

NASA/TM-20240008077



GlennICE Manual 4.2.0

*Christopher Porter, Mark Potapczuk, Thomas Ozoroski, and Zaid Sabri
Glenn Research Center, Cleveland, Ohio*

*Eric Galloway, David Rigby, and William Wright
HX5, LLC, Brook Park, Ohio*

*Paul Tsao
Ohio Aerospace Institute, Brook Park, Ohio*

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.**
Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.**
Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- **CONTRACTOR REPORT.**
Scientific and technical findings by NASA-sponsored contractors and grantees.
- **CONTRACTOR REPORT.**
Scientific and technical findings by NASA-sponsored contractors and grantees.
- **CONFERENCE PUBLICATION.**
Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.**
Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.**
English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>



GlennICE Manual 4.2.0

*Christopher Porter, Mark Potapczuk, Thomas Ozoroski, and Zaid Sabri
Glenn Research Center, Cleveland, Ohio*

*Eric Galloway, David Rigby, and William Wright
HX5, LLC, Brook Park, Ohio*

*Paul Tsao
Ohio Aerospace Institute, Brook Park, Ohio*

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

This report is available in electronic form at <https://www.sti.nasa.gov/> and <https://ntrs.nasa.gov/>

NASA STI Program/Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

Contents

Contents	i
List of Abbreviations	vii
List of Symbols	viii
About This Document	1
Acknowledgment	2
1 Introduction	3
2 Installation	5
2.1 Supported Platforms	5
2.2 Requirements	5
2.3 Build Instructions	5
2.4 Installation	6
2.5 Usage	6
2.6 Prebuilt Modules	7
3 Command Line Input - Serial Build	8
3.1 Normal Run from Scratch	8
3.2 Controlling Scope of Simulation with JobType	8
3.3 Restart Run	8
3.3.1 Volume Restart	8
3.3.2 Surface Restart	9
3.3.3 Refinement Restart	9
3.3.4 Skip Writing of Volume Restart	9
3.4 Informational Output	10
3.5 Version Information	10
4 Command Line Input - MPI Builds	11
4.1 Normal Run from Scratch	11
4.2 Controlling Scope of Simulation with JobType	11
4.3 Restart Run	11
4.3.1 Volume Restart	11
4.3.2 Surface Restart	12
4.3.3 Refinement Restart	12
4.3.4 Skip Writing of Volume Restart	13
4.4 Informational Output	13
4.5 Version Information	13
5 Getting Started	14

5.1	Namelist File	14
5.2	Specification of the CFD Solution	15
5.2.1	One Solution CFD Methodology	15
5.2.2	Two Solution CFD Methodology	15
5.3	Controlling Trajectories	16
5.4	Specification of Cloud Distribution	16
5.5	Output Files	17
6	Input Files	19
6.1	Namelist File	19
6.1.1	&files Namelist	19
6.1.2	&surface_count Namelist	21
6.1.3	&surface_nml Namelist	21
6.1.4	&trajectory_solver Namelist	22
6.1.5	&release Namelist	25
6.1.6	&mass_flux_solver Namelist	26
6.1.7	&adaptive_refinement Namelist	27
6.1.8	&bin_count Namelist	28
6.1.9	&distribution Namelist	29
6.1.10	&freestream Namelist	29
6.1.11	&htc_augmentation Namelist	30
6.1.12	&volume_output Namelist	31
6.1.13	&conversions Namelist	31
6.1.14	&variable_properties Namelist	32
6.1.15	&rotation Namelist	33
6.1.16	&runback Namelist	34
6.2	Supported CFD File Formats	34
6.2.1	List of Variables Read from CFD Solution File	36
6.2.2	Specification of Input Variable Locations	38
6.3	Trajectories Job Input File	38
7	Output Files	40
7.1	Volume Restart File	40
7.2	Surface Restart File	40
7.3	Refinement Restart File	40
7.4	Stereolithography Files	40
7.5	Tecplot File	41
7.5.1	List of Output Variables	42
7.6	Refinement Out Files	43
7.6.1	List of Output Variables	44
8	Example Cases	45
8.1	ONERA M6 Input Flow Solution	45
8.2	Single Bin Ice Accretion	47
8.3	Refinement Restart	53

8.4	Droplet Distribution (Multi-Bin) Ice Accretion	55
8.5	Surface Restart	59
8.5.1	Multiple Single-Bin Combination	61
8.6	Trajectory Visualization	65
9	User Tips	68
9.1	Compiling	68
9.2	Namelist	68
9.3	Adaptive Refinement	68
9.3.1	Convergence and Efficiency Parameters	72
9.3.2	Convergence Metrics	73
9.3.3	Efficiency Metrics	75
9.4	Distribution	77
9.4.1	Comparison of Multiple Drop Diameters	77
9.5	Heat Transfer Augmentation	78
9.6	Ice Density	78
9.7	Conversions	79
9.8	Flow Solvers	79
9.9	FUN3D Considerations and Workflow	80
9.9.1	Tecplot Output	80
9.9.2	FUN3D Workflow	80
9.9.2.1	FUN3D Simulation Requirements	80
9.10	Seed Points and Release Points	82
9.11	Uniformly Refining Release Points	83
9.12	Optimizing Initial Release Points	84
9.13	Restart	84
9.14	Write Trajectories	84
9.15	Fraction_contained_tol	85
9.16	Parallel Computing	85
9.16.1	Static vs Dynamic Scheduling	87
9.17	Non-inertial Reference Frame	89
9.18	Known Issues	89
9.18.1	Known Issue 1	89
9.18.2	Known Issue 2	90
9.18.3	Known Issue 3	90
9.18.4	Known Issue 4	90
9.18.5	Known Issue 5	91
9.18.6	Known Issue 6	91
	Bibliography	92
	GlennICE Publications	94

List of Figures

5.1	Flow chart of the GlennICE process.	14
5.2	The iced geometry as stored in a Stereolithography output file.	18
5.3	The ice only and clean geometry as stored in a Stereolithography output file.	18
5.4	Collection efficiency of a single bin distribution on a NACA 0012 at a 4 degree angle of attack.	18
8.1	ONERA M6 Computational Grid (3D) with Symmetry Plane	45
8.2	C_p contours for the ONERA M6 case listed in Table 8.1	47
8.3	Single-Bin Collection Efficiency / beta_total	51
8.4	ONERA M6 Single-Bin 3D Ice Shape	52
8.5	pct_converged_limit effect on ONERA M6 3D Ice Shape	55
8.6	Results of changing pct_converged_limit at a slice location of $Y = 0.75$ m.	55
8.7	7-Bin MVD = $20\text{ }\mu\text{m}$ Langmuir-D Droplet Distribution Collection Efficiency.	57
8.8	7-Bin MVD = $20\text{ }\mu\text{m}$ Langmuir-D Droplet Distribution Collection Efficiency at $Y = 0.75$ m.	57
8.9	MVD = $20\text{ }\mu\text{m}$ 7-Bin Langmuir-D Droplet Distribution Ice Shape.	58
8.10	Single-Bin versus 7-Bin Langmuir-D Ice Shape Comparison.	58
8.11	Ice Shape for a 7-Bin MVD = $20\text{ }\mu\text{m}$ Langmuir-D icing_temperature adjustment.	62
8.12	Visualization of a Sample of $20\text{ }\mu\text{m}$ Droplet Trajectories.	67
9.1	Geometry and surface mesh of the annular test case.	69
9.2	Schematic of the "minimum wall distance" associated with a trajectory.	69
9.3	A two dimensional contour of minimum wall distance on the seed plane.	70
9.4	A cloud of seed points and the resulting mesh.	70
9.5	The global minimum search as performed on two sequential levels of adaptive refinement.	71
9.6	Collection Efficiency Contours for Annulus Case.	72
9.7	Collection Efficiency at Azimuthal Locations for Annulus Case.	73
9.8	Seed Point Grid for Annulus Case.	73
9.9	Hit Surfaces on Seed Plane Grid.	74
9.10	Collection Efficiency Variation Based on Efficiency Metrics.	75
9.11	Symmetry of Collection Efficiency Result.	76
9.12	Symmetry of Surface Pressure.	76
9.13	Refinement and collection efficiencies of iteration sixteen of the mass flux solver for three different discrete drop diameters with identical &release and &adaptive_refinement parameters.	78

9.14 Rectangular seed point grid and sample geometry.	83
9.15 Projection of seed points onto the Inlet surface.	83
9.16 Schematic showing definition of Fraction Contained.	85
9.17 Parallel Scalability (Efficiency).	89

List of Tables

6.1	List of CFD File Variants and Their Support Status within GlennICE.	36
6.2	Input variables, GlennICE recognized spellings, variable descriptions, map index.	37
6.3	Input variables and whether it is needed on the surface or volume.	38
8.1	CFD Run Conditions	45
8.2	Icing Conditions	48
8.3	Langmuir-D MVD = 20 μm Droplet Distribution	56
9.1	Comparison of number of trajectories and total impinging mass flow for the annular test case	77

List of Abbreviations

AATT	Advanced Air Transport Technology
AAVP	Advanced Air Vehicle Program
ARM D	Aeronautics Research Mission Directorate
CFD	Computational Fluid Dynamics
CGNS	CFD General Notation System
FUN3D	Fully Unstructured Navier-Stokes 3D
GlennICE	Glenn Icing Computational Environment
IRT	Icing Research Tunnel
MCCS	Maximum Combined Cross Section
MVD	Median Volume Diameter
MPI	Message Passing Interface
STL	Stereolithography
SZL	Tecplot Subzone Data
TACP	Transformative Aeronautics Concepts Pro- gram
TRL	Technology Readiness Level
TTT	Transformational Tools and Technologies
TWC	Total Water Content

List of Symbols

ρ	Density
ρ_{ice}	Ice Density
p	Pressure
p_0	Stagnation Pressure
u	x-Component of Velocity
v	y-Component of Velocity
w	z-Component of Velocity
x	x-Coordinate
y	y-Coordinate
z	z-Coordinate
T_{Wall}	Wall Temperature
Q_{Wall}	Wall Heat Flux
$x_{Wall\ Shear}$	x-Component of Wall Shear Stress
$y_{Wall\ Shear}$	y-Component of Wall Shear Stress
$z_{Wall\ Shear}$	z-Component of Wall Shear Stress
htc	Heat Transfer Coefficient
T_{AW}	Adiabatic Wall Temperature
T_0	Stagnation Temperature
V_∞	Freestream Velocity
AOA	Angle of Attack

About This Document

This manual is intended to provide the information necessary for an applications engineer to configure, compile, and execute the GlennICE software. The intent of this document is not to detail the algorithms of the software. The intent of this document is to detail how a user can interact and configure the algorithms of this software in order to produce a desired simulation.

Please contact the GlennICE team at GlennICE-support@lists.nasa.gov for any software suggestions and improvements, or if you find any errors or have any difficulties with the contents of this document.

Acknowledgment

The contributors of GlennICE would like to thank the Advanced Air Transport Technology (AATT) project of the Advanced Air Vehicle Program (AAVP) as well as the Transformational Tools and Technologies (TTT) project of the Transformative Aeronautics Concepts Program (TACP) for the funding for this effort. Both of these programs are under the Aeronautics Research Mission Directorate (ARMD) at NASA.

This document was generated thanks to the contributions of various members of the GlennICE effort. These contributors are:

Eric Galloway	– v0.1 - Present	– HX5 LLC
Thomas Ozoroski	– v2.2 - Present	– NASA Glenn Research Center
Christopher Porter	– v0.1 - Present	– NASA Glenn Research Center
Mark Potapczuk	– v0.1 - v2.1	– NASA Glenn Research Center
David Rigby	– v0.1 - Present	– HX5 LLC
Zaid Sabri	– v2.2 - Present	– NASA Glenn Research Center
Paul Tsao	– v0.1 - Present	– Ohio Aerospace Institute
William Wright	– v0.1 - Present	– HX5 LLC

1 Introduction

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

For those readers familiar with NASA's other computational icing tools, LEWICE [1] and LEWICE3D [2], they should note that GlennICE is being designed to address a generalized three dimensional icing problem. This differs from LEWICE and LEWICE3D, both of which relied on assumptions that are well suited for performing analysis on special cases, such as cantilevered wings, but not necessarily the optimal approach for the general case. To date GlennICE has been applied to several cases of interest such as the CRM, the Ice Prediction Workshop, as well as aiding in the design of wind-tunnel models. Some of these applications included comparisons to LEWICE3D, in which the results compared qualitatively favorably to the legacy tool. However, these comparisons are largely unpublished. It is appreciated that LEWICE3D has a significant amount of usage and acceptance as a computational icing tool for use in design and certification. Due to the lack of usage and validation of GlennICE the Technology Readiness Level (TRL) [3] is considered low, however the tool has proved quite useful and less cumbersome than LEWICE3D. The authors encourage the usage of GlennICE but note that care should be taken when being used as a design and certification tool.

For those readers not familiar with NASA's other computational icing tools, it may be useful to appreciate that history. In the 1980's NASA began the development of a computational tool for ice accretion modeling which would eventually become the LEWICE software. LEWICE is a two-dimensional ice accretion code which is based upon a potential flow solution, a Lagrangian droplet trajectory calculation, and the Messinger model [4] for ice growth. This software has been validated through comparisons to a large database of experimental ice shapes and collection efficiency measurements [5]. LEWICE runs quickly and efficiently on a standard laptop computer and version 3.2.3 is currently available in the NASA software repository [6].

The next step in development of ice accretion software at NASA was the LEWICE3D software. This software performs ice growth calculations along user specified two-dimensional cuts or along surface streamlines using a computational approach for the ice growth calculation derived from the two-dimensional LEWICE software. LEWICE3D takes a three-dimensional so-

lution from a user supplied CFD solution and performs a three-dimensional Lagrangian droplet trajectory calculation to determine the water impingement at all locations on the geometry of interest. The aforementioned ice growth calculation method is then performed to get ice shape profiles at discrete locations on the surface. LEWICE3D has also been validated through comparison to a more limited database of three-dimensional ice shapes [7]. Version 3.6.3 is currently available in the NASA software repository [8].

GlennICE is the next step in the development of ice accretion software at NASA. In order to enable users to utilize the CFD tool that they have the most experience with and the greatest confidence in, GlennICE has been developed to be extensible to use various common CFD solution file formats as input. See **Section 6.2 - Supported CFD File Formats** for a discussion on the currently supported file formats, as well as file formats that are currently unsupported, but future support is planned.

The software then proceeds to calculate the ice shape for the geometry used in the CFD analysis. GlennICE calculates the full three-dimensional droplet trajectory using the Lagrangian method and then proceeds to perform the mass and energy balance on all discrete surface elements that have had water input, whether through direct impact or through runback from adjacent elements. The water runback along the surface is determined by the shear stress imparted to the water from the surrounding airflow. The ice growth at each element is then used to create a new three-dimensional ice surface. The new surface is output in a format suitable for use by the grid generation tool employed by the user in order to perform the next time step in the ice growth process.

From a software development standpoint, the development team chose to build GlennICE from the ground up. This enabled the use of more modern software development practices which have made GlennICE modular in nature and well-documented. This has allowed the software to be created in a team based approach thus preserving continuity of development as the composition of the team varies over the life of the software. As both LEWICE and LEWICE3D have evolved over decades of development, it is important that this sustainability be incorporated into GlennICE development from the beginning.

As a consequence of the decision to create a completely new code, there are still some features that are not currently available in the early versions of the software. The main features added in version 4.2.0 include the following:

- Expanded variable output for the `refinement_out` file.
- Addressed bugs that were identified.
- Improved robustness of multiple routines.

A full list of current and previous changes can be found in the `change_log.md` file included within the GlennICE 4.2.0 distribution.

2 Installation

GlennICE must be built from source; it is not available as a pre-built binary.

2.1 Supported Platforms

GlennICE builds are supported on these Unix-like platforms:

- Apple OSX
- Linux
- Windows Subsystem for Linux

2.2 Requirements

Requirements for compiling GlennICE from source:

- CMake
- GNU Make
- A Fortran compiler
- A C compiler
- A C++ compiler

The GNU Fortran and Intel Fortran compilers have been used for development; their use will be supported. GlennICE uses some Fortran 2008 features. As such, a fairly recent version of a Fortran compiler is required. GNU Fortran 11.x and Intel 19.x were used for development of version 4.2.0. There were occasional segmentation faults when compiling with GNU Fortran 11.1 that did not occur with GNU Fortran 10.3 or GNU Fortran 11.2. If the user encounters errors of this kind, they should report them to GlennICE-support@lists.nasa.gov.

2.3 Build Instructions

The GlennIce source distribution is available as a zip file. For version 4.2.0, the distribution will be named `glennice-v4.2.0.zip`. To build GlennICE version 4.2.0:

```
$ unzip glennice-v4.2.0.zip
$ cd glennice-v4.2.0
$ mkdir build
$ cd build
$ cmake ../
$ make
```

This will create an executable named `glennice` in the build directory. If the user wants to build GlennICE for multiple processors, simply add `-DENABLE_MPI=YES` to the `cmake` command:

```
$ cmake ../ -DENABLE_MPI=YES
$ make
```

Additionally, the user can specify the compiler to use at the `cmake` command:

```
$ FC=gfortran CC=gcc CXX=gcc cmake ../
$ make
```

2.4 Installation

Once you have successfully compiled the software, a desire to move the executable location may exist depending on your system requirements or workflow. A few options exist for the user and have been described below.

Option 1: Do nothing. Since the GlennICE executable has been successfully created, the user can simply execute GlennICE from anywhere using the executable generated at the install location with the usage of:

```
$ ./<path-to-install-directory>/build/glennice glennice.nml
```

Option 2: Copy the GlennICE executable that is created `glennice` to a desired executable location:

```
$ cp glennice <desired-executable-directory>
```

Option 3: Setup an alias or environment variable that points to the `glennice` executable. If you are not using a `bash` shell, the files accessed change slightly. To see which shell you are using type `echo $SHELL` into your command line. Add the following line to your shell startup file (`.bashrc`) that is sourced upon loading a command window:

```
$ alias glennice='<path-to-glennice-directory>/build/glennice'
```

There is an extension of this workflow to setup `glennice` as an environment variable which will behave similarly to Option 3. This process has not been provided since it can change significantly depending on the environment being used and will likely be more difficult to implement for most users.

2.5 Usage

The following executions of the GlennICE software assume that you have generated an alias or environment variable that directly accesses the `glennice`

executable. If this has not been setup using Option 3 in **Section 2.4 - Installation**, then you will need to prepend **glennice** with the executable location. Execution of **glennice** is controlled by a namelist file:

```
$ glennice <glennice-namelist-file>
```

Execution of **glennice** using mpi is invoked using the following command:

```
$ mpiexec -np x glennice <glennice-namelist-file>
```

Where **x** is replaced by the number of processors requested. An extended sample namelist file can be found in the **doc** directory.

2.6 Prebuilt Modules

Prebuilt GlennICE modules are available on the NASA Advanced Supercomputing (NAS) environment. Please contact GlennICE support GlennICE-support@lists.nasa.gov to get instructions and information on how to use these modules.

3 Command Line Input - Serial Build

3.1 Normal Run from Scratch

The simplest way to execute a job is to input the executable name followed by the name of the namelist file. The namelist file contains the user provided input necessary to configure a GlennICE simulation. **Section 6.1 - Namelist File** provides a detailed discussion on the specifics of the available user input.

```
$ glennice <glennice-namelist-file>
```

3.2 Controlling Scope of Simulation with JobType

By specifying `jobType` on the command line the user can control the extent of the simulation. Choices for `jobType` include `full`, `collection`, `trajectories` and `mass_and_energy`.

For `jobType=full` (the default), all phases of a normal run are completed. For `jobType=collection`, only surface collection is completed, without any energy balance or ice growth. For `jobType=trajectories`, only individual trajectories are calculated from the file `particles_input.txt`. For `jobType=mass_and_energy`, the collection and energy balance are completed but no ice growth is attempted.

To specify a collection only run at the command line, the command looks like:

```
$ glennice <glennice-namelist-file> jobType=collection
```

3.3 Restart Run

Three types of restart are possible. A **volume restart** which saves recalculation of connectivity, a **surface restart** which saves recalculation of collection and a **refinement restart** which allows the user to add more trajectory refinement to an existing run.

3.3.1 Volume Restart

Once an initial run has completed, it will write out a volume restart. The name of the file is determined by the parameter `glennice_file_in` in the files namelist. If this file exists then the code can be run as a volume restart job. By running as a volume restart, time can be saved by not recalculating connectivity. Whether the time savings is significant will depend on the size of the problem.

To run a volume restart job, the command line looks like:

```
$ glennice <glennice-namelist-file> restartJob=Volume
```


3.3.2 Surface Restart

Once an initial run has completed, GlennICE will write out a file called `GlennICE_Surface_Restart_xpx.dat` where `xpx` refers to the particle size from the input file. For example if the user specified a 20 micron particle size the file would be named `GlennICE_Surface_Restart_20p0.dat`

The code can be then be run as a surface restart job. By running as a surface restart, significant time can be saved by not recalculating collection efficiency.

Parameters that do not affect collection can be modified, allowing the mass balance and ice growth to be recalculated. For example, the user could adjust `twc`, `relhumidity`, `icing_temperature`, and `time`.

To run a surface restart job, the command line looks like:

```
$ glennice <glennice-namelist-file> restartJob=Surface
```

For a case with a drop distribution, the user can run each individual bin as a separate case and then perform a surface restart case where the entire drop distribution is supplied. This process is described in detail within **Section 8.5.1 - Multiple Single-Bin Combination**.

3.3.3 Refinement Restart

Once an initial run has completed, GlennICE will write out a file called `GlennICE_Refinement_Restart_xpx.dat` where `xpx` refers to the particle size from the input file. For example if the user specified a 20 micron particle size the file would be named `GlennICE_Refinement_Restart_20p0.dat`

The code can be then be run as a refinement restart job. By running as a refinement restart, significant time can be saved by using the existing trajectories to refine collection efficiency. This restart case differs from the surface restart case. In the refinement restart case, additional trajectories will be calculated based on the parameters in the adaptive refinement namelist which will change mass impingement and collection efficiency. A refinement restart job can only be conducted on single bin jobs, the ability to restart a multi-bin job has not been implemented yet.

To run a refinement restart job, the command line looks like:

```
$ glennice <glennice-namelist-file> restartJob=Refinement
```

3.3.4 Skip Writing of Volume Restart

If it is known that a volume restart will not be needed, then time and disk space can be saved by not bothering to write the volume restart file.

To inhibit the writing of the volume restart file the command looks like:

```
$ glennice <glennice-namelist-file> restartJob=  
skip_restart_write
```

3.4 Informational Output

If the executable is run with no arguments, or with the argument `-h` or `-help`, an informational message will be output to the screen.

3.5 Version Information

If the executable is run with the `-v` or `-version` argument, then the version number and release date are printed to the screen.

4 Command Line Input - MPI Builds

4.1 Normal Run from Scratch

The simplest way to execute a job is to input the executable name followed by the name of the namelist file. The namelist file contains the user provided input necessary to configure a GlennICE simulation. **Section 6.1 - Namelist File** provides a detailed discussion on the specifics of the available user input.

```
$ mpiexec -np x glennice <glennice-namelist-file>
```

Where **x** is replaced by the number of processors requested.

4.2 Controlling Scope of Simulation with JobType

By specifying **jobType** on the command line the user can control the extent of the simulation. Choices for **jobType** include **full**, **collection**, **trajectories** and **mass_and_energy**.

For **jobType=full** (the default), all phases of a normal run are completed. For **jobType=collection**, only surface collection is completed, without any energy balance or ice growth. For **jobType=trajectories**, only individual trajectories are calculated from the. For **jobType=mass_and_energy**, the collection and energy balance are completed but no ice growth is attempted.

To specify a collection only run at the command line, the command looks like:

```
$ mpiexec -np x glennice <glennice-namelist-file> jobType=
  collection
```

4.3 Restart Run

Three types of restart are possible. A **volume restart** which saves recalculation of connectivity, a **surface restart** which saves recalculation of collection and a **refinement restart** which allows the user to add more trajectory refinement to an existing run.

4.3.1 Volume Restart

Once an initial run has completed, GlennICE will write out a volume restart. The name of the file is determined by the parameter **glennice_file_in** in the files namelist. If this file exists then the code can be run as a volume restart job. By running as a volume restart, time can be saved by not recalculating connectivity. Whether the time savings is significant will depend on the size of the problem.

To run a volume restart job, the command line looks like:

```
$ mpiexec -np x glennice <glennice-namelist-file> restartJob=
Volume
```

4.3.2 Surface Restart

Once an initial run has completed, GlennICE will write out a file called `GlennICE_Surface_Restart_xpx.dat` where `xpx` refers to the particle size from the input file. For example if the user specified a 20 micron particle size the file would be named `GlennICE_Surface_Restart_20p0.dat`. Note that if the user specifies additional significant decimal places, then more digits will appear to the right of the "p".

The code can be then be run as a surface restart job. By running as a surface restart, significant time can be saved by not recalculating collection efficiency.

Parameters that do not affect collection can be modified, allowing the mass balance and ice growth to be recalculated. For example, the user could adjust `twc`, `relhumidity`, `icing_temperature`, and `time`.

To run a surface restart job, the command line looks like:

```
$ mpiexec -np x glennice <glennice-namelist-file> restartJob=
Surface
```

For a case with a drop distribution, the user can run each individual bin as a separate case and then perform a surface restart case where the entire drop distribution is supplied. This process is described in detail within **Section 8.5.1 - Multiple Single-Bin Combination**.

4.3.3 Refinement Restart

Once an initial run has completed, GlennICE will write out a file called `GlennICE_Refinement_Restart_xpx.dat` where `xpx` refers to the particle size from the input file. For example if the user specified a 20 micron particle size the file would be named `GlennICE_Refinement_Restart_20p0.dat`

The code can be then be run as a refinement restart job. By running as a refinement restart, significant time can be saved by using the existing trajectories to refine collection efficiency. This restart case differs from the surface restart case. In the refinement restart case, additional trajectories will be calculated based on the parameters in the adaptive refinement namelist which will change mass impingement and collection efficiency. A refinement restart job can only be conducted on single bin jobs, the ability to restart a multi-bin job has not been implemented yet.

