ the particle is not in the current cell. In this case the PIC returns the face f intersected and the search direction is therefore defined, by moving to the other cell sharing the same face. This way impact with the boundaries are easily detected from the knowledge of boundary faces. The extended PIC the algorithm mapping position $\mathbf{x}_p^{n+1}$ to cell $\mathcal{C}^{n+1}$ does the following:

1. mark the current cell $\mathcal{C}^n$ as the cell to be checked $\mathcal{C}^k$
2. compute PIC $\left(\mathbf{x}_p^n, \mathbf{x}_p^{n+1}, \mathcal{C}^k\right)$. If the particle is in the current cell assign $\mathcal{C}^{n+1} = \mathcal{C}^k$ and exit the procedure.
3. get the exit face form the output of the PIC and assign the neighbouring element to $\mathcal{C}^k$. Return to 2

The computation of a valid intersection depends on the type of faces considered. In two dimensions faces are always segments as the elements considered are linear. In 3D we consider only the most common elements used by finite volume solvers: tetrahedra, pyramids, wedges, hexahedra. All this elements have planar (triangular) faces or quadrilateral faces that in this work are parametrized as non-planar bilinear patches. For planar 2D or 3D faces, the distance between $\mathbf{p}_0$ and the intersection point is computed as

$$d_I = \frac{(\mathbf{p}_f - \mathbf{p}_0) \cdot \mathbf{n}}{\mathbf{t} \cdot \mathbf{n}}$$

where $\mathbf{p}_f$ is a point on the face, $\mathbf{n}$ the outward pointing normal, and $\mathbf{t}$ is the direction of the ray $(\mathbf{p}_1 - \mathbf{p}_0)$. As noted in [20], the above distance is computed only if the sign of the denominator is positive thus saving some computations. The intersection is valid if the point belongs to the face and not just to the plane or line on which it lays. In two dimensions for a face defined between nodes $\mathbf{f}_0$ and $\mathbf{f}_1$, the intersection is valid if $(\mathbf{r}_s + d_I\mathbf{t} - \mathbf{f}_0) \cdot (\mathbf{r}_s + d_I\mathbf{t} - \mathbf{f}_1) \leq 0$, where $\mathbf{r}_s$ and $\mathbf{t}$ are the starting point and the tangent of the ray representing the particle trajectory, and d_I the intersection distance. For triangular faces, the intersection is valid if, for each edge of the face, the scalar product between the normal of the edge and the vector from the edge midpoint to the intersection point is negative or equal to zero. The normal of the edge $\mathbf{n}$ is defined in the plane of the face and points outward. The condition therefore reads:

$$\left(\mathbf{r}_s + d_I\mathbf{t} - \frac{\mathbf{e}_0^i + \mathbf{e}_1^i}{2}\right) \cdot \mathbf{n}_i \leq 0 \qquad i = 1:3$$

where $\mathbf{e}_0^i$ and $\mathbf{e}_1^i$ are the coordinates of the vertices of the i edge of the face. A bilinear patch can be parametrized as $\mathbf{r}(u,v) = uv\mathbf{a} + u\mathbf{b} + v\mathbf{c} + \mathbf{d} \qquad u, v \in [0,1]$ where $\mathbf{a} = \mathbf{r}_{00} - \mathbf{r}_{10} + \mathbf{r}_{11} - \mathbf{r}_{01}$, $\mathbf{b} = \mathbf{r}_{10} - \mathbf{r}_{00}$, $\mathbf{c} = \mathbf{r}_{01} - \mathbf{r}_{00}$, $\mathbf{d} = \mathbf{r}_{00}$ and $\mathbf{r}_{ij}$ are the vertices of the non-planar face. By substituting the ray $\mathbf{r}(u,v)$, the intersection distance is obtained as

$$d_I = (uv\mathbf{a} + u\mathbf{b} + v\mathbf{c} + \mathbf{d} - \mathbf{r}_s) \cdot \mathbf{t} \tag{6}$$

Multiplying the above equation by $\mathbf{t}$ and subtracting the result, one obtains $uv\hat{\mathbf{a}} + u\hat{\mathbf{b}} + v\hat{\mathbf{c}} + \hat{\mathbf{d}} - \hat{\mathbf{r}}_s = \mathbf{0}$ where $\hat{(\cdot)}$ represents the vector component normal to the direction of the ray, namely $\hat{(\cdot)} = (\cdot) - (\cdot)\cdot\mathbf{t}$. A two by two square system is finally obtained by subtracting the third row to the other two, which can be solved for u and v by substitution. The resulting equation is a second order one and can yield up to two solutions. For a solution to be valid, both u and v must lie in the interval $[0,1]$. If two valid solutions are found, and the further intersection distance is smaller than d, then the face can be discarded, as the trajectory would exit and then re-enter the same cell.

