Workflow for predictor–corrector simulations of in-flight ice accretion, with applications on swept wings

Emmanuel Radenac, Ghislain Blanchard, Thomas Renaud, Quentin Duchayne

Workflow for Predictor-Corrector simulations of in-flight ice-accretion, with applications on swept wings

Emmanuel Radenac[1*], Ghislain Blanchard[1†], Thomas Renaud[2†] and Quentin Duchayne[1†]

[1*]ONERA / DMPE, Université de Toulouse, F-31055 Toulouse - France.
[2]DAAA, ONERA, Université Paris Saclay, F-92190, Meudon - France.

*Corresponding author(s). E-mail(s): emmanuel.radenac@onera.fr;
Contributing authors: ghislain.blanchard@onera.fr; thomas.renaud@onera.fr; quentin.duchayne@onera.fr;
†These authors contributed equally to this work.

Abstract

A workflow for the numerical prediction of in-flight ice accretion on 3D structures is presented. The method is based on the predictor-corrector approach, which has been so far mainly developed in 2D codes and assessed in straight-wing test-cases. The adaptations made for the 3D implementation are thus described. Among other developments, a remeshing technique based on the Dragon method is presented. The new methods are verified against reference numerical data. The whole workflow is validated by using several test-cases for which experimental measurements of ice shapes are available. Both straight and swept wings are therefore investigated. The numerical results are encouraging for low-sweep angles. Some recommendations are made for improving the results, including the need for new ice-density models. The predictor-corrector method is poorly adapted for cases exhibiting large scallop-like structures, which tends to be reinforced by high sweep angles. In this article, the test cases with highest sweep angle are effectively calculated less accurately. Costly multi-step ice-accretion simulations should be preferred for such conditions, unless a special ice bulk density model is developed.

Keywords: icing, CFD, predictor-corrector, Dragon method

1 Introduction

In-flight icing is a significant threat to aircraft. This hazard is faced when the aircraft flies through clouds of supercooled water droplets. In order to mitigate this risk, manufacturers carry out flight and wind-tunnel tests and use numerical simulations. This set of techniques is useful both for the design of protection systems and for certification. Regarding numerical simulation tools, the implementation of 3D softwares is a strong need, as shown by the configurations investigated in the AIAA 1st ice prediction workshop (IPW1) [1].

For both design and certification purposes, a large number of simulations must be performed to cover the full range of icing conditions encountered by the aircraft. Therefore, the computational time is a key issue from a practical point of view. This question is even more important for the 3D codes discussed in this article than for 2D codes. The exposure time t_{exp} to icing conditions being very long compared to the characteristic time of the aerodynamic flow, a sequential approach is generally used [2–4] (see figure 1). In the latter, the steady-state aerodynamic flow, the supercooled-water-droplet trajectories and the flow of the water collected on the exposed surfaces are solved successively. The main result of this sequence is an ice-accretion rate at each point of the exposed surfaces. An ice thickness is deduced, corresponding to a physical accretion time Δt. A very basic approach, rather fast but not very accurate, is to perform this loop only once and deduce the ice thickness by using $\Delta t = t_{exp}$. This so-called predictor approach does not take into account the fact that ice growth affects the aerodynamic flow and the collection of water droplets by the surface. To do so, the multi-step approach [4] splits the exposure time into N steps. N successive loops are then performed, allowing the ice to grow gradually over physical accretion times $\Delta t = t_{exp}/N$. This approach is more realistic, but it can be costly for N large. The issue of the robustness of each step of the loop also becomes even more important for N large. The approach chosen for this article is a predictor-corrector approach. The latter has been developed and widely used in 2D [4]. It consists in reducing the total number of loops to $N = 2$, the first one being a simple predictor approach ($\Delta t = t_{exp}$), the second one being a correction step. We should indeed point out that, for the same computational cost, a predictor-corrector simulation generally gives better agreement with experiment than a multi-step simulation with $N = 2$ steps. Figure 2 shows this for two 2D cases presented in section 6, and calculated here with IGLOO2D [4]. For the (cold) rime-ice case RUN405, the multi-step calculation provides an ice shape that is a little too similar to the predictor shape. For the (warmer) glaze-ice run 06-27-91/1, the ice horn in the upper part extends too far downstream with the 2-step approach. In the lower

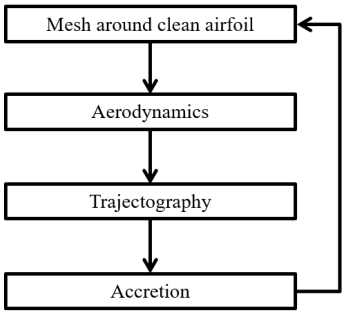


Fig. 1: Sequential approach for ice accretion simulations

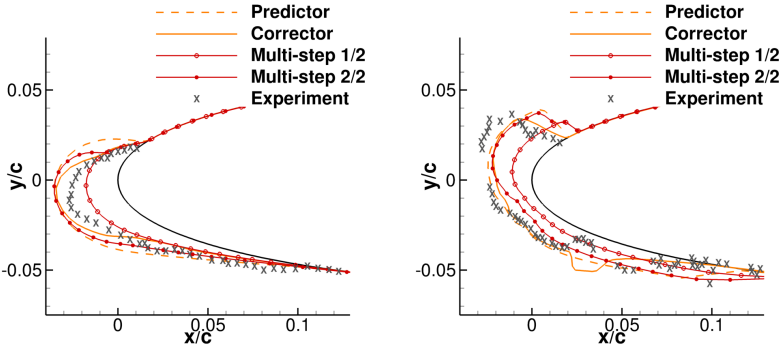


Fig. 2: Comparison between the predictor-corrector method and a multi-step method with $N = 2$ steps. RUN405 case on the left. Run 06-27-91/1 on the right. Simulations with IGLOO2D. Both predicted and corrected ice shapes are displayed for the predictor-corrector approach. The ice shapes are also displayed for the 2 steps of the multi-step method.

part, the ice shape provided by this method is too thick in the most downstream part. The predictor-corrector calculation significantly improves the ice shape in both respects. The reason for this is attributed to the fact that the shading effect progressively exerted by the growing ice shape on droplet trajectories is better modelled (section 5.2 will show that the predictor-corrector approach allows for additional sub-iteration, whereas the 2-step method only models ice growth in two steps with no sub-stepping).

For multi-step approaches as for predictor-corrector approaches, it is necessary to update the fluid volume at each additional loop because of the growth of the ice shape. The deformation of the exposed surface can be done by a Lagrangian displacement of the surface mesh nodes [3–10], or by a level-set resolution [6, 10–12]. The update of the fluid volume can be done by remeshing [3–6, 9, 10], by mesh deformation [7, 8] or by an IBM method [11, 13, 14].

In this article, both Lagrangian deformation and level-set resolution will be discussed, while the remeshing approach will be investigated.

This paper deals with a 3D framework for taking into account the retroaction of ice growth on the flow by predictor-corrector approach. The IGLOO3D icing suite is presented in section 2. The three solvers involved will be discussed in sections 2.2 (aerodynamic flow), 2.3 (droplet trajectories) and 2.4 (ice accretion). More precisely, the aerodynamic and trajectory solvers will be treated quite briefly while the accretion solver MESSINGER3D will be much more thoroughly discussed. The deformation of the exposed surface and the meshing tool based on the Dragon method are discussed in sections 3 and 4, respectively. The predictor-corrector method is presented in section 5. The method is assessed first in cases of 2D airfoils (section 6), then for swept wings (section 7). For both types of cases, the icing conditions used will be those of Appendix C [15]. The considered cloud is thus composed of supercooled water droplets small enough to be locally deposited on the exposed surfaces. It is then useless to account for phenomena such as splashing (or sticking and erosion for the case of ice crystals), which add a degree of modeling error that is not desirable for evaluating the method. For the same reason, surfaces not protected against ice are investigated.

2 Presentation of the numerical tools

2.1 Ice-accretion computational strategy

The sequential approach of figure 1 is used in the ice accretion suite IGLOO3D. This tool, which was designed to be modular, allows different solvers to communicate with each other (through CGNS file inputs and outputs [16]) in order to achieve all the steps necessary for ice-accretion simulations. Several types of mesh can be used. For this article, unstructured meshes are mainly employed. In particular, the meshes around the clean geometries of the IPW1 cases are the ones that have been provided by the IPW1 Committee. The Dragon method discussed later on was also used to generate several grids.

As shown in figure 3, the CEDRE solvers, well suited for unstructured meshes, are used for the airflow and droplet-trajectory simulations (sections 2.2 and 2.3). Other solvers could also be used as it was presented in previous articles [17, 18] (For example, the elsA solver for the airflow computation). The ice accretion is solved by MESSINGER3D, which is based on Messinger's approach [19] for the thermodynamic balance applied to the water deposited on the exposed surfaces. This method is one of the traditional approaches used for ice-accretion simulation, while another widespread technique is the use of thin-film flow simulation methods (SWIM method – Shallow Water Icing Model – [20–22], solution of the Saint-Venant equations [23], method of Chauvin *et al.* [24] considering the possible presence of three layers locally: runback water, ice and liquid water trapped under the ice). More details will be given in section 2.4, where it will become apparent that an iterative method is employed, quite similar in this respect to the work of Zhu *et al.* [25]. According

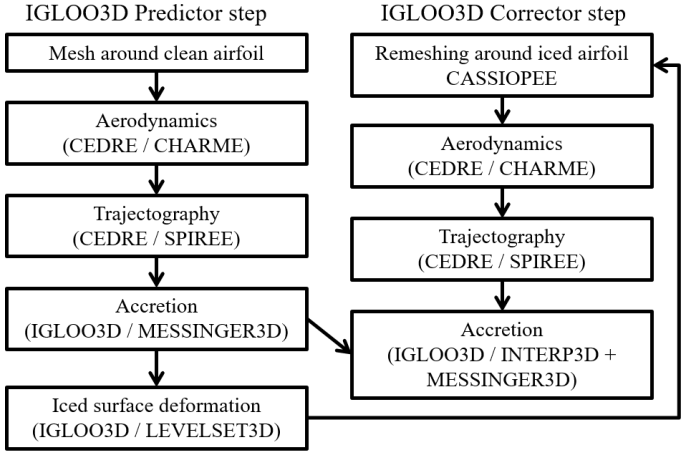


Fig. 3: IGLOO3D computational loop

to the articles of Chauvin *et al.* [24] and Lavoie *et al.* [22], very similar results would be expected between our Messinger's approach and a method based on the resolution of thin-film flows for the genuine ice-accretion cases addressed here. MESSINGER3D produces an ice-deposition mass-rate, from which an ice shape is inferred.

Figure 3 also shows that the "predictor-corrector" approach is the strategy chosen in this article in order to deal with the ice growth and its retroaction on the airflow and droplet trajectories. This approach consists in running:

1. a first "predictor" step, based on the sequential call to the three codes. The first ice-deposit deformation is thus directly computed from the ice-deposition-mass rate produced by the code and the time t_{exp} of the whole accretion process;
2. a second step for which the airflow and droplet trajectories are computed around the predicted ice shape, in order to correct the computation of the ice-deposition mass rate. This correction method is explained in section 5. For this approach, it is necessary to generate a mesh around the predicted ice shape. This is done by first modifying the shape of the iced surface as outlined in section 3 and then remeshing the fluid volume as described in section 4.

2.2 Aerodynamic solution

CEDRE is a multi-physics numerical tool, including unstructured solvers for several physical problems (airflow simulations, droplet trajectories, conductive heat transfer, radiative heat transfer,...). [26]

Among these solvers, the airflow solver CHARME was used [26, 27]. CHARME is based on the finite volume method on unstructured grids. Various numerical options are available for the space and time discretization and for the turbulence modeling for instance. In the present study, the k-ω SST model was used for turbulence modeling, along with the model developed by Aupoix [28] for modeling rough walls whenever necessary. For this model, a constant wall temperature is imposed and the equivalent sand-grain roughness height k_s was set to a constant value:

$$k_s = L/1000, \tag{1}$$

where L is a reference length. For 2D airfoils, it is common practice in ONERA's codes to set L equal to the chord length. For 3D objects, it is less obvious to link the roughness height to a unique reference length, and the validity of the empiric correlation (1) is questionable. Nevertheless, this relation was used because swept airfoils are investigated and it is still possible to define an equivalent chord length.

Regarding the numerical schemes, a second-order space discretization was used based on the HLLC scheme [29] and a MUSCL reconstruction (with a hybrid limiter [30]). For the time discretization, a basic first-order Euler implicit scheme was used with local time-stepping, since the steady-state solution is sought. The CFL numbers employed reached around 10 to 50 according to the test-case investigated. The large linear system arising from the implicit time scheme was solved with a GMRES method.

2.3 Computation of the droplet trajectories

The Eulerian droplet-trajectory solver SPIREE [31–33] was employed to compute the droplet trajectories. SPIREE is also based on the finite volume method on unstructured grids. The Schiller and Naumann model was used for the drag modeling. Full deposition was assumed on walls (only Appendix C test-cases are investigated). As in CHARME, a first-order Euler implicit scheme was used with local time-stepping, and the GMRES method was used. A second-order space discretization was also used, based on a Godunov-like scheme [31] and a MUSCL reconstruction (with a hybrid limiter [30]).