To run a refinement restart job, the command line looks like:

```
$ mpiexec -np x glennice <glennice-namelist-file> restartJob=
Refinement
```

4.3.4 Skip Writing of Volume Restart

If it is known that a volume restart will not be needed, then time and disk space can be saved by not bothering to write the volume restart file.

To inhibit the writing of the volume restart file the command looks like:

```
$ mpiexec -np x glennice <glennice-namelist-file>
restartJob=skip_restart_write
```

4.4 Informational Output

If the executable is run with no arguments, or with the argument `-h` or `-help`, an informational message will be output to the screen.

4.5 Version Information

If the executable is run with the `-v` or `-version` argument, then the version number and release date are printed to the screen.

5 Getting Started

This section is intended to describe the workflow of an initial GlennICE run to aid the user in becoming familiar with the GlennICE software. The complete workflow for the GlennICE software is illustrated in Figure 5.1. The white boxes in the process are external to GlennICE and accomplished with third party software. Surface redefinition The red, orange, yellow, green, and gray boxes denote work accomplished by the GlennICE software.

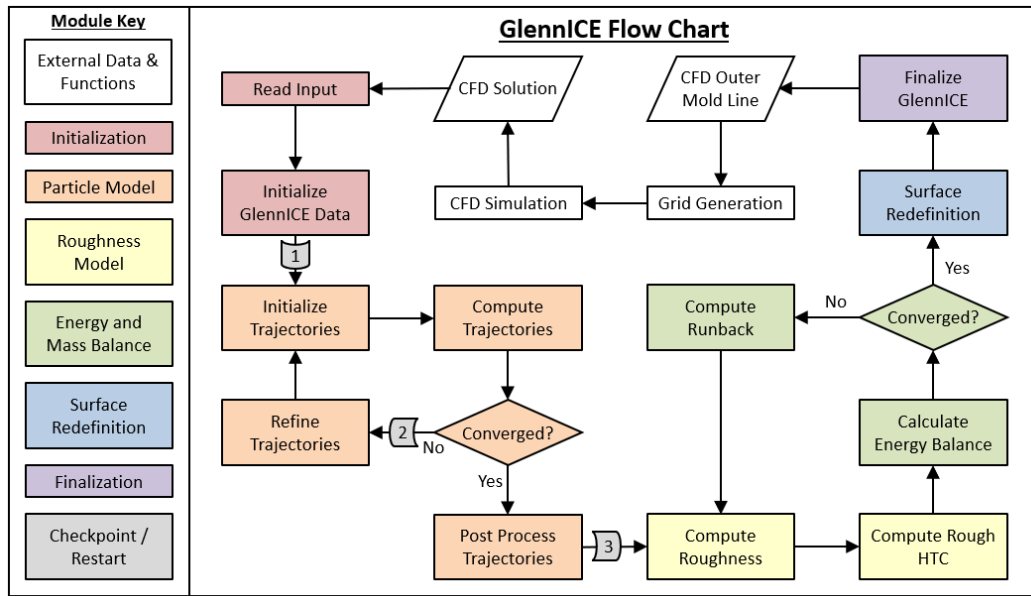


Figure 5.1: Flow chart of the GlennICE process.

This flow chart illustrates a multi-shot simulation strategy where the user is responsible for remeshing the new iced geometry and rerunning the Computational Fluid Dynamics (CFD) solver; this is shown in Blue. This multi-shot capability is optional, and single shot computation can be run if the user does not wish to perform additional meshing/CFD.

5.1 Namelist File

Prior to reading this section, the reader is encouraged to read **Chapter 3 - Command Line Input - Serial Build** for details on interfacing with GlennICE from the command line. A namelist file is used to allow the user to communicate with GlennICE. The example below assumes that GlennICE is in the environments `PATH`. Here, the user points to a file called `glennice_4degAoA.nml`. This file is in the same working directory as GlennICE. Also the output is optionally redirected to a log file, `glennice.log`, rather than to the screen.

```
$ glennice glennice_4degAoA.nml > glennice.log
```

Section 6.1 - Namelist File of this document contains a detailed description of the parameters that can be specified in the namelist file.

5.2 Specification of the CFD Solution

GlennICE uses a discretized flow field generated from a Computation Fluid Dynamics (CFD) solver. To maintain flexibility with a wide variety of commercial and in-house CFD solvers, the coupling between GlennICE and the CFD software is currently all one way coupling. As such GlennICE can be classified as a CFD post-processor. Currently this level of coupling has been sufficient for external icing problems.

Because of the one way coupling, GlennICE can theoretically accept solutions from a wide variety of CFD solvers. It does this by supporting a handful of widely used CFD file formats that a majority of CFD solvers utilize. However, it should be noted that there are many storage variants. **Section 6.2 - Supported CFD File Formats** of this document describes the variants that are currently supported.

5.2.1 One Solution CFD Methodology

A user can specify a single CFD solution for GlennICE post processing. This is specified in the `&files` namelist. The example below specifies the filename of a file corresponding to a NACA 0012 solution at an angle of attack of 4 degrees with a 20°C wall temperature. This file is in the current working directory.

```
&files
  solution_in = "AoA4_Twall_20.fvuns"
/
```

The single solution methodology assumes the CFD solver provided a value of heat transfer coefficient, and that this value accounts for the icing considerations that impact the prediction of heat transfer coefficient.

5.2.2 Two Solution CFD Methodology

Alternatively, the user can provide GlennICE with two CFD solutions differing only by the wall thermal boundary condition. GlennICE uses this information to calculate heat transfer coefficient and adiabatic wall temperature. It is assumed that the heat transfer coefficient, h_{tc} , and adiabatic wall temperature, T_{aw} , are independent of wall thermal condition. Therefore, at every location, given (T_1, q_1) and (T_2, q_2) two simultaneous equations are solved for h_{tc} and T_{aw} . This technique relies on the definition of h_{tc} from the equation:

$$q = h_{tc} * (T_{wall} - T_{aw}) \quad (5.1)$$

The user can specify these two files using the `&files` namelist similarly to the example below. Note that the two files consist of identical meshes, however, the two CFD solutions differ in their specification of the temperature of the surface. One solution enforces a 20°C surface temperature, while the other solution enforces a surface temperature of 30°C.

```
&files
  solution_in  = "AoA4_Twall_20.fvuns"
  solution2_in = "AoA4_Twall_30.fvuns"
/
```

5.3 Controlling Trajectories

The `&release` namelist can be used to provide control over the number and location of the release points used to generate trajectories. The current version of the software assumes that the flow direction is primarily in the x-direction. Through the `&release` namelist, the user can specify the location and refinement level of a box of equally spaced trajectories in the *YZ* plane. If the user does not specify the location of the box, then the maximum and minimum *Y* and *Z* values of the bounding box of the entire domain are utilized to define the size and location of this box.

The default values of the `&release` namelist (seen in **Section 6.1.5 - `&release` Namelist**) construct nine seed points in a three by three configuration within a box in a *YZ* plane defined by the bounding box of the domain.

These trajectories are then projected on the **Inlet** surface(s) and the solution of the trajectory begins from those location(s) on the **Inlet** surface(s). It should be noted that the user must specify the Inlet surfaces in the `&surface_nml` namelist as there currently is not a method for GlennICE to automatically glean this information from the CFD solution file. **Section 9.10 - Seed Points and Release Points** provides some diagrams that illustrate the differences between seed points and the release points they generate on the **Inlet** surface(s).

5.4 Specification of Cloud Distribution

GlennICE has the ability to run a single discrete drop diameter or approximate a continuous distribution with the use of multiple discrete bins, each represented by a discrete drop diameter. This is done through the use of two namelists: the `&bin_count` namelist, and the `&distribution` namelist. The example below illustrates the specification of a Langmuir "D" distribution with a Median Volume Diameter (MVD) of 20 microns.

```
&bin_count
  nbins = 7
```



```

/
&distribution
  diameter          = 6.2, 10.4, 14.2, 20., 27.4, 34.8,
    44.4
  mass_fraction     = 0.05, 0.10, 0.20, 0.30, 0.20, 0.10,
    0.05
/

```

The first namelist, the `&bin_count` namelist, specifies the number of discrete droplet bins that discretely represent a continuous cloud distribution. This number sets the size of the arrays in the `&distribution` namelist.

The `&distribution` namelist specifies size and mass fraction of each bin. Note that a given bin's size is specified using droplet diameter in microns. Previous versions of GlennICE allowed for an input variable called `reference_diameter` which scaled the diameter values, but users found this confusing.

5.5 Output Files

In order to visualize the simulated data, the user must specify the filenames where the user wants these pieces of information stored. If the filenames are not specified by the user, the simulated data is not saved. **Chapter 7 - Output Files**, of this document describes the contents and purpose of the output files in greater detail. The example below generates three solution files in the current working directory. The first file is an Stereolithography (STL) file that contains the entire geometry surface with the OML being redefined to include the ice shape. The second file is an STL file that contains just the ice shape portion of the geometry that was solved for within GlennICE. Finally, the third file is a Tecplot Subzone Data (SZPLT) file that contains surface based impingement data such as collection efficiency.

```

&files
  stl_outer_mold_line_out = "Iced_Geometry_OML.stl"
  stl_ice_only_out = "Iced_Geometry_Ice_Only.stl"
  tecplot_out = "Surface_Impingement_Data.szplt"
/

```

Figure 5.2 shows the iced geometry as output in the `stl_outer_mold_line_out` file. Note that the iced geometry is a singular surface that is the outer mold line of the clean plus iced geometry. Figure 5.3 shows the ice shape only output from the `stl_ice_only_out` variable in blue. The clean surface defined by the `tecplot_out` file has been shown as well in grey for simplification of viewing the differences between them.

Geometric information is only one piece of useful information a user may desire for the purposes of post processing the simulation data. GlennICE has the

ability to output surface based information, such as impingement related data like collection efficiency. This data is output in the `tecplot_output` file which is a Tecplot Subzone Data file. Figure 5.4 depicts a contour of collection efficiency of a single bin distribution on a NACA 0012 at a 4 degree angle of attack.

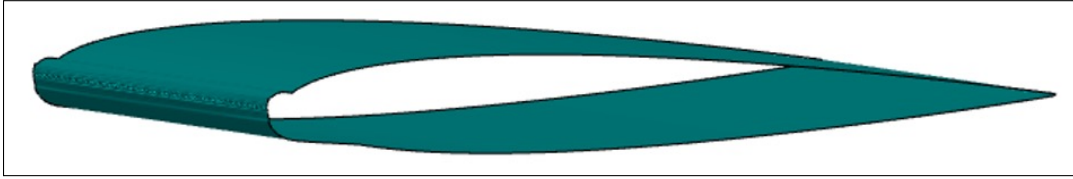


Figure 5.2: The iced geometry as stored in a Stereolithography output file. This simulation corresponds to a NACA 0012 at a 4 degree angle of attack.

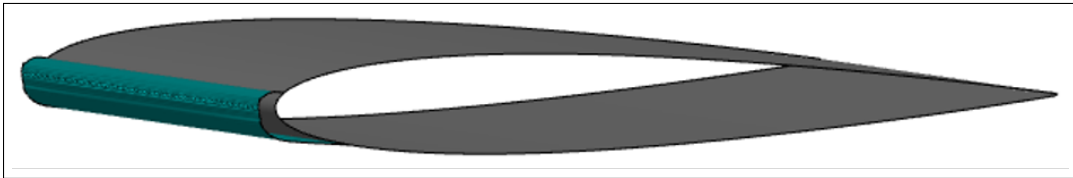


Figure 5.3: The iced geometry and the clean geometry simultaneously plotted. This simulation corresponds to a NACA 0012 at a 4 degree angle of attack.

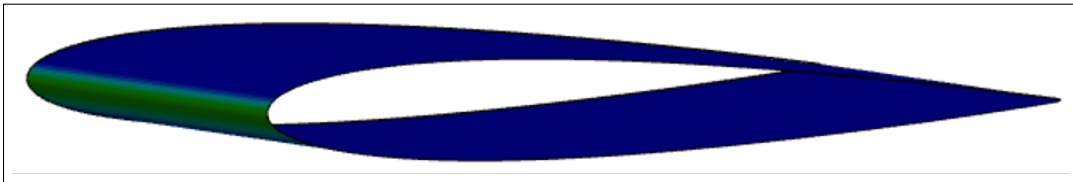


Figure 5.4: Collection efficiency of a single bin distribution on a NACA 0012 at a 4 degree angle of attack.

6 Input Files

A variety of input data is needed for GlennICE to execute. The required data can vary depending on the mode of operation. Some typical input files are the namelist file and the Computational Fluid Dynamics (CFD) solution file(s).

6.1 Namelist File

The namelist file contains all the ASCII based user input for specifying various filenames, controlling various settings, or inputting various numerical data into GlennICE.

The details of the namelists are provided in the subsections below and contain three distinct parts to aid the user in understanding the contents of each namelist. The first part optionally describes in greater detail the high level description of each namelist. The second part gives a sample implementation of the namelist with default values assigned. The third part describes in greater detail the purpose of each of the variables contained in that namelist, as well as the available option values if there is a list of option values for that particular namelist variable.

6.1.1 &files Namelist

This namelist controls the specification of paths, filenames, and extensions of various input/output files. The paths are relative paths from the current working directory GlennICE was executed in.

```
&files
  solution_in           = "solution_in.szplt"
  solution_surface_in   = "solution_surface_in.szplt"
  solution2_in          = ""
  solution2_surface_in  = ""
  stl_outer_mold_line_out = ""
  stl_ice_only_out      = ""
  tecplot_out           = ""
  refinement_out        = ""
  volume_restart_file    = "GlennICE_Volume_Restart.dat"
  refinement_restart_file = "GlennICE_Refinement_Restart.
dat"
  surface_restart_file   = "GlennICE_Surface_Restart.dat"
/
```

solution_in = "solution_in.szplt"

Input File. The CFD combined grid and solution file. This file contains both the volume and surface values or just the volume values. The extension must

be one of the following to signify the CFD file format (support for additional formats are planned, but not yet available):

<code>.fvuns</code>	—	Fieldview Unstructured
<code>.szplt</code>	—	Tecplot Subzone Load-on-Demand ¹

`solution_surface_in = "solution_surface_in.szplt"`

Input File. The CFD combined grid and solution file for the surface. If a file name is not supplied, the `solution_in` file must contain both volume and surface data. The extension must be one of the following to signify the CFD file format (support for additional formats are planned, but not yet available):

<code>.fvuns</code>	—	Fieldview Unstructured
<code>.szplt</code>	—	Tecplot Subzone Load-on-Demand ¹

`solution2_in = ""`

Input File. The second CFD combined grid and solution file. This file is used when two simulations are performed of differing wall temperature in order to compute the heat flux. It follows the same extension requirements as `solution_in`.

`solution2_surface_in = ""`

Input File. The CFD combined grid and solution file for the surface. If a file name is not supplied, the `solution_in` file must contain both volume and surface data. This file is used when two simulations are performed of differing wall temperature in order to compute the heat flux. It follows the same extension requirements as `solution_surface_in`.

`stl_outer_mold_line_out = ""`

Output File. Outer mold line ice shape ready for remeshing.

`stl_ice_only_out = ""`

Output File. Ice shape without the surface.

`tecplot_out = ""`

Output File. The Tecplot Subzone Loadable (`.szplt`) file. This file contains the tetrahedral volume and triangular surface meshes as well as input and computed variables used by GlennICE.

¹ Only *FETRIANGLE*, *FEQUADRILATERAL*, *FETETRAHEDRON*, and *FEBRICK* types are currently supported.

¹ Only *FETRIANGLE* and *FEQUADRILATERAL* types are currently supported.

refinement_out = ""

Output File. This file contains the Delaunay triangulation of the point cloud of release points for a given bin and mass flux solver iteration. Currently this file must be of extension `.szplt`. Meta data is added to the file name to specify the bin and mass flux solver iteration of the refinement. E.g. if `refinement_out="refinement.szplt"`, then the filename for the refinement of the first mass flux solver iteration of the first bin will have a file name of `'refinement_bin_1_iteration_1.szplt'`.

volume_restart_file = "GlennICE_Volume_Restart.dat"

Output File. This file contains the volume restart

created from the current run. It will have a default name of `GlennICE_Volume_Restart.dat`. If a volume restart is requested at the command line, this is the type of file required. It contains the required information extracted from the initial CFD input files, plus the grid has been converted to all tetrahedrons. The file includes the connectivity information which saves time on subsequent restarts.

refinement_restart_file = "GlennICE_Refinement_Restart.dat"

Output File. This file contains the refinement restart that will be used if the command line option for a refinement restart is used. It has a default name of `GlennICE_Refinement_Restart.dat`. A refinement restart will use the existing trajectory results and add refinement iterations specified by the user.

surface_restart_file = "GlennICE_Surface_Restart.dat"

Output File. This file contains the surface restart that will be used if the command line option for a surface restart is used. It has a default name of `GlennICE_Surface_Restart.dat`. A surface restart will use the existing collection efficiency results and perform the ice accretion specified by the user.

6.1.2 &surface_count Namelist

```
&surface_count
  number_of_surfaces = 1
/
```

number_of_surfaces = 1

The number of surfaces that bound the volume. This number controls the array size for the variables in `&surface_nml`.

6.1.3 &surface_nml Namelist

```
&surface_nml
```

```

    label      = "nolabel"
    category   = "nocategory"
/

```

label = "nolabel"

The label of a surface. This value is unused in the code and is for user reference to differentiate various surfaces of the same category.

category = "nocategory"

The surface category that is used to discriminate surface types. The wall distance user for the feature finding scheme will only be calculated for a surface designated "icing". By doing this, the user can discriminate between surfaces desired for analysis while eliminating the need to run additional trajectories. The following categories are recognized by GlennICE and used for discrimination:

- "Inlet" — The inlet of the domain. Currently release points are only only released from the `Inlet` and it is required that at least one surface is designated as an `Inlet`
- "Icing" — The surface(s) that a user wishes to impinge and subsequently grow ice on. At least one surface must be designated as an `Inlet` surface for the software to generate an ice shape.
- "nocategory" — Used to represent additional surfaces that are not "Inlet" or "Icing"

6.1.4 &trajectory_solver Namelist

```

&trajectory_solver
  trajectory_summary_output = "none"
  write_histories           = .false.
  time_statistics           = .true.
  t_start                   = 0.0      ! (expert parameter)
  t_stop                    = -1.0     ! (expert parameter)
  stop_time_safety_factor  = 10.0     ! (expert parameter)
  max_find_cell_tries       = 10       ! (expert parameter)
  abs_tol                   = 1.e-08   ! (expert parameter)
  rel_tol                   = 1.e-07   ! (expert parameter)
  chunk_percent             = 0.0      ! (expert parameter)
/

```

trajectory_summary_output = "none"

Prints out a variety of parameters of interest for trajectories to a file called `trajectory_summaries.dat` which will contain different values depending on the argument choice. The valid argument choices are:

- "none" — No trajectory information is printed and the file `trajectory_summaries.dat` is not written out.
- "volume_exit_error_only" — The same as "all" but only is written to the `trajectory_summaries.dat` file if the trajectory had an invalid exit face.
- "all" — Prints the following parameters of interest to the `trajectory_summaries.dat` file:
 - the trajectory index
 - the initial cell index
 - the starting coordinate of the trajectory
 - the initial particle velocity
 - the particle diameter
 - the stopping coordinate of the trajectory
 - the stopping time of the trajectory
 - the face index the trajectory intersected
 - the surface index the trajectory intersected
 - the number of integration steps
 - the exit index generated by dopri5
 - the initial cell volume
 - the minimum wall distance

`write_histories = .false.`

Setting this boolean to `.true.` writes trajectories to the file `trajectories.dat`, an ASCII Tecplot file. This functionality is not intended for use with a normal simulation. It is intended to enable the visualization of a handful of trajectories for analysis.

`time_statistics = .true.`

Setting this boolean to `.true.` writes time statistics for the trajectories computed

The following parameters are for advanced users and should **not** be adjusted unless directed to do so by the GlennICE support team.

t_start = 0.0  **EXPERT PARAMETER**

A trajectory's initial condition with respect to time in seconds.

t_stop = -1.0  **EXPERT PARAMETER**

The upper bound of a trajectory's time in seconds. The difference between t_stop and t_start sets the bound on the length of time a trajectory is computed. If t_stop is not set, the calculated value is:

$$t_{\text{stop}} = \text{stop_time_safety_factor} \cdot \left(\frac{x_{\text{max}} - x_{\text{min}}}{V_{\text{inlet}}} \right)$$

where

x_{max} = maximum x-location in domain

x_{min} = minimum x-location in domain

V_{inlet} = average velocity at inlet

stop_time_safety_factor = 10.0  **EXPERT PARAMETER**

'Safety factor' used to scale t_stop if t_stop is not set in this file.

max_find_cell_tries = 10  **EXPERT PARAMETER**

The number of previous trajectory locations used to find the volume cell that contains the current trajectory location.

abs_tol = 1.e-08  **EXPERT PARAMETER**

Absolute tolerance for the trajectory integration method.

rel_tol = 1.e-07  **EXPERT PARAMETER**

Relative tolerance for the trajectory integration method.

chunk_percent = 0.0  **EXPERT PARAMETER**

Percent of trajectories given to each processor.

chunk_percent = "0.0"	⇒	Purely Dynamic Scheduler
"0.0" < chunk_percent < "100.0"	⇒	Blend between Static and Dynamic
chunk_percent = "100.0"	⇒	Purely Static Scheduler

for more details on parallel computing efficiency, the user is directed to **Section 9.16 - Parallel Computing**.

6.1.5 &release Namelist

The release namelist specifies information pertaining to the initial seed grid. If additional refinement iterations are specified in the &mass_flux_solver namelist, then the initial state of the refinement algorithm will be the initial seed grid specified by the values in this namelist.

NOTE: Version 4.2.0 has a requirement that the number of seed points and release points are equal. This effectively requires that the box specified by the &release namelist must be equal to or bounded by the edges of the Inlet surface(s). Reference Section 9.10 - Seed Points and Release Points for more detail.

```
&release
  inlet_coverage      = "rectangular"
  n_ytraj             = 3
  n_ztraj             = 3
  x_min               = bounding box x_min
  x_max               = bounding box x_max
  y_min               = bounding box y_min
  y_max               = bounding box y_max
  z_min               = bounding box z_min
  z_max               = bounding box z_max
/
```

inlet_coverage = "rectangular"

Defines the inlet coverage algorithm to be used. Valid choices are:

- "full" — Will ignore the other values in this namelist and the initial release points will be the node centered locations of the inlet surface(s). (Recommended)
- "rectangular" — Limit release locations to the rectangular bounding box with the options described below.

n_ytraj = 3

The number of trajectories in the y-direction. *Note: This value must be greater than one.*

n_ztraj = 3

The number of trajectories in the z-direction. *Note: this value must be greater than one.*

x_min = bounding box x_min

Sets `x_min` in meters. *Note: Currently this value does nothing but will be useful as the nomenclature shifts away from an assumed primary flow pointing in the x direction.*

`x_max = bounding box x_max`

Sets `x_max` in meters. *Note: Currently this value does nothing but will be useful as the nomenclature shifts away from an assumed primary flow pointing in the x direction.*

`y_min = bounding box y_min`

Sets `y_min` in meters

`y_max = bounding box y_max`

Sets `y_max` in meters.

`z_min = bounding box z_min`

Sets `z_min` in meters.

`z_max = bounding box z_max`

Sets `z_max` in meters.

6.1.6 `&mass_flux_solver` Namelist

```
&mass_flux_solver
  max_iterations      = 100
  refinement          = "adaptive"
  refine_face_center  = .false.
  refine_face_edges   = .true.
/
```

`max_iterations = 100`

The maximum number of iterations (or refinements) of the trajectory release point cloud. The default value is set with the notion that the user will reach the `pct_converged_limit` before the `max_iterations` value is reached.

`refinement = "adaptive"`

Defines the refinement algorithm to be used. Valid choices are:

- "adaptive" — This algorithm adaptively adds trajectories to limit the amount of computational work wasted computing trajectories that miss the icing surface of interest.
- "uniform" — This algorithm uniformly adds trajectories.

refine_face_center = .false.

Setting this flag to `.true.` will place a new seed point at the center of an existing seed point face that has been set to refine due to one of the criteria in the `adaptive_refinement` namelist.

refine_face_edges = .true.

Setting this flag to `.true.` will place new seed points at the center of each edge of an existing seed point face that has been set to refine due to one of the criteria in the `adaptive_refinement` namelist.

6.1.7 &adaptive_refinement Namelist

The adaptive refinement namelist controls various efficiency and robustness parameters with respect to the adaptive refinement algorithm. For a more detailed discussion of these parameters the reader is directed to **Section 9.3 - Adaptive Refinement**.

```
&adaptive_refinement
  min_face_count           = 20
  max_face_count           = 600
  mwd_scale_factor         = 1.0
  turn_off_bounded_faced   = .true.
  fraction_contained_tol   = 0.5
  mass_flow_tol            = 0.0
  min_feature_size         = -infinity
  hit_percent_limit        = 25.0
  use_nonuniform_tolerance = .true.
  pct_converged_limit      = 90.0
/
```

min_face_count = 20

Sets a minimum face count for the `mass_flow_tolerance_criteria`. At least this many hits need to occur on a face before the `mass_flow_tol` metric will be checked.

max_face_count = 600

Stops the adaptive refinement algorithm from refining around trajectories that hit a face that has already been impinged by the number of trajectories specified here.

turn_off_bounded_faces = .true.

Do not turn off refinement of seed point faces that are contained in a single CFD face.

`fraction_contained_tol = 0.5`

Shuts off refinement of CFD faces that have the fraction of seed point faces contained on the CFD face above this value. A value of 1.0 would mean faces will never be turned off due to this metric.

`mass_flow_tol = 0.0`

Shuts off refinement of CFD faces when the mass impacting the face changed from value in the current refinement iteration to the value in the previous refinement iteration by less than the tolerance given. The default value means faces will never be turned off due to this metric.

