NASA/CR—2001-210965

ICEG2D (v2.0)—An Integrated Software Package for Automated Prediction of Flow Fields for Single-Element Airfoils With Ice Accretion

David S. Thompson and Bharat K. Soni
Mississippi State University, Mississippi State, Mississippi

Prepared under Grant NAG3–2235

National Aeronautics and
Space Administration

Glenn Research Center

June 2001

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

Contents

List of Figures

List of Tables

1. Introduction

Prediction of ice accretion on a full aircraft configuration and the resulting performance degradation is a long-term objective of icing research. To this end, a three-year research and development program has been initiated to address issues related to computational fluid dynamics simulations for single- and multi-element airfoils with ice accretions. The first-year objective of the effort was to develop and evaluate a software package to perform the preprocessing necessary to generate NPARC [1] solutions automatically for single-element airfoil geometries with ice accretion. The efforts of the first year were reported in [2]. During the second year, the capability to generate solution adaptive grids was implemented. Additionally, the capabilities to generate semistructured, generalized grids [3, 4] and compute flow solutions on these grids utilizing the flow solver HYBFL2D [5] were incorporated into the software. Below is a brief summary of the activities accomplished to date:

- The software framework was developed along with the driver program, ICEG2D (v2.0), which serves as the interface to the framework and the various component modules. The interface was written in Perl and uses a vocabulary of directives to associate external data with the execution of appropriate program modules. ICEG2D (v2.0) can be executed in batch mode using the directives as a scripting language or in an interactive mode from a command line prompt.
- The geometry module, ICE_DIST_2D, was developed to generate a discrete surface definition for a given iced airfoil configuration. ICE_DIST_2D was written in C with a FORTRAN90 namelist routine and utilizes NURBS routines from GGLib - a geometry/grid library developed at MSU [6]. Automatic distribution of points on the airfoil surface is accomplished using an equidistribution algorithm in the iced region (with the weight function based on surface curvature) and algebraic stretching for the remainder of the airfoil.
- The grid module, ICE_GRID_2D, was developed to generate a nearly orthogonal grid from a previously defined surface distribution. ICE_GRID_2D was written in FORTRAN90 with a C driver routine and uses a parabolic grid generation scheme [7–9]. For structured grids, both single- and double-block grid topologies can be generated automatically. The semistructured grids feature point insertion and deletion based on cell geometry [3, 4]. The capability to generate solution adaptive grids based on velocity gradients was also implemented.
- The modules to generate an NPARC namelist input file and an NPARC initial guess file, NPARC_INPUT and NPARC_INITIAL respectively, were developed. A module to interpolate an NPARC solution to a new grid, NPARC_RESTART, was also developed for solution-adaptive grid applications.
- The module to generate the HYBFL2D input file, HYBFL2D_INPUT, was developed. Because of problems associated with HYBFL2D, a three-dimensional version of the code is actually used here. A preprocessor converts the two-dimensional generalized grid into a three-dimensional grid by extrusion. The process occurs in a seamless manner that is transparent to the user. Efforts are currently under way to rectify the problems with the two-dimensional code. These difficulties slowed the evaluation of the generalized grid techniques. Therefore, the results reported here should be considered to be preliminary, at best.
- A preliminary evaluation of the ICEG2D (v2.0) software package has been performed. Included in the evaluation is an assessment of grid quality by comparison of NPARC and HYBFL2D solutions computed using the automatically generated grids with experimental data (NASA GRC data for NLF0414 [10]).

The resulting software package has proven to be efficient and reliable for automated geometry processing and grid generation for single-element iced airfoils with ice accretion. To date, the emphasis has been on developing automated capability assuming the airfoil geometry and ice shape are known.

In the sections that follow, the technology employed in ICEG2D (v2.0), the installation procedure, and a description of the operation of the software package are discussed. Procedures used to evaluate the software package are also discussed. It should be noted that since this report also serves as a de facto users' manual, much of the material appearing in the first year report [2] is repeated here for convenience.