The above algorithm provides a robust way of tracking particles in 3D unstructured meshes. It is based on the existing literature on geometry based known vicinity algorithms and proposes to fix some shortcomings that were highlighted in earlier studies. The checking of a valid interface allows to track particles even in concave cells, whereas the sole computation of the intersection distance, although less computationally expensive, led to algorithms working only on convex elements [20]. The use of the bilinear patch to represent non planar faces allows a robust treatment of quadrilateral faces. In geometry based algorithms particles were being lost due to the use of the planar PIC equations to treat slightly non-planar faces. Many authors proposed to split those faces in two triangles in order to fix the problem. It was noted [21] that spitting non-planar faces can result in gaps being created if the splitting on one side of the face is not the same as that used on the other, and particles can be lost in that space. By representing the

actual non-planar faces none of the above problems were found. Another shortcoming of geometry based search algorithms is due to the finite precision of floating point numbers. When comparing two values, the errors arising from the finite precision of floating point numbers can yield the wrong result if the numbers are close enough. This can lead the algorithm to mistake a non valid intersection with a valid one or the other way around. Some authors [20] propose to use a tolerance when comparing two floating point values to account for these errors. In this work a different approach is used. The computations that are prone to floating point errors (i.e. selecting the trajectory exit face) are encapsulated in a routine (the extended PIC that will be presented later). A fatal floating point error is detected if, at the successive tracking step, no intersections can be computed. In such a case the computation is repeated with increased precision until the exact solution is recovered. The multiple precision floating point type provided by the MPFR library [22] is used for such computations.

4 Code assessment

The code developed was used to replicate the collection efficiencies obtained during an experimental campaign conducted by Papadakis et al [3] and aimed at providing the first extensive impingement database for Supercooled Large Droplets (SLD) as well as for smaller droplets. The authors of [3] measured the collection efficiency on a NACA 23012 airfoil. Catch efficiency was obtained for the clean geometry as well as for the iced geometries resulting from the LEWICE computation of 5, 10, 15, 22.5 and 45 minutes glaze ice shapes. The tests were conducted at the NASA Glenn Icing Research Tunnel (IRT) and the results obtained were compared with those computed numerically with the LEWICE code. The experiments were conducted at a total air temperature of 283 - 294 K and an air speed of 78 m/s, corresponding to a Reynolds number of approximately 5.25 million per meter. The profile used had a chord of 91.44 cm; collection efficiencies were measured for MVDs of 20, 52, 111, 154, and 236 micrometers.

The carrier fluid solution was obtained by solving the RANS equations with the CFD code SU2. The Spalart-Allmaras model was used to model the turbulent viscosity. The values for the collection efficiency were obtained by simulating a cloud of 20 000 droplets impinging the airfoil and their diameters were set equal to the MVDs of droplet distribution used in the experimental campaign. The wall impingement model of LEWICE [5] was implemented in the code in order to compare the results of the one presented earlier. Figure 3 shows the computed catch efficiency for two values of MVD (52 and 236 μm). The computation with the LEWICE model yields a good prediction of both impingement limits and maximum catch efficiency while the original model, although it retrieves the same values near the limits, strongly underestimates the efficiency near the nose of the profile. This is possibly due to the mass loss coefficient of the model being a function of only the Cossali parameter and not the incidence angle. In the LEWICE model splashing is inhibited for near normal impact thus explaining the large difference in the prediction between the two models. The same computation was performed on the