`min_feature_size = -infinity`

Shuts off refinement of trajectories that miss by less than the value input. Stops extraneous refinement of trajectories near the impingement limit.

`mwd_scale_factor = 1.0`

Scales refinement of trajectories that miss by setting the local seed point spacing on the seed plane to be a function of the minimum wall distance of the trajectory. Stops extraneous refinement of trajectories near the impingement limit.

`hit_percent_limit = 25.0`

Uses the percentage of trajectories that hit an Icing surface to provide a user adjustable metric to adaptively stop the feature finding algorithm. Limits extraneous refinement of trajectories near the impingement limit.

`use_nonuniform_tolerance = .true.`

Relaxes the convergence tolerance on CFD faces which have less mass flow. It uses the CFD face with maximum mass flow as a scale.

`pct_converged_limit = 90.0`

The impinging mass flow is considered converged when the selected convergence criteria (`fraction_contained_tol` and/or `mass_flow_tol`) reaches this level.

6.1.8 `&bin_count` Namelist

```
&bin_count
  nbins = 1
/
```

`nbins = 1`

The number of bins in the discretized cloud distribution.

6.1.9 &distribution Namelist

```
&distribution
  diameter          = 20.0
  mass_fraction     = 1.0
/
```

diameter = 20.0

An array of length `nbins` that stores the diameter array in microns.

mass_fraction = 1.0

An array of length `nbins` that stores the fraction of mass each bin contains. This fractional mass is the ratio of the bin's water content to the Total Water Content (`twc`). The summation of this array shall equal 1.0.

6.1.10 &freestream Namelist

```
&freestream
  twc               = 0.5
  relhumidity       = 100.0
  time              = 60.0
  icing_temperature = 0.0
  ice_density       = 0.0
  gravity           = 0.0, 0.0, 0.0
/
```

twc = 0.5

Total Water Content (TWC) of the entire discretized distribution in units of g/m^3 .

relhumidity = 100.0

The freestream relative humidity with units of percent.

time = 60.0

The total icing time, in units of seconds (s), in which ice accretion will be calculated for.

icing_temperature = 0.0

The freestream temperature, in units of Kelvin (K), used to perform the surface energy balance. An assumption is made that small changes in freestream temperature do not affect the velocity vector field, resulting in identical impingement regardless of what this value is set to. If this parameter is a value

of 0.0 (the default), then the freestream temperature from the supplied CFD solution file will be used. The freestream temperature of the CFD solution file is defined as the area averaged value of temperature of the **Inlet** surface.

ice_density = 0.0

The "density" of the ice mass in kg/m^3 . This term can capture the effect of changes in density due to trapped air bubbles, or it can capture the effect of macroscopic voids such as those within the Maximum Combined Cross Section (MCCS) of three dimensional scalloped ice shapes. When the default value of zero is specified, the value of ice density is computed internally as a linear function of freezing fraction.

gravity = 0.0, 0.0, 0.0

The gravity vector specified in units of m/s^2 . To neglect the gravitational effects set to 0.0, 0.0, 0.0

6.1.11 &htc_augmentation Namelist

```
&htc_augmentation
  augmentation_type      = "none"
  laminar_htc_augmentation = 1.0
  turbulent_htc_augmentation = 5.0
  ideal_rime_limit        = 0.009
  x_transition            = -infinity
  num_augmentation_steps  = 6
/
```

augmentation_type = "none"

Selects the method in which the heat transfer coefficient is being augmented to account for iced surface roughness. The valid argument choices are:

- "none" — No augmentation is applied to the extracted value of HTC.
- "fixed_transition" — Laminar and turbulent augmentation of HTC based on the value of **x_transition** specified.
- "McClain_correlation" — Augmentation of HTC based on average roughness height as defined in McClain et al. [9].

laminar_htc_augmentation = 1.0

A factor that scales the laminar heat transfer coefficient.

turbulent_htc_augmentation = 5.0

A factor that scales the turbulent heat transfer coefficient. The default value of 5 provided the best prediction as determined in Ref [20].

ideal_rime_limit = 0.009

Upper limit on Ideal Rime Thickness as defined in Equation 27 of reference [9]. At some point, ice growth stops being roughness and becomes ice shape. The default value of 0.009 m (9 mm) was chosen as defined in Ref [20].

x_transition = -infinity

An **x** location in meters that forces a transition from laminar to turbulent heat transfer. Heat transfer values upstream of this location are assumed to be laminar and are multiplied by the value of the variable **laminar_htc_augmentation**. Heat transfer values down stream of this location are assumed to be turbulent and are multiplied by the value of the variable **turbulent_htc_augmentation**. The default value is set to the smallest possible floating point number.

num_augmentation_steps = 6

The number of augmentation steps for the McClain correlation. Roughness depends on freezing fraction which depends on heat transfer coefficient which is affected by augmentation which depends on roughness. A value of 1 will result in no augmentation and a value of 2 is predictor-corrector. Cases performed thus far show good convergence in 6 iterations.

6.1.12 &volume_output Namelist

```
&volume_output
  write_volumes = .false.
/
```

write_volumes = .false.

If this flag is set to **.true.**, the volume solution will be written to the Glen-ICE solution file. The values in the volume are unchanged except for unit conversions which are set in either **Section 6.1.13 - &conversions Namelist** or **Section 6.1.14 - &variable_properties Namelist**.

6.1.13 &conversions Namelist

This namelist provides the ability to dimensionalize a non-dimensional variable set to SI, or to convert from a non-SI unit set into SI. Note that for variables that use an offset to translate the value, the offset is applied before the variable is scaled. That is to say that the offset is always in the units being converted (or non-dimensional).

```

&conversions
  pressure_offset      = 0.0
  temperature_offset   = 0.0
  temperature_scale_factor = 1.0
  length_scale_factor  = 1.0
  mass_scale_factor    = 1.0
  time_scale_factor    = 1.0
/

```

pressure_offset = 0.0

A temperature translation value. Used for providing the translational value for the temperature conversion.

temperature_offset = 0.0

A temperature translation value. Used for providing the translational value for the temperature conversion.

temperature_scale_factor = 1.0

The scale factor for temperature. This value has units of kelvin in the numerator and the units being converted from in the denominator.

length_scale_factor = 1.0

The scale factor for length. This value has units of meters in the numerator and the units being converted from in the denominator.

mass_scale_factor = 1.0

The scale factor for mass. This value has units of kilograms in the numerator and the units being converted from in the denominator.

time_scale_factor = 1.0

The scale factor for time. This value has units of seconds in the numerator and the units being converted from in the denominator.

6.1.14 &variable_properties Namelist

This namelist provides the ability to specify variable properties for an individual variable. This includes properties such as the variable's label from the CFD solver, or to dimensionalize a non-dimensional variable set to SI, or to convert from a non-SI unit set into SI. Note that for variables that use an offset to translate the value, the offset is applied before the variable is scaled. That is to say that the offset is always in the units being converted (or non-dimensional).


```

&variable_properties
  map_index      = 0
  spelling       = ""
  offset         = infinity
  scale_factor   = infinity
/

```

map_index = 0

The map index that specifies the variable that these properties are associated with. See **Section 6.2.1 - List of Variables Read from CFD Solution File** for details on the variable mapping.

spelling = ""

GlennICE recognizes various spellings internally from a subset of CFD solvers. If GlennICE doesn't recognize a variable's label as output from the CFD solver, it can be specified here.

offset = infinity

A user specified offset for a variable. Note that specification of this value will have priority over the offset specified in the **&conversions** namelist described in **Section 6.1.13 - &conversions Namelist**. Note that the default value of infinity will ultimately result in an offset of zero being enforced.

scale_factor = infinity

A user specified scale factor for a variable. Note that specification of this value will have priority over the scale factor specified in the **&conversions** namelist described in **Section 6.1.13 - &conversions Namelist**. Note that the default value of infinity will ultimately revert to a default value of one set within **&conversions**.

6.1.15 &rotation Namelist

This namelist specifies if the supplied CFD solution is in the absolute or relative frame. If the supplied CFD solution is in the relative frame then omega is used to specify the rotation rate, in radians/second, about the x-axis.

```

&rotation
  in_absolute_frame = .true.
  omega             = 0.0
/

```

in_absolute_frame = .true.

The logical `in_absolute_frame` specifies if the supplied CFD flow solution is in the relative or absolute frame of reference.

omega = 0.0

The parameter `omega` specifies the rotation rate, in *rad/s*, about the x-axis.

6.1.16 &runback Namelist

This namelist specifies parameters controlling surface water runback.

```
&runback
  max_runback_iterations      = 0
  runback_tolerance           = 1.e-16
/
```

max_runback_iterations = 0

Maximum number of runback iterations. For the default value of 0, the number of runback iterations will be equal to the number of faces on the icing grid surfaces specified. For very warm conditions, water may not freeze and this section of the software can take a long time to complete. The solution can be interrupted and a surface restart case can be executed with an adjustment to this value specified.

runback_tolerance = 1.e-16

Tolerance for water runback convergence. Once the `runback_tolerance` value is satisfied, no more runback iterations will be calculated. If the `runback_tolerance` is not achieved, the solution will cease iterating based upon the value specified by `max_runback_iterations`.

6.2 Supported CFD File Formats

In order to compute water impingement in a Lagrangian particle trajectory framework, a vector field must be provided. GlennICE uses a discretized vector field, a common output of Computational Fluid Dynamic (CFD) software. Discretization is limited to hexahedrons, tetrahedrons, prisms and pyramids. I.E. GlennICE does not support and has no plans to support an arbitrary polyhedral discretization scheme.

Note that while GlennICE interfaces with multiple discretization types, GlennICE post processes the discretization to be purely tetrahedral using the methods of Dompierre [12]. Thus, the algorithms in the GlennICE software only need to consider volume tetrahedrons and surface triangles.

There are many different ways a discretized CFD mesh and solution can be stored, both in organization and encoding. While GlennICE has a desire to support all of these different variations, robust implantation is time consuming. Table 6.1 illustrates the current support status of the various CFD file variants. If support for any of the unsupported variants is particularly of high importance, please email the GlennICE team at GlennICE-support@lists.nasa.gov and communicate this need to aid us in our prioritization of supporting the unsupported elements.

Table 6.1: List of CFD File Variants and Their Support Status within GlennICE

File Format	Support Status
Fieldview Unstructured	Yes
Tecplot Subzone Loadable	Yes
CGNS	No
Plot3D	No
Solution Location	
Node Centered	Yes
Cell Centered	No
Grid and Solution File Storage	
Combined File	Yes
Separate Files	No
Volume and Surface File Storage	
Combined File	Yes
Separate Files	Yes ¹
Discretization Type	
Tetrahedral Unstructured	Yes
Mixed-Element Unstructured	Yes
Hexahedral Structured	No
Solution Units	
SI	Yes
Non-Dimensional	Yes
Arbitrary Units	Yes

6.2.1 List of Variables Read from CFD Solution File

GlennICE can read 17 variables from a solution file. Which variables are used depends on whether the user chooses a one solution methodology or a two solution methodology as described in **Section 5.2 - Specification of the CFD Solution**. The default units are SI. The required variables are listed below.

The `&variable_properties` namelist described in **Section 6.1.14 - &variable_properties Namelist** allows the user to specify individual variable-based properties for simulation configuration. The user chooses the desired variable by use of a map index. Table 6.2 contains the input variables that GlennICE currently recognizes, and the variable's associated map index.

For example, if your variable is named `wall heating`, you would need to input information within the `&variable_properties` section such that you

¹ Only Valid for .szplt format

would have `map_index = 10` , `spelling = "wall heating"` with the necessary `scale_factor` value for unit consistency. Additionally, if you have a variable name that is natively recognized by GlennICE but there is a need to scale it; you would use the `map_index` variable to point to the variable being scaled and then include the appropriate value for `scale_factor`.

Table 6.2: Input variables, GlennICE recognized spellings, variable descriptions, map index.

Map Index	Variable Name	GlennICE Recognized Spellings	Description
1	<code>x</code>	<i>x</i>	<i>X-coordinate</i>
2	<code>y</code>	<i>y</i>	<i>Y-coordinate</i>
3	<code>z</code>	<i>z</i>	<i>Z-coordinate</i>
4	<code>rho</code>	<i>rho</i> <i>Density</i>	<i>Density</i>
5	<code>u</code>	<i>u</i> <i>x-Velocity</i>	<i>X-component of velocity</i>
6	<code>v</code>	<i>v</i> <i>y-Velocity</i>	<i>Y-component of velocity</i>
7	<code>w</code>	<i>w</i> <i>z-Velocity</i>	<i>Z-component of velocity</i>
8	<code>p</code>	<i>p</i> <i>Pressure</i> <i>Absolute Pressure</i>	<i>Pressure</i>
9	<code>Twall</code>	<i>temperature</i> <i>Wall Adjacent Temperature</i> <i>surface temperature</i> <i>TWall</i>	<i>Wall temperature</i>
10	<code>Qwall</code>	<i>heating</i> <i>Wall Heat Flux</i> <i>surface heat flux</i> <i>QWall</i>	<i>Wall heat flux</i>
11	<code>xWallShear</code>	<i>shear_x</i> <i>x-Wall Shear</i> <i>Wall Shear</i> <i>x surface shear</i> <i>xWallShear</i>	<i>X-component of wall shear stress</i>
12	<code>yWallShear</code>	<i>shear_y</i> <i>y-Wall Shear</i> <i>Wall Shear</i> <i>y surface shear</i> <i>yWallShear</i>	<i>Y-component of wall shear stress</i>
13	<code>zWallShear</code>	<i>shear_z</i> <i>z-Wall Shear</i> <i>Wall Shear</i> <i>z surface shear</i> <i>zWallShear</i>	<i>Z-component of wall shear stress</i>
14	<code>htc_from_input</code>	<i>Wall Heat Transfer Coefficient</i> <i>surface heat transfer coefficient</i> <i>htc_from_input</i>	<i>Heat transfer coefficient</i>
15	<code>Taw_from_input</code>	<i>Wall Adiabatic Temperature</i> <i>Adiabatic Surface Temperature</i> <i>Taw_from_input</i>	<i>Adiabatic wall temperature</i>

`xWallShear`, `yWallShear`, and `zWallShear` are only required on the surface. `Twall` and `Qwall` are only needed on the surface for the two solution methodology described in **Section 5.2.2 - Two Solution CFD Methodology** while `Htc` and `Taw` are only needed on the surface for the one solution methodology

in **Section 5.2.1 - One Solution CFD Methodology**.

6.2.2 Specification of Input Variable Locations

Table 6.3 describes whether a variable, listed in Table 6.2 is needed within the surface and volume input file. **Twall** and **Qwall** are only needed on the surface for the two solution methodology described in **Section 5.2.2 - Two Solution CFD Methodology** while **Htc** and **Taw** are only needed on the surface for the one solution methodology in **Section 5.2.1 - One Solution CFD Methodology**.

Table 6.3: Input variables and whether it is needed on the surface or volume.

Map Index	Variable Name	Surface	Volume
1	x	Yes	Yes
2	y	Yes	Yes
3	z	Yes	Yes
4	rho	Yes	Yes
5	u	No	Yes
6	v	No	Yes
7	w	No	Yes
8	p	Yes	Yes
9	Twall	2-Solution Method	No
10	Qwall	2-Solution Method	No
11	xWallShear	Yes	No
12	yWallShear	Yes	No
13	zWallShear	Yes	No
14	htc_from_input	1-Solution Method	No
15	Taw_from_input	1-Solution Method	No

6.3 Trajectories Job Input File

A feature exists within GlennICE where the user is able to specify the release location of particle trajectories for visualization purposes. This method is useful when there are bad trajectories identified within the execution of GlennICE and allows the user to gain insight into what aspect of the solution is causing the error. This file is called the **particles_input.txt** file and is read automatically when **jobtype=trajectories** is specified within the command line input. One difficulty associated with this routine is the knowledge of what to put into the required columns: a user is highly unlikely to know this off-hand. It is suggested that the user generates a **trajectories_summaries.dat** file to obtain the necessary input information for the **particles_input.txt** file. Specifying an output of the **trajectory_summaries.dat** file is discussed in **Section 6.1.4 - &trajectory_solver Namelist**. If the **trajectory_summaries.dat** file is being used as a reference, only the first 9 columns are read

by GlennICE as an input in `particles_input.txt`. The `id` value is an integer value that allows for bookkeeping of the trajectory within GlennICE, this value is not capable of naming or labeling the trajectories. An example of a particle input being specified is shown below with the header line being included. More about how to run a trajectory job is discussed in **Section 3.2 - Controlling Scope of Simulation with JobType** and extended information on trajectory visualization within GlennICE are described in **Section 8.6 - Trajectory Visualization**.

Example formatting of the `particles_input.txt` file:

id	start_cell	x_start	y_start	z_start	vx_p_start	vy_p_start	vz_p_start	diameter
1	137171	-0.1531E+02	-0.4000E+01	-0.3750E+01	0.4642E+01	0.9344E-03	0.3487E-02	0.9999E-05
2	137228	-0.1531E+02	-0.3999E+01	-0.3355E+01	0.4687E+01	-0.2376E-03	0.9260E-04	0.9999E-05

7 Output Files

After execution, a restart file can be output to limit the computational expense relating to certain interactions with the software. Also, desirable information and simulation results can be viewed utilizing third party visualization software. Currently GlennICE has the ability to output two visualization files at the user's discretion. These files contain information about the geometric ice shape, as well as impingement related data and are described in more detail below.

7.1 Volume Restart File

GlennICE currently provides a volume restart capability of limited scope. The binary volume restart file is based on a rigid file type and preserves the grid and flow solution along with GlennICE-specific indexing. The primary use of this restart file is to save computation time by allowing GlennICE to read in initial information rather than recomputing it.

7.2 Surface Restart File

GlennICE currently provides a surface restart capability of limited scope. The binary surface restart file is based on a rigid file type and preserves the surface based impingement data. The primary use of this restart file is to facilitate a parametric temperature sweep. Note that the validity of the parametric temperature sweep is reliant on the assumption that small changes in freestream temperature don't significantly perturb the velocity vector field. Or stated another way, the assumption that the impingement results are not sensitive to changes in temperature.

7.3 Refinement Restart File

GlennICE currently provides a refinement restart capability of limited scope. The binary refinement restart file is based on a rigid file type and preserves the results of the trajectory computation. The primary use of this restart file is to add additional trajectory refinement to an existing case. For example, if a case was performed using 20 refinement iterations and the user wants to add additional refinement, a case can be submitted with `restartJob=Refinement` to perform a 21st iteration. In this case, the number of iterations in the user input file represents the number of additional iterations requested.

7.4 Stereolithography Files

One desirable set of data is that pertaining to the geometric ice shape. A binary stereolithography (STL) file is used to output this information for visualization. An STL file stores a triangularly tessellated surface and is supported

by a majority of visualization packages. Since GlennICE operates on a purely tetrahedral volume with triangular surfaces, the use of an STL file to store the iced plus clean surface is natural. This file could be used within a script to create a new volume mesh for multi-shot ice accretion prediction. A second STL file is generated that only contains the iced regions. This file would be useful for 3D printing ice shapes produced by GlennICE. The user is forewarned that this geometry is not guaranteed to be watertight, although it usually meets this requirement.

It should be noted that the wide range of support for STL files is based on a simple but rigid file format that can be seen below [13]. The primary detriment of this rigid file format is that the nodal locations of the tessellated surface are stored in single precision. However, for most applications, single precision is acceptable.

```
UINT8[80] - Header
UINT32 - Number of triangles

foreach triangle
REAL32[3] - Normal vector
REAL32[3] - Vertex 1
REAL32[3] - Vertex 2
REAL32[3] - Vertex 3
UINT16 - Attribute byte count
end
```

7.5 Tecplot File

Surface based information is output in a binary Tecplot Subzone Loadable (SZPLT) file. It is noted that usage of this file format introduces a requirement that the user gain access to the commercial visualization software Tecplot [11]. This is a current limitation of the GlennICE software, and there exists a desire to implement the outputting of this same data set in CFD General Notation System (CGNS) for use with a wider array of visualization packages. If this limitation directly affects the user, please email the GlennICE team at GlennICE-support@lists.nasa.gov and communicate this need to aid us in our prioritization of supporting the unsupported elements.

This file outputs the GlennICE volume and surface meshes. The volume is represented by the tetrahedral discretization that GlennICE utilizes internally. That is to say, if a mixed element mesh is provided to GlennICE, the mixed element mesh is not returned. What is returned is the tetrahedral mesh the mixed element mesh is split into. Note that no new nodes are added during the element splitting.

The surface meshes are also output, as well as the variable data associated with these meshes. Note that only the surfaces given the `category` of "Icing" in the `&surface_nml` name list will have impingement data, such as collection efficiency, associated with them.

7.5.1 List of Output Variables

- `x`, x-coordinate, m
- `y`, y-coordinate, m
- `z`, z-coordinate, m
- `rho`, air density, kg/m^3
- `u`, x-velocity, m/s
- `v`, y-velocity, m/s
- `w`, z-velocity, m/s
- `p`, pressure, N/m^2
- `Twall`, Wall temperature, K
- `QWall`, Wall heat flux, W/m^2
- `xWallShear`, x-component of wall shear stress, N/m^2
- `yWallShear`, y-component of wall shear stress, N/m^2
- `zWallShear`, z-component of wall shear stress, N/m^2
- `htc_from_input`, heat transfer coefficient (if supplied), W/m^2K
- `WallDistance`, distance to nearest viscous surface, m
- `Twall2`, Wall temperature from second solution file (if it was supplied), K
- `QWall2`, Wall Heat flux from second solution file (if it was supplied), W/m^2
- `Taw`, adiabatic wall temperature, K
- `htc`, heat transfer coefficient, W/m^2K
- `impinging_water_bin_1`, water impinging for bin 1, kg/s
- `impinging_water_fraction_contained_bin_1`, water impinging from contained stream tubes for bin 1, kg/s
- `impinging_water_diff_bin_1`, difference in previous two variables for bin 1, kg/s
- `hit_counts_bin_1`, number of particles hitting each face for bin 1

- `seed_area_fraction_contained_bin_1`, fraction of stream tube areas contained
- `converged_bin_1`, faces that have converged using the metrics supplied.
Valid options:
 - 1 = no impingement
 - 0 = impingement not met (not converged)
 - 1 = impingement metrics met (converged)
- `impinging_water_total`, total water impinging for all bins, kg/s
- `beta_total`, collection efficiency for all bins
- `beta_bin_1`, collection efficiency for bin 1
- `ice_mass`, ice mass on each face, kg/s
- `ice_volume`, ice volume for each face, m^3
- `ice_thickness`, prism ice thickness (ice volume divided by face area), m
- `augmentation`, heat transfer augmentation (dimensionless ratio)
- `roughness`, roughness for McClain method, m
- `freezing_fraction`, fraction of impinging water that freezes on a face.
Freezing fraction = 1 on all faces that have runback ice and is = -1 where there is no ice.
- `temperature`, surface temperature, K
- `bad_extrusions`, faces where extrusion method failed
 - 0 = extrusion successful
 - 1 = no positive cubic root (negative height calculated)
 - 2 = quadratic root less than cubic root (non-physical height calculated)

7.6 Refinement Out Files

Each refinement iteration GlennICE performs will output the seed plane locations that are projected onto the inlet. These files are useful to see where the trajectories that hit the surface come from, as those regions will have the the finest mesh. These files are only output if the user supplies a filename with the `refinement_out` variable and the filename which is written is indexed but the iteration number.

7.6.1 List of Output Variables

- **x_start**, starting x-coordinate of a seed point, m
- **y_start**, starting y-coordinate of a seed point, m
- **z_start**, starting z-coordinate of a seed point, m
- **vx_start**, starting x-velocity of trajectory, m
- **vy_start**, starting y-velocity of a trajectory, m
- **vz_start**, starting z-velocity of a trajectory, m
- **x_stop**, stopping x-coordinate of a trajectory, m
- **y_stop**, stopping y-coordinate of a trajectory, m
- **z_stop**, stopping x-coordinate of a trajectory, m
- **vx_stop**, stopping x-velocity of a trajectory, m
- **vy_stop**, stopping y-velocity of a trajectory, m
- **vz_stop**, stopping z-velocity of a trajectory, m
- **min_wall_distance**, minimum wall distance for trajectory released from this node, m
- **mass_flow**, mass flow of the release area controlled by this node, kg/s
- **intersected_face**, index of surface face hit by trajectory
- **intersected_surface**, index of surface hit by trajectory
- **number_of_time_steps**, number of time steps for the trajectory
- **iteration**, refinement iteration when this node was created

8 Example Cases

8.1 ONERA M6 Input Flow Solution

In this chapter, the operation of GlennICE will be demonstrated through the description of several example cases. The cases will use the same grid and flow solutions but will illustrate different functionality of the software. Two flow solutions are used for the example cases. See **Section 5.2.2 - Two Solution CFD Methodology** for more information on the option to read two flow solutions.

Table 8.1: CFD Run Conditions

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

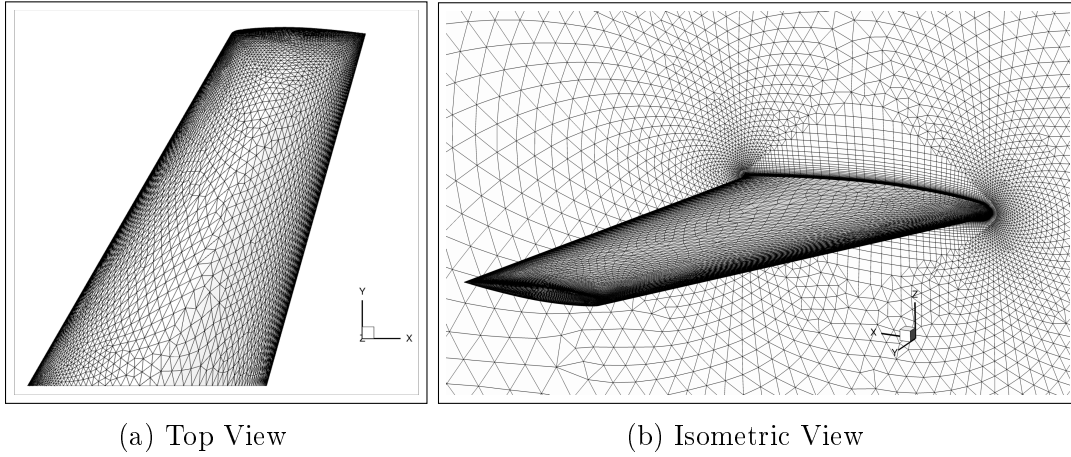


Figure 8.1: ONERA M6 Computational Grid (3D) with Symmetry Plane.