2. The ICEG2D (v2.0) Software Package

The ICEG2D (v2.0) software package was developed to automatically perform the preprocessing tasks necessary to generate computational fluid dynamic simulations of flow fields around single-element airfoils with ice accretion using the structured grid flow solver NPARC [1] or the generalized grid flow solver HYBFL2D [5]. ICEG2D (v2.0) includes a framework in which various operations are performed to generate the CFD solutions. The user interface to the ICEG2D (v2.0) framework is a Perl program that utilizes a system of file associations and directives to perform the basic functions shown in Table 1. A listing of all ICEG2D (v2.0) directives and a brief description of each directive

Table 1: Modules in ICEG2D (v2.0)

Function	**Module**	**Details**
Generate discrete definition of airfoil surface	ICE_DIST_2D	The surface definition module takes the iced airfoil definition, the clean airfoil definition, and default or user-specified parameters to generate the grid ready, point distribution defining the iced airfoil.
Generate weight function for solution adaptive grid	WEIGHT_GEN	The weight generation module uses the solution obtained from a flow solver and computes a scalar weight function to be used to generate a solution adaptive grid.
Generate grid	ICE_GRID_2D	The grid generation module takes the iced airfoil surface definition and default or user-specified parameters and blocking specification to generate a nearly orthogonal grid.
Display surface distribution or grid	ICE_PLOT_2D	This module displays the surface distribution and the grid.
Generate NPARC input file	NPARC_INPUT	This module generates an NPARC namelist input file using the grid and a user-defined namelist file.
Generate NPARC initial file	NPARC_INITIAL	This module generates an NPARC initial condition file using the grid and a user-defined namelist file.
Generate NPARC restart file	NPARC_RESTART	This module generates an NPARC restart file by interpolating the solution onto a new solution-adaptive grid.
Generate HYBFL2D input file	HYBFL2D_INPUT	This module generates an HYBFL2D input file using the generated grid and a user-defined namelist file.
Generate HYBFL2D initial file	HYBFL2D_PRE	This module generates a HYBFL2D initial grid file using the grid.
Generate CFD solution		A batch process is started to generate an NPARC or HYBFL2D solution using the input and restart files.

are included in Section 2.3.

Based on discussions with NASA GRC personnel and within the context of automation, it was decided to minimize the user input needed to run ICEG2D (v2.0). Third year plans include coupling LEWICE [11] with ICEG2D (v2.0) for ice accretion predictions. ICEG2D (v2.0) may be executed in batch or interactive mode. In batch mode, a text file containing a stream of ICEG2D (v2.0) directives is used as input via redirection. In the interactive mode, the user executes ICEG2D (v2.0) and issues directives from a command line within a Unix shell. Further enhancing the automation of ICEG2D (v2.0) are the FORTRAN namelists that define the various parameters used to generate the surface distribution and the grid. The namelist variables have been assigned default values that work for a wide variety of iced airfoil applications.

A typical ICEG2D (v2.0) session consists of taking an iced airfoil definition, available from LEWICE [11] or IRT data [12], and a clean airfoil definition as input to the module ICE_DIST_2D to generate a discrete surface distribution. The surface definition is then used as initial data for the grid generation module, ICE_GRID_2D, to generate a grid. The grid is used along with a problem definition namelist file as input to appropriate modules to generate an NPARC restart file and an NPARC input file. Finally, an NPARC solution is started as a background process. A similar process is used to generate a HYBFL2D solution. It should be noted that the process could be initiated at any point provided the necessary data files are available and have been defined appropriately within the ICEG2D (v2.0) framework.

The ICEG2D (v2.0) software package was developed to run on SGI platforms. The software has not been installed or tested on any other platform.