2.4 Messinger balance

The Messinger balance solved in MESSINGER3D is very similar to the one of IGLOO2D [4]. It consists of a mass and energy balance for water on the icing surface. Thus it accounts for the deposited droplets (the mass rate is provided by the droplet trajectory solver), liquid-water runback on the surface, phase changes (evaporation, sublimation, solidification), convective heat transfer between the water and the air flow. As mentioned earlier, only Appendix C is addressed in this article although some features are available for Appendices O (Supercooled Large Droplets) and P (ice crystals). Therefore, no distinction

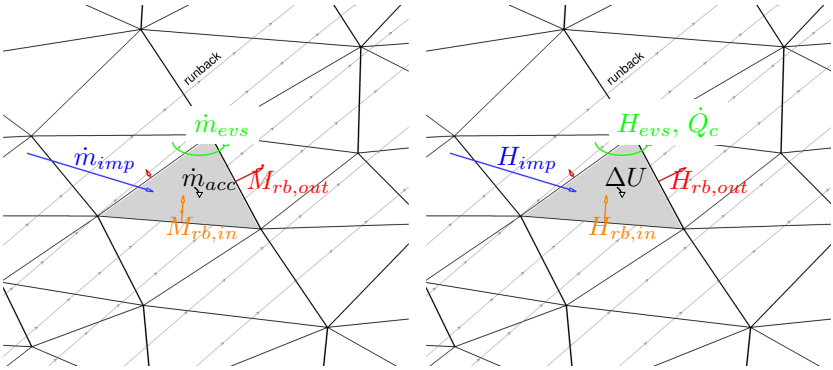


Fig. 4: Mass and energy balance in Messinger approach

is made between impacting mass and deposited mass. The main outputs are the icing regime (rime, glaze or running-wet regime), the ice growth rate $\dot{m}_{acc}$, the wall liquid mass fraction f_L and the water (or wall) temperature T_w.

2.4.1 Mass and energy balance

The mass and energy balance are sketched in figure 4 for a given "control volume" (actually a surface element). Liquid water is incoming through runback and impingement and outgoing through runback, evaporation and solidification. Condensation or melting could occur instead of evaporation and solidification, respectively, for specific conditions, which will not be considered here for the sake of simplicity (although it does not affect the validity of the equations presented here). Regarding the energy balance, runback, droplet impingement, phase changes and convection are accounted for. It is assumed that the wall is adiabatic, which is a fairly good assumption for unheated walls.

It is obvious that the runback direction plays a major role in the balance. In 2D codes, there is no ambiguity about this direction. In three dimensions, it is assumed here that the runback is mainly driven by the skin friction exerted by the air flow. It means that the effects of pressure gradients and gravity are neglected, as well as inertial forces (fixed airfoils are considered). The "2D-like balance" exposed here is thus applied along the friction lines.

Along these lines, the mass and energy balances for the liquid water write, respectively:

$$S\dot{m}_{imp} + \dot{M}_{rb,in} - \dot{M}_{rb,out} - S\dot{m}_{evs} = S\dot{m}_{acc} \tag{2}$$

$$S\dot{m}_{imp}H_{imp} + \dot{M}_{rb,in}H_{rb,in} - \dot{M}_{rb,out}H_{rb,out} - S\dot{m}_{evs}H_{evs} - S\dot{Q}_c = \Delta U \tag{3}$$

where S is the area of the surface element and:

- $\dot{m}_{imp}$ is the impinging surface mass rate (kg.m^{-2}.s^{-1}). H_{imp} is the impinging mass enthalpy (J/kg);
- $\dot{M}_{rb,in}$ and $\dot{M}_{rb,out}$ are the incoming and outgoing runback mass rates (kg.s^{-1}), respectively. $H_{rb,in}$ and $H_{rb,out}$ are the incoming and outgoing runback water mass enthalpy (J/kg). The runback terms are related to each other via fluxes between the cells of the mesh, as shown in section 2.4.2;
- $\dot{m}_{evs}$ is the evaporated or sublimated surface mass rate (kg.m^{-2}.s^{-1}). H_{evs} is the evaporated or sublimated mass enthalpy (J/kg);
- $\dot{Q}_c = h_{tc}\,(T_w - T_r)$ is the convective heat flux (W/m^2), related to the heat transfer coefficient h_{tc}, the recovery temperature T_r and the iced wall temperature T_w;
- $\dot{m}_{acc}$ is the solidified surface mass rate or surface ice growth rate (kg.m^{-2}.s^{-1}). This is a major output of the Messinger code, which then allows to compute the ice thickness. It is assumed that no liquid water is accumulated locally: the surface mass rate corresponding to liquid accumulation is zero;
- ΔU is the variation of the total energy of the water system (ice and liquid water) during the accretion process (W). ΔU is a function of T_w. Since there is no accumulation of water, ΔU is linked to the variation of total energy of the growing ice:
$$\Delta U = S\dot{m}_{acc}c_s\,(T_w - T_m) \tag{4}$$
where c_s is the ice specific heat capacity and T_m is the water melting temperature.

The system of equations (2) and (3) thus depends on three unknowns: the wall temperature T_w, the surface ice growth rate $\dot{m}_{acc}$ and the runback mass rate (more specifically $\dot{M}_{rb,out}$ is chosen). One additional relation is thus required to close the system. This is done by successively considering four mutually exclusive icing regimes, each regime being associated with one equation:

1. Rime ice regime: there is only ice, provided that $T_w < T_m$. It means that no runback occurs and the additional relation is:
$$\dot{M}_{rb,out} = 0 \tag{5}$$

2. Glaze ice regime: there are both liquid and solid water, which requires that $\dot{M}_{rb,out} > 0$, and the additional equation is:
$$T_w = T_m \tag{6}$$

3. Running-wet regime: there is only liquid water (if $T_w > T_m$). No ice accretion occurs and the additional equation is:
$$\dot{m}_{acc} = 0 \tag{7}$$

4. Dry-wall regime: there is no water ($\dot{M}_{rb,out} = 0$), and again $\dot{m}_{acc} = 0$.

Evaporation

The evaporated or sublimated surface mass rate $\dot{m}_{evs}$ required in equations (2) and (3) is given by:

$$\dot{m}_{evs} = -\rho_g h_m \left(Y_v(T_e) - Y_v(T_w) \right) \tag{8}$$

where ρ_g is the gas density, Y_v stands for the steam mass fraction and h_m is the mass transfer coefficient. h_m is derived from the heat transfer coefficient through the Chilton-Colburn analogy [34, 35]:

$$\frac{h_m}{h_{tc}} = \frac{\mathrm{Le}^{-2/3}}{\rho_g c_p} \tag{9}$$

where c_p is the air specific heat capacity at constant pressure and Le is the Lewis number for air-water vapor.

Convective heat transfer

After the airflow simulation, the heat transfer coefficient, which plays a key role in the modeling of glaze ice shapes, is computed thanks to the following formula:

$$h_{tc} = \frac{\phi_{wa}}{T_{wa} - T_r} \tag{10}$$

where ϕ_{wa} is the wall heat flux provided by the airflow solver, T_{wa} is the wall temperature imposed as a boundary condition to the airflow solver. The recovery temperature is calculated via the recovery coefficient $r_{rec} = \mathrm{Pr}^{1/3}$ (in turbulent regime, where $\mathrm{Pr} = 0.7$ is the Prandtl number of air):

$$T_r = T_e \left(1 + r_{rec} \frac{\gamma - 1}{2} M_e^2 \right) \tag{11}$$

where $\gamma = 1.4$ is the heat capacity ratio of air. T_e and M_e are the temperature and the Mach number at the edge of the boundary layer, respectively. M_e is computed as the isentropic Mach number. T_e is reconstructed from the assumed conservation of the total temperature in the flow outside of the boundary layer.

There are other methods available in IGLOO3D to compute h_{tc}, as discussed in reference [17]. The first one is based on two simulations of the airflow with two different wall temperatures. h_{tc} and T_r are derived from the linearization of the heat flux with respect to the wall temperature. This approach is expensive because two Navier-Stokes simulations are required per loop. This method has therefore been left aside. The other methods consist in imitating the integral-boundary-layer models available in IGLOO2D [4], by relating h_{tc}

to the skin friction coefficient C_f:

$$h_{tc} = \rho_g c_p u_e \frac{C_f/2}{\mathrm{Pr}_t + \sqrt{C_f/2}/\left(1.92{k_s}^{+-0.45}\mathrm{Pr}^{-0.8}\right)} \tag{12}$$

where u_e is the airflow velocity at the edge of the boundary layer (related to M_e), $\mathrm{Pr}_t = 0.9$ is the turbulent Prandtl number and ${k_s}^+$ is the equivalent sand-grain roughness height k_s made non-dimensional in wall coordinates. This method is based on a Reynolds analogy modified to take into account the rough state of the iced surface. It is thus possible to use the skin friction coefficient C_f produced by the RANS solver. In order to reproduce the IGLOO2D method as closely as possible, it is also possible to compute:

$$\frac{C_f}{2} = \frac{0.1681}{\ln^2\left(\frac{864\theta}{k_s} + 2.568\right)} \tag{13}$$

where θ is the momentum thickness of the boundary layer simulated by the RANS solver. Unless explicitly specified, the methods based on equation (12) were also discarded for the simulations of this article because they are assumed to be less accurate than the method based on Navier-Stokes computations in 3D (there are especially some 2D assumptions made in this method).

In addition, it was possible to activate a laminar-turbulent transition model for the 2D cases. This model consists in computing the recovery factor and the heat transfer coefficient differently in the laminar area. First, the laminar area is identified as the region in which

$$\frac{k_s u_e}{\nu} \leq 600 \tag{14}$$

where ν is the kinematic viscosity of the airflow. This is a simple version of sharp transition based on Braslow's criterion. Second, in the laminar area, the recovery coefficient is:

$$r_{rec} = \sqrt{\mathrm{Pr}}, \tag{15}$$

while the heat transfer coefficient is computed after Smith and Spalding's formula, as in IGLOO2D [4]. This equation was derived for an incompressible boundary layer over a 2D body with an arbitrarily varying free-stream velocity. It links the heat transfer coefficient to the evolution of the velocity at the edge of the boundary layer along a streamline (s is the wrap distance from the attachment line) [36, 37]:

$$h_{tc} = \rho_g c_p u_e \frac{0.2926\nu}{u_e \mathrm{Pr}} \sqrt{u_e^{2.87} / \int_0^s \nu(s') u_e(s')^{1.87} ds'} \tag{16}$$

It is not straightforward to assess the integral term $\int_0^s \nu(s')u_e(s')^{1.87}ds'$ over 3D surfaces. A partial differential equation (PDE) approach was thus developed to easily make the computation on unstructured surface meshes. The integral term has the following form:

$$\psi = \int_0^s \phi(s')\mathrm{d}s' = \int_0^s \mathrm{d}\psi' \tag{17}$$

where the function to be integrated is:

$$\phi(s) = \nu(s)u_e(s)^{1.87} \tag{18}$$

A pseudo time step $\mathrm{d}t$ is defined to move along the streamline so that:

$$\mathrm{d}t = \kappa \mathrm{d}\psi \tag{19}$$

where κ is a unit scalar whose units are compatible with the fact that $\mathrm{d}t$ is a time step (κ could take any value). This is equivalent to $\mathrm{d}t = \kappa\phi\mathrm{d}s$ after equation (17). The following velocity can thus be defined locally:

$$\vec{V_\psi} = \frac{\mathrm{d}s}{\mathrm{d}t}\vec{e_s}(s) = \frac{1}{\kappa\phi(s)}\vec{e_s}(s) \tag{20}$$

where $\vec{e_s}$ is the unit direction vector of the streamline (*i.e.* $\vec{e_s}(s) = \frac{\vec{u_e}(s)}{\|\vec{u_e}(s)\|}$).

Moreover, equation (19) implies

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{1}{\kappa} = \frac{\partial\psi}{\partial t} + \vec{V_\psi}\cdot\vec{\nabla}\psi \tag{21}$$

Under conservation form, this equation reads:

$$\frac{\partial\psi}{\partial t} + \vec{\nabla}\cdot\left(\psi\vec{V_\psi}\right) = \frac{1}{\kappa} + \psi\vec{\nabla}\cdot\vec{V_\psi} \tag{22}$$

Equation (22) is an hyperbolic PDE for which the propagation velocity is $\vec{V_\psi}$. It can thus easily be solved with a Finite Volume method.