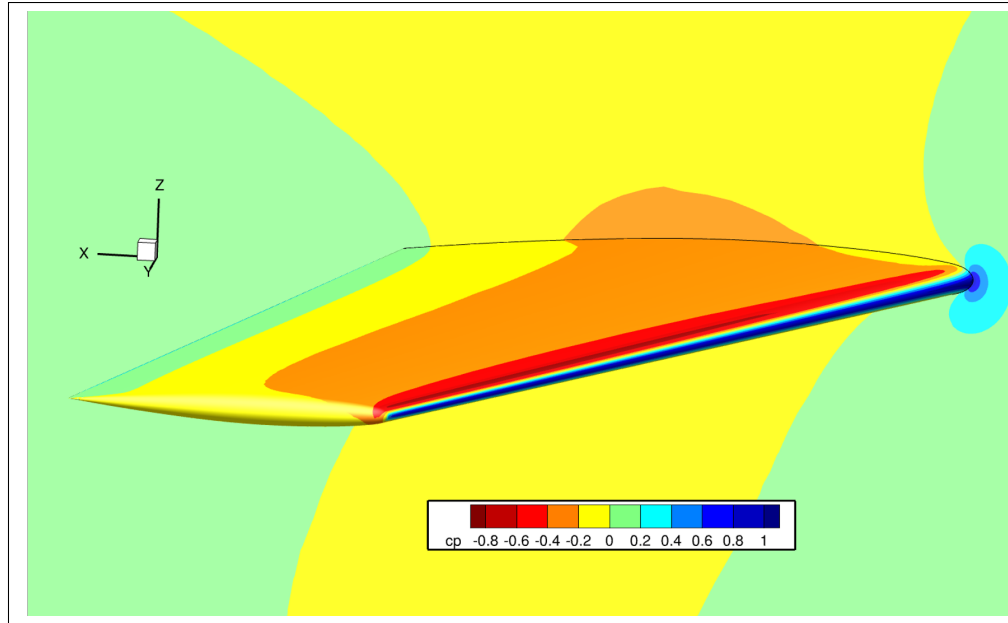
The geometry simulated is an ONERA M6 swept wing with a chord length of 1.0 m cantilevered off a symmetry plane with a span of approximately 1.5 m. The inlet and exit planes are located approximately 200 body lengths upstream and downstream. The domains are farfield boundaries that can be represented through fluxes or freestream boundary conditions to achieve the freestream conditions provided. The grid was generated using Heldenmesh [21] and contains a mixed element mesh based upon local surface curvature and contains 955,027 nodes. This case contains adequate spacing to produce a $y^+ \leq 1$ on the surface of the grid when run under the conditions provided. Due to the freestream conditions, three-dimensionality, and fully viscous nature of this case, a reduction in grid size does not produce a grid converged flow solution. It is also assumed that the application of this tool for problems of interest to the user will necessitate access to distributed memory systems such as an HPEC resource and so a grid of this size is not burdensome. If a smaller case is needed or a user cannot run a mixed-element mesh, accommodations

can be made for the user on a case-by-case basis. The necessary grid files has been included within the GlennICE distribution and are found in the `Example_Cases/CFD_Grids` directory.

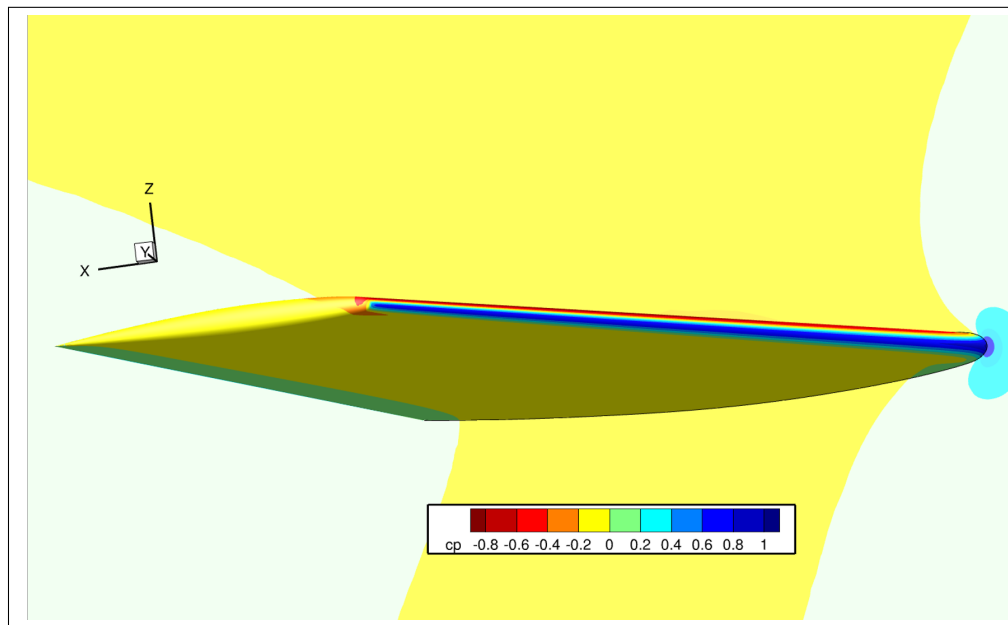
The flow solution was generated using FUN3D v14.0.1 with a steady RANS perfect-gas finite-volume discretization with the SA-neg turbulence model [23, 22]. The solutions utilized the two-temperature wall temperature process with a nondimensional wall temp of $W_{T,1} = 1.00$ and $W_{T,2} = 1.04$ under the freestream conditions given in Table 8.1. These wall temperatures do not need to be identical to the ones utilized and may need to be represented differently within your CFD solver. The different wall temperatures are used in GlennICE to compute the heat transfer coefficient that is described in **Section 5.2 - Specification of the CFD Solution**. For example, within ANSYS-CFX they could be represented as a constant wall temperature of 20°C and 30°C [15]. It is important to note that this case was run nondimensionally with the Mach number, Temperature, and Reynolds number defining the freestream conditions with the farfield freestream boundaries being specified via fluxes. If your flow solver requires dimensional values, utilizing Table 8.1 will produce an appropriate outcome that is still applicable to the proceeding instruction. Adjustments to the turbulence model being used will produce different results and is most notable when it comes to the surface heating, shear stresses (vector components only), and occasionally the surface pressures. This is not to say that the solution is insensitive to other off-body values such as velocity, but that cases with strong turbulence model sensitivities can produce different ice shapes. If an alternate turbulence model is used for the example cases, the obtained solution will still be valid, but do not anticipate that you will obtain the same ice shapes as to what is shown.

The final outcome from the CFD run is to obtain a boundary and flow field solution at two different wall temperatures in a `.fvuns` or `.szplt` format. For the workflow shown, the Tecplot `.szplt` format was utilized. The type of files allowed as inputs to GlennICE can be found in Table 6.1. If the solver being utilized does not allow for direct exporting of allowable formats, loading the data into Tecplot and then subsequently writing it out as a `.szplt` format is recommended.

The solution for a single wall temperature is shown in Figure 8.2 with contours for the coefficient of pressure being seen along the model surface and symmetry plane. This case should not be difficult for the user to converge and does not need complex solver parameters to converge appropriately. The results shown converged the meanflow and turbulence equations to an RMS residual of $1e-14$ which approximately represents an 11-orders-of-magnitude reduction. It should be recognized that there is noticeable three-dimensionality within the flow field and will be appropriate for highlighting the fully three-dimensional ice accretion capabilities.



(a) Isometric Top



(b) Isometric Bottom

Figure 8.2: C_p contours for the ONERA M6 case listed in Table 8.1.

8.2 Single Bin Ice Accretion

This example case will utilize the flow solutions generated for the ONERA M6 case previously described. A single bin ice accretion case will be conducted to generate an ice shape at the conditions shown in Table 8.2. An example `glennice.nml` file has been included within the GlennICE distribution and can be found in the `Example_Cases/Single_Bin_Ice_Accretion` directory and is

the same as to what is provided below. This `glennice.nml` file will be utilized for each of the example cases that will be worked through with only the changes necessary for each example being written out.

Table 8.2: Icing Conditions

Diameter (μm)	LWC (g/m^3)	Spray Time (s)	Icing Temp. (K)	ρ_{ice} (kg/m^3)
20	0.51	2700	267.15	450

```
&files
  solution_in = "om6ste_volume_T1.szplt"
  solution_surface_in = "om6ste_boundary_T1.szplt"
  solution2_in = "om6ste_volume_T2.szplt"
  solution2_surface_in = "om6ste_boundary_T2.szplt"
  tecplot_out = "om6ste_tecplot_output.szplt"
  stl_outer_mold_line_out = "om6ste_IceShape.stl"
  volume_restart_file = "om6ste_Volume_Restart.dat"
  refinement_restart_file = "om6ste_Refinement_Restart.dat"
  surface_restart_file = "om6ste_Surface_Restart.dat"
/
&release
  inlet_coverage = "full"
/
&distribution
  diameter = 20
/
&bin_count
  nbins = 1
/
&adaptive_refinement
  fraction_contained_tol = 0.7
  max_face_count = 10000
  pct_converged_limit = 80
/
&surface_count
  number_of_surfaces=4
/
&surface_nml
  category = "Inlet" , "nocategory" , "nocategory" , "Icing"
/
&freestream
  icing_temperature = 267.150
  twc = 0.51
  time = 2700.0
  ice_density = 450
```



```

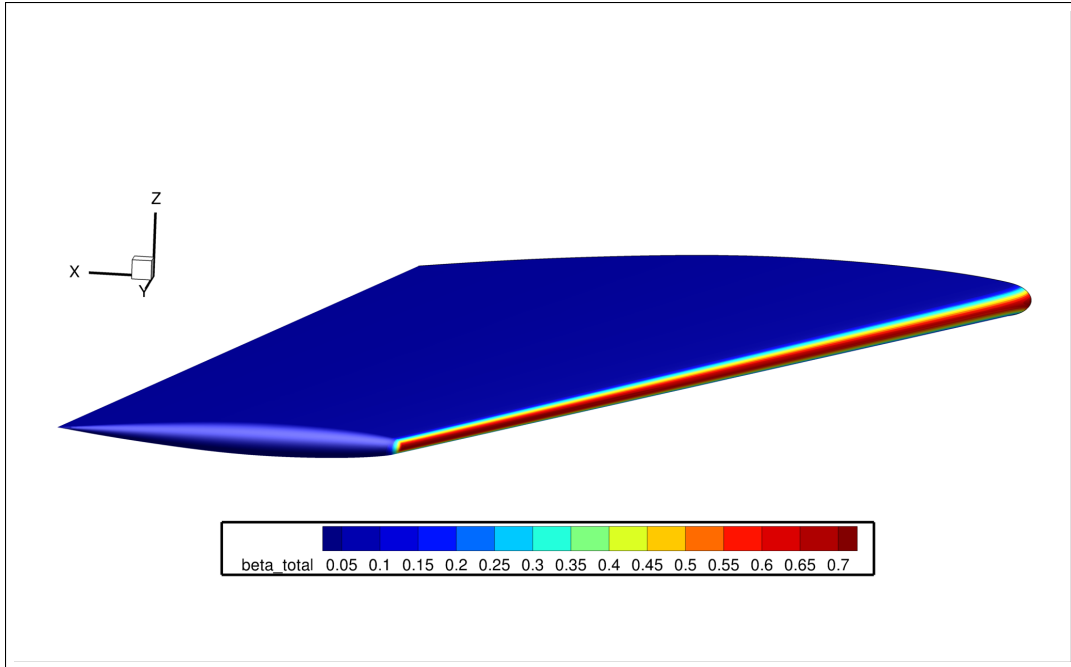
/
&htc_augmentation
  augmentation_type = "McClain_roughness"
  turbulent_htc_augmentation = 3.0
  ideal_rime_limit = 0.004
/
&variable_properties
  ! Scaling density
  map_index = 4
  scale_factor = 0.7710
/
&variable_properties
  ! Scaling u-velocity
  map_index = 5
  scale_factor = 322.2565
/
&variable_properties
  ! Scaling v-velocity
  map_index = 6
  scale_factor = 322.2565
/
&variable_properties
  ! Scaling w-velocity
  map_index = 7
  scale_factor = 322.2565
/
&variable_properties
  ! Scaling pressure
  map_index = 8
  scale_factor = 80068.45220 !  $\rho_{\infty} * a_{\infty}^2$ 
/
&variable_properties
  ! Scaling temperature
  map_index = 9
  scale_factor = 267.150
/
&variable_properties
  ! Scaling heating
  map_index = 10
  spelling = "heating"
  scale_factor = -10000.0 !  $W/cm^2$  to  $W/m^2$  and a sign change
/

```

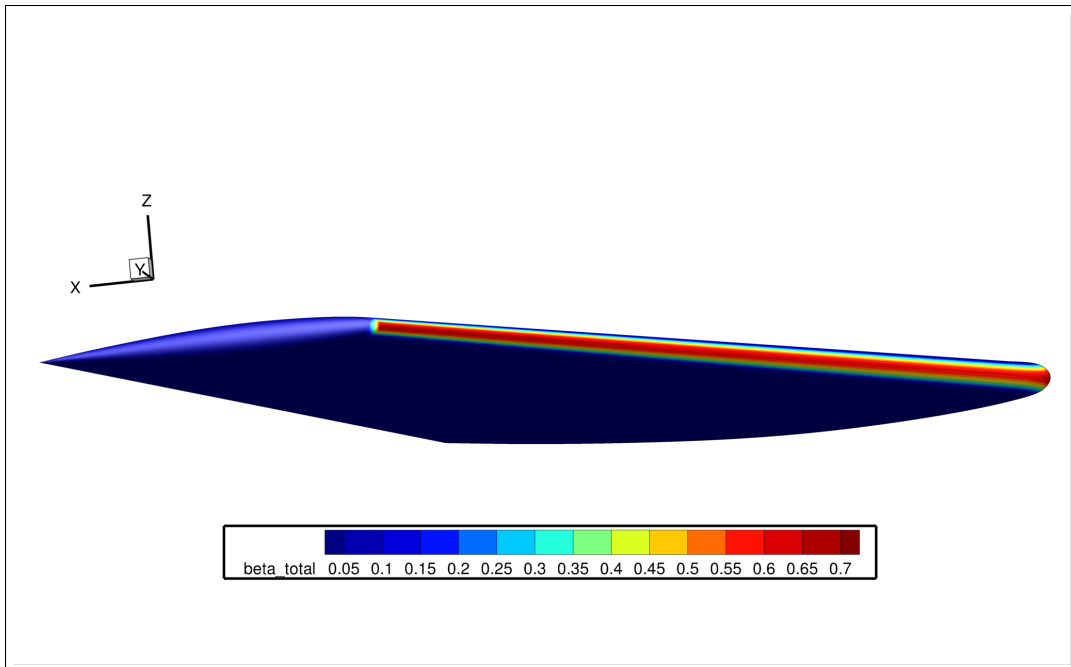
In this example it is assumed that the solution files are in the working directory,

but this is not necessary. The solution files can be accessed by GlennICE using the working directory as the starting point and being directed to where they are located. This can be done as such: `solution_in = "../Flow/solution_file_1.szplt"`. This case is being run from a PBS job submission script with the MPI based execution line as such:

```
$ mpiexec -np 128 glennice glennice.nml >  
    single_bin_ice_accretion_log.txt
```

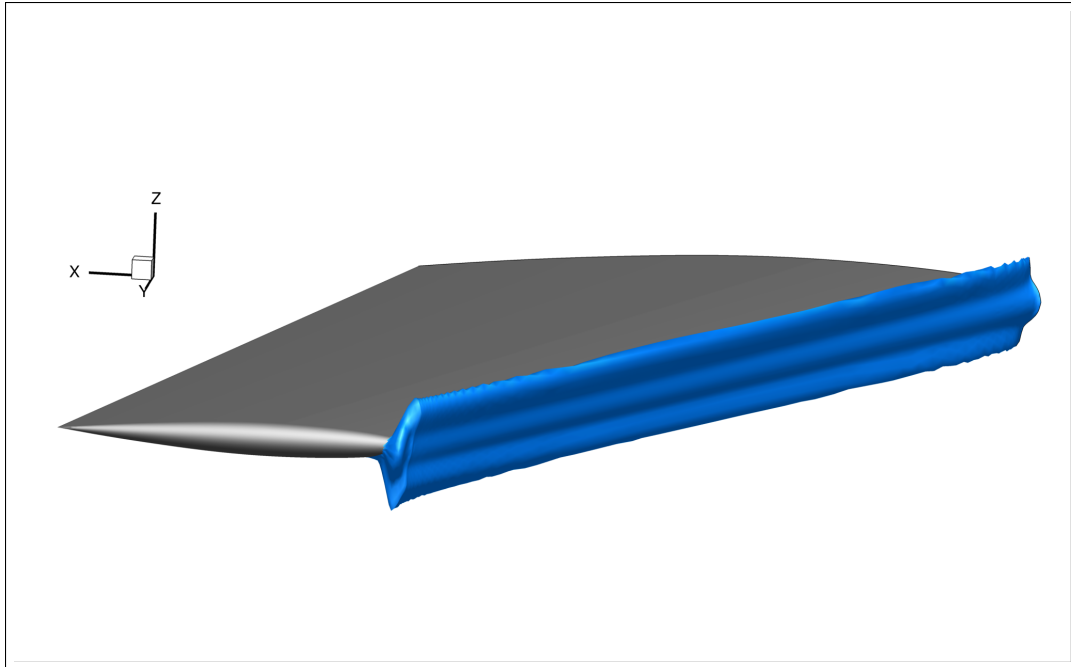


(a) Isometric Top View

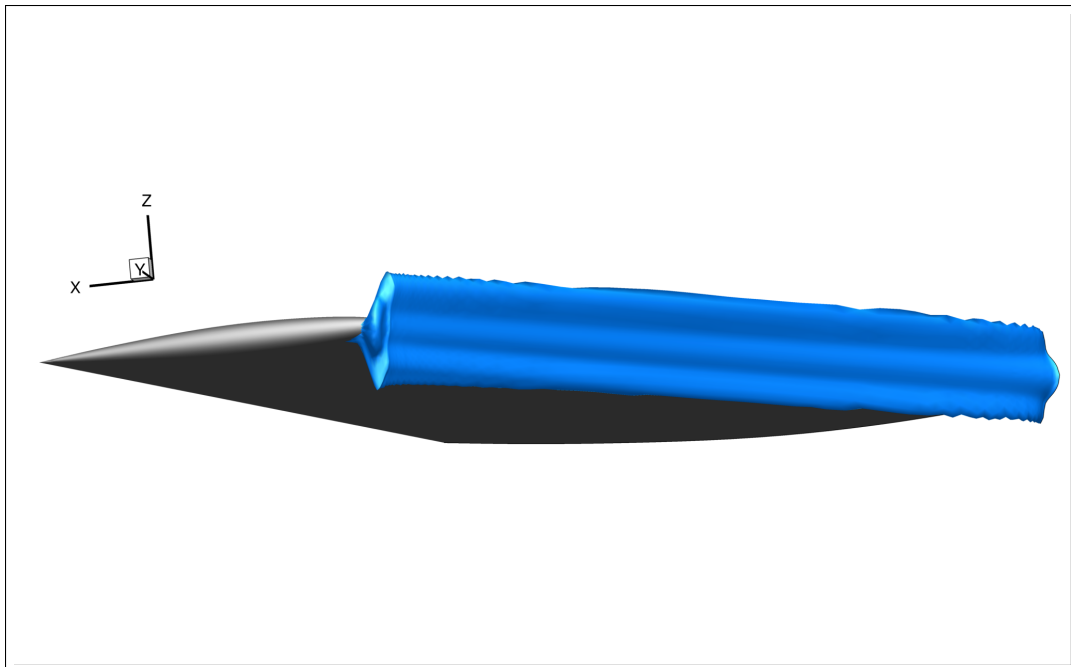


(b) Isometric Bottom View

Figure 8.3: Collection Efficiency / `beta_total`.



(a) Isometric Top View



(b) Isometric Bottom View

Figure 8.4: ONERA M6 3D Ice Shape.

8.3 Refinement Restart

This example case will utilize the solution solved for in **Section 8.2 - Single Bin Ice Accretion**. It is important to keep in mind that the users exact values obtained when running this case are likely to be different than what is provided. This is because the CFD solutions that are being given to GlennICE are certain to be slightly different. It is expected though that the global behavior of the users and the provided solution are qualitatively similar. Within **Section 8.2 - Single Bin Ice Accretion**, it was seen that the value selected for the `pct_converged_limit` was set to 80%.

The following is the output from the final iteration for **Section 8.2 - Single Bin Ice Accretion**:

```
Refinement iteration: 16, number release points: 9868242
  Surface 4: "boundary 4 Airfoil"
    Number faces hit: 5274, min hit count: 1, max hit count:
    9589, total hits: 9556528
    Total mass flow: 2.380617939E-03, 12 convergence:
    9.537353706E-05
    Percent of impacted faces converged (by metric):
    fraction_contained_tol: 86.04%, mass_flow_tol: 0.00%,
    total: 86.04%
```

It is seen that for the final iteration, GlennICE reports a `fraction_contained_tol` metric of 86.04%. This is above the 80% `pct_converged_limit` threshold that was specified in the `glennice.nml` file and is indicative of GlennICE completing successfully.

For the purpose of demonstrating the tools and capabilities, let us assume that there is a desire to increase the final iteration value of the `fraction_contained_tol` metric to be above a `pct_converged_limit` of 90% instead. A refinement restart will allow for GlennICE to compute additional trajectories and improve upon the convergence metrics without the need to start from scratch. This can save the user a significant amount of time since there is not a need to recompute trajectories that already exist. Additional information about the refinement restart capabilities can be found in **Section 4.3 - Restart Run**.

While keeping everything else the same, the following adjustments need to be made to the `glennice.nml` file. The `glennice.nml` file that will capture the changes shown below can be found in the `Example_Cases/Refinement_Restart` directory. These changes are as follows:

```
/
&adaptive_refinement
  fraction_contained_tol = 0.7
```

```
max_face_count = 10000
pct_converged_limit = 90 ! The restart adjusted value
/
```

The addition of `restartJob=Refinement` to the execution call of GlennICE is also required. GlennICE will be looking for a refinement restart file name defined in the `glennice.nml` file by the `refinement_restart_file` variable within the directory being run. The following line will be used for the refinement restart execution of GlennICE:

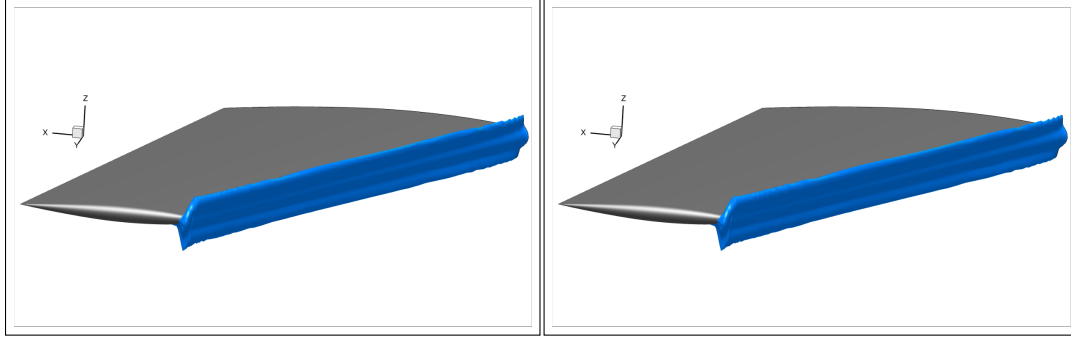
```
$ mpiexec -np 128 glennice glennice.nml restartJob=Refinement >
  refinement_restart_log.txt
```

Once the refinement restart is completed, the following output is obtained from the final iteration where the `pct_converged_limit` of 90% was utilized:

```
Refinement iteration: 17, number release points: 12892366
  Surface 4: "boundary 4 Airfoil"
    Number faces hit: 5288, min hit count: 1, max hit count:
    17762, total hits: 12321239
    Total mass flow: 2.380496000E-03, 12 convergence:
    1.961877346E-05
    Percent of impacted faces converged (by metric):
    fraction_contained_tol: 95.92%, mass_flow_tol: 0.00%,
    total: 95.92%
```

It can be seen now, that after performing the refinement restart, with the adjusted `pct_converged_limit`, that the final `fraction_contained_tol` value reported by GlennICE is 95.92%. Now, it is seen that the resultant ice shape is qualitatively the same with only minor differences being visible between the two variations in Figure 8.5, but were included for completeness. This process though can easily be extended to a solution that had a `pct_converged_limit` = 60 and the changes become more drastic.

To further illustrate what did occur doing this run, the `seed_area_fraction_contained_bin_1` variable is plotted at a mid-span slice location for both of the tolerances that were used in Figure 8.6. It is seen that the majority of the improvements to the solution within the restart were made closer to the impingement limits where more trajectories were released. The impacts to the solution under different values in the `glennice.nml` file for the `pct_converged_limit` and `fraction_contained_tol` variables is not guaranteed to behave in the same manner for every case.



(a) `pct_converged_limit` = 80%

(b) `pct_converged_limit` = 90%

Figure 8.5: `pct_converged_limit` effect on ONERA M6 3D Ice Shape.

