ICE3D ------- Three-dimensional solver of water film flow and ice accretion

This program numerically solves two Partial Differential Equations (PDEs) which govern water film flow and ice growth on arbitrary three-dimensional curved surface. After calculating the local thickness of the water film and ice layer, geometric reconstruction method is used to generate a new icing surface which serves as a boundary interface for the meshing module.

Physical model and numerical method

This program uses a set of PDEs derived from the lubrication theory (a kind of perturbation method) as the governing equations and an implicit-explicit conservative finite difference method based on the hyperbolic conservation law as the numerical method. The detailed derivation and implementation can be referred to a paper “Extension to the Myers model for calculation of three-dimensional glaze icing”.

The program uses explicit and implicit methods for time marching of the water film governing equation (through the configuration parameter `solutioncontrol`), the actual application on aircraft surface is often accompanied by the occurrence of separated flow. The limit of Courant number makes the implicit method having better robustness in cases of separated flow. The detailed discrete method can also refer to aforementioned paper. Due to the quasi-linearity of the hyperbolic water film governing equations, the thickness of water film may be a shock-like discontinuous solution (representing the advancing front of water film downwind, namely that, the interface between the regions of glaze ice and rime ice), so the numerical scheme of the flow flux at the cell interface is crucial. According to the research of numerical method of the quasi-linear conservation law, this program uses an extended first-order upwind scheme or a TVD (Total Variation Diminishing) scheme which combines first-order upwind and second order Lax-Wendroff scheme for calculating the numerical flux at cell interface (by switching the configuration parameter `discretecontrol`). Although the TVD scheme can obtain second-order precision for water film thickness in continuous regions and a steeper shock in discontinuous regions, a large number of numerical experiments show that the stability of the TVD scheme is very poor in extreme case (time marching divergences, regardless of using explicit or implicit method) due to the complexity of the input parameter distribution on the aircraft surface. Therefore, the extended first-order upwind scheme is the first choice for calculating the numerical flux in cases that problems of numerical instability happen.

For implicit method, there are two methods in this program for solving the linear equations. One is a multi-block point iteration based on Gauss-Seidel method, the other is the iterative methods in a sparse linear solver - HYPRE library of Lawrence Livermore National Laboratory (LLNL). The codes for calling HYPRE library is shown in the source file `Hypresolver.f90`. If you want to use HYPRE library, you should add `Hypresolver.f90` as dependency and provide `HYPREf.h` header file and `libHYPRE.a` or `HYPRE.lib` library file, then modify the options of compiler and linker in `Makefile`.

In the multi-step calculation of ice accretion, changes in ice shape will cause the air flow near the wall to separate at some point. If the water film arrives near the separation point, the larger gradient of air shear stress causes the water film thickness near the separation point to increase dramatically with time. As a result, the lubrication approximation may be invalid. For aircraft icing, this program uses a simple way to deal with this situation, namely that, the part that the water film thickness beyond a certain critical thickness will be completely ignored. Although this sacrifices some mass conservation, it avoids the failure of lubrication approximation and the occurrence of numerical instability. The comparisons with the experiments also show that the ice shape downstream of the separation point is rather irregular, but the effect on the overall ice shape is small due to the thinner local thickness downstream the separation point.

Finally, it should be noted that the numerical solution of the water film thickness is not the actual distribution of the unfrozen water on the aircraft surface (the actual situation is much more complex and depends on the surface physicochemical properties). It can be considered as the average thickness of water films, rivulets and beads over a large scale. Taking into account the input parameters of the model - the air-droplets flow field near the wall obtained by the RANS method is also a large-scale average (Reynolds average), it is reasonable to do the same for runback unfrozen water on icing surface, which result in the large scale icing thickness and ice shape. For the TVD scheme employed at cell interface, the significance of second-order numerical precision and steep shock is not obvious in the sense of the average thickness distribution.

Boundary conditions and time steps

Since the three-dimensional curved surface representing the aircraft surface is mainly closed, the program only considers a symmetrical boundary condition, using a special interface conditions to deal with. This program does not need entering specific boundary parameters for a computational example. The symmetrical boundary condition is applicable to the case of a semi-closed surface with a symmetric surface.

The time step used in this program can only be constant. For the explicit method, in order to maintain numerical stability it should be set to a small enough value. For the implicit method, the limit of the time step will be relaxed and can be set to a larger value.