2.4.2 Numerical resolution on unstructured grids

Messinger balance

The Messinger balance is solved by a steady-state finite-volume approach on an unstructured surface mesh (figure 5). The runback terms are conservative fluxes between neighboring faces. An iterative process is thus used. At step $n+1$, the incoming runback term of face i , $\dot{M}_{rb,in,i}^{n}$, is supposed to be known from the previous step n. The following system of equations, derived from

equations (2) and (3), is solved successively in each face i, using Newton's method.

$$\begin{cases} \dot{M}_{rb,out}{}_{,i}^{n+1} + S_i \dot{m}_{evs}(T_w{}_{,i}^{n+1}) + S_i \dot{m}_{acc}{}_{,i}^{n+1} & = S_i \dot{m}_{imp,i} + \dot{M}_{rb,in}{}_{,i}^{n} \\ \Delta U(T_w{}_{,i}^{n+1}, \dot{m}_{acc}{}_{,i}^{n+1}) + \dot{M}_{rb,out}{}_{,i}^{n+1} H_{rb,out}(T_w{}_{,i}^{n+1}) & \\ + S_i \dot{m}_{evs}(T_w{}_{,i}^{n+1}) H_{evs}(T_w{}_{,i}^{n+1}) - S_i \dot{Q}_c(T_w{}_{,i}^{n+1}) & = S_i \dot{m}_{imp,i} H_{imp,i} \\ & \quad + \dot{M}_{rb,in}{}_{,i}^{n} H_{rb,in}{}_{,i}^{n} \\ \mathcal{C}\left(T_w{}_{,i}^{n+1}, \dot{m}_{acc}{}_{,i}^{n+1}, \dot{M}_{rb,out}{}_{,i}^{n+1}\right) & = 0 \end{cases} \tag{23}$$

$\mathcal{C}$ is the closure relation, dependent on the icing regime. This produces the three unknowns $T_w{}_{,i}^{n+1}$, $\dot{m}_{acc}{}_{,i}^{n+1}$ and $\dot{M}_{rb,out}{}_{,i}^{n+1}$ in all faces i at time step $n+1$. In order to update $\dot{M}_{rb,in}{}_{,i}^{n+1}$ for the next iteration in all the faces i in a conservative way, the outcoming rates $\dot{M}_{rb,out}{}_{,i}^{n+1}$ are dispatched over the neigbouring faces. For instance, faces $j=1$ and $j=2$ of figure 5 receive some incoming runback from face i. There is also some incoming runback stemming from another face in face $j=2$. To achieve the computation of $\dot{M}_{rb,in}{}_{,i}^{n+1}$, the ratio of runback which is assigned to each neighbouring face is first computed. For face i, the ratio is primarily given by the orientation between the runback direction $\vec{d_i}$ and the unit normal vector $\vec{n_{ij}}$ (in the tangent plane of face i, pointing outwards) to the edge Γ_{ij} between face i and its neighbor face j (like in any other finite-volume method). Each neighbor face thus receives a ratio r_{ij} of the overall outgoing runback mass rate, constructed to conserve the mass flow rate:

$$r_{ij} = \frac{\dot{M}_{rb,out,i\to j}}{\dot{M}_{rb,out,i}} = \frac{x_{ij}^{+}}{\sum_{k\in\mathcal{N}(i)} x_{ik}^{+}} \tag{24}$$

where $\mathcal{N}(i)$ is the set of edges of face i and

$$x_{ij}^{+} = \begin{cases} \vec{d_i} \cdot \vec{n_{ij}} \mid \Gamma_{ij} \mid & \text{if } \vec{d_i} \cdot \vec{n_{ij}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where $\mid \Gamma_{ij} \mid$ is the length of edge Γ_{ij}.

Thus, in the case of figure 5, face $j=1$ receives $100 r_{i1}$ % of $\dot{M}_{rb,out,i}$. Face $j=2$ receives $100 r_{i2}$ % of $\dot{M}_{rb,out,i}$, which corresponds to $100(1-r_{i1})$ % of $\dot{M}_{rb,out,i}$ because face $j=3$ does not receive any water mass flow rate from face i. On the contrary, face $j=3$ is a source of water runback for face i.

$\dot{M}_{rb,in}{}_{,i}^{n+1}$ is then the sum of all the mass flows received from neighbouring faces:

$$\dot{M}_{rb,in}{}_{,i}^{n+1} = \sum_{j\in\mathcal{N}(i)} r_{ji} \dot{M}_{rb,out,j} y_{ji}^{+}, \tag{26}$$

where $y_{ji}^{+} = 1$ if $\vec{d_j} \cdot \vec{n_{ji}} > 0$, 0 otherwise.

The iterative process is continued until convergence to steady-state.

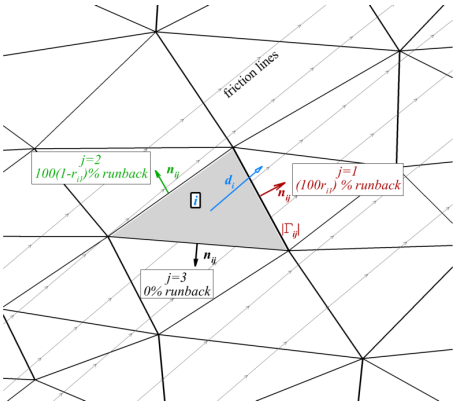


Fig. 5: Computation of runback on unstructured grid

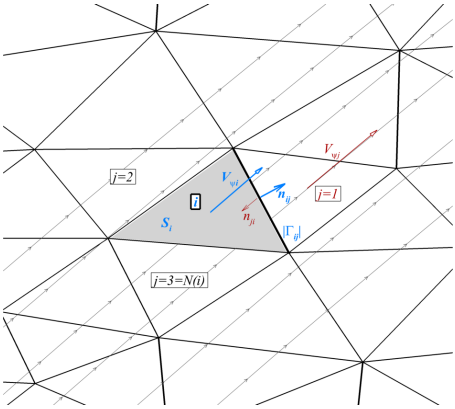


Fig. 6: Notations used for the surface Finite Volume method, face i and its neighbours $j \in \mathcal{N}(i) = \{1, N(i)\}$

Computation of integrals along streamlines

Regarding the solution of equation (22), the notations of figure 6 are retained. After applying the divergence theorem, the integration of this equation over face i reads:

$$\frac{\partial U_i}{\partial t} + \frac{1}{S_i} \int_{\partial S_i} \psi \vec{V_\psi} \cdot \vec{n} \mathrm{d}l = \mathcal{S}_i \tag{27}$$

where the conservative term (solved term, mean value of ψ), the convective term and the source term are, respectively:

$$U_i = \frac{1}{S_i} \int_{S_i} \psi \mathrm{d}S \tag{28}$$

$$\frac{1}{S_i} \int_{\partial S_i} \psi \vec{V_\psi} \cdot \vec{n} \mathrm{d}l = \sum_{j \in \mathcal{N}(i)} G_{ij} \tag{29}$$

$$\mathcal{S}_i = \frac{1}{S_i} \int_{S_i} 1 + \psi \vec{\nabla} \cdot \vec{V_\psi} \mathrm{d}S \tag{30}$$

G_{ij} is the numerical flux through the jth edge of face i, Γ_{ij}. Basic upwind fluxes are used according to the orientation of the velocity $\vec{V_\psi}$:

$$G_{ij} = \frac{\left(V_{ij}^+ U_i + V_{ij}^- U_j\right) \mid \Gamma_{ij} \mid}{S_i} \tag{31}$$

where:

$$V_{ij}^+ = \max\left(0, V_{ij}\right) \quad \text{and} \quad V_{ij}^- = \min\left(0, V_{ij}\right), \tag{32}$$

and V_{ij} is the velocity at the edge Γ_{ij}, that is interpolated as follows:

$$V_{ij} = \frac{1}{2}\left(\vec{V}_{\psi_i} \cdot \vec{n}_{ij} - \vec{V}_{\psi_j} \cdot \vec{n}_{ji}\right) = V_{ij}^+ + V_{ij}^- \tag{33}$$

The source term of equation (30) is simplified as

$$\mathcal{S}_i \simeq 1 + \frac{U_i}{S_i} \int_{S_i} \vec{\nabla} \cdot \vec{V_\psi} \mathrm{d}S = 1 + \frac{U_i}{S_i} \sum_{j \in \mathcal{N}(i)} V_{ij} \mid \Gamma_{ij} \mid \tag{34}$$

Introducing equations (29), (31), (33) and (34) in equation (27) leads to:

$$\frac{\partial U_i}{\partial t} + \frac{1}{S_i} \sum_{j \in \mathcal{N}(i)} \left(V_{ij}^+ U_i + V_{ij}^- U_j\right) \mid \Gamma_{ij} \mid = 1 + \frac{U_i}{S_i} \sum_{j \in \mathcal{N}(i)} \left(V_{ij}^+ + V_{ij}^-\right) \mid \Gamma_{ij} \mid \tag{35}$$

Simplifying the factor terms of V_{ij}^+ in both LHS and RHS, and considering steady-state, the latter equation becomes:

$$\frac{1}{S_i} \sum_{j \in \mathcal{N}(i)} V_{ij}^- U_j \mid \Gamma_{ij} \mid - \frac{U_i}{S_i} \sum_{j \in \mathcal{N}(i)} V_{ij}^- \mid \Gamma_{ij} \mid = 1 \tag{36}$$

This equation is solved via a Gauss-Seidel iterative algorithm: at iteration $n+1$, a loop is made over the faces i to compute U_i from the values U_j already known from iteration n in the neighboring faces j:

$$U_i^{n+1} = \frac{\sum\limits_{j \in \mathcal{N}(i)} V_{ij}^- \mid \Gamma_{ij} \mid U_j^*}{\sum\limits_{j \in \mathcal{N}(i)} V_{ij}^- \mid \Gamma_{ij} \mid} - \frac{S_i}{\sum\limits_{j \in \mathcal{N}(i)} V_{ij}^- \mid \Gamma_{ij} \mid} \tag{37}$$

where U_j^* is either U_j^{n+1} if it has already been computed at $(n+1)$th iteration during the loop over the faces, or U_j^n otherwise.

Moreover, a specific treatment is made in the vicinity of stagnation points, where $U_i^{n+1} = 0$, to ensure that $\psi = 0$. Three conditions are tested to check that face i is a stagnation point:

- for all the edges of face i, the fluxes are outgoing from face i: test if $\sum_{j \in \mathcal{N}(i)} V_{ij}^- \mid \Gamma_{ij} \mid = 0$ (no incoming flux),
- for at least one edge of face i, the flux is outgoing on both sides of the edge: test if $\vec{V_{\psi}}_i \cdot \vec{V_{\psi}}_j < 0$ for at least one edge Γ_{ij},
- an inlet boundary condition is detected at one edge of face i.

2.5 CASSIOPEE

CASSIOPEE (CFD Advanced Set of Services In an Open Python Environment) [38] is an open-source library developed at ONERA. It is a suite of Python modules dedicated to the pre- and post-processing of CFD simulations. These modules interface the data through the CGNS standard representation. Among all functionalities, CASSIOPEE is able to create/modify structured or unstructured meshes (extrusion, intersection, smoothing, adaptation...), prepare the required information for the CFD simulation (i.e. connectivities, overset transfers, immerged boundary conditions...) and post-process a solution (interpolation, slice, iso-surface...). The Python-CGNS architecture allows to easily setup complete and automatic workflows or develop research algorithms. All the CASSIOPEE functionalities can be used in a sequential openMP or a distributed MPI environment. For this work, CASSIOPEE has been used mainly for the re-meshing and Level-Set methods presented later on.

3 Ice shape

3.1 Ice thickness

The ice thickness h_{ice} is derived from the ice growth rate $\dot{m}_{acc}$ thanks to the following relation:

$$h_{ice}(\Delta t) = \frac{\dot{m}_{acc} \Delta t}{\rho_i} \tag{38}$$

where Δt is the physical accretion time and ρ_i is the density of ice. For glaze conditions, $\rho_i = 917$ kg.m^{-3}. For the rime regime, the ice is especially porous and the following correlation for the bulk density is used [39]:

$$\rho_i = 378 + 425 \log(R_m) - 82.3 \left(\log(R_m)\right)^2 \tag{39}$$

where R_m is the Macklin parameter [40]:

$$R_m = -\frac{1}{2} D_{43,imp} \times 10^6 \|\vec{u_p}\| \frac{1}{T_w - T_m} \tag{40}$$

where $D_{43,imp}$ is the de Brouckere mean diameter of impinging droplets and $\|\vec{u_p}\|$ is their velocity. Besides, ρ_i is limited to values larger than 750 kg.m^{-3}.

This correlation has been quite widely and successfully used for 2D geometries with IGLOO2D [4].

3.2 Surface deformation

For most of the simulations of this article, a basic Lagrangian approach is used for the iced surface deformation. The nodes N_i of the surface mesh are simply displaced over a distance h_{icei} in the normal (outwards) direction to the surface $\overrightarrow{n_{Si}}$:

$$\overrightarrow{ON_i^1} = \overrightarrow{ON_i^0} + h_{icei}\overrightarrow{n_{Si}} \tag{41}$$

where O is the origin of the reference frame, N_i^0 is the initial position of node N_i (clean profile) and N_i^1 is the new position of this node (iced profile). Linear interpolations are performed to compute h_{icei} at node N_i^0 from the ice thicknesses known at the neighbouring face centers.