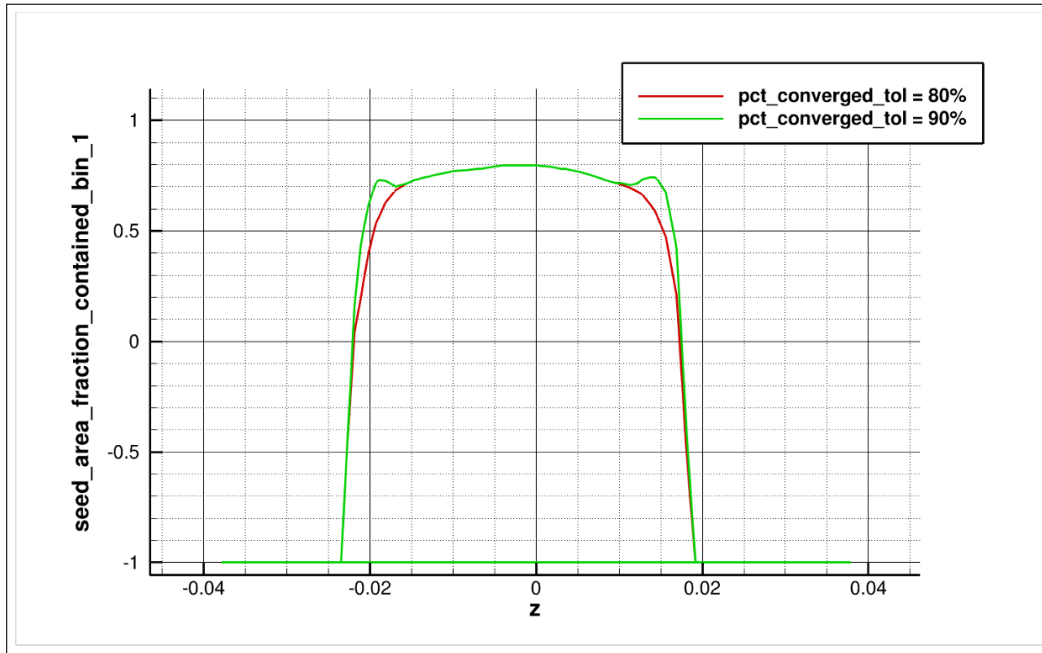


Figure 8.6: Results of changing `pct_converged_limit` at a slice location of $Y = 0.75$ m.

8.4 Droplet Distribution (Multi-Bin) Ice Accretion

Within **Section 8.2 - Single Bin Ice Accretion** and **Section 8.3 - Refinement Restart**, a single-bin droplet diameter simulation was calculated using a diameter of $20\mu\text{m}$. This example case will utilize the multi-bin capabilities within GlennICE to calculate a $20\mu\text{m}$ Langmuir-D droplet distribution [24]. The values for the droplet diameters and mass fractions have been computed and provided within Table 8.3. There does not currently exist a capability within GlennICE to set different convergence metrics for different droplet bins during a multi-bin calculation. This could cause issues when attempting to converge certain problems if the set tolerances for all bins cannot be satis-

fied. A workflow that can get around this deficiency is provided in **Section 8.5.1 - Multiple Single-Bin Combination**. Again, for this example case, all 7 bins will be computed within the same execution of GlennICE with extended capabilities being discussed in **Section 8.5.1 - Multiple Single-Bin Combination**.

Table 8.3: Langmuir-D MVD = 20 μm Droplet Distribution

Diameter (μm)	6	10	14	20	27	35	44
Mass Fraction	0.05	0.1	0.2	0.3	0.2	0.1	0.05

The necessary additions to the `glennice.nml` file are included below, but can also be found in the `Example_Cases/Multi_Bin_Ice_Accretion` directory. Only the `&distribution` and `&bin_count` sections need to be adjusted to the following lines.

```

/
&distribution
  diameter = 6.0, 10.0, 14.0, 20.0, 27.0, 35.0, 44.0
  mass_fraction = 0.05, 0.1, 0.2, 0.3, 0.2, 0.1, 0.05
/
&bin_count
  nbins = 7
/

```

This case is being run from a PBS job submission script with the MPI based execution line as such:

```

$ mpiexec -np 128 glennice glennice.nml >
  multi_bin_ice_accretion_log.txt

```

It is seen in Figure 8.9, the resultant ice shape obtained is significantly different than that of the single-bin analysis. This is highlighted by a photo comparing a slice at $Y = 0.75$ m for the single-bin and multi-bin analyses in Figure 8.10. The icing limits move further back on the leading edge, the horns get adjusted, and the attachment line ice thickness increases. Furthermore, it is seen in Figure 8.7 where a slice is taken at the mid-span location of the wing. Since GlennICE will provide a value of `beta_total` for each bin, it is straightforward to provide a comparison of each at this slice location. In Figure 8.8, the different values for `beta_total` are plotted, with the variable `beta_total_bin_#` being plotted for each diameter. Additionally, the combined value of `beta_total` for the 7-bin distribution is plotted alongside the results as well.

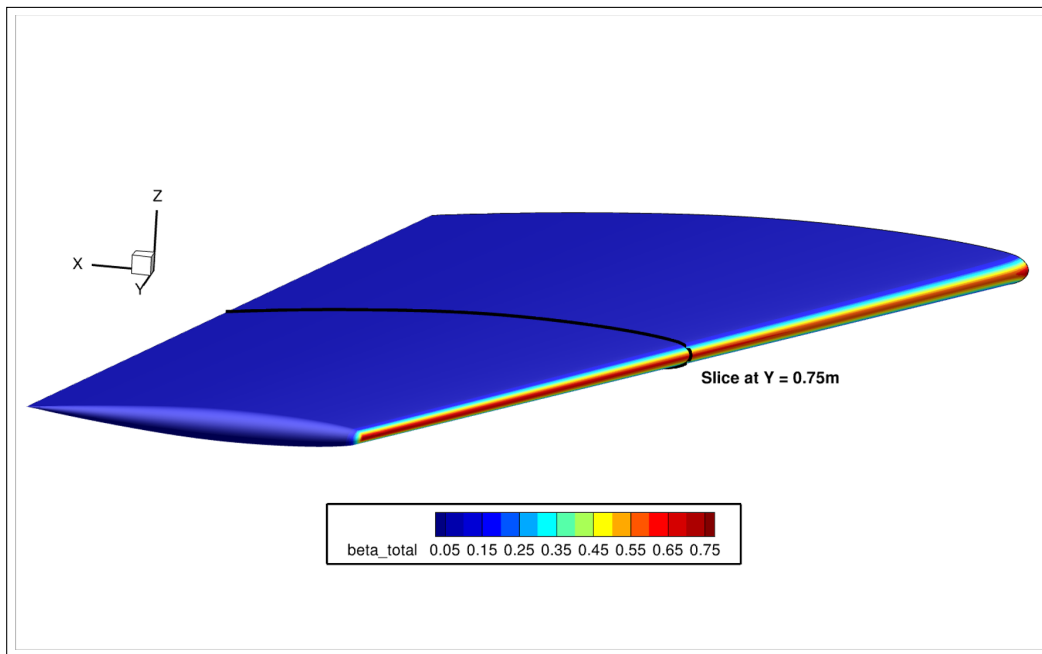


Figure 8.7: 7-Bin MVD = 20 μm Langmuir-D Droplet Distribution Total Collection Efficiency. A slice location of $Y = 0.75\text{ m}$ is shown.

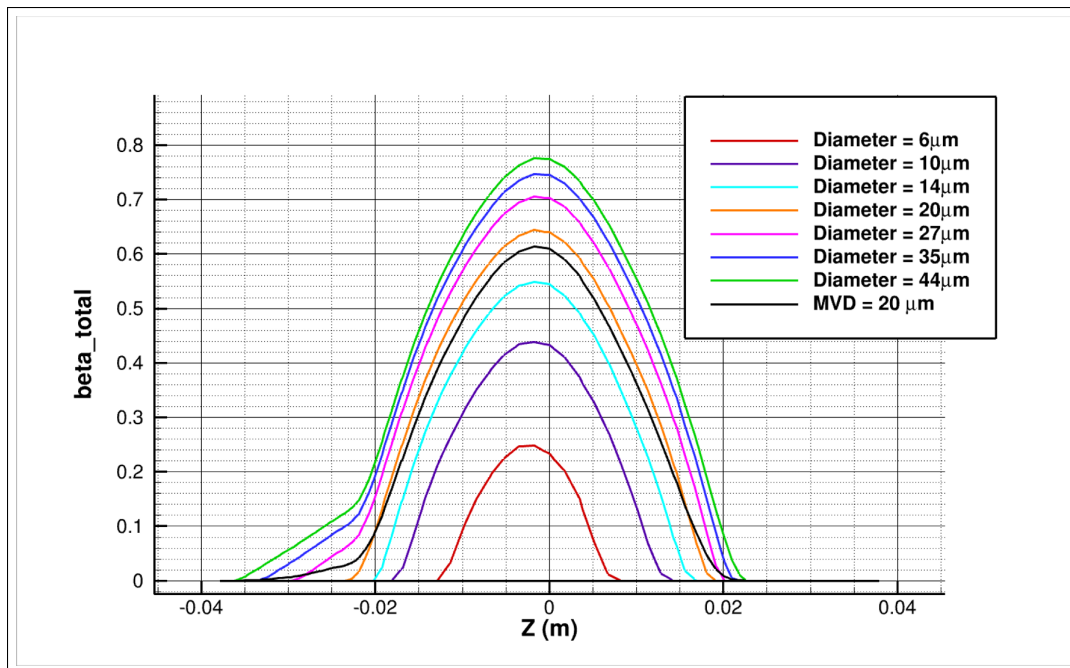


Figure 8.8: 7-Bin MVD = 20 μm Langmuir-D Droplet Distribution Collection Efficiency at $Y = 0.75\text{ m}$.

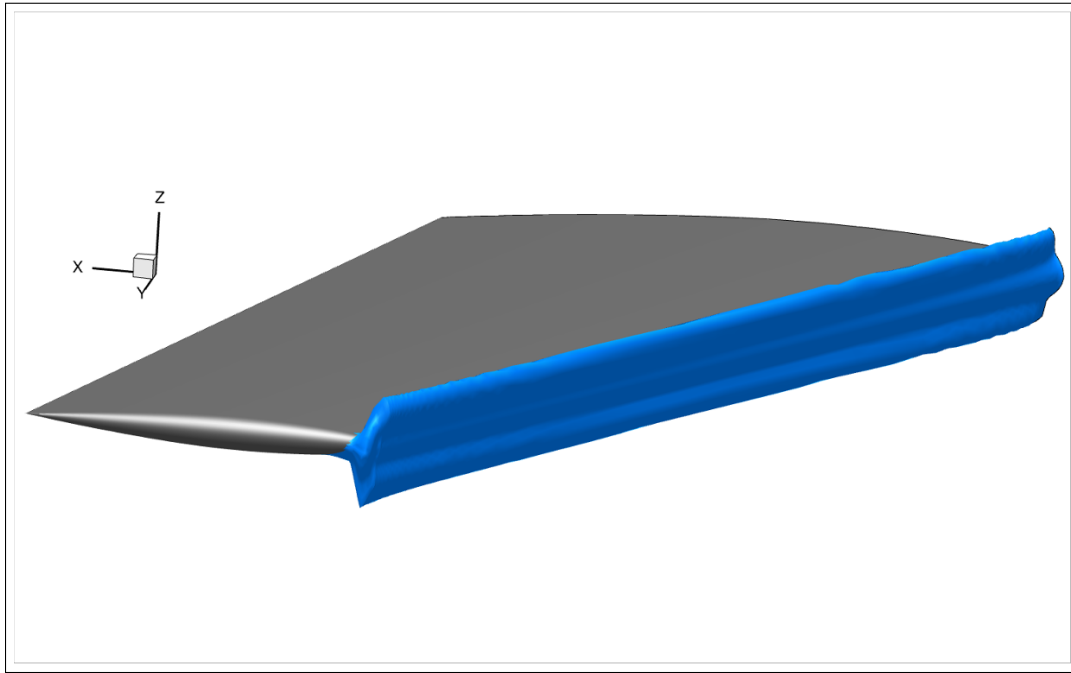


Figure 8.9: 7-Bin MVD = 20 μm Langmuir-D Droplet Distribution Ice Shape.

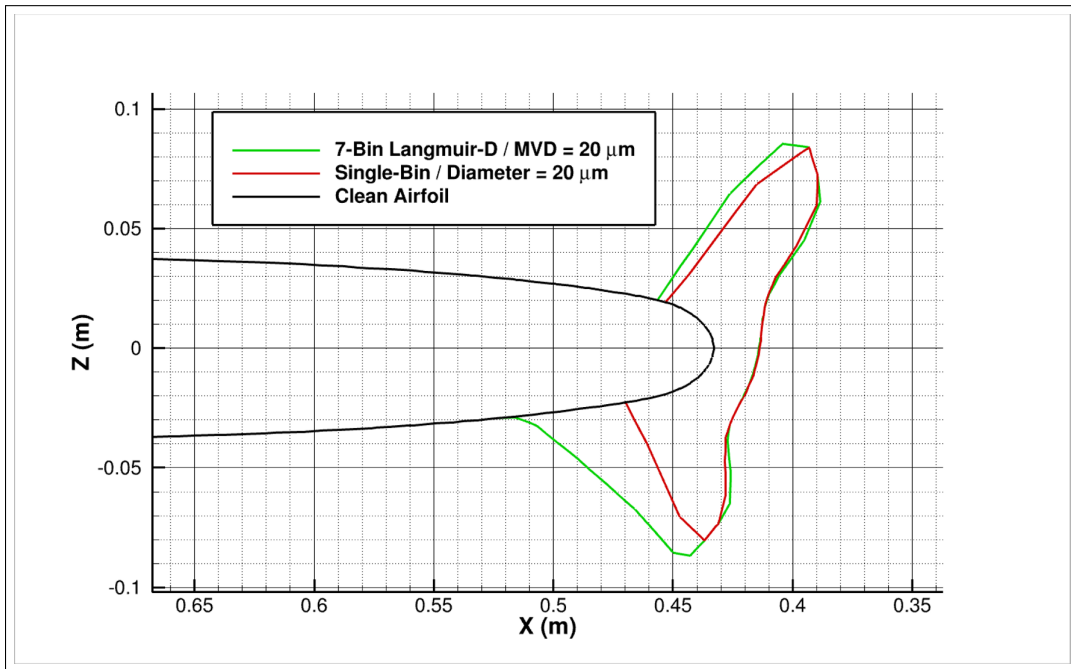


Figure 8.10: Single-Bin 20 μm diameter versus 7-Bin MVD = 20 μm Langmuir-D Ice Shape Comparison at Y = 0.75 m.

8.5 Surface Restart

GlennICE has the ability to use the results of a previous case in order to expedite parameter studies by conducting a Surface Restart. For **Section 8.4 - Droplet Distribution (Multi-Bin) Ice Accretion**, a 7-bin droplet distribution case was conducted to generate an ice shape at a temperature of `icing_temperature = 267.150 K`. For **Section 8.5 - Surface Restart**, it will be assumed that we want to get an ice shape at a different temperature, but utilize the same trajectories as were calculated before. This method allows for GlennICE to recompute the ice shape and surface variables for a new `icing_temperature` in a fraction of the amount of time it would take to do this same analysis from scratch.

The necessary additions to the `glennice.nml` namelist file are included below, but can also be found in the `Example_Cases/Surface_Restart` directory. Only the `&freestream` section needs to be adjusted to the following lines for this example.

```
/
&freestream
  icing_temperature = 257.15 ! Surface Restart Adjusted Value
  twc = 0.51
  time = 2700.0
  ice_density = 450
/
```

The addition of `restartJob=Surface` to the execution call of GlennICE is also required. GlennICE will be looking for a surface restart file name defined in the `glennice.nml` file by the `surface_restart_file` variable within the directory being run. The following line will be used for the surface restart execution of GlennICE:

```
$ mpiexec -np 128 glennice glennice.nml restartJob=Surface >
  surface_restart_log.txt
```

After completion of the surface restart, the following information was extracted from the GlennICE output for **Section 8.4 - Droplet Distribution (Multi-Bin) Ice Accretion** and the surface restart that was just conducted for **Section 8.5 - Surface Restart**. The information is meant to highlight numerical differences computed within the energy balance along with the newly computed ice shape.

For **Section 8.4 - Droplet Distribution (Multi-Bin) Ice Accretion**, the following information was obtained based on an `icing_temperature` of 267.150 K.

```
Final energy balance
```

```

Initial mass balance for surface 4: "boundary 4 Airfoil"
Water mass in          =      0.237777708457856E-02 kg/s
Ice formed on impact   =      0.639804617968059E-03 kg/s
Water evaporated       =      0.301341110651104E-03 kg/s
Water runback         =      0.143663135595940E-02 kg/s
Global mass imbalance  =      0.000000000000000      kg/s
RMS mass imbalance     =      0.000000000000000      kg/s
Max face imbalance     =      0.000000000000000      kg/s
Water runback converged after 198 runback iterations
Total remaining runback =      0.898136108879612E-16 kg/s after
    198 runback iterations
    RUNBACK_TOLERANCE =      0.100000000000000E-15

Final mass balance for surface 4: "boundary 4 Airfoil"
Total ice formed       =      0.163324125571289E-02 kg/s
Water leaving surface  =      0.663914044394877E-05 kg/s
Water stuck on face    =      0.436555564245496E-03 kg/s
Remaining runback      =      0.898136108879612E-16 kg/s
Global mass imbalance  =      0.000000000013525      kg/s
RMS mass imbalance     =      0.0000000000009512      kg/s
Max face imbalance     =      0.0000000000006980      kg/s
Runback iterations     =      198
Computing ice growth

```

For **Section 8.5 - Surface Restart**, the following information was obtained based on an `icing_temperature` of 257.150K.

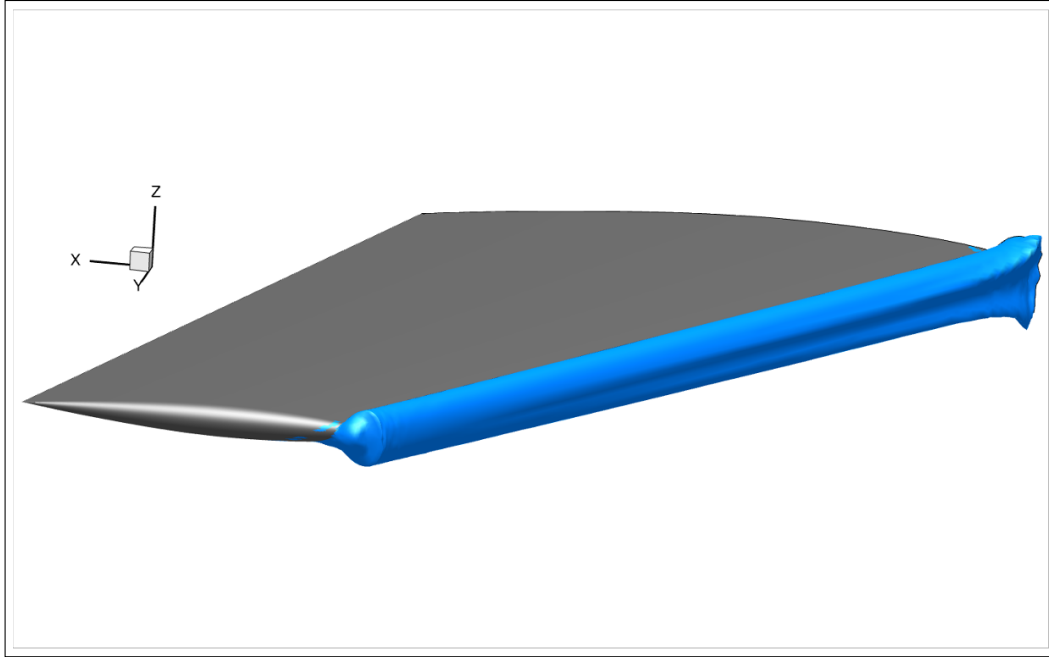
```
Final energy balance
Initial mass balance for surface 4: "boundary 4 Airfoil"
Water mass in           =      0.221734128591442E-02 kg/s
Ice formed on impact    =      0.189234548743984E-02 kg/s
Water evaporated        =      0.283971979747082E-03 kg/s
Water runback           =      0.410238187275019E-04 kg/s
Global mass imbalance   =     -0.0000000000000000 kg/s
RMS mass imbalance      =      0.0000000000000000 kg/s
Max face imbalance      =      0.0000000000000000 kg/s
Water runback converged after 43 runback iterations
Total remaining runback =      0.657653341073575E-16 kg/s after
    43 runback iterations

Final mass balance for surface 4: "boundary 4 Airfoil"
Total ice formed        =      0.193286008199333E-02 kg/s
Water leaving surface   =      0.509224173939785E-06 kg/s
Water stuck on face     =      0.226228022713313E-20 kg/s
Remaining runback       =      0.657653341073575E-16 kg/s
Global mass imbalance   =     -0.0000000000000000 kg/s
RMS mass imbalance      =      0.0000000000000000 kg/s
Max face imbalance      =      0.0000000000000000 kg/s
Runback iterations      =      43
Computing ice growth
```

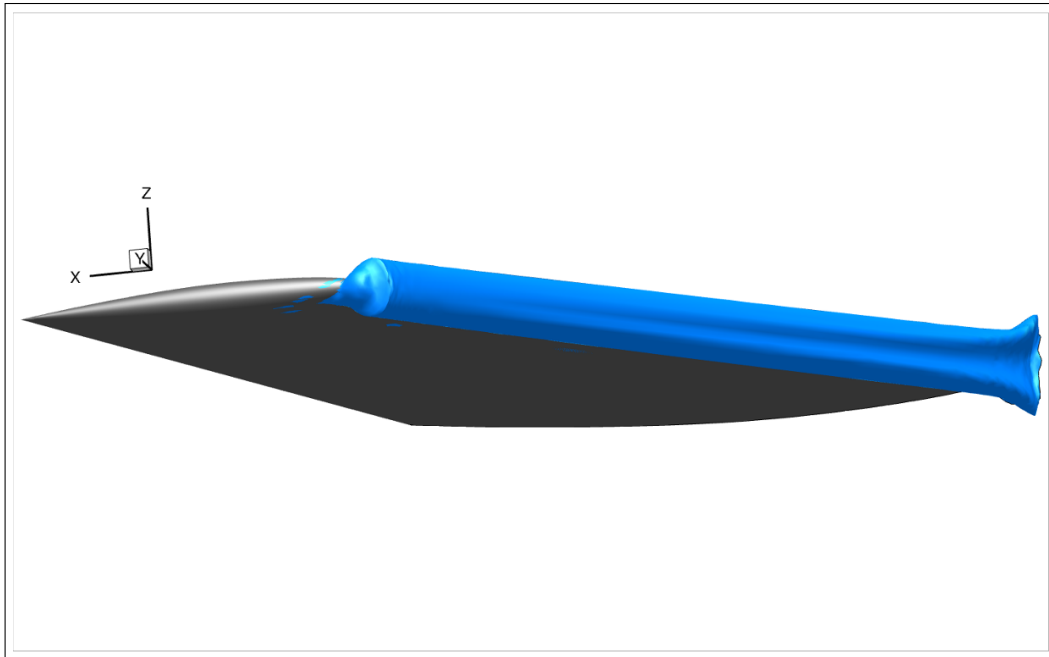
It is seen that the amount of ice formed on impact becomes much larger due to the case being at a more rime ice condition than a glaze ice condition. Additionally, it is seen that the amount of water runback is significantly larger for the warmer temperature case for the same reason. The resultant ice shape can be seen in Figure 8.11 where a much more rime ice shape has been calculated. Again, this solution was obtained by simply adjusting the `icing_temperature` value to be 10 K lower and conducting a surface restart.

8.5.1 Multiple Single-Bin Combination

An extended capability of GlennICE is the ability to combine multiple, separately calculated droplets, into a single combined result using a Surface Restart. This process can be advantageous if a certain particle diameter struggles to satisfy the same `pct_converged_limit` tolerances for all diameters. This can occur when the equations being solved for by GlennICE become more stringent; small diameter particles often succumb to this. Additionally, no capability exists within GlennICE that allows for the user to specify different `pct_converged_limit` values for different bins, as mentioned previously.



(a) Isometric Top View



(b) Isometric Bottom View

Figure 8.11: Ice Shape for a 7-Bin $MVD = 20 \mu m$ Langmuir-D `icing_temperature` adjustment.

The example cases should not struggle with the 7-bin Langmuir-D distribution that was provided, but it can be seen that the number of iterations before satisfying the `pct_converged_limit` changes for different droplet diameters. Additionally, the following instructions are provided to the user as an infor-

mative manner in which to describe the process.

For each droplet diameter, a single-bin computation would be conducted (as done in **Section 8.2 - Single Bin Ice Accretion**), for each droplet diameter listed in Table 8.3. Each single-bin computation will print out a surface restart file with a name that was specified within `&files:surface_restart_file`. This file name gets appended with the diameter specified within the `glennice.nml` file such that a specification of `om6ste_Surface_Restart.dat`, with a `&distribution:diameter = 20`, will get printed out as `om6ste_Surface_Restart_20p0.dat`.