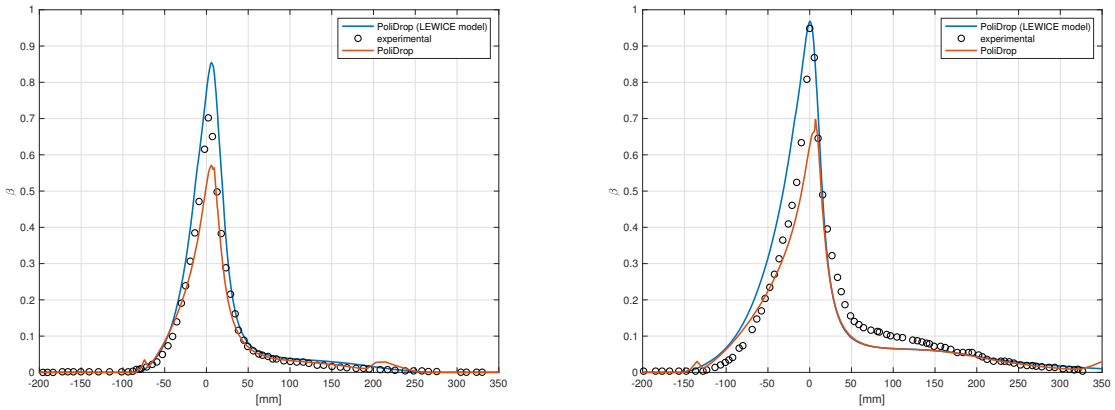

Figure 3: Collection efficiency on the clean naca 23012. Left MVD 52 μm. Right MVD 236 μm.

iced geometries reported in [3]. In Figure 4 are reported the results for the same values of MVD of the clean profile. As in the clean case, near the impingement limits both models yield accurate results for the collection efficiency. On the iced part, the LEWICE model, due to the correlation used for the mass loss coefficient, overestimates the catch efficiency by as much as 50% for the 52μm case. Instead the model presented earlier yields results much closer to the experimental data.

5 Conclusions

A particle tracking algorithm was presented to solve the particle trajectories of the supercooled cloud droplets for in-flight ice accretion simulations. A one-way coupling description of the particle laden flow is assumed because of the relatively small concentration of droplets within the clouds. State-of-the-art models for the aerodynamic forces acting on the particles are implemented. Considered wall interaction mechanics mechanics include sticking, splashing, rebounding and spreading particles and benefits from a detailed description of the boundary geometry. A robust particle tracking algorithm is implemented, which make use of arbitrary precision algebra to determine the intersection of the particle trajectories with the grid edges and nodes. Numerical simulations of a typical ice accretion problem confirmed the validity of the present approach.

REFERENCES

[1] T. Cebeci, F. Kafyeke, *Aircraft Icing*, Annual Review of Fluid Mechanics, Vol. 35, 2003.

[2] S. Elghobashi, *On predicting Particle-Laden Turbulent Flows*, Applied Scientific Research, Vol. 52, 1994.

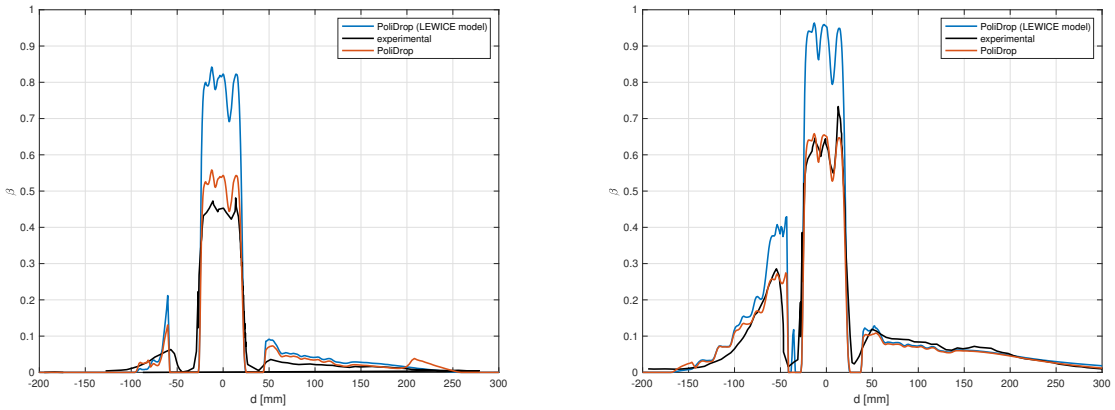

Figure 4: Values for the collection efficiency on a naca 23012 profile after an exposure time of 10 minutes. Left MVD 52 μm. Right MVD 236 μm.