A Level-set approach is also available in IGLOO3D in order to deal with the potential robustness issues of this basic Lagrangian approach. The LEVELSET3D module indeed uses a Level-Set field to calculate the ice growth direction. The main steps of the algorithm are the following:

- Compute the distance ϕ to the iced surface (i.e. the Level-Set field) in cells of the surrounding volume mesh.
- Compute the ice growth direction $\overrightarrow{n}_\phi$ in the volume mesh which is defined as (see figure 7):

$$\overrightarrow{n}_\phi = \frac{\vec{\nabla}\phi}{\|\vec{\nabla}\phi\|} \tag{42}$$

  This growth direction field is based on the Level-Set whose theoretical properties ensure that the displacement of a concave surface would not lead to self-intersection in a pure Eulerian formalism.
- Move the nodes of the iced surface mesh by doing N_L sub-iterations:

$$\forall k \in [1, N_L], \overrightarrow{(\Delta x)}_n^k = \frac{h_{icen}}{N_L}\overrightarrow{n}_\phi(\vec{x}_n^k) \tag{43}$$

  where $\overrightarrow{(\Delta x)}_n^k$ is the displacement of the node n at iteration k, h_{icen} is the ice thickness calculated according to equation (38) at the node n, $\overrightarrow{n}_\phi(\vec{x}_n^k)$ is the interpolated value of the steady ice growth direction field $\overrightarrow{n}_\phi$ at the current position x_n^k of the node n at sub-iteration k. This interpolated value is updated at each sub-iteration.
  As here we use a Lagrangian displacement method, self-intersections are still possible. However, as this Lagrangian displacement is driven by the Eulerian field $\overrightarrow{n}_\phi$, the risk of self-intersections can be reduced by increasing the number of sub-iterations N_L. We point out that the resulting surface mesh may then have nodes that collapse when dealing with difficult configurations with highly concave features. In these cases, geometric post-processing may be necessary to remove these collapsing nodes. This step is not addressed

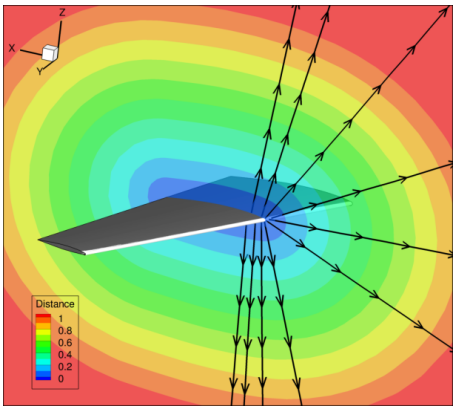


Fig. 7: Ice growth direction defined as the gradient of the distance to the iced surface

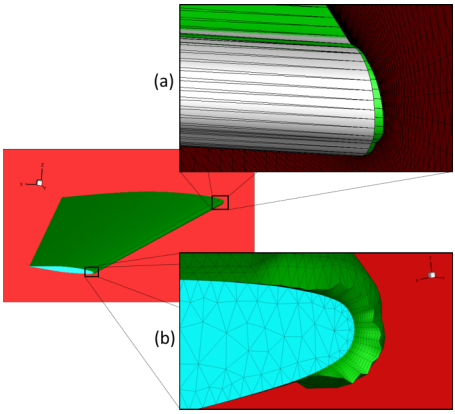


Fig. 8: Imposing the topological consistency of the connections between the deformed iced surface and the adjacent boundary conditions by projecting (a) and extruding (b) the edges of the iced surface

here as this issue was not encountered for the simulations described in the present study.

- Ensure topological consistency of the connections between the iced surface and other domain boundary conditions adjacent to it (see figure 8).

3.3 Verification

The RUN405 case of table 1 was used in order to check that IGLOO3D produces the expected ice shape. To do so, IGLOO2D was run first, as a reference, with default options [4] on an unstructured mesh with a characteristic mesh size near the wall equal to $\Delta x/c \simeq 2 \cdot 10^{-3}$ (c is the chord length). Then,

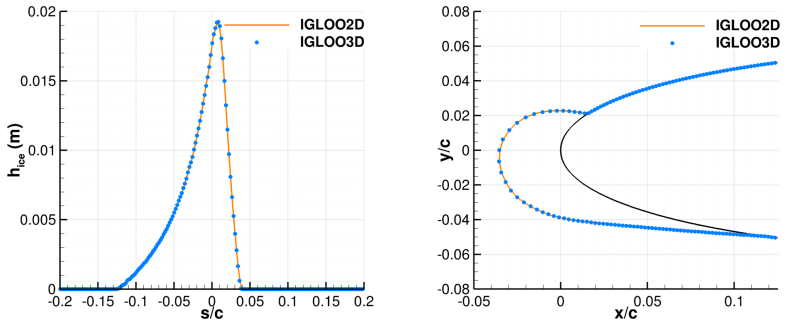


Fig. 9: Verification of ice shape, RUN405 case. On the left, predicted ice thicknesses. On the right, ice shapes.

IGLOO3D was fed with the exact same input data as IGLOO2D: the surface mesh, the aerodynamic inputs (h_{tc}, T_r, etc.) and the droplet inputs (especially $\dot{m}_{imp}$) are the same for both codes. Predictor simulations were run with the two codes. Figure 9 shows that IGLOO3D produces the same results as IGLOO2D (with the simple Lagrangian displacement method). There is indeed a perfect agreement between the two codes on the ice thickness h_{ice} for all curvilinear abscissas s (set equal to 0 at the stagnation point). The ice shape is also logically the same for both codes.

The level-set method is evaluated in figure 10 with the same unstructured mesh was used as for figure 9 (after verifying that the solution is unchanged for finer volume meshes). Figure 10 shows that the result is the same as that of the basic Lagrangian approach, whether the number of iterations of the method N_L is equal to 10 or 30. This is expected because the airfoil surface is convex. For the predictor-corrector approach, in the cases investigated in this paper, there is no concavity effect to take into account either, and in the following, the basic Lagrangian method will be preferred.

4 Remeshing with CASSIOPEE

4.1 Dragon method

An automatic grid generation strategy called "Dragon mesh" [41] is evaluated using the open-source CASSIOPEE software [38]. This approach creates a hybrid grid composed of prism cells extruded from the walls and hexahedra cells in the near field with a junction layer made of tetrahedra. The mesh generation process can be decomposed in 6 steps:

1. The geometry can be imported in CASSIOPEE, either as an IGES/STEP file or a surface mesh. In the present configurations, a surface mesh (unstructured with triangles elements) is provided. This mesh can be refined or smoothed (for example, for too sharp iced geometries).

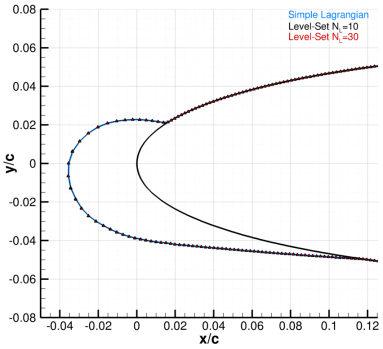


Fig. 10: Verification of the level-set solution, RUN405 case

2. The second step concerns the creation of a prismatic layer by extruding the surface mesh. CASSIOPEE allows growing layers of prisms with a given distribution along the normals of the surface mesh. Here, a simple geometric law is applied where the user gives the ratio and the first prism height.
3. An octree grid is automatically generated by CASSIOPEE from the external prism layer. The minimum cell size of the octree is calculated from the averaged cell size of the surface mesh and the ratio between two refinement levels is equal to 2. Other reference surfaces for the creation of the octree can be added during this process, in order to refine locally, for example around a body wake or a separation area.
4. The next step consists in blanking the unstructured octree grid around the external prisms layer. An offset of this external surface is applied to ensure a given thickness of the blanking hole.
5. Finally, the gap, previously created between the octree blanked grid and the prism layers mesh is filled with tetrahedra thanks to the open source TetGen tool, integrated in CASSIOPEE. The automatic processing of the previous steps with respect to a mean cell size value will ensure that the tetrahedra are almost isotropic.
6. The last step allows assembling the three previous meshes (prisms, octree and tetrahedra) which are converted to a conformal polyhedra representation. The boundary conditions can also be automatically added and the final block can be split for parallel distribution.

This meshing strategy has already been applied on different external configurations like a simple CRM aircraft configuration or the LAGOON landing gear [42]. The Dragon mesh strategy has been extended to constrained geometries (with symmetry plane(s) for example) or internal configurations. The problematic was to manage the extrusion step with the plane constraints along which the extruded prisms have to be projected. Figures 11, 12 and 13 present three examples:

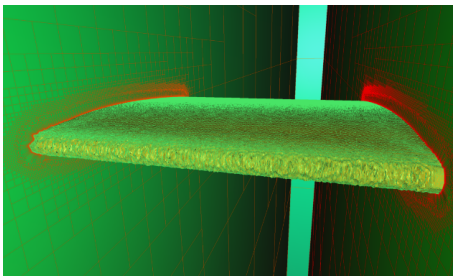


Fig. 11: Example of Dragon mesh strategy applied on a NACA0012 iced wing with an ice shape (RUN405 conditions) and constrained by two planes

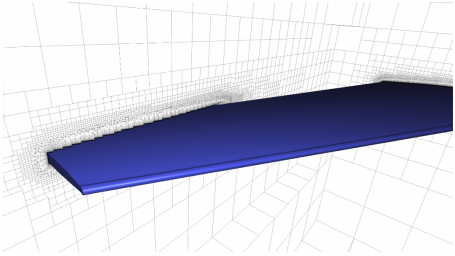


Fig. 12: Example of Dragon mesh strategy applied on a GLC305 geometry with an ice shape

- a NACA0012 iced wing constrained by two planes on which the extruded prisms are projected (figure 11). For this case, the algorithm has been embedded in a global CASSIOPEE function for cases without or with one or two constraining planes.
- the embedded algorithm also allows generating meshes on wings constrained by a wall on one side, such as the GLC305 wing of figure 12.
- for the Rotor67 NASA test case (figure 13), the iced blade, spinner and carter surfaces are extruded with the constraints of periodic lateral and in/out surfaces.

4.2 Verification

The Dragon method is assessed by generating a mesh around a clean unswept NACA0012 geometry (figure 14). Two kinds of meshes were used for the airfoil surface, one composed of around 113500 triangles, and another one composed of 44800 quadrangles (the considered span length is equal to the chord length). The Dragon volume meshes are composed of 5.5 million cells for the first mesh, and of 3 million cells for the second one. Even though the overall number of cells is smaller, the mesh based on quadrangles is almost twice as fine in the streamwise direction as the mesh based on triangles in the vicinity of the stagnation line. The prism layers used are thin enough at the wall

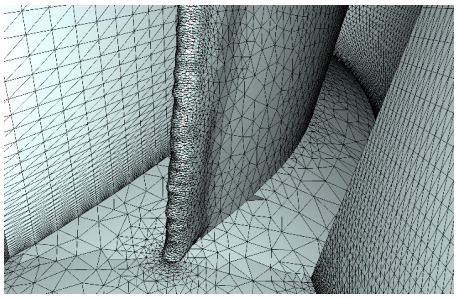


Fig. 13: Example of Dragon mesh strategy applied on a Rotor67 NASA geometry with an ice shape

that the y^+ of the first cells is everywhere lower than 3, which is correct for the low-Reynolds approach used for modelling the wall boundary conditions. For the RUN405 case, as shown in figures 15 and 16, the solutions produced by IGLOO3D ("Unstructured") on the Dragon meshes are compared against IGLOO2D (using the default options) and a 3D reference solution obtained on a structured mesh manually generated with ICEM-CFD ("IGLOO3D Structured"). For the latter solution, the airflow simulation was performed with the solver elsA [43] using the same conditions as the ones exposed in [17]. The agreement between all the solutions is quite good for pressure distribution on the airfoil. There is however a slight difference in pressure on suction side especially for the mesh based on quadrangles on the surface. This deserves further analysis, but the solution is already good enough for a key parameter of the simulations, the collection efficiency $\beta = \dfrac{\dot{m}_{imp}}{\mathrm{LWC}u_\infty}$ (where LWC is the liquid water content of the cloud, while u_∞ is the farfield airflow velocity), to be very similar to the other solutions (computed with a single-bin droplet-size distribution). The other main conclusion is that for the surface mesh composed of triangles, the solution is quite good but there is a little bit more variability from one cell to another than with the surface mesh made of quadrangles (the illustrations appear as slightly thicker curves). Figure 17 confirms that there is less variability in the predictor ice shape produced by IGLOO3D for the surface mesh composed of quadrangles than for the one composed of triangles.

Finally, an important parameter for the calculations is the heat transfer coefficient. In figure 18, it is plotted for the mesh composed of triangles on the wing surface, for two methods: the default method (equation (10)) and the one based on the skin friction coefficient C_f (equation (12)). For this case, the spreading of the IGLOO3D results is quite important, even more so for the default method. In addition, the C_f-based method is somewhat more similar to IGLOO2D in the immediate vicinity of the leading edge, which is expected. The agreement is worse further away (while for the default method it remains a little better). Considering the differences in methods, the agreement between all results can already be judged as satisfactory (viscous-inviscid coupling for IGLOO2D, Navier-Stokes approach for IGLOO3D, h_{tc} computed

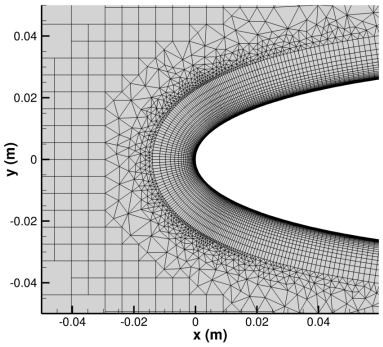


Fig. 14: NACA0012 RUN405 test-case: Dragon mesh of the clean wing

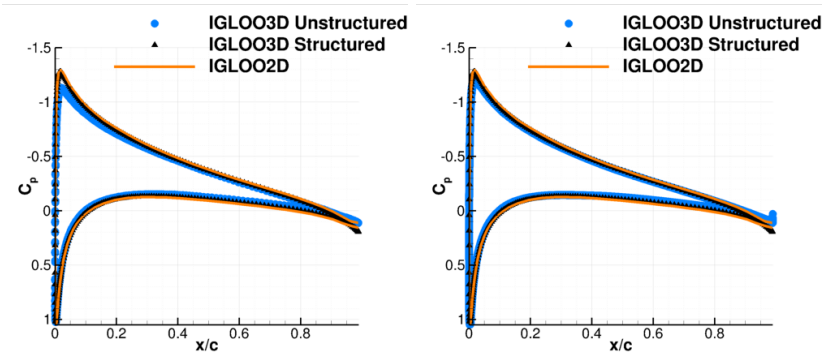


Fig. 15: NACA0012 RUN405 test-case: pressure coefficient for two simulations using the Dragon mesh of the clean wing. Comparison against simulations with IGLOO2D and with IGLOO3D on a reference mesh. On the left, surface mesh composed of quadrangles. On the right, surface mesh composed of triangles.

with equation (10) or equation (12) for IGLOO3D, with equation (13) for IGLOO2D).