Once each single-bin droplet diameter is calculated (likely in different directories), the surface restart files for each bin will need to be copied into a new directory to be collocated. This would mean, that if this process were to be done for the diameters in Table 8.3, the following restart file names would need to all be located within the same directory:

- `om6ste_Surface_Restart_6p0.dat`
- `om6ste_Surface_Restart_10p0.dat`
- `om6ste_Surface_Restart_14p0.dat`
- `om6ste_Surface_Restart_20p0.dat`
- `om6ste_Surface_Restart_27p0.dat`
- `om6ste_Surface_Restart_35p0.dat`
- `om6ste_Surface_Restart_44p0.dat`

Then, the adjustments to the `glennice.nml` file would be consistent with the 7-bin distribution given in **Section 8.4 - Droplet Distribution (Multi-Bin) Ice Accretion** and the execution of GlennICE would mimic what is done for **Section 8.5 - Surface Restart**. That is to say that the following lines should be a part of what is in the `glennice.nml` file that is provided to GlennICE. It is important to note that the values in `&distribution:diameter` need to be consistent with the values appended to the surface restart files within the run directory. If they are not the same, the correct file will not be included by GlennICE. Furthermore, note that you do not need to supply GlennICE with the names of each surface restart file in `&files:surface_restart_file`, only the file name that prepends the droplet diameter surface restart files calculated separately.

```
/
&files
  solution_in = "om6ste_volume_T1.szplt"
  solution_surface_in = "om6ste_boundary_T1.szplt"
  solution2_in = "om6ste_volume_T2.szplt"
  solution2_surface_in = "om6ste_boundary_T2.szplt"
```

```

tecplot_out = "om6ste_tecplot_output.szplt"
stl_outer_mold_line_out = "om6ste_IceShape.stl"
volume_restart_file = "om6ste_Volume_Restart.dat"
refinement_restart_file = "om6ste_Refinement_Restart.dat"
surface_restart_file = "om6ste_Surface_Restart.dat"
/
&distribution
  diameter = 6.0, 10.0, 14.0, 20.0, 27.0, 35.0, 44.0
  mass_fraction = 0.05, 0.1, 0.2, 0.3, 0.2, 0.1, 0.05
/

```

Once these steps are completed, GlennICE will produce a result that combines the seven single-bin computations into a result that is consistent with **Section 8.4 - Droplet Distribution (Multi-Bin) Ice Accretion**, by utilizing a surface restart. Further restarts can also be conducted utilizing this multi-bin result as was described earlier on in **Section 8.5 - Surface Restart**. Results based on the analysis methodology discussed are not provided since they would be the same as results obtained using previous methods.

8.6 Trajectory Visualization

The visualization of trajectories can be useful for a variety of reasons, such as analysis and/or debugging. Currently GlennICE allows the user to visualize trajectories by writing the trajectory history to an ASCII Tecplot file called `trajectories.dat`.

This functionality is not intended to be run in conjunction with a full GlennICE simulation, due to the impact on computational efficiency as well as file size requirements to store a significant number of trajectories. Instead, this functionality is intended to visualize a reasonable sample of trajectories to better understand the behavior of trajectories in a region of interest, or to aid in debugging bad trajectories.

Currently the trajectory visualization implementation is not algorithmically decoupled from the algorithms that compute mass flux. As such the requirement that `n_ytraj` and `n_ztraj` of the `&release` namelist must be greater than one are required here as well. However, one can emulate a line rake quite easily. Note that for the release domain specified, each trajectory is essentially duplicated, with the end result emulating a single line rake. The results of the computed trajectories around the mid-span location of the wing can be seen in Figure 8.12.

The necessary adjustments to the `glennice.nml` namelist file are included below, but can also be found in the `Example_Cases/Trajectory_Visualization` directory. The user may find it beneficial to check between the namelist they are to run with the one provided to ensure a proper execution of GlennICE. The `&release`, `&mass_flux_solver`, `&diameter`, and the `&trajectory_solver` sections of the namelist need to be adjusted to the following lines for this example. This methodology does work for multiple bins as well with each trajectory, for each bin, being generated; a 7-bin case would generate 280 trajectory zones total for these parameters. For simplicity, we have adjusted the diameter back to what was used for **Section 8.2 - Single Bin Ice Accretion**.

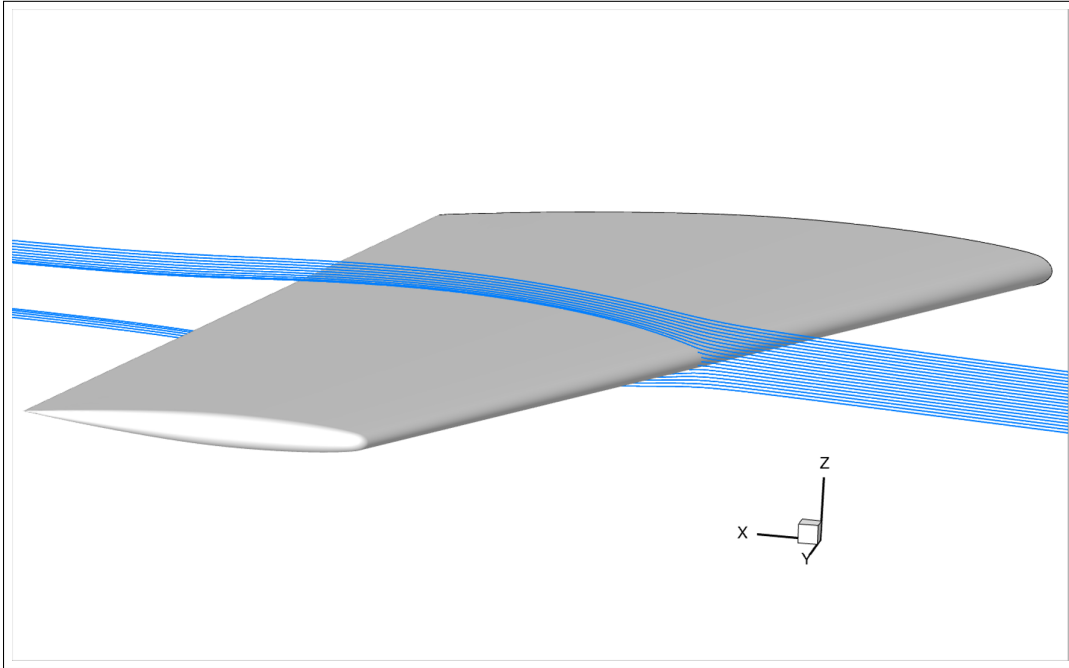
```
&release
  n_ytraj = 2
  n_ztraj = 20
  z_min = -7.05
  z_max = -6.95
  y_min = 0.74
  y_max = 0.76
/
&mass_flux_solver
  max_iterations = 1
/
&trajectory_solver
```

```
write_histories = .true.  
trajectory_summary_output = "all"  
/  
&distribution  
  diameter = 20.0  
/
```

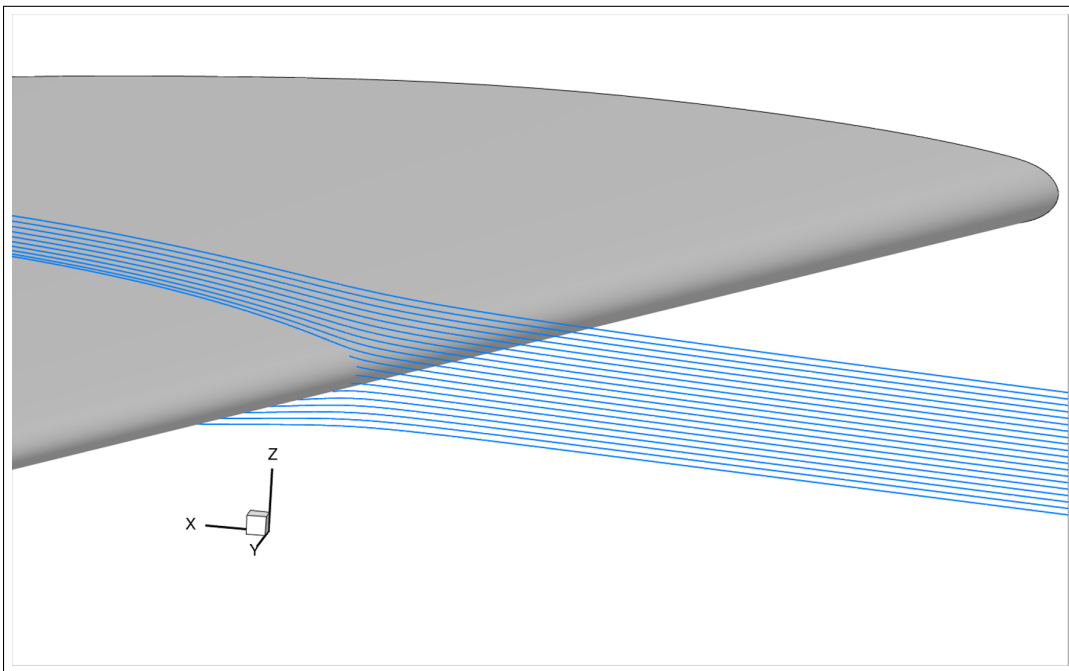
Additionally, it is important to note that GlennICE does not currently support the writing of trajectory histories in parallel. Future versions of GlennICE may support this option, but it is not currently available. This means that a more significant change is needed for the execution of GlennICE that will likely need to occur either on a login/compute node or on a local machine. The execution call will depend on how you have GlennICE setup, but the following is valid:

```
$ ./glennice glennice.nml > trajectory_visualization_log.txt
```

In order to visualize the results, it is recommended the user load in both the surface boundary file and the `trajectories.dat` file into Tecplot. Each trajectory gets put into a single Tecplot zone that can be turned on and off by itself. When the Mesh is turned on, the trajectories will appear on the Tecplot screen for visualization. It is recommended the user also turns off the boundary mesh in the Zone Style menu. Trajectories (and their subsequent particles) that impact the surface will be seen to terminate at the boundary surface and can be more easily seen in Figure 8.12b.



(a) Expanded View



(b) Close-Up View

Figure 8.12: Visualization of a Sample of 20 μm Droplet Trajectories.

9 User Tips

This section is intended to provide users with advice on running cases with this version of GlennICE. If you need additional guidance, send your questions to GlennICE-support@lists.nasa.gov.

9.1 Compiling

GlennICE uses many features of modern Fortran. Older compilers may not successfully build the executable.

9.2 Namelist

It is best to start with the namelist provided in **Section 8.2 - Single Bin Ice Accretion** as it contains a valid input for running an ice accretion case. It is most likely the case that this namelist is not sufficient for most user workflows. It is recommended to work through the example cases or reference the **Section 6.1 - Namelist File** section to garner a better understanding of the GlennICE workflow.

Namelist format requires that the variable names are spelled correctly. If the user puts a variable in a namelist that cannot be read, the entire namelist is ignored. As an example, if the user spells variable `relhumidity` in the `&freestream` namelist as `Rel_Humidity`, then all of the variables in namelist `freestream` are ignored. The software will stop based on this error and inform the user which variable was not read. However, it will stop on the first error it encounters. An input file with multiple misspellings would require multiple runs to debug.

9.3 Adaptive Refinement

An annular geometry is utilized in this section to demonstrate the algorithm. The annulus is defined by an ellipse in the xz -plane rotated a full revolution about the x axis. The ellipse has coefficients a and b of 0.1 and 0.5 respectively with a center located at $(0, 0, 1)$. The semi-major axis is in the x direction and the semi-minor axis is in the z direction. Figure 9.1 depicts the annulus and the discretized surface mesh used for the simulation.

The adaptive refinement methodology employed here is an algorithm that is solved in the domain of the two dimensional seed plane. For more detail on the concept of a two dimensional seed plane, the reader is referred to **Section 9.10 - Seed Points and Release Points**.

For each trajectory, the minimum distance that trajectory comes to a body of interest is identified and stored. Currently GlennICE computes the value of wall distance internally based on the icing surface specified within **Section 6.1.3 - &surface_nml Namelist**. Therefore the bodies of interest are the

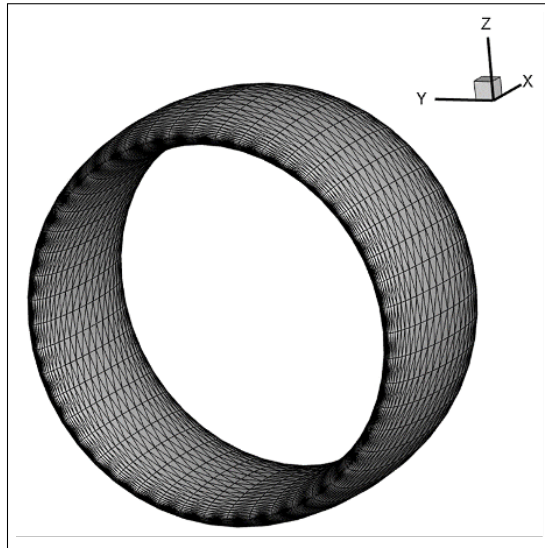


Figure 9.1: Geometry and surface mesh of the annular test case.

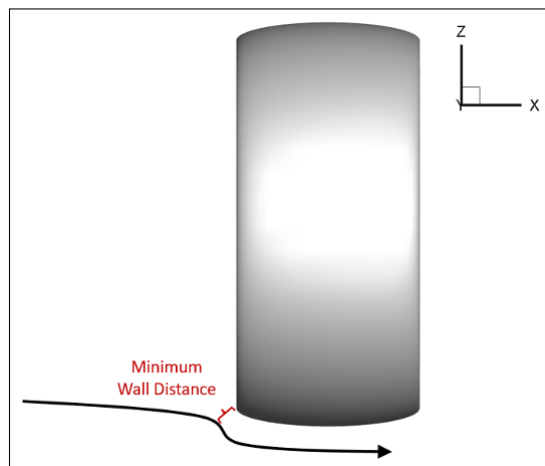


Figure 9.2: Schematic of the "minimum wall distance" associated with a trajectory.

viscous walls specified in the CFD simulation. The value of minimum wall distance for a given trajectory is associated with the seed point it was generated from. This results in a two dimensional contour which can be seen in Figure 9.3. However, one does not know the analytic solution of minimum wall distance a priori. Thus, the algorithm operates on the available discrete point cloud present at a given iteration. Figure 9.3 also illustrates this discretized data set.

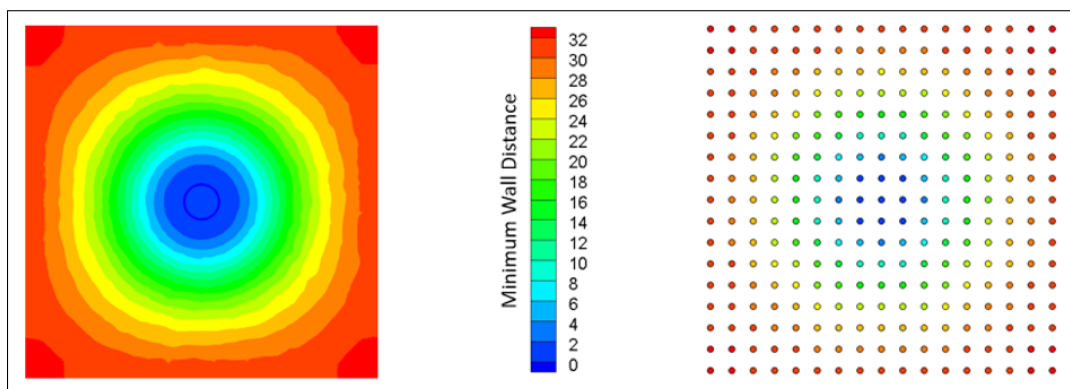


Figure 9.3: A two dimensional contour of minimum wall distance on the seed plane (left), and an example of the discrete point cloud the adaptive refinement algorithm considers (right).

The point cloud is organized into a mesh using delaunator-cpp [17], a Delaunay triangulation [18] library written in C++. The connectivity of the mesh is leveraged in multiple portions of the adaptive refinement algorithm, such as the determination of a seed point's neighbors, generation of the subsequent seed point refinement, and the computation of each trajectory's stream tube area. An illustration of the triangularization process can be seen in Figure 9.4.

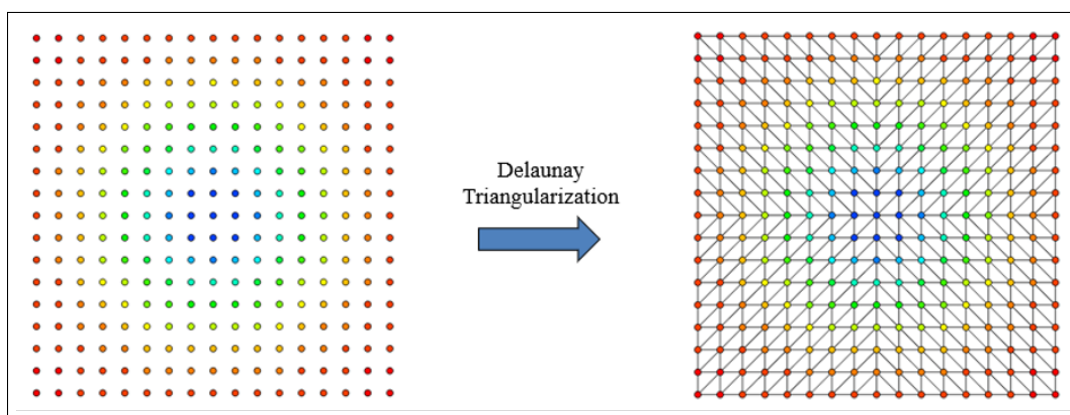


Figure 9.4: A cloud of seed points (left), and the resulting mesh generated from a Delaunay triangularization of this point cloud (right).

The desire of this algorithm is to only release trajectories that will result in impingement, i.e. minimum wall distances of exactly zero. These regions

are the global minimum of the two dimensional data set described above. A global minimum search is first performed on the discrete data set to identify regions where trajectories are likely to impinge. Figure 9.5 illustrates the result of the global minimum search on two sequential levels of refinement from the namelist below. The annulus in Figure 9.5 is where the global minimum of zero occurs. At this level of refinement, there are still relatively few hit trajectories (zero minimum wall distance). However, the region of impingement is clearly identifiable.

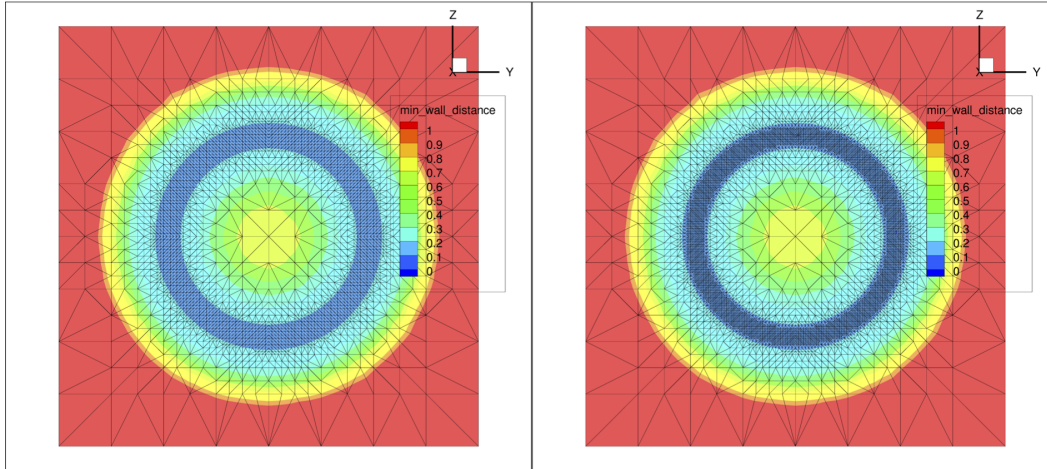


Figure 9.5: The global minimum search as performed on two sequential levels of adaptive refinement.

It should be noted that due to the sink-like behavior of the minimum wall distance, a local minimum occurs when neighboring seed points lead to trajectories that pass on separate sides of a geometry. Thus the local minimums are regions of interest as there is likely geometry of interest in this region of the mesh. Also note how a seed point that was a local minimum on one refinement level may not be a local minimum on a subsequent refinement level. There will be a point as the algorithm progresses where there will be no more local minimums. This occurs when the level of refinement has progressed to the point where all of the trajectories released hit the surface. The **adaptive_refinement** section of the namelist file is shown below. The sections of the namelist not being shown are consistent with the example cases.

```
&mass_flux_solver
  Max_iterations = 16
/
&adaptive_refinement
  max_face_count=100000
  fraction_contained_tol=1.
  mass_flow_tol=0.
  hit_percent_limit=100.
  turn_off_bounded_faces=.false.
```

```

use_nonuniform_tolerance=.false.
pct_converged_limit=100.
/

```

9.3.1 Convergence and Efficiency Parameters

The values in the `adaptive_refinement` namelist above were deliberately chosen so the baseline case would not converge such that the benefit of changing the `adaptive_refinement` inputs in this section could be demonstrated. With these inputs, 16 iteration resulted in 9,548,213 trajectories of which 8,811,741 hit the surface but convergence was not achieved as it was not possible to meet the criteria chosen.

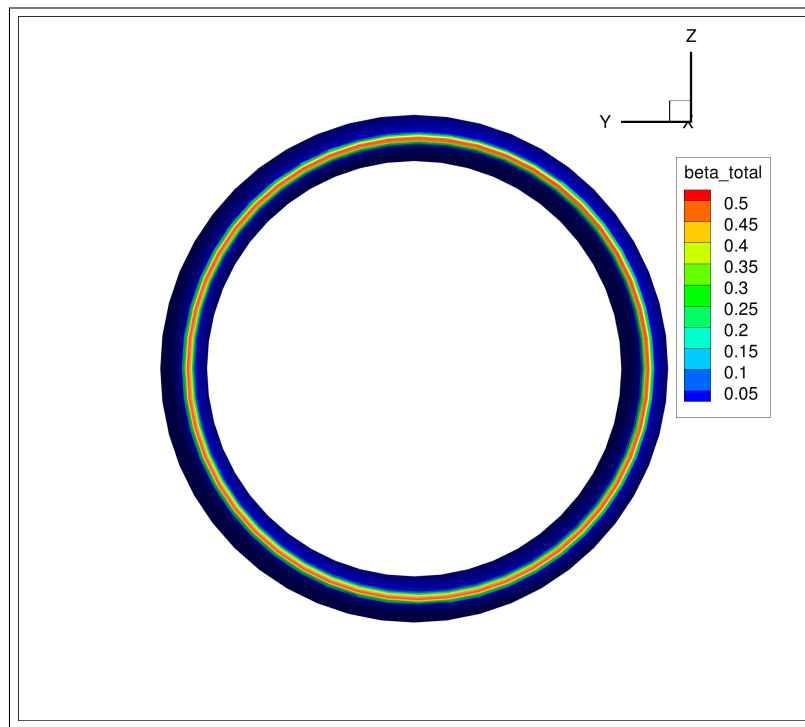


Figure 9.6: Collection Efficiency Contours for Annulus Case.

The first step is to examine the baseline case generated with the partial namelist supplied in this section. Figure 9.6 shows the collection efficiency contours for the case described. The contours look uniform around the annulus as expected and the results qualitatively look like what a knowledgeable icing analyst would expect for this geometry and the conditions supplied. This uniformity is confirmed by the collection efficiency slices at the azimuthal locations as shown in Figure 9.7. Further confirmation is supplied by the plot of seed point locations in Figure 9.8. These points are also uniform and concentrated at locations that hit the surface. It is seen that these seed points result in trajectories that hit the surface and can be confirmed by plotting

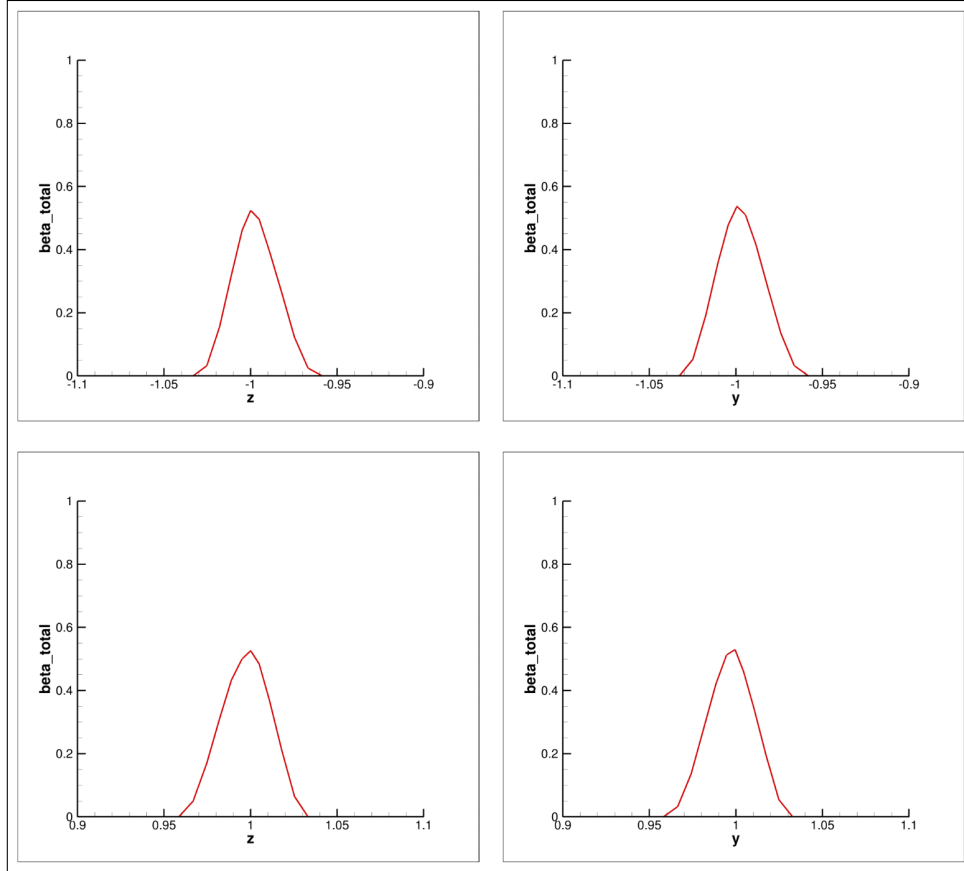


Figure 9.7: Collection Efficiency Contours at Azimuthal Locations for Annulus Case.

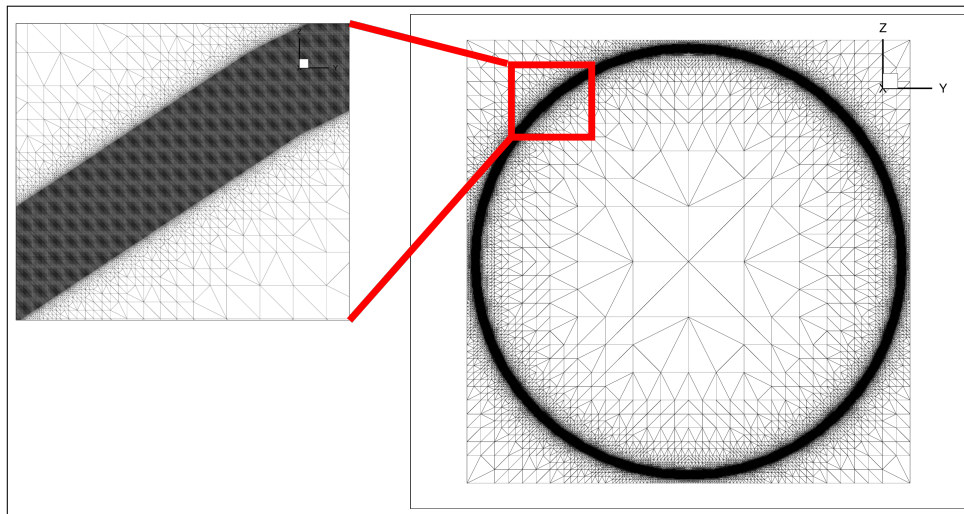


Figure 9.8: Seed Point Grid for Annulus Case.

the variable `intersected_surface` in the seed plane output file for the last iteration as shown in Figure 9.9.