2.1 Installation

Installation of ICEG2D (v2.0) can be accomplished by following these steps:

1. Change to the directory where ICEG2D (v2.0) will be installed.
2. Unzip the gzipped file using "gunzip ICEG2D.tar.gz"
3. Untar the file using "tar -xvf ICEG2D.tar"
4. Execute "Install_ICEG2D"

The executable "Install_ICEG2D" compiles and links the various modules in their respective directories. Note: Warnings will occur as the NURBS library is being compiled. This is normal and does not affect execution of the code.

During installation, a dozen subdirectories are created in the main ICEG2D (v2.0) directory ($ICEG2D_DIR). The directories and their contents are given below:

1. $ICEG2D_DIR/common - FORTRAN90 routines common to several modules
2. $ICEG2D_DIR/examples - four subdirectories containing example cases
3. $ICEG2D_DIR/geometry - ICE_DIST_2D module
4. $ICEG2D_DIR/grid_gen - ICE_GRID_2D modules
5. $ICEG2D_DIR/help_files - help file
6. $ICEG2D_DIR/hybfl2d_gen - one subdirectory for the HYBFL2D_INPUT module
7. $ICEG2D_DIR/nparc_gen - three subdirectories for NPARC_INPUT, NPARC_INITIAL, and NPARC_RESTART modules
8. $ICEG2D_DIR/nurbs_lib - GGLib
9. $ICEG2D_DIR/parameter_files - FORTRAN namelist files
10. $ICEG2D_DIR/plot_2d - ICE_PLOT_2D module
11. $ICEG2D_DIR/report - contains this document as a pdf file
12. $ICEG2D_DIR/weight_gen - WEIGHT_GEN module

These directory names must be left unaltered because of internal file associations made within the ICEG2D (v2.0) framework.

To execute ICEG2D (v2.0), the following four lines must be included in your .cshrc file:

```
setenv $ICEG2D_DIR (path to directory where ICEG2D is located)
setenv $NPARC_DIR (path to directory where nparc2ds is located)
setenv $HYBFL2D_DIR (path to directory where HYBFL2D is located)
set path = ($path $ICEG2D_DIR)
setenv TRAP_FPE "ALL=COUNT;UNDERFL=ZERO;
       OVERFL=IEEE,TRACE(5),ABORT(100); DIVZERO=ABORT;
       INVALID=TRACE(1),ABORT(1)"
```

The first three lines define environment variables that specify the path to the main ICEG2D (v2.0) directory, the path to the NPARC executable, and the path to the HYBFL2D executable. The fourth line includes the ICEG2D (v2.0) directory in the shell path variable. The fifth line defines how floating point exceptions are treated and is important for error trapping in the fully automatic mode.

2.2 The ICEG2D (v2.0) Framework

The ICEG2D (v2.0) framework is based on a system of internal associations with default or user-prescribed file names. The primary association is the *case_name*. By defining the association *case_name*, a set of automatically defined file names is utilized by the framework for module input and output. These file names are defined using *case_name* and a default extension based on the file type. Below is a table containing the default file extensions and their associated file types used in the ICEG2D (v2.0) framework.

Table 2: Default file extensions

Extension	Data	Format
.cln	Clean airfoil input	ASCII file, number of points followed by (x,y) data pairs
.dat	Iced airfoil input	ASCII file, number of points followed by (x,y) data pairs
.dst	Airfoil point distribution	FAST formatted, multizone, 2d grid data
.grd	Grid file	ASCII file, topology dependent Structured grid - FAST formatted, multizone, 2d grid data Semistructured grid - hybrid grid format
.inf	Problem information file	ASCII file, solver dependent
.ini	Initial condition file	Solver dependent
.inp	Input file	ASCII file, solver dependent
.rst	Restart file	Solver dependent restart files
.out	Solver output	Output from flow solver
.wt	Weight function file	FAST formatted, multizone, 2d function data

Additionally, there are seven FORTRAN namelists used to define parameters used by ICEG2D (v2.0) modules. The default namelist files are located in the directory $ICEG2D_DIR/parameter_files:

1. The file containing data for the FORTRAN namelist **DISTRIBUTION** (associated with *dist_parm_file*) is either **dist_smooth.ini** or **dist_rough.ini**, based on the *surface_type* association, or a user-defined file. These parameters control the distribution of points on the iced airfoil surface.