The agreement between IGLOO3D and IGLOO2D on h_{tc} is even better for the Case 241 of IPW1, which will be discussed in section 6.3 (figure 19). The difference between the two codes on h_{tc} is also much smaller than the dispersion observed between all the codes of IPW1 [1]. The IGLOO3D results, computed with the default method (equation (10)), are very good both for the mesh provided by the IPW1 committee and for a Dragon mesh (there is however a bit more variability on the latter, shown by all the points in light blue corresponding to the solution in the surface elements). Besides, there is a difference at the stagnation point, with IGLOO3D predicting a significantly lower h_{tc} for the mesh of the IPW1 committee (the mesh is finer for IGLOO3D

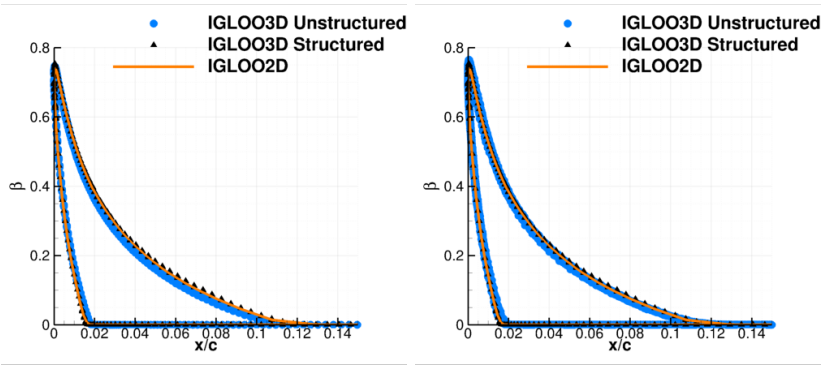


Fig. 16: NACA0012 RUN405 test-case: collection efficiency for two simulations using the Dragon mesh of the clean wing. Comparison against simulations with IGLOO2D and with IGLOO3D on a reference mesh. On the left, surface mesh composed of quadrangles. On the right, surface mesh composed of triangles.

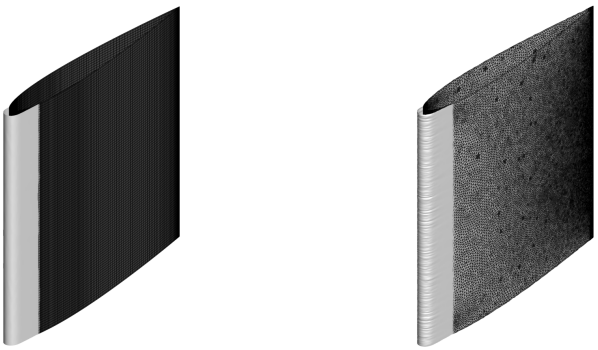


Fig. 17: NACA0012 RUN405 test-case: predictor ice shape for two simulations using the Dragon mesh of the clean wing. On the left, surface mesh composed of quadrangles. On the right, surface mesh composed of triangles.

than IGLOO2D at this location). At this location, the discrepancy is critical to accretion modeling and may introduce a deficit in freezing (because the ice is less cooled), and thus additional liquid-water runback.

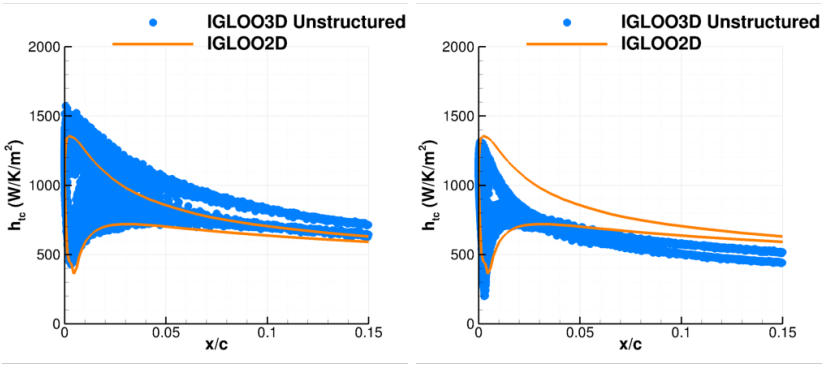


Fig. 18: NACA0012 RUN405 test-case: heat transfer coefficient for simulations using the Dragon mesh of the clean wing (surface mesh composed of triangles). On the left, h_{tc} computed with equation (10). On the right, h_{tc} computed with equation (12).

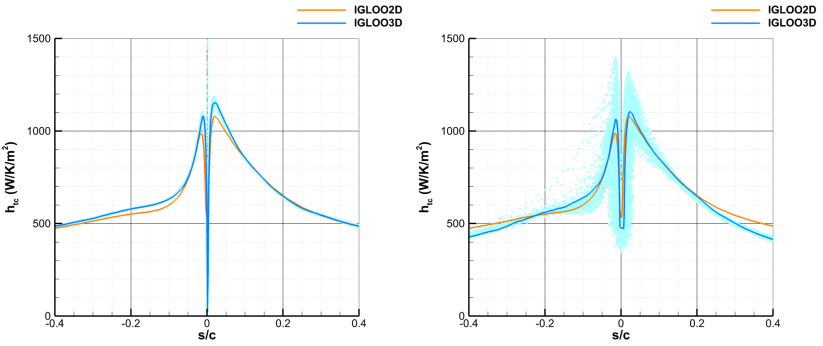


Fig. 19: Case 241 test-case: heat transfer coefficient $h_{tc}(W/K/m^2)$ produced by IGLOO2D and IGLOO3D on the clean airfoil, with respect to the curvilinear abscissa s ($s = 0$ at the stagnation point). Solution on the mid-slice for IGLOO3D (and solution in the surface elements in light blue). On the left, mesh provided by the IPW1 committee. On the right, Dragon mesh.

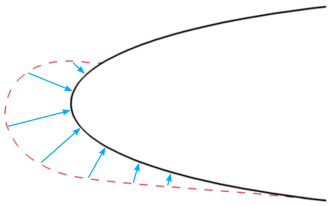


Fig. 20: Schematic view of the interpolation (projection) performed by INTERP3D from the iced profile (red) to the initial profile (black).

5 Predictor-Corrector method

The predictor-corrector method is an approach that models the retroaction of the ice shape on the airflow and droplet trajectories at the cost of only two computational loops of figure 1 (unlike the multi-step method which typically requires several tens of loops). This method is available in the 2D code IGLOO2D [4]. In this article, we describe the implementation of this method for a 3D icing suite, IGLOO3D. As explained in section 2.1, the method consists of correcting the predicted ice shape thanks to the aerodynamic and trajectography solutions around the predicted ice shape. This means that, once these "corrector" aerodynamic and trajectography solutions are obtained, the following tasks need to be done:

1. project and interpolate corrector data from the iced profile to the initial profile (clean profile),
2. run a Messinger calculation in corrector mode on the initial profile using both predictor and interpolated corrector data.

Step 1 is performed thanks to the INTERP3D module whose operating principle is detailed in the paragraph 5.1. MESSINGER3D computations in corrector mode are described in section 5.2.

5.1 Data interpolation with INTERP3D module

INTERP3D interpolates the surface data from the (predicted) iced profile (also called corrector surface) to the initial (clean) profile (also called predictor surface), see figure 20. For this purpose, the following steps are performed by the INTERP3D module:

- Extraction of the surface geometry and (aerodynamic and trajectography) data for both predictor and corrector sets,
- Interpolation of corrector surface data at face centers whenever necessary,
- Projection of corrector surface data on predictor surface,
- Interpolation of projected corrector surface data at nodes if necessary.

All computations are performed based on face-centered data values. To calculate a projected value at a face center of the initial profile mesh, it is

necessary to identify the subset of faces of the predicted profile mesh that will contribute to the result. This is done in two steps:

- First, a ray is launched from the center of the considered face in the normal direction. The face of the predicted mesh profile that it intersects is identified (see figure 21-a). Note that faces of the predicted profile mesh are temporarily triangulated and stored in a background Cartesian grid to accelerate the computation and make it more robust.
- Second, the other donor faces of the predicted profile mesh are identified: those are the faces adjacent to the node of the intersected face which is closest to the intersection point (see figure 21-b).

Once the donor faces are identified, their centers are projected on the tangent plane of the considered face and their values are interpolated to the center of the considered face by a least-square interpolation method in the tangent plane (see figure 21-c). Note that if the number N_d of donor faces is strictly less than three (at a mesh boundary for example), a least-square interpolation is not possible. Thus, a particular treatment is used for the interpolation in such cases: inverse-distance interpolation for $N_d = 2$, interpolation of order 0 for $N_d = 1$.

5.2 Corrector mode of MESSINGER3D

In corrector mode, the Messinger computation is made iteratively. N successive solutions of the Messinger balance exposed in section 2.4 are computed (for the simulations of this article, $N = 10$). At each loop $i \in \{1, ..., N\}$, the (aerodynamic and trajectography) inputs of the Messinger balance are linearly interpolated from the predictor INPUT^{pre} and interpolated corrector INPUT^{cor} data as follows:

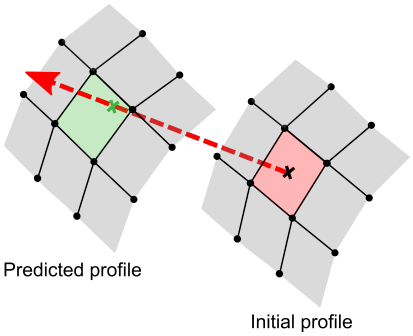
$$\text{INPUT}^{i} = (1 - \frac{t^i}{\Delta t})\text{INPUT}^{pre} + \frac{t^i}{\Delta t}\text{INPUT}^{cor} \tag{44}$$

where:

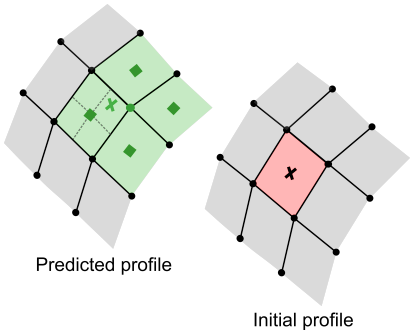
$$t^i = \frac{i-1}{N-1}\Delta t \tag{45}$$

The physical accretion time considered for the corrector simulation is consistent with the one used for the predictor simulation: $\Delta t = t_{exp}$ (exposure time to the icing cloud). Consequently, the first corrector loop is made with predictor inputs (INPUT^{pre}), whereas the last loop is performed with corrector inputs (INPUT^{cor}). The final ice thickness is the sum of the ice thicknesses resulting from each loop:

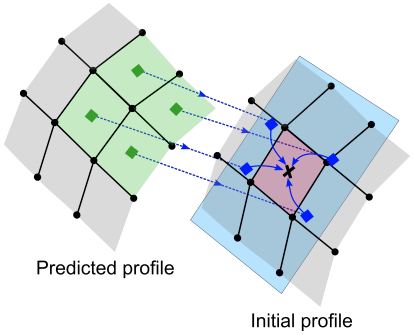
$$h_{ice} = \sum_{i=1}^{N} h_{ice}(t^{i-1} \rightarrow t^i) \tag{46}$$



(a) Ray-Face intersection



(b) Donor faces identification



(c) Data projection and in-plane least square interpolation

Fig. 21: Illustration of the projection process.

5.3 Verification

For the verification of the predictor-corrector method, the RUN405 and Run 06-27-91/1 cases of table 1 are used. For RUN405, which is a rime case, figure 22 shows that the ice thickness converges quickly with the number of loops N. The difference between the thicknesses obtained for $N = 50$ and $N = 5$ is less than 70 microns. For $N = 10$, which is the default value in the following,

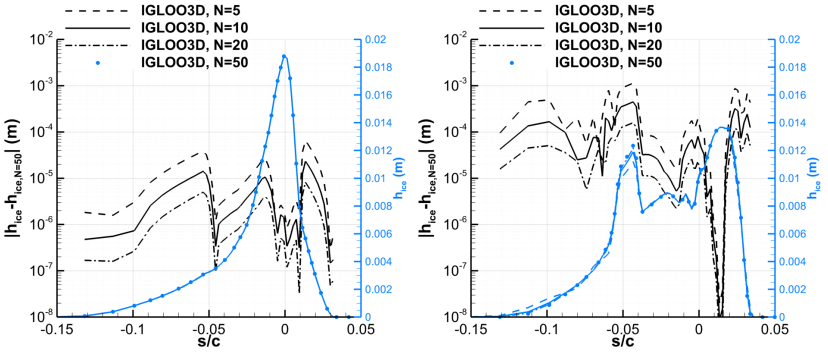


Fig. 22: Impact of the number of loops N on the predictor-corrector solution. On the left, RUN405 case. On the right, Run 06-27-91/1.

it is around 10 microns at most. This limited effect of N is expected: in rime regime, the accretion rate $\dot{m}_{acc}$ is directly proportional to β because freezing is immediate. The interpolation performed on β by INTERP3D as well as equation (44) both being linear, no effect of N is expected (the variation of the ice density bringing however a non-linearity regarding h_{ice}). In the glaze regime, the process is less linear. For the glaze-ice run 06-27-91/1, the convergence is slower and the error remains larger for $N = 10$. However, it is less than 0.5 mm (corresponding to less than a 5% error).