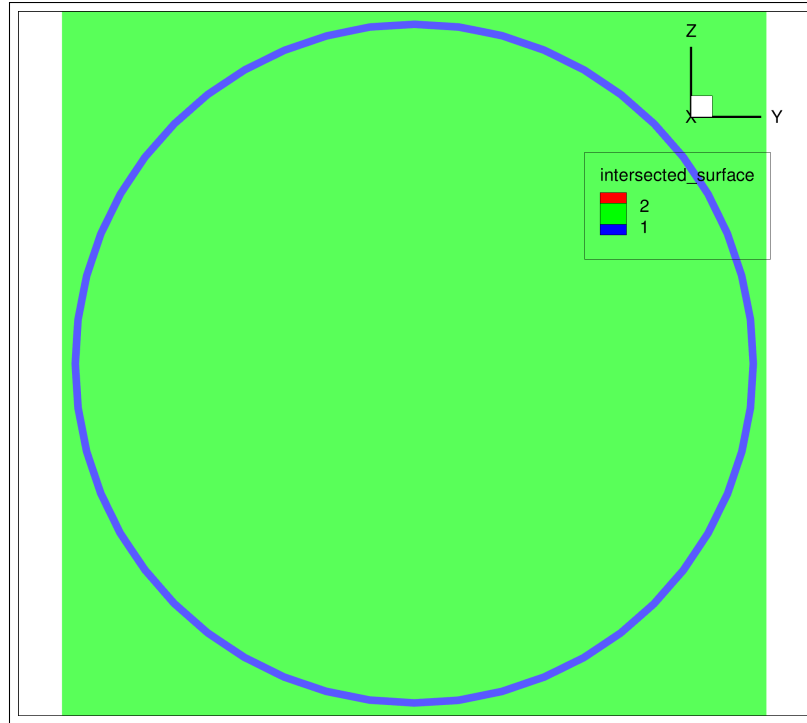


Figure 9.9: Hit Surfaces on Seed Plane Grid.

9.3.2 Convergence Metrics

Even though the results look qualitatively acceptable, the user would want to have some indication that the result is converged. The two parameters that can be used to set convergence are `fraction_contained_tol` and `mass_flow_tol`. The `fraction_contained_tol` parameter looks at the fraction of seed point faces that are contained on a CFD face. Larger fractions mean that more of the seed point faces lie entirely within the CFD face. The `mass_flow_tol` parameter looks at the difference in impinging mass flow from one iteration to the next. Lower values mean that the results are not changing from one iteration to the next. Two additional cases were performed with each metric changed to show the ability of GlennICE to converge on a result. The first case used a `fraction_contained_tol` of 0.3 and the second used a `mass_flow_tol` of 0.01 (1% difference). CFD faces that meet these criteria will no longer be refined in GlennICE. The resulting collection efficiencies appear nearly identical to the baseline case. There is no visible difference in collection efficiency at the azimuthal locations as shown in Figure 9.10. The maximum difference in collection efficiency at any location for the `fraction_contained_tol` case is less than 0.018 and for the `mass_flow_tol` case it was 0.015. These differences are less than the differences in collection efficiency around the annulus, which is a symmetric geometry as shown in Figure 9.11. The collection efficiency results are not symmetric because the flow field is not symmetric as shown in Figure 9.12. After 16 iterations, the `fraction_contained_tol` case reached

99.88% convergence using 1,897,968 trajectories, 1,161,496 of which hit the surface. This result would be considered converged using the default `pct_converged_limit` of 90%. The `mass_flow_tol` case reached 79.16% convergence using 3,913,487 trajectories and 3,177,015 hits. In either case, if full convergence is desired, the user could continue the simulations to 100% convergence by submitting a `refinement_restart` case.

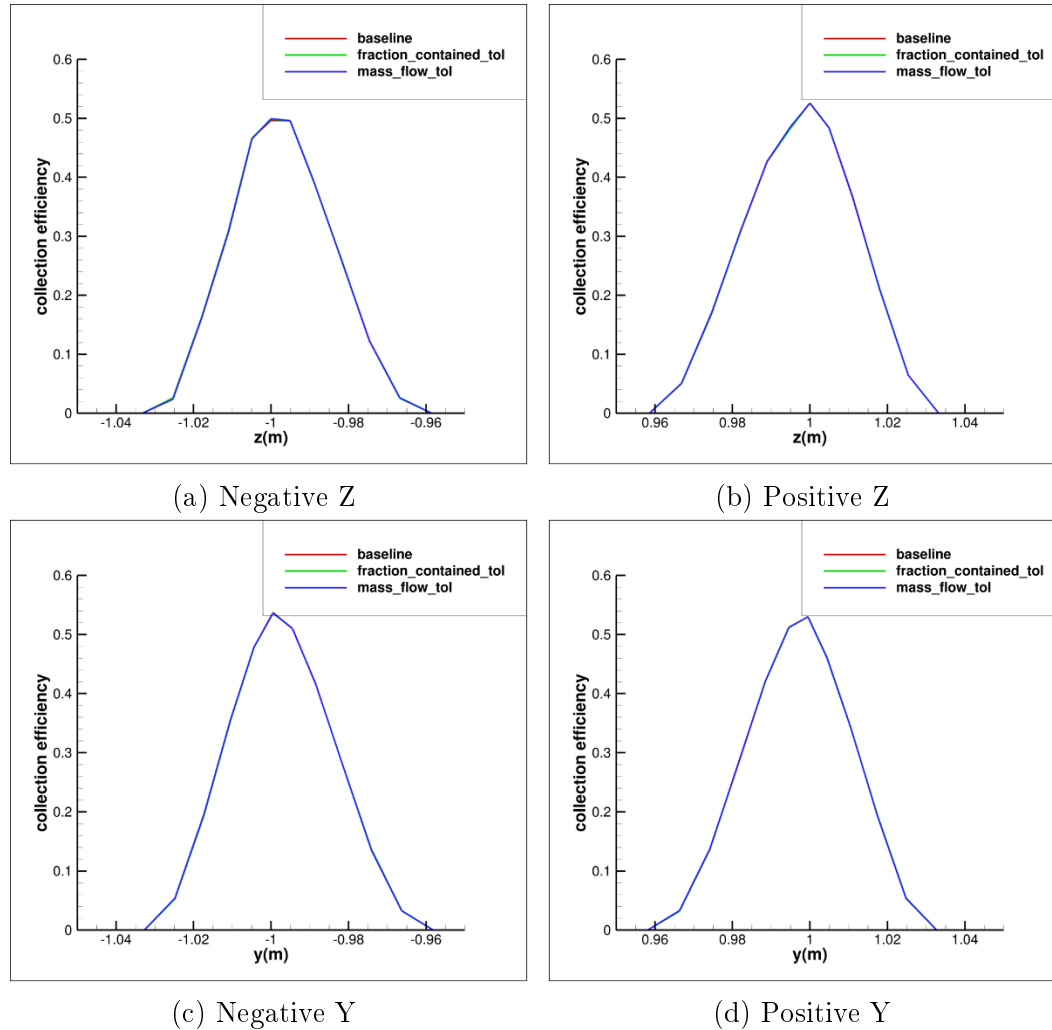


Figure 9.10: Collection Efficiency Variation Based on Efficiency Metrics.

9.3.3 Efficiency Metrics

GlennICE also contains several inputs in the `adaptive_refinement` namelist that can be employed to improve the efficiency (by reducing the number of trajectories run) without affecting the quality of the solution. The parameters that are explored in this example are `max_face_count`, `turn_off_bounded_faces`, `hit_percent_limit` and `use_nonuniform_tolerance`. `max_face_count` will stop refinement on a CFD face once it reaches the specified number

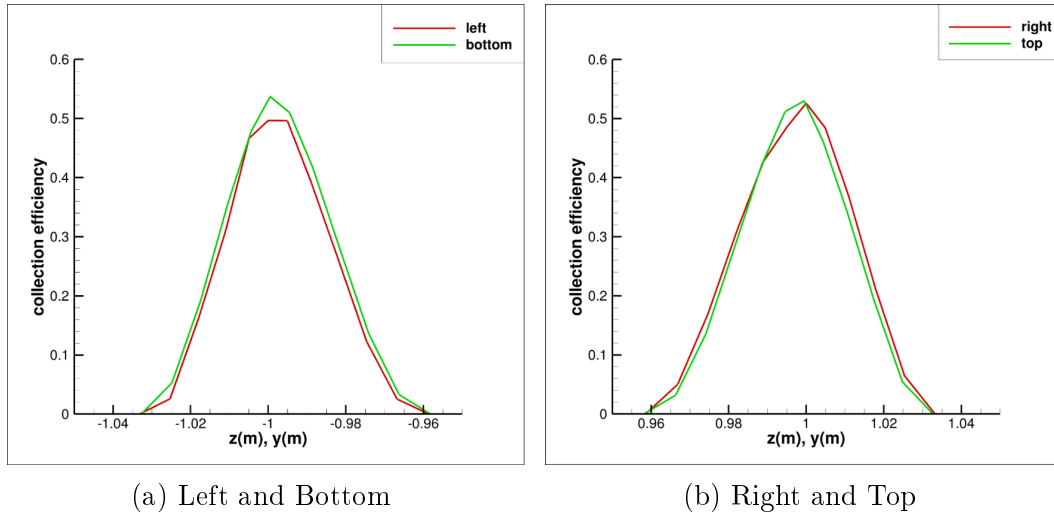


Figure 9.11: Symmetry of Collection Efficiency Result.

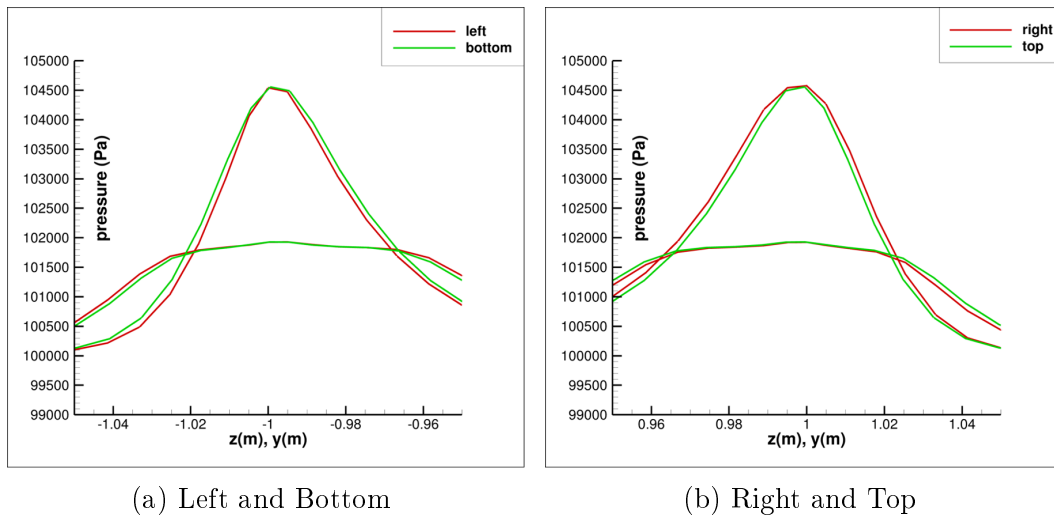


Figure 9.12: Symmetry of Surface Pressure.

of particles hit. `turn_off_bounded_faces` will stop refinement of seed point faces that lie entirely within a CFD face. This is different from `fraction_contained_tol` because in the case of `turn_off_bounded_faces` only those seed point faces will cease refinement while `fraction_contained_tol` will cease refinement within the CFD face that meets the `fraction_contained_tol` set. `hit_percent_limit` will cease the feature-finding algorithm after the specified percentage of hit trajectories has been met. The user should avoid setting this parameter too low or GlennICE could miss features of the geometry that contain impingement. This example uses a value of 25% hits. This means that after 25% of the computed trajectories hit, GlennICE will no longer refine missed trajectories. `use_nonuniform_tolerance` relaxes convergence near the impingement limit. If this parameter is set to false, GlennICE may run a large

number of trajectories trying to converge on a region that contains few hits. However, these metrics could lead GlennICE to falsely decide convergence has been met for complex cases. For example, surface geometries with long, thin faces will need more hits on the face to converge, so the default value for `max_face_count` of 600 may not be enough. The parameter `use_nonuniform_tolerance` may affect collection efficiency near the impingement limit. Table 9.1 shows the number of trajectories ran for each individual criterion as well as using all the metrics at their current default values. This exercise is not a conclusive result. The convergence tolerance chosen and the efficiency metrics chosen can greatly affect the total number of trajectories computed and the metrics needed to ensure a quality result could be dependent on the complexity of the geometry as well. Lowering `mass_flow_tol` or increasing `fraction_contained_tol` will cause more trajectories to run.

Table 9.1: Comparison of number of trajectories and total impinging mass flow for the annular test case.

Case	Trajectories	Hits	Total Mass Flow	l2 convergence	Pct convergence
baseline	9548213	8811741	4.595680873E-03	3.104993562E-05	00.00%
<code>fraction_contained_tol = 0.3</code>	1897968	1161496	4.595680274E-03	1.896289995E-05	99.88%
<code>fraction_contained_tol = 0.3 + max_face_count</code>	1675927	939455	4.595680274E-03	2.041321713E-05	96.18%
<code>fraction_contained_tol = 0.3 + hit_percent_limit=25</code>	1255704	1074876	4.595580510E-03	1.011205723E-05	99.88%
<code>fraction_contained_tol = 0.3 + turn_off_bounded_faces</code>	1744957	1008485	4.595680573E-03	1.823035737E-05	99.88%
<code>fraction_contained_tol = 0.3 + use_nonuniform_tolerance</code>	1762893	1026421	4.595680274E-03	2.104860552E-05	99.88%
<code>fraction_contained_tol = 0.3 + all_metrics</code>	870189	729021	4.595555181E-03	6.652031977E-05	97.11%
<code>mass_flow_tol = 0.01</code>	3913487	3177015	4.595680573E-03	8.605236973E-05	79.17%
<code>mass_flow_tol = 0.01 + max_face_count</code>	1705499	969027	4.595680573E-03	3.021969982E-05	78.36%
<code>mass_flow_tol = 0.01 + hit_percent_limit = 25</code>	3373639	3175851	4.596515331E-03	8.666827945E-05	80.32%
<code>mass_flow_tol = 0.01 + turn_off_bounded_faces</code>	2805733	2069261	4.595680573E-03	7.802494742E-05	79.98%
<code>mass_flow_tol = 0.01 + use_nonuniform_tolerance</code>	3916472	3180000	4.595680573E-03	8.751383558E-05	79.98%
<code>mass_flow_tol = 0.01 + all_metrics</code>	985240	854495	4.596323625E-03	1.896866355E-05	98.73%

9.4 Distribution

A single or a multi-bin distribution can be supplied. The suggested behavior is to provide the diameter array in the `&distribution` namelist directly in microns with an associated `mass_fraction` array specifying the fractional amount of total water content to provide to that respective bin.

9.4.1 Comparison of Multiple Drop Diameters

The impingement limit is dependent on multiple factors, such as geometry, drop diameter, and the velocity vector field. The primary benefit of the adaptive refinement algorithm is the limited user input required to identify the impingement limit for an arbitrary simulation. To illustrate this the drop diameter, one of the independent variables of the impingement limit, was chosen and varied. Identical `&adaptive_refinement` namelists were used, except for the drop size.

Drop diameters of $20\mu\text{m}$, $60\mu\text{m}$ and $180\mu\text{m}$ were utilized here. Figure 9.13 illustrates the impact of the drop diameters on the adaptive refinement results, as well as the collection efficiency. Note the ability of the adaptive refinement to find the impingement limits, and once found aggressively release trajectories that primarily impinge.

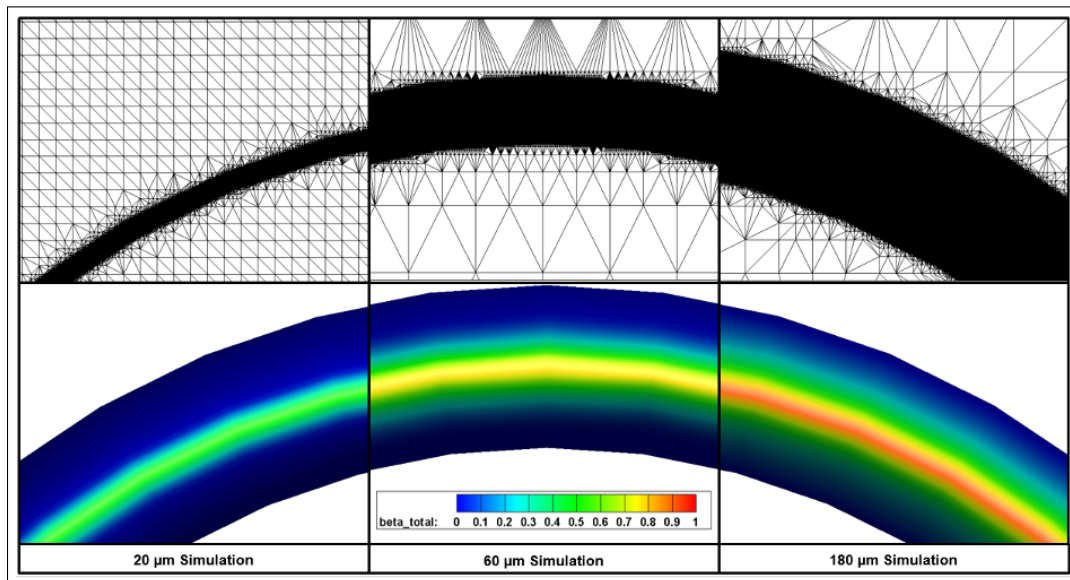


Figure 9.13: Refinement and collection efficiencies of iteration sixteen of the mass flux solver for three different discrete drop diameters with identical `&release` and `&adaptive_refinement` parameters.

9.5 Heat Transfer Augmentation

Many turbulence models do not account for surface roughness. This version currently allows the user to multiply the incoming heat transfer coefficient to account for this effect on ice shapes. The effect will be greatest for glaze ice conditions. The fixed transition method allows the user to increase both laminar and turbulent heat transfer by a multiplier. The McClain roughness method used the empirical Equation 27 of reference [9] to predict the roughness height. The augmentation is then a linear multiplier based on the turbulent heat transfer coefficient. While this can be tuned to specific ice shapes from experiment, it is not known if this value will give accurate predictions for a full range of conditions and geometries.

9.6 Ice Density

Ice density can be input in the `&freestream` namelist to mimic the voids found in scallop ice formation. Available data from the Icing Research Tunnel using swept wing models suggests that an ice density of 450 is needed to approximate

scallop ice shapes. If the user does not input a value for ice density or sets it to zero, the local ice density is calculated using the formula:

$$\rho_{ice} = 917 - (917 - 820) \cdot N_f \quad (9.1)$$

where N_f is the freezing fraction of a face.

9.7 Conversions

GlennICE assumes the flow solution input is in consistent units. The user must input a compatible flow solution for accurate analysis. Thus if the grid geometry is in *inches*, the software will be expecting velocities that are *in/s* not *ft/s* or *knots*. Additionally, the user can provide unit conversions from **&conversions**. This namelist assumes the flow solution variables are in a consistent set of units which are not metric and the values entered would be used for unit conversion.

Some sample conversion formulae using the **&conversions** namelist:

- Convert temperature from °F to *K*:
 - `temperature_offset = 459.67`
 - `temperature_scale_factor = 0.55555555`
- Convert dimensionless temperature to *K*:
 - `temperature_offset = 0`
 - `temperature_scale_factor = reference_temperature_used_in_flow_solver`
- Convert airspeed in *knots* to *m/s*:
 - `length_scale_factor = 0.51444444` (note: this will convert all lengths, including grid geometry as the software will assume all lengths are in *(knots · s)*!)

Additionally, the section **&variable_properties** is used to provide unit conversions on a single variable and can be used when the flow solution does not have consistent units. As an example, FUN3D provides the variable heating in W/cm² even when other variables are metric and can output normalized values for pressure and temperature. This methodology can be seen within **Section 8.2 - Single Bin Ice Accretion**.

9.8 Flow Solvers

At this time, the number of flow solvers that have a tested workflow is limited. Currently workflows exist for CFX solutions in Fieldview Unstructured

format as well as FUN3D solutions in Tecplot Subzone Load-on-Demand format. GlennICE can internally interpret variable spellings emanating from these workflows, if the user is attempting to utilize a workflow different than above, the naming convention of the variables as output from the CFD solver may not be immediately recognized by GlennICE. Should this be the case, the user can specify the proper spelling utilizing the `&variable_properties` namelist.

It should be noted that GlennICE will notify the user if it did not find variables required to perform impingement. However, due to the multiple workflows for computation of surface heat transfer coefficient, the software is currently not intelligent enough to determine if the data provided by the user is adequate and it may unexpectedly fail. See **Section 6.2 - Supported CFD File Formats** for additional information.

9.9 FUN3D Considerations and Workflow

This section is intended to provide some guidance for leveraging FUN3D solutions in GlennICE.

9.9.1 Tecplot Output

The precision of Tecplot solution output can be manually changed by modifying the value assigned to the variable `isdouble` in `nml_global` for FUN3D v14.0+. It is currently unknown if the usage of this has appreciable changes to a subsequent GlennICE solution.

9.9.2 FUN3D Workflow

This section describes the FUN3D workflow using the two wall temperature approach for the computation of heat transfer coefficient and adiabatic wall temperature. The user runs cases with two different wall temperatures. If the user is only interested in collection efficiency, only one simulation is required, with an adiabatic condition being specified at the wall.

It should be noted that for certain variables (e.g. `heating`) FUN3D makes the assumption that the grid is in meters. Since GlennICE leverages this variable, simulations utilizing FUN3D should use grids that are in units of meters.

9.9.2.1 FUN3D Simulation Requirements

FUN3D needs to be built against the TECIO library to enable the writing of Tecplot Subzone Load-on-Demand (`*.szplt`) files. It is required that the FUN3D simulation outputs volume and surface data in one zone. For instance if FUN3D is linked against the serial version of the TECIO library and run in parallel, FUN3D writes '`n`' number of solution files written where '`n`' is the

number of processors utilized. Each file contains a partition of the geometry. If this is the configuration of FUN3D in the user's environment, they are instructed to re-run FUN3D in restart mode with zero extra iterations with 'n' equal to one to facilitate the generation of a single volume and boundary *.szplt file.

A sample of the FUN3D volume and boundary output namelists are provided below for user reference:

```
&volume_output_variables
  export_to      = "tecplot"
  x              = .true.
  y              = .true.
  z              = .true.
  primitive_variables = .true.
/

&boundary_output_variables
  number_of_boundaries = -1
  boundary_list        = "1-x" ! Where x is the total number of
    boundaries
  x                    = .true.
  y                    = .true.
  z                    = .true.
  primitive_variables = .true.
  temperature          = .true.
  shear_x              = .true.
  shear_y              = .true.
  shear_z              = .true.
  heating              = .true.
/
```

FUN3D outputs certain variables as non-dimensional and certain values as dimensional. Below is a sample namelist input for use with GlennICE to enable the proper unit conversion. As mentioned previously it is required that the mesh provided to FUN3D is in units of meters and the namelists below makes that assumption as well. This suggestion is made since FUN3D will output heating in units of W/cm^2 but this is only true if the grid units are in meters. Additionally, pressure is often output as a dimensionless variable whereas shear stress is output in Pascals. Unit conversions for FUN3D are thus provided for each variable and the **&conversions** namelist is not recommended for this flow solver. Note that the spellings in the namelist below are not required, but are provided for clarity. Also note that values of $1.225 \text{ kg}/m^3$, $343 \text{ m}/s$, and 273 K are being used in the example below for reference density, speed of sound, and temperature.

```
&variable_properties
  map_index      = 4
```

```

    spelling      = "rho"
    scale_factor = 1.225
/

&variable_properties
    map_index     = 5
    spelling      = "u"
    scale_factor = 343.
/

&variable_properties
    map_index     = 6
    spelling      = "v"
    scale_factor = 343.
/

&variable_properties
    map_index     = 7
    spelling      = "w"
    scale_factor = 343.
/

&variable_properties
    map_index     = 8
    spelling      = "p"
    scale_factor = 144120.025 ! rho_ref * (a_ref)^2
/

&variable_properties
    map_index     = 9
    spelling      = "temperature"
    scale_factor = 273.
/

&variable_properties
    map_index     = 10
    spelling      = "heating"
    scale_factor = -10000. ! Converts from cm^2 to m^2 with the negative
                          ! accounting for an orientation change.
/

```

9.10 Seed Points and Release Points

The `&release` namelist defines the initial bounding box and number of seed points used for the first mass flux solver iteration. If a large bounding box is

input, some of the seed points will not project onto the Inlet surface as shown in Figure 9.14 and Figure 9.15. Trajectories will only be computed on the locations that project onto an Inlet surface.