2. The file containing the namelists **NUMBER_OF_BLOCKS** and **BLOCK_PARAMETERS** (associated with *block_parm_file*) is either **single_block.ini**, **double_block.ini**, or a user defined file. These parameters define the blocking scheme.

3. The file for namelist **GRID_PARAMETERS** (associated with *grid_parm_file*) is **grid_gen.ini** or a user defined file. These parameters control the grid generation process.

4. The namelists used in the NPARC problem definition— **TITLES**, **INPUTS**, **TURBIN**, **SEQDT**, and **BLOCK**— are contained in the problem definition file **NPARC_Info.inf** (associated with *problem_info_file*) or a user defined file. Typically, the user will copy this file to the local directory and make modifications appropriate for the problem.

5. The namelist used in the HYBFL2D problem definition—**INPUTS**—is contained in the problem definition file **HYBFL2D_Info.inf** (associated with *problem_info_file*) or a user defined file. Typically, the user will copy this file to the local directory and make modifications appropriate for the problem.

Again, these associations may be changed using appropriate ICEG2D (v2.0) directives. Minimally, the user must provide the iced and clean airfoil input files and the problem definition file. **Default values for all parameters are defined within ICEG2D (v2.0)**. These default values have proven effective for the cases examined to date. In rare cases, the user may desire to modify one or more of the namelist variables. The suggested procedure is to copy the needed files into the directory containing the airfoil definition files and make the necessary modifications. The modified file can then be associated with the internal representation using the appropriate ICEG2D (v2.0) directive.

2.3 The ICEG2D (v2.0) Interface

The ICEG2D (v2.0) framework provides a series of associations between internal data and external files. Some associations are created automatically while others are defined using the ICEG2D directives. The ICEG2D (v2.0) directives all have the same form:

[directive - action] [object - what the directive produces/acts on] [additional info (if needed)]

Each directive **must** start in the first column, i.e., there can be no leading blanks. Either upper or lower case letters may be used for the directives and the objects. File names, however, are case sensitive. In the descriptions below, the directives and objects appear in bold while the data entries appear in italics. Also, characters appearing in parentheses are optional.

1. # - Indicates that a comment follows. A # is needed for each line of the comment
2. **def(ine)** - defines something. Note: The **def(ine)** directive does not actually do anything other than define associations internal to ICEG2D (v2.0).

    - **adapt** *adapt_type* [type of solution adaptive grid - *none*, *refine*, *redist*, or *both*, default is *none*]
    - **block_parm** *grid_block_file* [file containing blocking parameters, *single* for the default single-block definition using **single_block.ini**, *double* for the default double-block definition **double_block.ini**, or *default* to use the default single-block definition (used to reset block definition)
    - **case** *case_name* [identifier used for default associations]
    - **clean_geom** *clean_geom_file* [clean geometry file name, default is *case_name.cln*]
    - **dist** *dist_file* [distribution file name, default is *case_name.dst*]
    - **dist_parm** *dist_parm_file* [file containing distribution parameters or *default* to use the default distribution parameter file for the specified *surface_type*]
    - **grid** *grid_file* [grid file name, default is *case_name.grd*]
    - **grid_parm** *grid_parm_file* [file containing grid parameters or *default* to use default grid parameter file *grid_gen.ini*]
    - **iced_geom** *iced_geom_file* [iced geometry file name, default is *case_name.dat*]
    - **info** *info_file* [NPARC information file used to define flight conditions, etc., default is **NPARC_Info.inf**]
    - **initial** *initial_file* [solver initial file name, default is *case_name.ini*]
    - **input** *input_file* [solver input file name, default is *case_name.inp*]
    - **input_grid** *input_grid_file* [input grid file name for grid adaptation, default is **fort.31**]
    - **input_q** *input_q_file* [input q file name for grid adaptation, default is **fort.30**]
    - **mode** *mode_type* [*int(eractive)* or *bat(ch)* with interactive being the default. Note: The program will exit if an error is encountered in batch mode]
    - **output** *output_file* [solver output file name, default is *case_name.out*]
    - **restart** *restart_file* [restart file name, default is *case_name.rst*]
    - **reflag** *reflag* [*yes* or *no* to specify restart for HYBFL2D, default is *no*]
    - **solver** *solver_type* [*str(uctured)* or *semi(structured)* to use either the default structured or semistructured solver.]
    - **surface** *surface_type* [*smooth* or *rough* with *smooth* being the default, sets the default distribution parameter file to **dist_smooth.ini** or **dist_rough.ini** respectively.]
    - **topo** *grid_topology* [*str(uctured)* or *semi(structured)* with the default being *str(uctured)*, also sets *solver_type* definition]