As shown in figure 23 for RUN405, the IGLOO3D result is successfully cross-checked against the reference IGLOO2D results, for identical inputs (surface mesh size, aerodynamic and droplet data). The ice temperature, which is strictly below 273.15 K, confirms that a purely rime ice is obtained. Therefore, the ice thickness is related to the droplet input (β), as well as to the ice density. The ice temperature is affected by the airflow, in particular by h_{tc}. The fact that both the ice thickness and the temperature are well captured by the sequence of INTERP3D and MESSINGER3D thus shows that the predictor-corrector method implemented in IGLOO3D faithfully reproduces the reference method of the article [4].

6 2D ice-accretion test-cases

A series of test cases is discussed to show the potential and limitations of the predictor-corrector method.

Several 2D cases are first discussed, as shown in table 1. The RUN405 case has been used earlier for verifications. The cold conditions of this case produce some rime ice. A NACA0012 airfoil is used, with a chord length equal to 0.5334 m and a 3.5° angle of attack. A NACA0012 profile with the same chord length is also used for the run 06-27-91/1. The angle of attack is 4°. The warmer conditions and the higher value of the LWC make it possible to have some glaze ice. Case 241 is a rime-ice configuration investigated in the IPW1 [1].

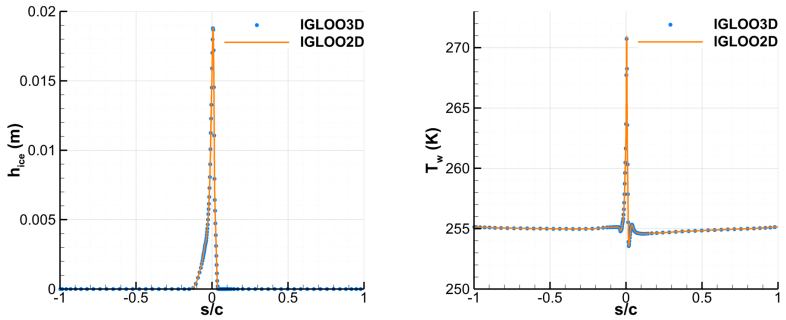


Fig. 23: Verification of the predictor-corrector method, RUN405 case

This case involves a NACA23012 airfoil with a chord length equal to 0.4572 m and a 2° angle of attack. Case 241 is therefore mainly a variation in geometry compared to RUN405. In all three cases, the geometry is extruded to employ 3D meshes.

In this section, IGLOO3D results are compared to both experimental data and IGLOO2D. This allows us to verify that the implementation in the 3D code allows a similar behavior to the 2D code, despite sometimes considerable differences on the numerical approaches and parameters. For example, for the IGLOO2D computations, the airflow simulations are performed with a coupled approach between an inviscid code and an integral-boundary-layer solver, on very coarse meshes. Moreover, the droplet trajectories are treated by a Lagrangian approach. Besides, this section dealing with relatively simple geometries is also an opportunity to present and compare approaches based on several types of meshes: those generated by the Dragon method, and some meshes created manually with ICEM-CFD already used in a previous paper [17].

Table 1: 2D ice-accretion cases investigated

Case	V_∞ (m/s)	P_∞ (Pa)	T_∞ (K)	MVD (μm)	LWC (kg/m^3)	t (s)
Case 241	102.889	92528.	250.15	30.	$0.42 \cdot 10^{-3}$	300.
Run 06-27-91/1	57.914	95610.	266.45	20.	$1.3 \cdot 10^{-3}$	480.
RUN405	102.827	93000.	250.37	20.	$0.55 \cdot 10^{-3}$	420.

6.1 RUN405

The NACA0012 RUN405 case was run with the predictor-corrector approach. The mesh employed for the predictor step was the Dragon grid discussed in section 4.2, for which the NACA0012 surface mesh is composed of around

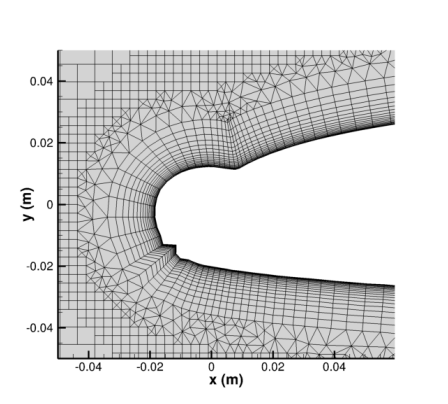


Fig. 24: NACA0012 RUN405 test-case: Dragon mesh for the corrector computation.

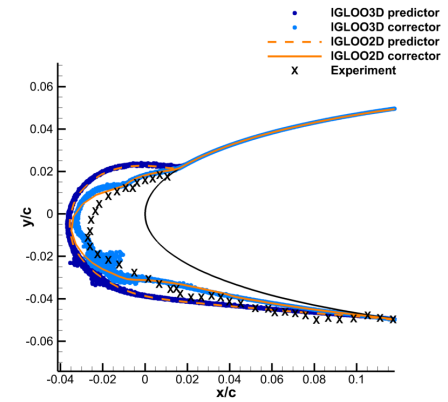


Fig. 25: NACA0012 RUN405 test-case: ice shape

113,500 elements. The mesh of the corrector step is also a Dragon mesh (figure 24) with the same number of surface elements and 4.4 million volume elements. For this case, the heat transfer coefficient is computed from the friction coefficient C_f (equation (12)). After equation (1), the equivalent sand-grain roughness height employed for the computation is $k_s = 0.5334$ mm. Regarding the simulation of droplet trajectories, a single-bin distribution was used, which is common practice for modeling in-flight icing of 2D airfoils under Appendix C conditions [44].

The predicted ice shape, shown in figure 25, is very similar to that computed with IGLOO2D (with default options). This was expected since rime-ice shapes are directly linked to the collection efficiency β – because the water freezes locally – and figure 16 shows the very good agreement between the codes on β (for the predictor step). However, the convective heat transfer being

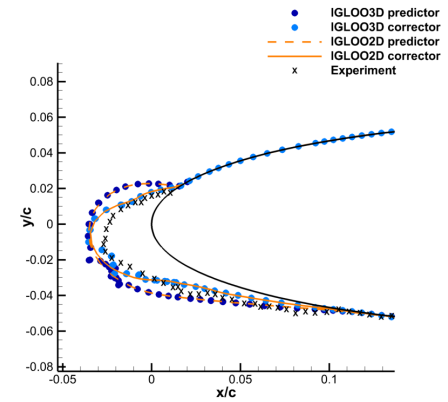


Fig. 26: NACA0012 RUN405 test-case: ice shape using a manually-generated structured mesh for IGLOO3D

lower for IGLOO3D than for IGLOO2D around the stagnation point (figure 18), IGLOO3D predicts the occurence of a bit of liquid-water runback there, which explains the slight discrepancy between the IGLOO2D and IGLOO3D ice shapes. The numerical and experimental results are very similar in all respects, except for the maximum thickness. The small inaccuracy there could be caused by several factors: the ice density model may be slightly inaccurate, a little bit of runback may occur in the experiment (which may be influenced by the convective transfer and the roughness height, which is difficult to predict). An accurate multi-step simulation may also help capturing better the ice thickness there. Consistently with what was discussed in section 4.2, there is some fluctuation in the ice shape due to the surface mesh used, composed of triangles.

As an element of comparison, figure 26 shows the results obtained with IGLOO3D using structured meshes manually generated with ICEM-CFD (similar to the ones of reference [17], with around 1300 quadrangles on the wing surface and 250000 hexaedra), instead of the Dragon meshes. The aerodynamic simulations were then performed with elsA, as in reference [17], which provides the momentum thickness of the boundary layer, from which the heat transfer coefficient was inferred (equation (13)). The results are very similar to the ones obtained with the Dragon meshes. The main difference is that there is no variability in the ice shape because the mesh is structured and the results are strictly identical for all cells of a given x/c.

6.2 Run 06-27-91/1

The same approach as for figure 26 was used to investigate the 2D glaze-ice case. A structured mesh made with ICEM-CFD was thus made with the same properties as in reference [17] both for the predictor and the corrector steps. The airflow simulations were then made with elsA and equation (13) was used

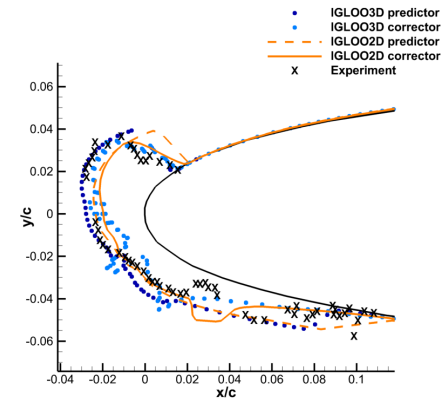


Fig. 27: NACA0012 run 06-27-91/1 test-case: ice shape using a structured mesh for IGLOO3D

to compute h_{tc}. The predictor step was already shown in reference [17] for this case but a fully-turbulent approach was used. Here, the laminar-turbulent transition is modeled. The predictor and corrector ice shapes are shown in figure 27. Again, the agreement between IGLOO3D and IGLOO2D is good. The biggest difference between the results of the two codes concerns the upper part of the profile: IGLOO3D predicts a slightly thicker and narrower horn than IGLOO2D. This difference is mainly due to the fact that IGLOO3D predicts a larger h_{tc} than IGLOO2D almost everywhere (except around the stagnation point), which reduces runback. Nevertheless, overall, all the simulations capture quite satisfactorily the experimental ice shape. IGLOO3D produces too much runback in the immediate vicinity of the stagnation point, causing the presence of two thin horns there. However, it must be emphasized that the corrector simulations improve the results in the vicinity of the accretion limits.

6.3 Case 241

For the rime-ice case 241, we used the "hybrid" approach retained for all the simulations of the IPW1 addressed in this article: the mesh of the clean airfoil provided by the organizing committee was used for the predictor step while a mesh was automatically generated with the Dragon method for the corrector step. The 7-bin droplet size distribution provided by the organizing committee was also used.

The agreement between IGLOO2D and IGLOO3D is very good both for predictor and corrector steps (figure 28). The predictor-corrector approach produces a rather satisfactory ice shape, compared to the experimental one (both the entire set of ice-shape scan points and the so-called Maximum Combined Cross-Section – MCCS – are provided for the experimental shape, with the MCCS resembling an envelope of the ice shape [1, 45]). The corrector ice

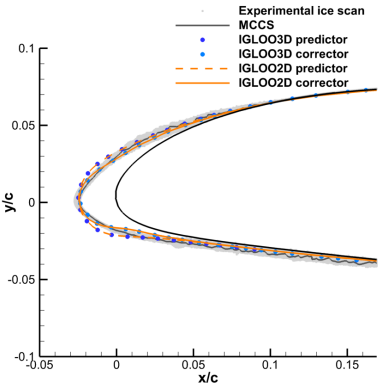


Fig. 28: Case 241, 2D rime-ice case. Ice shapes produced by IGLOO2D and IGLOO3D. Comparison against the experimental ice shape.

shape is again sharper than the predictor shape and it is in better agreement with the experimental shape.

7 3D ice-accretion test-cases

Table 2 shows the icing conditions for the five 3D test-cases investigated. First, a test-case involving a GLC305 airfoil with 28° sweep angle at the leading edge (and 15.6° at the trailing edge) is investigated, Case 4 (a glaze-ice case). The experimental data is available in reference [46]. A single-bin droplet-size distribution was used for this case. The roughness height was set to 0.952 mm.

The other four cases are taken from the IPW1 base. NACA0012 airfoils are employed for these test-cases with a chord length equal to 0.91438 m and a 0° angle of attack. Two different sweep angles are used, $\Lambda = 30°$ and $\Lambda = 45°$. The interested reader will find further information in the synthesis paper of IPW1 [1]. The 7-bin droplet size distributions provided by the organizing committee were used for these cases. After equation (1), the roughness size was set to 0.91438 mm.

Table 2: 3D ice-accretion cases investigated

Case	Λ (°)	V_∞ (m/s)	P_∞ (Pa)	T_∞ (K)	MVD (μm)	LWC (kg/m^3)	t (s)
Case 4	28.	112.	97511.	263.	20.	$0.68 \cdot 10^{-3}$	120.
Case 361	30.	103.	92321.	257.	34.7	$0.5 \cdot 10^{-3}$	1200.
Case 362	30.	103.	92321.	266.	34.7	$0.5 \cdot 10^{-3}$	1200.
Case 371	45.	103.	94463.	257.	32.	$0.5 \cdot 10^{-3}$	1200.
Case 372	45.	103.	94463.	266.	32.	$0.5 \cdot 10^{-3}$	1200.

7.1 Case 4

Regarding the GLC305 Case 4, the grid of the predictor simulation is an unstructured mesh for which the surface mesh is composed of 6,636 triangles. Figure 29 shows the mesh and its level of refinement in the vicinity of the stagnation point. The two cuts, A and B, employed to compare between IGLOO3D and the experiments are also shown in figure 29. The mesh of the corrector run is an automatically generated Dragon grid (figure 12). The surface grid is composed of 59,260 triangles and the volume grid is made of 3.2 million elements. Additionally, the heat transfer coefficient was computed with equation (12) for this case.