NOTE: Version 4.2.0 has a requirement that all seed points project on the Inlet. This effectively requires that the box specified by the `&release` namelist must be equal to or bounded by the edges of the Inlet surface(s). Should this requirement not be met, the code will gracefully exit with the following message:

```
STOPPING: <XXX> seed points do not project onto the inlet.
```

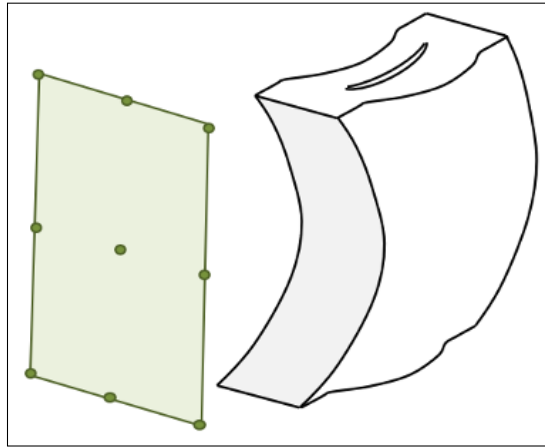


Figure 9.14: Rectangular seed point grid and sample geometry.

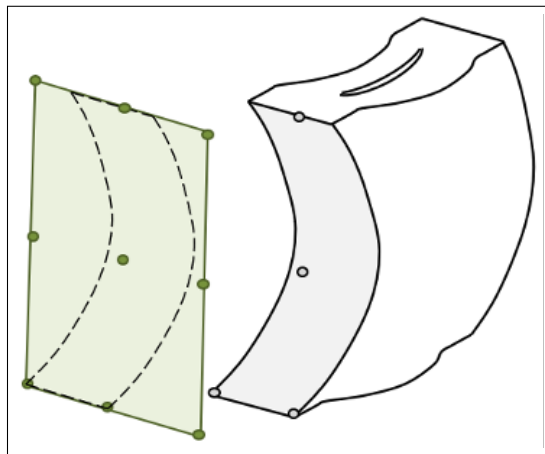


Figure 9.15: Projection of seed points onto the Inlet surface.

This error can be rectified by reducing the bounding box so that all trajectories project onto the inlet.

9.11 Uniformly Refining Release Points

GlennICE contains an ability to uniformly refine release points. This method of refinement is the most conservative, however it not very computationally efficient if the initial seed grid specified in the `&release` namelist contains a large proportion of area that results in missed trajectories. As such, the user is cautioned against using this option.

Additionally, it is not easy to realize how many total trajectories will be computed if the user selects the multiple iteration option. The following equation can be very useful:

$$\text{Number of Seed Points} = [2^{m-1}(n_y - 1) + 1] [2^{m-1}(n_z - 1) + 1] \quad (9.2)$$

In this equation m is the number of iterations specified in the `&mass_flux_solver` namelist, n_y is the initial number of seed points in the y-direction and n_z is the number of seed points in the z-direction as specified in the `&release` namelist.

9.12 Optimizing Initial Release Points

GlennICE will work best when the Initial seed point grid is evenly spaced, meaning that the point spacing in the y-direction is the same as the z-direction. This can be adjusted by altering the bounding box in the `&release` namelist but is usually adjusted by changing the initial values of `n_ytraj` and `n_ztraj`. For example, collection efficiencies on a long, thin wing will converge faster if the initial spacing is more uniform.

9.13 Restart

The user has the ability to use various restart capabilities to enhance performance. Surface restarts are used to perform parameter studies and can be performed quickly as the software will use the results already obtained. See the Restart section in the description of inputs. Also see **Section 8.3 - Refinement Restart** and **Section 8.5 - Surface Restart**. Volume restarts can be used to save computation time as the initialization of a simulation can take several minutes for a complex geometry and this feature will use the initialized volume from a previous case. Refinement restarts are also very useful as the simulation will stop after the specified number of refinement iterations are performed and the solution may not be converged. The user can then submit an additional case that will resume the simulation of trajectory refinement. At this juncture the user can also change convergence criteria or other parameters in the `adaptive_refinement` namelist to improve the collection efficiency results. Additionally, it should be noted that the command line input for the restart calls are case insensitive. This means that the usage of `restartjob=surface` will produce the same output as `restartJob=Surface`.

9.14 Write Trajectories

The user can now output trajectory paths to a file by setting the boolean `write_trajectories=.true.` in the `trajectory_solver` namelist. However, writing a large number of trajectories slows execution and creates a large ASCII file that will load slowly into Tecplot.

9.15 Fraction_contained_tol

Figure 9.16 shows a schematic of the definition of the variable, fraction contained. In the figure, the large triangle drawn in gray represents a CFD face on an Icing surface. The small circles each represent a trajectory, with the lines connecting them showing the connectivity from the release plane. The dashed red lines indicate the stream tube area for each of the four hit points. The green point in the center has all its neighbors hitting the same CFD Face. It is, therefore, assumed that the stream tube associated with the green point is completely contained within the CFD Face. The three blue points surrounding the green point, have some neighbors with hits on the same CFD Face, but also some that do not. Therefore, it is assumed that only part of the stream tubes from the blue points is contained in the CFD Face. With the above figure, and explanation, we can now define fraction contained. The variable, fraction contained, is defined as the ratio of contained mass flow to all mass flow on a CFD Face. Note that the schematic in Figure 9.16 is representative of the very beginnings of convergence. For a reasonably converged result, it is expected that there would be many contained stream tubes surrounded by partially contained stream tubes around the edge of the CFD Face.

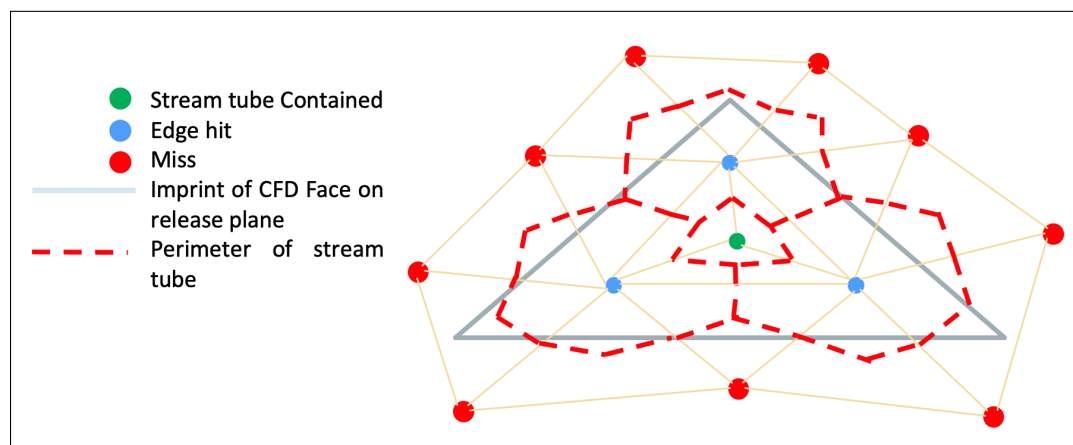


Figure 9.16: Schematic showing definition of Fraction Contained.

9.16 Parallel Computing

Chapter 4 - Command Line Input - MPI Builds explains how to build and execute GlennICE in parallel. After executing the simulation, and assuming `time_statistics` in `&trajectory_solver` namelist is set to `.true.`, an output for the trajectory routine job statistics is written to screen for each refinement level (for example):

```
Trajectory Routine Job Statistics
Trajectories run:           2571316
Number of Processors:       768
Time (sec):                 445.130100000000027
Time per trajectory:        0.000173113728534
Trajectories per second:    5776.549372868740647
Per Processor Statistics
Time per trajectory:        0.132951343514372
Trajectories per second:    7.521548662589506
Efficiency Statistics
Overall Efficiency:         0.999934122724271
Minimum Efficiency:         0.999821849836711
```

- `Trajectories run` refers to the number of subsequent trajectories computed on the current iteration
- `Number of Processors` refers to the total number of processors (`-np x`) used in the run
- `Time` refers to the total run time (in seconds) for the subsequent iteration
- `Time per trajectory` looks at the average run time (in seconds) it takes to release a trajectory:

$$\text{Time per trajectory} = \frac{\text{Time}}{\text{Trajectories run}}$$

- `Trajectories per second` refers to the number of trajectories released per second:

$$\begin{aligned}\text{Trajectories per second} &= \frac{\text{Trajectories run}}{\text{Time}} \\ &= \frac{1}{\text{Time per trajectory}}\end{aligned}$$

The statistics is also be provided on a per-processor basis:

- `Time per trajectory` looks at the average run time (in seconds) it takes to release a trajectory per processor:

$$\text{Time per trajectory (per processor)} = \frac{\text{Time}}{\left(\frac{\text{Trajectories run}}{\text{Number of Processors}}\right)}$$

- **Trajectories per second** refers to the number of trajectories released per second per processor:

$$\begin{aligned} \text{Trajectory per second (per processor)} &= \frac{\left(\frac{\text{Trajectories run}}{\text{Number of Processors}} \right)}{\text{Time}} \\ &= \frac{1}{\left(\begin{array}{c} \text{Time per trajectory} \\ \text{(per processor)} \end{array} \right)} \end{aligned}$$

Efficiency statistics are also provided by GlennICE

- **Overall Efficiency** refers to the average amount of time a processor (x) is conducting work:

$$\text{Overall Efficiency} = \frac{\sum_i^{(x-1)} (\text{Computation Time of a Processor})_i}{(x-1) \cdot \max (\text{Computation Time of a Processor})_i}$$

- **Minimum Efficiency** refers to the least efficient processor:

$$\text{Minimum Efficiency} = \frac{\min (\text{Computation Time of a Processor})_i}{\max (\text{Computation Time of a Processor})_i}$$

9.16.1 Static vs Dynamic Scheduling

The management of trajectory distribution among processors in parallel computing can be achieved through the implementation of either a static scheduling scheme or a dynamic scheduling scheme. Static scheduling is a method used in parallel computing where jobs are given to processors at the beginning of a program's execution. It determines task assignments and execution orders based on predetermined criteria, such as task priorities or a fixed schedule.

Dynamic scheduling, on the other hand, is a method in parallel computing where task assignments and execution orders are determined dynamically during runtime based on real-time factors such as workload distribution, resource availability, and task priorities. This adaptive method allows for efficient utilization of computational resources and is well-suited for jobs with uneven workload balance. Under this configuration, each trajectory is initially allocated an address within the shared memory of the initialized GlennICE data. Thus, all trajectories have simultaneous access to and a 'view' of the entire domain. Afterwards, a designated processor responsible for decision-making and workload allocation manages the dynamic scheduling process. This processor will have the following responsibilities:

- Communicating trajectories to each processor
 - Involves providing an integer array of where the calculation begins and how many trajectories are to be computed.

- Tracking the status of completed trajectories.
- Monitoring processors engaged in active computations.
- Managing processors that have finished a task on a trajectory and require additional work.

The manager processor however, will not engage in the calculation of any trajectories. While it may appear counter-intuitive to sacrifice one processor's capacity for efficiency gains, as the number of processors increases, the impact of losing one processor diminishes and the solver's efficiency approaches 100%. The current workflow for dynamic scheduling is structured as follows:

1. The manager processor initiates the distribution of an initial subset of trajectories $[x]$ among processors. The size of this subset depends on the namelist parameter `chunk_percent` from **Section 6.1.4 - &trajectory_solver Namelist**:

- (a) For serial runs:

$$[x] = \max[(\text{chunk_percent} \times \text{Total Number of Trajectories}), 1]$$

- (b) For parallel runs:

$$[x] = \max\left[\left(\text{chunk_percent} \times \frac{\text{Total Number of Trajectories}}{\text{Number of Processors}-1}\right), 1\right]$$

2. Each processor independently calculates its allocated subset of trajectories.
3. Whenever **any** processor completes its work on the assigned subset of trajectories, it requests an additional subset from the manager processor.
4. Steps (1), (2), and (3) are repeated iteratively until all specified **Trajectories run** have been calculated.

Figure 9.17 illustrates efficiency comparisons between static scheduling (v3.2.0) and dynamic scheduling (`chunk_percent` = 0.0) vs. number of processors. As depicted in the figure, dynamic scheduling initially exhibits a sharp efficiency drop due to the involvement of one manager processor. Nevertheless, as the number of processors increases, the impact of losing one processor diminishes, and the solver's efficiency approaches 100%. In contrast, static scheduling starts with high efficiencies initially, thanks to moderately even load balancing among processors. However, as the number of processors grows, workload imbalances emerge, leading to an inevitable drop in efficiency. Although not recommended, modifying the namelist parameter `chunk_percent` = 100 would replicate a similar behavior to that observed with static scheduling.

It is crucial to acknowledge that the efficiency outcomes of static scheduling depicted in Fig. 9.17 are case specific, given the variability in load balance

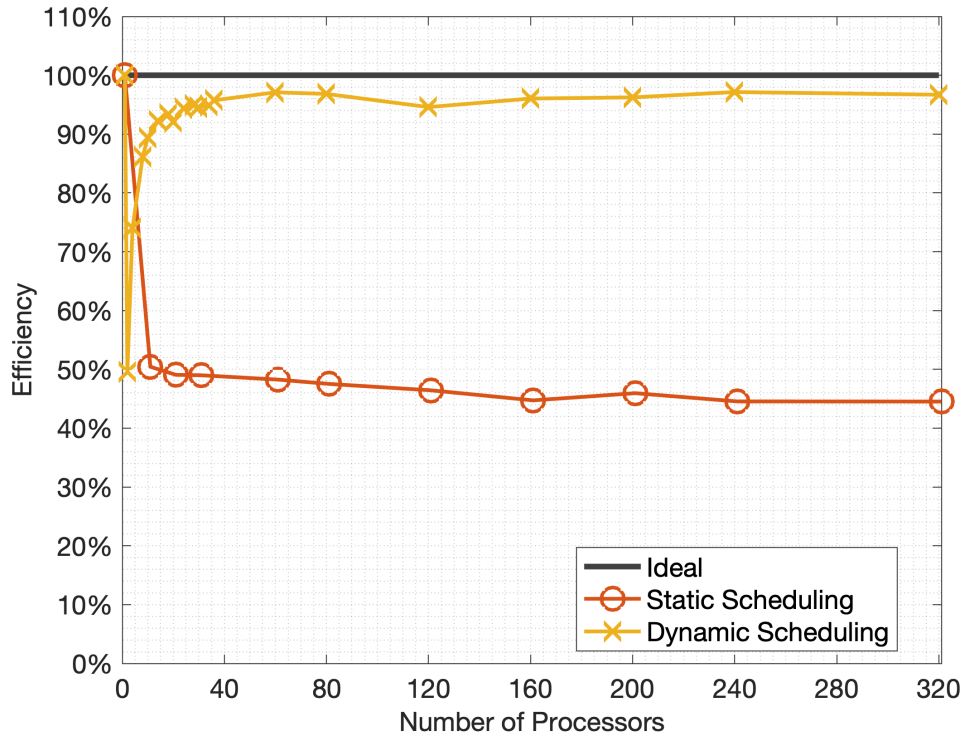


Figure 9.17: Parallel Scalability (Efficiency).

across cases [25]. Static scheduling efficiencies for other geometries have been documented to span from 5% to 90%. The inconsistency observed across test cases for static scheduling, coupled with the uncertainty regarding the efficiency metric beforehand highlights dynamic scheduling (`chunk_percent` = 0) as a more resilient option across various geometries, yielding consistent efficiency results.

In the context of serial runs, the manager processor coincides with the processor responsible for both distribution and conduction of work. Thus the value of `chunk_percent` will **not** have an effect on the efficiency of the solver.

9.17 Non-inertial Reference Frame

Since the runback model only uses shear to set direction, only Rime cases should be run (i.e. very cold so no runback). If specified rotation rate is non-zero the freestream conditions will be calculated only using the x-direction value of velocity.

Supplied solution can be in absolute or relative frame. You just need to indicate which in the `&rotation` namelist, using the logical variable `in_absolute_frame`. The rotation rate, in radians per second, is specified in the `rotation` namelist with the variable `omega`. Output solution and trajectories

will be in the relative frame.

9.18 Known Issues

9.18.1 Known Issue 1

```
WARNING: XXX trajectories ended without an observed intersection
        with
a surface bounding the volume.
```

There are two known causes for this issue. One cause is the value of `tstop` in the `&trajectory_solver` namelist is not large enough to track the entire trajectory through the volume. This error can be mitigated by increasing the value of `tstop`.

9.18.2 Known Issue 2

The other issue is related to GlennICE "losing" the trajectory. GlennICE uses an algorithm that jumps from tetrahedron to neighboring tetrahedron to keep track of a trajectory's current location in the mesh. In very rare circumstances, the trajectory seems to stop in mid-air regardless of the value used for `tstop`. The cause of this case is still being investigated.

```
Bad triangulation. Making another attempt.
```

This warning occurs when the Delaunay triangulation scheme fails. This is a known issue for these algorithms when there are large numbers of collinear points. The routine can make two checks on the triangulation. The first check looks for triangles with zero area (collinear points). The second check confirms that the sum of the seed point triangles equals the area of the bounding box. The software will attempt to increase an internal offset to get a valid triangulation. The maximum offset is 0.1. The user is encouraged to report a case that reaches the maximum offset to GlennICE-support@lists.nasa.gov.

9.18.3 Known Issue 3

```
Temperature below lower bound in energy balance
0.229998167369471E+03
Temperature lower bound =      0.2300000000000000E+03
```

This warning can occur when a low temperature is supplied to GlennICE either from the flow solution or from the `Icing_temperature` input. The correlation for evaporative mass loss is not valid below a temperature of 230 K. GlennICE will use a temperature of 230 K to calculate the evaporation term. Since the term is small at this temperature, it may not be a concern.

9.18.4 Known Issue 4

```
Warning. No real, positive root found
Number of negative roots 210
Warning. Quadratic root less than cubic root.
Number of bad roots 1
All icing surface faces that contain the icing surface node
    index below could not find an extrusion.
This means there will be no ice in the STL output by GlennICE at
    this node
    node index = 7319
node location =      0.168039975585937E+02      0.557326291503906E
    +01      0.671746594238281E+01
```

These warnings occur in locations where the extrusion process used to create the new ice shape failed. The first two messages occur when a face extrusion could not be obtained for a given face and the third message occurs when all faces connected to a node failed to find a proper extrusion. As such, no ice will be shown for these nodes even though the mass balance predicted ice. This usually occurs in corners such as where the pylon meets the engine inlet. Contact GlennICE-support@lists.nasa.gov if this is a concern. The GlennICE solution file will show these locations using the variable `bad_extrusions`.

9.18.5 Known Issue 5

When using the `&conversions` variable, the shear stress values that are within the `&tecplot_output` file become incorrectly scaled if they are not of consistent units as Pressure. To address this, include the scaling of shear stress within the `&variable_properties` namelist. If you are not using the `&conversions` variables, this will not be an issue. Additionally, the current methodology does not care about the dimensional value of shear stress, so this error does not change the resultant ice shape or accretion properties; only the output variable value.

9.18.6 Known Issue 6

Currently, there are discrepancies being seen between Apple based ARM systems when being compared to x86 systems. The scope and implication of these issues are still being characterized and understood. For the time being, it is recommended that users use x86 based systems until we fully resolve and understand the discrepancies.

Bibliography

- [1] Wright, W. B. "User's Manual for LEWICE Version 3.2", [NASA CR-2008-214255](#), Aug. 2008
- [2] Bidwell, C. S., and Potapczuk, M. G., "User's Manual for the NASA Lewis Three-Dimensional Ice Accretion Code (LEWICE 3D)", [NASA TM-105974](#), Dec., 1993.
- [3] "Technology readiness level" Wikipedia, Wikimedia Foundation, 29 September 2020, https://en.wikipedia.org/wiki/Technology_readiness_level
- [4] Bernard L. Messinger. "Equilibrium temperature of an unheated icing surface as a function of air speed." Journal of the aeronautical sciences . 20.1 (1953): 29-42.
- [5] Wright, W. B., "Validation Results for LEWICE 3.0", [AIAA-2005-1243/NASA-20050160961](#). Mar., 2005.
- [6] LEWICE, NASA Glenn Research Center, Cleveland, OH; software available at <https://software.nasa.gov/software/LEW-18573-1>
- [7] Bidwell, C. S., "Icing Analysis of a Swept NACA0012 Wing Using LEWICE3D Version 3.48", [AIAA-2014-2200/NASA-20150000867](#). June, 2015.
- [8] LEWICE3D, NASA Glenn Research Center, Cleveland, OH; software available at <https://software.nasa.gov/software/LEW-19433-1>
- [9] McClain, S. T., Vargas, M., Tsao, J., and Broeren, A. P., "A Model for Ice Accretion Roughness Evolution and Spatial Variations", [AIAA-2021-2641/NASA-20210017767](#). June, 2021.
- [10] Fieldview, Intelligent Light, Rutherford, NJ; software available at www.ilight.com/en
- [11] Tecplot 360, Tecplot Inc., Bellevue, WA; software available at <https://www.tecplot.com/>
- [12] Dompierre, Julien, et al. "How to Subdivide Pyramids, Prisms, and Hexahedra into Tetrahedra." IMR 99 (1999): 195.
- [13] "STL (File Format)" Wikipedia, Wikimedia Foundation, 13 June 2020,[https://en.wikipedia.org/wiki/STL_\(file_format\)](https://en.wikipedia.org/wiki/STL_(file_format))
- [14] Pointwise, Pointwise Inc., Fort Worth, TX; software available at <https://www.pointwise.com>
- [15] Ansys-CFX, Ansys Inc., Canonsburg, PA; software available at <https://www.ansys.com>
- [16] Menter, F. "Zonal Two Equation $k-\omega$ Turbulence Models for Aerody-

- namic Flows." [24th AIAA Fluid Dynamics Conference](#), Orlando, FL (1993).
- [17] Delaunator-cpp, Vladi Bilonenko, GitHub repository, 30 October 2020, <https://github.com/delfrrr/delaunator-cpp>
 - [18] "Delaunay Triangulation" Wikipedia, Wikimedia Foundation, 2 February 2021, https://en.wikipedia.org/wiki/Delaunay_triangulation
 - [19] Proceeds of the 1st AIAA Ice Prediction Workshop, <https://icepredictionworkshop.wordpress.com>, 2021.
 - [20] William B. Wright, Thomas A. Ozoroski, David Rigby. "Roughness Parameter Optimization of the McClain Model in GlennICE," SAE 2023-01-1468. SAE International Conference on Icing of Aircraft, Engines, and Structures. [2023-01-1468/NASA-20230003729](#). June, 2023.
 - [21] Heldenmesh v4.14, Helden Aerospace, Acworth, GA; software available at <https://heldenaero.com/>
 - [22] Allmaras, S. R. and Johnson, F. T. and Spalart, P. R., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," International Journal of Heat and Fluid Flow, Big Island, Hawaii, pp. ICCFD7-1902, 9-13 July 2012.
 - [23] FUN3D v14.0.1, NASA Langley Research Center, Hampton, VA; software available at <https://software.nasa.gov/software/LAR-20188-1>
 - [24] Langmuir, I. and Blodgett, K., "Mathematical Investigation of Water Droplet Trajectories," AAFTR 5418, 1946.
 - [25] Sabri, Z. and Porter, C., "Scalability of GlennICE in a Parallel Environment," SAE 2023-01-1482. SAE International Conference on Icing of Aircraft, Engines, and Structures. [2023-01-1482/NASA-20230004013](#). June, 2023.

GlennICE Publications

This section is intended to capture publications that describe and/or illustrate the capabilities of the GlennICE software. Some of these publications may be also listed in the references section but are duplicated here for the reader's benefit.

2020

Porter, Christopher E., and David L. Rigby. "Three Dimensional Surface Redefinition Method for Computational Ice Accretion Solvers." *AIAA Aviation 2020*. [AIAA2020-2831/NASA-20205001501](#), June, 2020.

2021

Wright, William B., Porter, Christopher E., Galloway, Eric T., and David L. Rigby. "An Automated Refinement Process for Particle Trajectory Methods in GlennICE." *AIAA Aviation 2021*. [AIAA2021-2631/NASA-20210017510](#), July, 2021.

Wright, William B., Porter, Christopher E., Galloway, Eric T., and David L. Rigby. "GlennICE 1.1 Results." *AIAA Ice Prediction Workshop 2021*. [Proceedings of the 1st AIAA Ice Prediction Workshop](#), July, 2021.

2022

Wright, William B., Porter, Christopher E., Galloway, Eric T., and David L. Rigby. "GlennICE 2.1 Capabilities and Results." *AIAA Aviation 2022*. [AIAA2022-3309/NASA-20220006017](#), June, 2022.

Rigby, David L. and Wright, William B., "Convergence Criteria for Lagrangian Collection Efficiency Simulations." *AIAA Aviation 2022*, [AIAA 2022-3693/NASA-20220006326](#), June, 2022.

Porter, Christopher E. "A Comparison of Trajectory Refinement Schemes for GlennICE." *AIAA Aviation 2022*. [AIAA 2022-3692/NASA-20220006461](#), June, 2022.

Chen, Ru-Ching and Porter, Christopher E., "Simulation of Fluid Flow and Collection Efficiency for a SEA Inc. Multi-Element Probe and Ice Crystal Detector Using GlennICE." *AIAA Aviation 2022*. [AIAA 2022-3694/NASA-20220006656](#), June, 2022.

2023

Stewart, E. and Barktus, T., "Computational Icing Analysis on NASA's SIDRM Geometry to Investigate Collection Efficiency ." *SAE International Conference on Icing of Aircraft, Engines, and Structures*. [SAE 2023-01-1476/NASA-20230003530](#). June, 2023.

Wright, William B. and Ozoroski, Thomas A. and Rigby, David L., "Roughness Parameter Optimization of the McClain Model in GlennICE," *SAE International Conference on Icing of Aircraft, Engines, and Structures*. [SAE 2023-01-1468/NASA-20230003729](#). June, 2023.

Rigby, David L. and von Hardenberg, Paul H., "Demonstration of Initial GlennICE Relative Frame Capability: Axial-Flow Propeller," *SAE International Conference on Icing of Aircraft, Engines, and Structures*. [SAE 2023-01-1457/NASA-20230003777](#). June, 2023.

Sabri, Zaid and Porter, Christopher E., "Scalability of GlennICE in a Parallel Environment," *SAE International Conference on Icing of Aircraft, Engines, and Structures*. [SAE 2023-01-1482/NASA-20230004013](#). June, 2023.