- **weight** *weight_file* [weight file name, default is *case_name.wt*]

3. **gen(erate)** - generates **object** by executing an ICEG2D (v2.0) module

   - **dist(ribution)** - generates distribution using defined information
   - **grid** - generates grid using defined information
   - **initial** - generates solver-appropriate initial file
   - **input** - generates solver-appropriate input file
   - **restart** - generates solver-appropriate restart file by interpolating old solution onto new grid
   - **solution** - runs selected solver as a batch job
   - **weights** - generates weight function file using defined information

4. . **display** - graphically displays something

   - **curv(ature)** - displays weight function used for distribution of points on surface
   - **dist(ribution)** - displays surface distribution
   - **grid** - displays grid

5. **show** - shows current file definition associations
6. **sys** "*unix_command*" - executes the Unix command in quotes
7. **help** - displays a help message
8. **exit, quit, bye** - exits ICG2D

Additional features of ICEG2D (v2.0) include:

- The user can initiate the process at any point. For example, if a surface definition already exists, the user can start the process by specifying the file containing the surface distribution and then generate the grid.
- Results of each part of the process are saved automatically either using the default name based on the case name or by file names specified by the user.
- In the interactive mode, the user can display the distribution of points on the surface as well as the generated grid in a pop-up window with translate and zoom capability.
- The program generates a history file in ICEG2D.HST

The batch mode of ICEG2D (v2.0) is invoked using

```
SysPrompt% ICEG2D < case#1.txt > case#1.out
```

where **case#1.txt** is an ASCII file containing the ICEG2D (v2.0)directives and **case#1.out** is the redirected output from ICEG2D (v2.0). The interactive mode is invoked using

```
SysPrompt% ICEG2D
```

and entering the ICEG2D (v2.0) directives from the ICEG2D (v2.0) prompt. Sample sessions for the batch and interactive modes are included in Section 9.

ICEG2D (v2.0) also prints informational messages to *stdout* indicating which associations are in effect, etc. Also printed are responses to the various ICEG2D (v2.0) directives.

Upon execution, ICEG2D (v2.0) prints a welcome message for the user that lists important information regarding the environment variables.

```
Magnolia[882] dst% ICEG2D

        ICEG2D - Integrated Geometry/Grid/Simulation Software (v2.0)

        ***** Welcome dst *****

        Fri Nov 10 14:42:31 CST 2000

        ICEG2D directory          - /ICEG2D
        NPARC directory           - /NPARC/2D
        HYBFL2D directory         - /HYBFL3D
        Current working directory - /ICEG2D/validation/nlf0441/623exp/aoa=3

ICEG2D>
```

The user would now enter a series of ICEG2D (v2.0) directives. The following sections describe the technology employed in and use of the various modules in ICEG2D (v2.0). Example directive streams are given in Section 9.