The numerical and experimental ice shapes are compared on the two cuts A and B in figure 30. The numerical ice shape is smooth (figure 29) and does not capture the fact that there are some scallop structures in the experimental ice shape. However, the simulation remains satisfactory. First, the ice thickness at the leading edge is captured quite well on both cuts. Second, except for the sharpness of the ice shape at its limit on suction side, the overall ice shape is predicted correctly by IGLOO3D. Besides, the ice shape obtained with the corrector run is closer to the experimental one than the ice shape of the predictor run. It is especially in the vicinity of the accretion limits that the ice

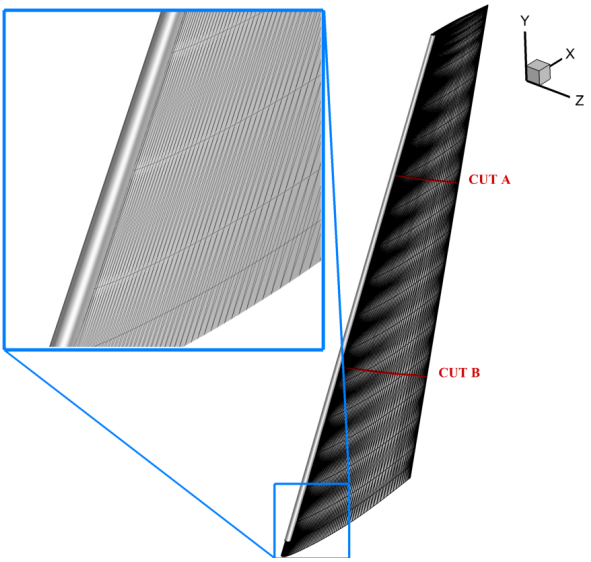


Fig. 29: GLC305 case 4: clean surface mesh, cuts A and B for experimental ice shape measurements, simulated ice shape (white surface).

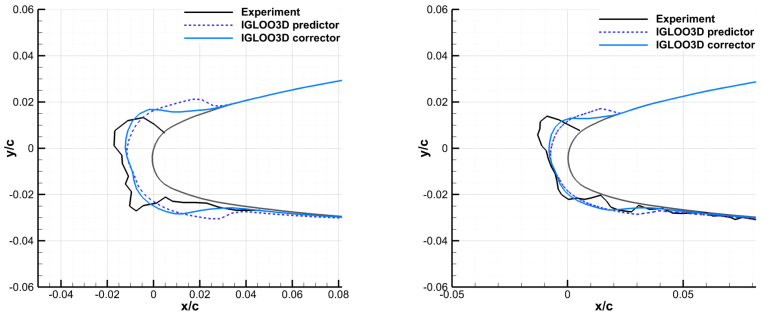


Fig. 30: GLC305 case 4: comparison between experimental and numerical ice shapes along cuts A and B. On the left, Cut A. On the right, Cut B.

shape is improved by the predictor-corrector simulation. It has to be mentioned that the effect of the number of corrector loops N has been tested for this 3D glaze-ice case with rather small ice thickness. For $N = 50$, the IGLOO3D result is almost superimposed to the curve obtained with the default $N = 10$ (which explains why we did not add the $N = 50$ curve in figure 30).

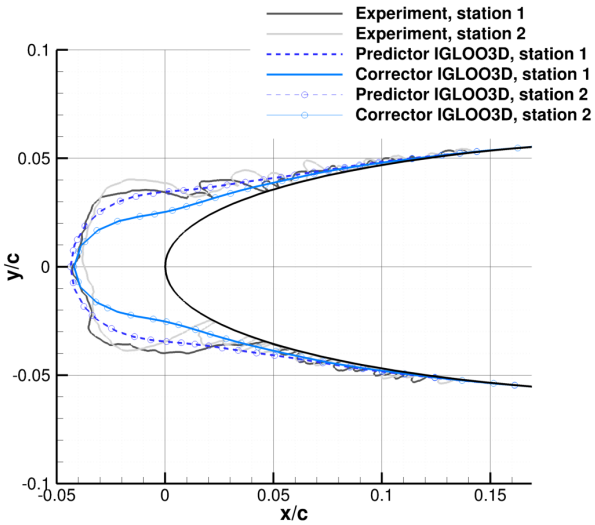


Fig. 31: Case 361: ice shapes produced by IGLOO3D. Comparison against the experimental ice shape [47].

7.2 Case 361

For the rime ice case 361 with a 30° sweep-angle NACA0012 airfoil, the ice shape produced by the predictor approach is reasonably good, compared to the experimental ice shape (figure 31). The predictor-corrector results do not even improve the ice shapes, which tend to be too sharp. For both the predictor and corrector solutions, there is no visible difference between the numerical ice shapes produced at the two experimental stations.

The pressure coefficient distribution has also been checked to ensure that it is correct compared to the experimental data (figure 32). Additionally, the ice shapes predicted by various solvers (including ours) proved to be very similar to each other on this test-case, despite different numerical approaches and meshes [1].

In order to improve the results, it could be necessary to work on the ice density model. It is our experience that this model is quite satisfactory for 2D wings but it is probably less reliable for 3D geometries. Additionally, if the ice shape exhibits a transverse pattern such as small scallops, it is unlikely that the predictor-corrector approach captures the shape, as confirmed by the identical results obtained for the two stations in figure 31. Such ice shapes are indeed largely due to shadowing effects that affect the droplet collection and that are poorly modeled by the predictor-corrector approach. A multi-step approach would be more appropriate.

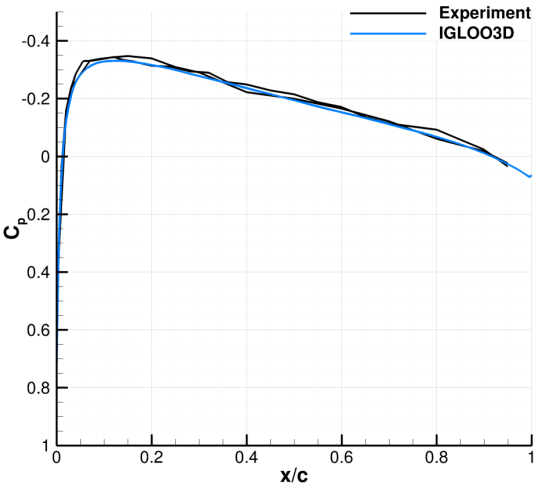


Fig. 32: Case 361: pressure coefficient produced by IGLOO3D and by the experiments [47].

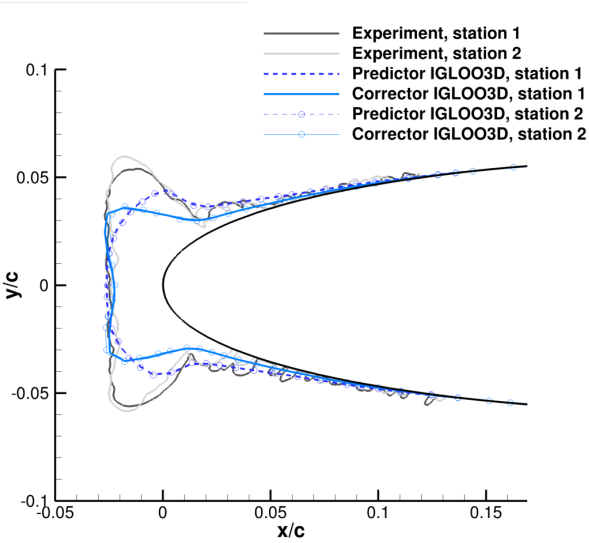


Fig. 33: Case 362: ice shapes produced by IGLOO3D. Comparison against the experimental ice shape [47].

7.3 Case 362

For the glaze-ice case 362, figure 33 shows that IGLOO3D captures very well the experimental ice shape on both sides of the separation line. However, the horn heights are not predicted correctly. As in case 361, the ice density model could be questioned and the use of a multi-step approach could help capturing some scallop-like patterns (which may be expected to be moderate as shown by the two rather similar experimental tracings of figure 33).

It is also possible that IGLOO3D produces too little runback, because h_{tc} is high close to the stagnation point. This is due to the fact that for the swept-wings, it was not possible to activate a transition model in IGLOO3D and the boundary layer is thus fully turbulent. Modeling the laminar-turbulent transition would probably increase the amount of runback. However, since the ice thickness predicted by the IGLOO3D simulation is not so bad along the separation line, the amount of runback should not be increased too much.

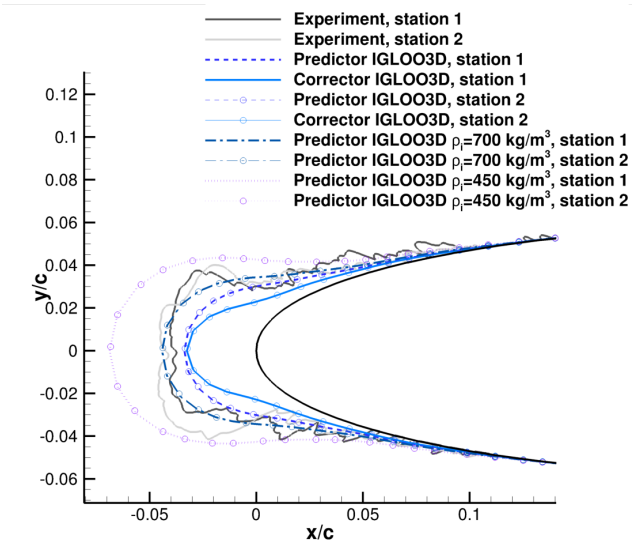


Fig. 34: Case 371: ice shapes produced by IGLOO3D. Comparison against the experimental ice shape [47].

7.4 Case 371

Figure 34 shows that the IGLOO3D results are worse for a 45° sweep-angle than in the same condition for a 30° sweep-angle (case 361). As in case 361, the ice shape predicted by the predictor approach is better than that given by the predictor-corrector approach, which is is too sharp. Figure 35 shows that the pressure distribution is well predicted again, which excludes the cause of a poorly calculated aerodynamic effect.

For swept wings with a high sweep angle corresponding to the SUNSET2 database, it was necessary to drastically change the ice density by taking $\rho_i = 450kg/m^3$ [18, 48, 49]. This was to model a bulk ice density taking into account the voids between the scallop structures. For test-case 371, the use of $\rho_i = 450kg/m^3$ predicts a too thick ice shape, whereas $\rho_i = 700kg/m^3$ seems to be a reasonable value to capture at least the ice thickness in the vicinity of the separation line (figure 34). However, it is not a proper model for the bulk ice density. Such a model still needs to be developed for 3D applications.

For high sweep angles, various kinds of scallop-like ice shapes can be obtained [50], which could explain the relatively high variability of the experimental tracings for this case. The use of multi-step approaches could thus also help to better model the ice shape, as mentioned earlier.

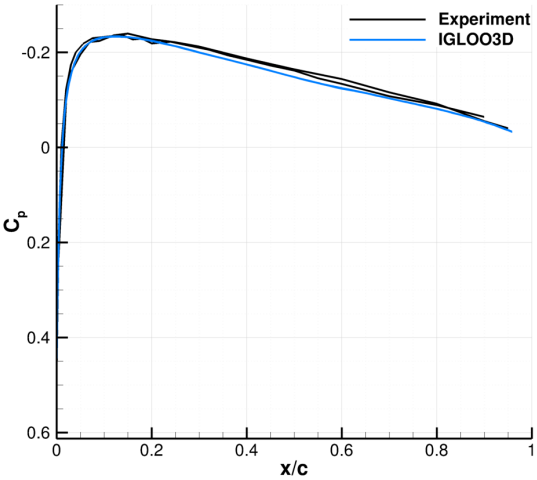


Fig. 35: Case 371: pressure coefficient produced by IGLOO3D and by the experiments [47].

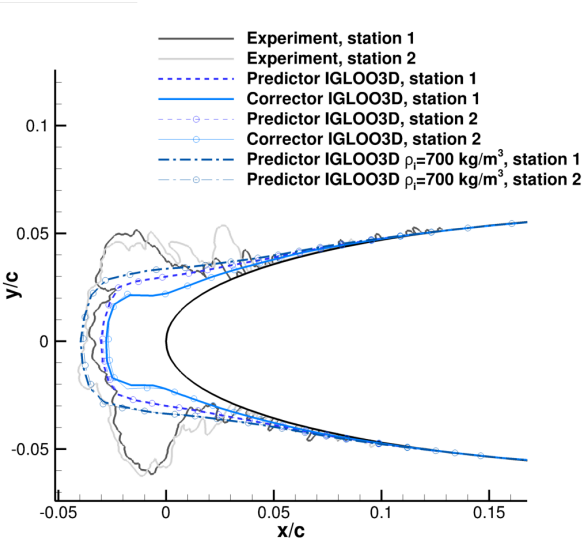


Fig. 36: Case 372: ice shapes produced by IGLOO3D. Comparison against the experimental ice shape [47].

7.5 Case 372

For case 372, the ice thickness predicted by the predictor IGLOO3D simulation is not so bad along the separation line but the use of $\rho_i = 700kg/m^3$ again produces slightly better results. Apart from that, the predicted ice shape is not very satisfactory for this case. As in case 362, the improvement of the heat transfer coefficient modeling (which mainly drives the amount of liquid-water runback) and the use of a better validated ice density could be interesting avenues for future progress. For this case where the scallop structure is quite pronounced, the use of a multi-step method is nevertheless the most relevant way to improve the results.