Input and output files

This program can only read two-dimensional PLOT3D grid files that do not contain the “Blank” tags, which describe the multi-block structured surface mesh that corresponds to each patch on arbitrary three-dimensional curved surface. For the quasi-steady single-step method, only the surface mesh file of initial clean solid wall is needed. For the quasi-steady multi-step method, you also need a surface mesh file of icing wall at previous step. In addition, the program also needs two data files provided by the upper-level solver as input, corresponding to a 2D PLOT3D format solution function file on the wall outputted by an aerodynamic solver (which stores air pressure,

shear stress and surface heat transfer coefficient, etc.), and a 2D PLOT3D format solution function file outputted by a droplet transport solver (which stores the droplet collection coefficient and impact velocity on the wall). The codes for reading the PLOT3D solution files from upper-level solver are shown in source file `Readdata.f03`. The user can modify these codes according to the data format outputted by their own upper-level solver.

After special intervals, the program outputs a block data file that records the numerical solution of the thickness of water film and ice layer at the center of each grid cell. After calculating the new icing shape, 2D PLOT3D format files are used to write the grid coordinates of the new surface and the local thickness of water film and ice layer at each cell, which include a grid coordinates file and a solution function file. At the beginning of the runtime, two data files are generated to write the surface driving force parameters and impingement characteristics for ease of inspection and debugging. After the time is marching, the program will generate a text file that records the state of the numerical solution for each time step and will output two files that record the iteration information at special intervals. One of them reports the iterative case of linear equations (implicit method) at current time step, and the other reports the calculation status of nonlinear heat coefficient for icing equation.

In addition, the block data file generated after a process is completed can be used as the input file for the next running, from which the thickness of water film and ice layer can be directly used as the initial conditions of the next time marching.

Data structure

The data structure of the program can be seen as a subset of the data structure of a 3D multi-block structured aerodynamic solver, which is extracted from the multi-block structured data on the 2D wall. A user-defined type that contains multiple two-dimensional arrays is used as the internal storage format of grid coordinates, the fluid field solution and the discrete matrices. The topological information between blocks and the boundary data for each block are stored using a dedicated user-defined type. The members of these user-defined types, i.e., the arrays are unallocated until the program is in run-time state. The dimension of each member array is allocated according to the content of PLOT3D grid file for minimizing the memory usage. The resulting multi-block data structure can be considered as a one-dimensional array of such user-defined types, and the dimension is determined by the number of blocks which is also allocated at run-time. Because the PLOT3D grid file does not include any topological information between blocks, all the connectivity processing needs to be carried out within the program. This program does not deal with all the possible topological connectivity between blocks, but instead the topology of a general surface patch on a curved surface is used to describe the connectivity. Therefore, this program has special requirements on the topology of a multi-block surface structured mesh. (The new version committed to GitHub has fixed this limitation) The details can be found in Section “Usage”.

Algorithm procedure

Read configuration parameters -> Read data files and memory allocation -> Set constant and free flow parameter -> Preprocessing -> Define topology information -> Calculate boundary field of geometry ->Calculate geometric parameters and external driven forces -> Calculate boundary field of driven forces -> Initialization -> Start time marching -> Calculate heat transfer parameters (optional steps, depending on configuration parameters) -> Calculate flow flux of water film at cell center -> Calculate boundary field of water flow flux -> Calculate wave velocity at cell center -> Calculate boundary field of wave velocity -> Calculate water flow flux at cell interface -> Calculate the thickness of the water film at the next timestep (the marching method is determined by the configuration parameters) -> Calculate the thickness of the ice layer at the next timestep (optional steps, depending on the configuration parameters) -> Calculate boundary field of solutions -> Reconstruct the ice shape (optional steps, depending on the configuration parameters) -> Output result files based on time interval -> Determine whether the water film thickness or ice layer thickness is divergent -> If yes, terminate the program, otherwise continue -> If reach the scheduled time, end the time marching, or else return to the step of "Start time marching"

Source file list

The programming language is Fortran 2003, taking advantage of the new language features such as user-defined types of run-time allocations and pointers. The contents of each source file are:

COM.f03 ---- public module COM, contains the declaration of the program's public data structure and variables
BCtransfer.f03 ---- Subroutine for boundary field calculation of geometric parameters and external driving forces
Constants.f03 ---- Subroutine for physical constants and free flow parameter setting
Cornersmooth.f03 ---- Subroutine for smoothing ice shape boundary of each block
dhbctransfer.f03 ---- Subroutine for boundary field calculation of water film thickness increment
Energyterm.f03 ---- Subroutine for heat transfer parameters calculation
Explicitcompute.f03 ---- Subroutine for marching the thickness of water film and ice layer using explicit method
Fluxf.f03 ---- Subroutine for water film flux calculation at cell interface
Fluxp.f03 ---- Subroutine for water film flux calculation at cell center
Forceparameter.f03 ---- Subroutine for geometric parameters and external forces calculation
Hypresolver.f90 ---- Subroutine for solving sparse linear equations by calling HYPRE library, the Makefile should be modified if this source file is included.
Iceshape.f03 ---- Subroutine for ice shape calculation
Implicitcompute.f03 ---- Subroutine for solving linear equations of water film thickness increment using point relaxation method
Impliciticecompute.f03 ---- Subroutine for marching the thickness of ice layer using implicit method
Initial.f03 ---- Subroutine for variables initialization

innerBCtransfer.f03 ---- Subroutine for calculating the boundary field of variable and water film flux
Main.f03 ---- Main program, root of the calling tree, calling the other subroutines to complete the function of this program, contains the main calculation procedure
Makefile ---- Makefile of this program which can be used by GNU Make.
Mbsolver.f03 ---- Subroutine for solving linear equations of water film thickness incremental using multi-block point iterative method
Pre.f03 ---- Subroutine for preprocessing
Readdata.f03 ---- Subroutine for reading grid coordinates from PLOT3D format mesh file and allocating memory for dynamic data in COM module
Readpara.f03 ---- Subroutine for reading configuration parameters, including the model, algorithm, boundary conditions and time steps
saBCtransfer.f03 ---- Subroutine for boundary field calculation of wave velocity
Saveresults.f03 ---- Subroutine for storage of result files
Topology.f03 ---- Subroutine for specifying the topology of block mesh
Wavevelocity.f03 ---- Subroutine for wave velocity calculation at cell center

Compile and link

This program can be compiled using any Fortran compiler which follows Fortran2003 standard. Because there is only one external dependent library, HYPRE library existing in the program (optional), you can consider adding the directory and name of the `HYPREf.h` header file and the `libHYPRE.a` or `HYPRE.lib` library file into the necessary compiler and linker options (the Makefile in the source package can be used for this purpose).

Usage

The topology connection of multi-block surface mesh in this program is similar with the wall mesh of a common 3D multi-block aerodynamic solver. A one-to-one conformal multi-block surface mesh must be generated on the walls. The whole boundary curve of each block must also be one-to-one conformal. It indicates that this program can not handle the situations that the boundary curve of one block embeds in the boundary curve of another block. But from a topological point of view, any kind of conformal multi-block surface mesh can be split into this form. (The new version committed to GitHub has fixed this limitation)

Because the PLOT3D file does not contain any information of model, algorithm and boundary condition, the configuration parameters and boundary conditions of this program must be set by the user. This function is implemented in the source file `Readpara.f03`. The current default form is the script mode, which means you need to prepare a script file named `Script.txt` which includes all the information you can find in `Readpara.f03`. Such a script file should be organized into an interlaced format, that is, a line of input data immediately following the message line. The detailed formats and contents can be found in the demo script file in the source code

package. In addition, this program can also accept parameters through an interactive mode, you only need to switch the string variable `Scpt` from 'Y' to 'N' in `Readpara.f03`, you can input data interactively through the command line terminal console when running the program.

Most of the data structures in the program are described in the public module `COM` in source file `COM.f03`. In order to facilitate understanding, the meanings of variable identifiers are given by the comments in `COM.f03`. In the process of using this program, many source-level details may be encountered, but they can't be mentioned here and you should find the answer from the codes in source files. The two demo examples shipped with the source code package are given in the form of their PLOT3D surface mesh files, data files from upper-level solver, and script files. They can be used for code debugging and validation. One of these examples is the calculation of ice accretion on the windward side of a sphere, and the other is the calculation of the icing at the leading edge of a GLC-305 swept wing.

Please report bugs or errors to hs180934@163.com. For any questions and comments, please also email hs180934@163.com.