3. Geometry Modeling Module - ICE_DIST_2D

The quality of the surface grid point distribution is critical to the ultimate success of a CFD simulation. This is particularly true for iced airfoils. Chung, et. al. [13] used an interactive software package, TURBO-GRD [14], to smooth and distribute points on a given iced airfoil surface. In [13], the number of control points could be decreased to reduce the surface complexity, i.e., smooth the surface.

The ICE_DIST_2D module is designed to generate automatically a grid-ready surface definition for an iced airfoil. The surface definition may have been generated by an analysis program such as LEWICE [11] or obtained experimentally from the IRT [12]. The basic philosophy employed here is to represent a viable surface as faithfully as possible for a given number of points. Points are clustered points in regions of high surface curvature to ensure geometric fidelity of the iced airfoil surface definition. At no time during the process is any explicit smoothing applied to the airfoil surface definition. However, as discussed below, the airfoil surface is represented internally as a NURBS curve. When the points are distributed on the surface, some smoothing is present due to the fact that the NURBS representation does not reproduce slope discontinuities in the surface description [15] . However, this smoothing is present once a NURBS representation is used to describe the geometry and is not a function of the distribution algorithm.

The module ICE_DIST_2D is written in C and utilizes the GGLib [6] geometry/grid library. ICEG2D (v2.0) passes four file names as command line arguments to ICE_DIST_2D: the iced airfoil input file (associated with *iced_geom.dat*), the clean airfoil input file (associated with *clean_geom.dat*), the output file for the surface definition (associated with *dist_file*), and the distribution parameter file (associated with *dist_parm_file*). These file names are defined using the associations made within ICEG2D (v2.0). In the sections that follow, the input for the module is described along with the methodology used to automatically distribute points on the airfoil surface. The module ICE_DIST_2D is executed using the ICEG2D (v2.0) directive **gen(erate) dist(ribution)**.

3.1 ICE_DIST_2D Input

The input for ICE_DIST_2D consists of a file containing the iced airfoil definition, a file containing the clean airfoil definition, and a file containing the parameters for the namelist **DISTRIBUTION**. The iced airfoil definition file, associated with *iced_geom_file* in the ICEG2D (v2.0) framework, is an ASCII file consisting of the integer number of points employed in the surface definition (*n_points_iced_nl*) followed by *n_points_iced_nl* (x, y) data pairs defining the surface. The data may or may not represent a full, closed airfoil definition. The algorithm to merge partial iced airfoil data with full clean airfoil data is described below. The clean airfoil file, associated with *clean_geom_file*, is defined similarly with the integer number of points employed in the surface definition (*n_points_clean_nl*) followed by *n_points_clean_nl* (x, y) data pairs defining the clean airfoil. The clean airfoil definition must be a full airfoil definition. **In both files, the surface is defined starting at the trailing edge on the lower surface proceeding clockwise to the trailing edge on the upper surface. The trailing edge point must be repeated as the first and last points in the file.** If the trailing edge is not closed, the trailing edge point is placed at the average position of the upper and lower surface trailing edge locations.

Input parameters used in the surface distribution are defined using a FORTRAN namelist. The namelist **DISTRIBUTION** is contained in the file associated with *dist_parm_file*. The variables and their default values are listed in Table 3 and are described in detail in the appropriate section.

3.2 Merging and Pruning Algorithms

Data from the IRT is obtained in terms of partial iced airfoil shapes [10]. It is therefore necessary to devise an algorithm to merge partial iced airfoil data with a clean airfoil definition to obtain a complete airfoil. Additionally, the iced airfoil data often contains twists or loops in which the curve describing the surface actually loops back onto itself. Here, the operation removing these undesirable loops is termed pruning. Both algorithms are described below.

In developing the merging algorithm, there are two primary issues that must be addressed: 1) The locations on the clean airfoil definition where the iced airfoil should be merged must be determined and 2) the curves used to form the transitions between the partial iced surface and the clean surface must be specified. It should be noted that the approaches used to address these issues are based purely on heuristics. The primary objective is to ensure that the merging process introduces no significant discontinuities in the surface definition.