8 Conclusion

This article presents a workflow for 3D ice-accretion simulations. It is based on the predictor-corrector method, which allows to perform only two loops of the usual quasi-steady computational sequence performed for in-flight-icing modeling. For the correction loop, a remeshing is performed with the Dragon method.

Several test cases are presented to evaluate the method, involving straight and swept wings in Appendix-C icing conditions (and not protected against icing). The predictor-corrector method is very interesting for ice-accretion modeling on straight wings, which is not a surprise since the method is very often used with the 2D icing suite IGLOO2D for example. It is also very encouraging for low-sweep angles. However, this method is not adapted for capturing

scallop-like ice shapes. For highly-swept wings (45° in this article), the method is noticeably less accurate. To correctly model such ice shapes, the multi-step approach is recommended (at the cost of a much higher number of loops of the computational sequence). As an alternative, the development of a specific model of bulk ice density may be able to support the future use of the predictor-corrector approach for the cases where there are well-developed scallops. More generally, the ice shapes could be improved by upgrading the ice-density model (that has been empirically estimated and validated on straight wings). There is also a need to improve liquid-water runback prediction, which implies better modeling the convective heat transfer on the rough surface of the ice.

To conclude, only wing geometries have been tested. It would be useful to use other geometries representative of 3D aircraft structures (however, experimental data are not always available to validate the simulations).

Acknowledgments. The authors would also like to thank François Caminade from DASSAULT-AVIATION for the mesh and data used for the predictor simulation of the GLC305 case 4.

Declarations

- Funding and/or Conflicts of interests/Competing interests: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824310. Partial financial support was also received from DGAC (GENOME project). The authors declare they have no financial interests.

References

[1] Laurendeau, E., Bourgault-Côté, S., Ozcer, I.A., Hann, R., Radenac, E., Pueyo, A.: Summary from the 1st aiaa ice prediction workshop. In: AIAA AVIATION 2022 Forum (2022). https://doi.org/10.2514/6.2022-3398

[2] Gent, R.W., Dart, N.P., Cansdale, J.T.: Aircraft icing. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **358**(1776), 2873–2911 (2000). https://doi.org/10.1098/rsta.2000.0689

[3] Wright, W.B.: User's manual for LEWICE version 3.2. CR 2008-214255, NASA (2008). https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080048307.pdf

[4] Trontin, P., Blanchard, G., Kontogiannis, A., Villedieu, P.: Description and assessment of the new onera 2D icing suite IGLOO2D. In: 9th AIAA Atmospheric and Space Environments Conference (2017). https://doi.org/10.2514/6.2017-3417

[5] Saeed, F., Gouttebroze, S., Paraschivoiu, I.: Modified Canice for Improved Prediction of Airfoil Ice Accretion. In: 8th Aerodynamic Symposium, 48th CASI Conference (2001)

[6] Bourgault-Côté, S., Hasanzadeh, K., Lavoie, P., Laurendeau, E.: Multi-Layer Icing Methodologies for Conservative Ice Growth. In: 7th European Conference for Aeronautics And Aerospace Sciences (EUCASS) (2017). https://doi.org/10.13009/EUCASS2017-258

[7] Gori, G., Zocca, M., Garabelli, M., Guardone, A., Quaranta, G.: PoliM-Ice: A simulation framework for three-dimensional ice accretion. Applied Mathematics and Computation **267**, 96–107 (2015). https://doi.org/10.1016/j.amc.2015.05.081. The Fourth European Seminar on Computing (ESCO 2014)

[8] Beaugendre, H., Morency, F., Habashi, W.: ICE3D, FENSAP-ICE'S 3D in-flight ice accretion module. https://doi.org/10.2514/6.2002-385

[9] Bidwell, C.S., Potapczuk, M.G.: Users manual for the NASA Lewis three-dimensional ice accretion code (LEWICE 3D). Technical report, NASA (1994). NASA/TM–105974

[10] Pena, D., Hoarau, Y., Laurendeau, E.: A single step ice accretion model using level-set method. Journal of Fluids and Structures **65**, 278–294 (2016). https://doi.org/10.1016/j.jfluidstructs.2016.06.001

[11] Lavoie, P., Radenac, E., Blanchard, G., Laurendeau, E., Villedieu, P.: Immersed boundary methodology for multistep ice accretion using a level set. Journal of Aircraft **59**(4), 912–926 (2022). https://doi.org/10.2514/1.C036492

[12] Donizetti, A., Bellosta, T., Rausa, A., Re, B., Guardone, A.: Level-set mass-conservative front-tracking technique for multistep simulations of in-flight ice accretion. Journal of Aircraft **0**(0), 1–11 (0). https://doi.org/10.2514/1.C037027

[13] Al-Kebsi, A., Mose, R., Hoarau, Y.: Multi-Step Ice Accretion Simulation Using the Level-Set Method. In: SAE International Conference on Icing of Aircraft, Engines, And Structure, Minneapolis, United States (2019). https://doi.org/10.4271/2019-01-1955

[14] Capizzano, F., Catalano, P., Carozza, A., Cinquegrana, D., Petrosino, F.: CIRA contribution to the first AIAA Ice Prediction Workshop. https://doi.org/10.2514/6.2022-3400

[15] Certification specifications and acceptable means of compliance for large aeroplanes CS-25. Amendment 17. https://www.easa.europa.eu/en/

downloads/66815/en (2015)

[16] Montreuil, E., Chazottes, A., Guffond, D., Murrone, A., Caminade, F., Catris, S.: ECLIPPS: 1. Three-dimensional CFD prediction of the ice accretion. In: 1st AIAA Atmospheric and Space Environments Conference (2009). https://doi.org/10.2514/6.2009-3969

[17] Radenac, E.: Validation of a 3D ice accretion tool on swept wings of the SUNSET2 program. In: AIAA AVIATION Forum (2016). https://doi.org/10.2514/6.2016-3735

[18] Radenac, E., Gaible, H., Bezard, H., Reulet, P.: IGLOO3D computations of the ice accretion on swept-wings of the SUNSET2 database. In: SAE Technical Paper (2019). https://doi.org/10.4271/2019-01-1935

[19] Messinger, B.L.: Equilibrium temperature of an unheated icing surface as a function of air speed. Journal of the Aeronautical Sciences **20**(1), 29–42 (1953). https://doi.org/10.2514/8.2520

[20] Bourgault, Y., Beaugendre, H., Habashi, W.G.: Development of a shallow-water icing model in FENSAP-ICE. Journal of Aircraft **37**(4), 640–646 (2000). https://doi.org/10.2514/2.2646

[21] Beaugendre, H.: A PDE-based 3D approach to in-flight ice accretion. PhD thesis, McGill University (2003)

[22] Lavoie, P., Pena, D., Hoarau, Y., Laurendeau, E.: Comparison of thermodynamic models for ice accretion on airfoils. International Journal of Numerical Methods for Heat And Fluid Flow **28**(5), 1004–1030 (2018). https://doi.org/10.1108/HFF-08-2016-0297

[23] Laurent, C., Bouyges, M., Charton, V., Bennani, L., Senoner, J.-M.: Ice crystals accretion capabilities of ONERA's 3D icing suite. In: AIAA AVIATION 2022 Forum (2022). https://doi.org/10.2514/6.2022-3537

[24] Chauvin, R., Bennani, L., Trontin, P., Villedieu, P.: An implicit time marching Galerkin method for the simulation of icing phenomena with a triple layer model. Finite Elements in Analysis and Design **150**, 20–33 (2018). https://doi.org/10.1016/j.finel.2018.07.003

[25] Zhu, C., Fu, B., Sun, Z., Zhu, C.: 3D ice accretion simulation for complex configuration basing on improved Messinger model. International Journal of Modern Physics: Conference Series **19**, 341–350 (2012). https://doi.org/10.1142/S2010194512008938

[26] Refloch, A., Courbet, B., Murrone, A., Villedieu, P., Laurent, C., Gilbank, P., Troyes, J., Tessé, L., Chaineray, G., Dargaud, J.B., Quémerais, E.,

Vuillot, F.: Cedre software. AerospaceLab Journal **2** (2011)

[27] Courbet, B., Benoit, C., Couaillier, V., Haider, F., Le Pape, M.-C., Péron, S.: Space discretization methods. AerospaceLab Journal **2** (2011)

[28] Aupoix, B.: Roughness corrections for the k-ω shear stress transport model: Status and proposals. Journal of Fluids Engineering **137**(2) (2014). https://doi.org/10.1115/1.4028122

[29] Toro, E.F., Spruce, M., Speares, W.: Restoration of the contact surface in the HLL-Riemann solver. Shock Waves **4**, 25–34 (1994)

[30] Le Touze, C., Murrone, A., Guillard, H.: Multislope MUSCL method for general unstructured meshes. Journal of Computational Physics **284**, 389–418 (2015). https://doi.org/10.1016/j.jcp.2014.12.032

[31] Murrone, A., Villedieu, P.: Numerical modeling of dispersed two-phase flow. AerospaceLab Journal **2** (2011)

[32] Rutard, N., Dorey, L.-H., Le Touze, C., Ducruix, S.: Large-eddy simulation of an air-assisted liquid jet under a high-frequency transverse acoustic forcing. International Journal of Multiphase Flow **122** (2020)

[33] Le Touze, C., Dorey, L.-H., Rutard, N., Murrone, A.: A compressible two-phase flow framework for large eddy simulations of liquid-propellant rocket engines. Applied Mathematical Modelling **84**, 265–286 (2020)

[34] Eckert, E.R.G., Drake Jr, R.M.: Analysis of Heat and Mass Transfer. Taylor & Francis, London (1987)

[35] Spalding, D.B.: Convective Mass Transfer: an Introduction. McGraw-Hill, New York (1963)

[36] Kays, W.M., Crawford, M.E.: Convective Heat and Mass Transfer. McGraw-Hill, New York (1993)

[37] Smith, A.G., Spalding, D.B.: Heat transfer in a laminar boundary layer with constant fluid properties and constant wall temperature. Journal of the Royal Aeronautical Society **62(565)**, 60–64 (1958). https://doi.org/10.1017/S0368393100067948

[38] Benoit, C., Péron, S., Landier, S.: CASSIOPEE: A CFD pre- and post-processing tool. Aerospace Science and Technology **45**, 272–283 (2015). https://doi.org/10.1016/j.ast.2015.05.023

[39] Makkonen, L., Stallabras, J.R.: Ice accretion on cylinders and wires. TR-LT 005, National Research Council of Canada (1984). https://nrc-publications.canada.ca/eng/view/ft/?id=

7678af66-db1a-430b-b0ff-b1a5dad6b9f9

[40] Macklin, W.C.: The density and structure of ice formed by accretion. Quart. J. Roy. Meteor. Soc. **88**, 30–50 (1962). https://doi.org/10.1002/qj.49708837504

[41] Liou, M.-S., Kao, K.H.: Progress in grid generation: From Chimera to DRAGON grids. Technical Memorandum 106709, NASA (1994). https://ntrs.nasa.gov/citations/19970031503

[42] Vuillot, F., De La Puente, F., Landier, S., Renaud, T., Benoit, C., Sanders, L.: New unstructured grid strategies for applications to aeroacoustic computations of the LAGOON, landing gear model, using the cedre unstructured flow solver. In: 23rd AIAA/CEAS Aeroacoustics Conference (2017). https://doi.org/10.2514/6.2017-3011

[43] Cambier, L., Veuillot, J.-P.: Status of the elsA CFD software for flow simulation and multidisciplinary applications. https://doi.org/10.2514/6.2008-664

[44] Bragg, M., Gregorek, G., Shaw, R.: An analytical approach to airfoil icing. https://doi.org/10.2514/6.1981-403

[45] Broeren, A.P., Potapczuk, M.G., Lee, S., Malone, A.M., Paul, B.P., Woodard, B.S.: Ice-accretion test results for three large-scale swept-wing models in the NASA icing research tunnel. In: 8th AIAA Atmospheric and Space Environments Conference. https://doi.org/10.2514/6.2016-3733

[46] Papadakis, M., Yeong, H.-W., Wong, S.-C., Vargas, M., Potapczuk, M.: Experimental investigation of ice accretion effects on a swept wing. Final Report DOT/FAA/AR-05/39, FAA-WSU-NASA (2005). http://www.tc.faa.gov/its/worldpac/techrpt/ar05-39.pdf

[47] Bidwell, C.S.: Icing Analysis of a Swept NACA 0012 Wing Using LEWICE3D Version 3.48. https://doi.org/10.2514/6.2014-2200

[48] Fujiwara, G.E., Bragg, M.B., Camello, S., Lum, C.: Computational and Experimental Ice Accretions of Large Swept Wings in the Icing Research Tunnel. https://doi.org/10.2514/6.2016-3734

[49] Broeren, A.P., Lee, S., Bragg, M.B., Woodard, B.S., Radenac, E., Moëns, F.: Experimental and computational icing simulation for large swept wings. TP 20210023843, NASA (2022). https://ntrs.nasa.gov/citations/20210023843

[50] Bragg, M.B., Yoshida, W., Broeren, A.P., Lee, S., Woodard, B.S.: Ice Shape Classification for Swept Wings. https://doi.org/10.2514/6.

2020-2845