[3] M. Papadakis et al., *Water Impingement Experiments on a NACA 23012 Airfoil with Simulated Glaze Ice Shapes*, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 2004, AIAA 2004-0565.

[4] A. Hamed , K. Das , D. Basu, *Numerical simulations of ice droplet trajectories and collection efficiency on aero- engine rotating machinery*, 43$^{\text{rd}}$AIAA Aerospace Sciences Meeting & Exhibit, January 10-13, 2005, Reno, NV.

[5] W.B. Wright, *User Manual for LEWICE, version 3.2*, NASA Technical Report, 2008.

[6] C.N. Aliaga, M.S. Aube, G.S. Baruzzi, W.G. Habashi,, *FENSAP-ICE Unsteady: Unified In-Flight Icing Simulation Methodology for Aircraft, Rotorcraft, and Jet Engines*, Journal of Aircraft, Vol. 48, No. 1, 2011.

[7] G. Gori et al., *PoliMIce: A simulation framework for three-dimensional ice accretion*, Applied Mathematics and Computation, Vol. 267, 2015.

[8] F. Palacios et al., *Stanford University Unstructured (SU2): Open-source Analysis and Design Technology for Turbulent Flows*, 52nd Aerospace Sciences Meeting, AIAA 2014-0243.

[9] E. Loth, *Numerical Approaches for Motion of Dispersed Particles, Droplets and Bubbles*, Progress in Energy and Combustion Science, Vol. 26, 2000.

[10] A. Morrison, *An Introduction to Fluid Mechanics*, Cambridge University Press, New York, 2013.

[11] R. Clift, J.R. Grace, and M.E. Weber, *Bubbles, drops, and particles*, Courier Corporation, 2005.

[12] R. Honsek, W.G. Habashi, *Eulerian modeling of droplet impingement in the SLD regime of aircraft icing*, AIAA Paper, No. 465, 2006.

[13] C. Bai and A.D. Gosman, *Development of Methodology for Spray Impingement Simulation*, SAE Technical Report 950283, 1995.

[14] A.L. Yarin, *Drop impact dynamics: splashing, spreading, receding, bouncing. . .*, Annu. Rev. Fluid Mech., Vol. 38, 2006.

[15] G. Cossali, A. Coghe, M. Marengo, *The impact of a single drop on a wetted solid surface*, Experiments in fluids, Vol. 22, No. 6, 1997.

[16] P. Trontin, P. Villedieu, *Revisited Model for Supercooled Large Droplet Impact onto a Solid Surface*, Journal of Aircraft, Vol. 54, No. 3, 2017.

[17] M.F. Trujillo, W.S. Mathews, C.F. Lee J.E. Peters, *Modelling and experiment of impingement and atomization of a liquid spray on a wall*, Int J Engine Research, Vol. 1, No. 1, 2000.

[18] J.R. Dormand, P.J. Prince, *A Family of Embedded Runge Kutta Formulae*, Journal of Computational and Applied Mathematics, Vol. 6, No. 1, 1980.

[19] R. Lohner, J. Ambrosiano, *A Vectorized Particle Tracer for Unstructured Grids*, Journal of Computational Physics, Vol. 91, 1990.

[20] A. Haselbacher, F.M. Najjar, J.P. Ferry, *An Efficient and Robust Particle-Location Algorithm for Unstructured Grids*, Journal of Computational Physics, Vol. 225, 2007.

[21] D.C. Cohen Stuart, C.R. Kleijn, S. Kenjeres, *An Efficient and Robust Method for Lagrangian Magnetic Particle Tracking in Fluid Flow Simulations on Unstructured Grids*, Computer and Fluids, Vol. 40, 2011.

[22] L. Fousse, G. Hanrot, V. Lefevre, P. Pelissier, P. Zimmermann, *MPFR: A Multiple-Precision Binary Floating-Point Library With Correct Rounding*, ACM Transactions on Mathematical Software, Vol. 33, No. 2, 2007.