The merging algorithm implemented here is based on identifying the points in the clean airfoil surface definition closest to the two endpoints of the partial iced airfoil definition. Once these two points are located, one for each endpoint, the slope of the iced airfoil surface at each endpoint is used to estimate the intersection of the ice surface with the clean airfoil surface. The clean airfoil surface is then deleted between the intersection points. C^1 continuous Hermite interpolation [16] is then used to generate transition curves between the two surfaces.

Table 3: Namelist **DISTRIBUTION** variables

Namelist variable	dist_smooth.ini	dist_rough.ini
prune_nl	1.0	1.0
order_nl	3	3
lower_surface_split_point_nl	0.4	0.4
upper_surface_split_point_nl	0.1	0.1
rpower_nl	6.0	8.0
n_points_aft_lower_region_nl	61	61
n_points_nose_region_nl	151	251
n_points_aft_upper_region_nl	101	101
int_max_nl	2000	4000
n_smooth_nl	200	800
cluster_nl	2.0	2.0
max_min_ratio_nl	5.0	5.0
delta_s_te_nl	0.005	0.005

Figure 1.a shows examples of the transitions for a merged surface definition corresponding to IRT data for the GLC305 airfoil section denoted as case 904EXP [10]. This case was artificially created by deleting points from a full surface definition. As can be seen from the figure, the transition on the lower surface is smooth. A close up of the region near the intersection between the horn and the upper surface is shown in Figure 1.b. Again, the transition is smooth and maintains the surface continuity.

The pruning algorithm is also based on heuristics. The basis for the pruning algorithm is the observation that when "looping" occurs the following inequality is satisfied:

$$|\mathbf{r}_{i+k} - \mathbf{r}_i| \le \beta |\mathbf{r}_{i+1} - \mathbf{r}_i| \quad \text{for} \quad k > 1 \tag{1}$$

where β is a proportionality constant of $O(1)$ defined via the **DISTRIBUTION** namelist parameter *prune_nl*. The above equation states that looping is a concern if a nonadjacent point, $\mathbf{r}_{i+k}$, is closer to the point under consideration, $\mathbf{r}_i$, than a specified tolerance that is based on the distance to the adjacent point, $\mathbf{r}_{i+1}$. This relation is implemented by searching a portion of the iced airfoil definition $k = 1, \ldots, K$ where K is some fraction of the points used in the initial definition of the airfoil surface. If Equation 1 is satisfied for some $1 < k \le K$, the points contained within the segment $\mathbf{r}_i$ through $\mathbf{r}_{i+k}$ are eliminated. Figure 2 shows a pruned surface generated using $\beta = 1$. The surface definition corresponds to the IRT data denoted as case 904EXP [10]. The pruning algorithm described here generates a usable surface definition from a clearly unacceptable surface. At this time, the effects of pruning on the flow solution have not yet been quantified. However, it seems unlikely that any negative effects would be significant.

3.3 Internal Surface Description

Once a full surface description of the iced airfoil is obtained, an internal representation of the surface must be generated. Routines from the geometry/grid library GGLib [6] are used to define a NURBS representation for the iced airfoil surface. The parametric description of the NURBS curve is given by

$$\mathbf{P}(u) = (x(u), y(u)) \tag{2}$$

where u is the parametric variable. The namelist variable *order_nl* defines the order of the NURBS approximation. An inversion routine from GGLib is used to compute the control points needed to produce the specified surface. It should be noted that this inversion is the single most time-consuming element of the process aside from the CFD solution itself but requires less than one minute on typical workstations. Using another GGLib routine, the NURBS description is then split into three parts—the aft lower surface $\mathbf{P}_L(u)$, the nose region $\mathbf{P}_N(u)$, and the aft upper surface $\mathbf{P}_U(u)$—based on the **DISTRIBUTION** namelist variables *lower_surface_split_point_nl* and *upper_surface_split_point_nl* which represent x/c locations on the lower and upper surfaces respectively as shown in Figure 3. These split points should

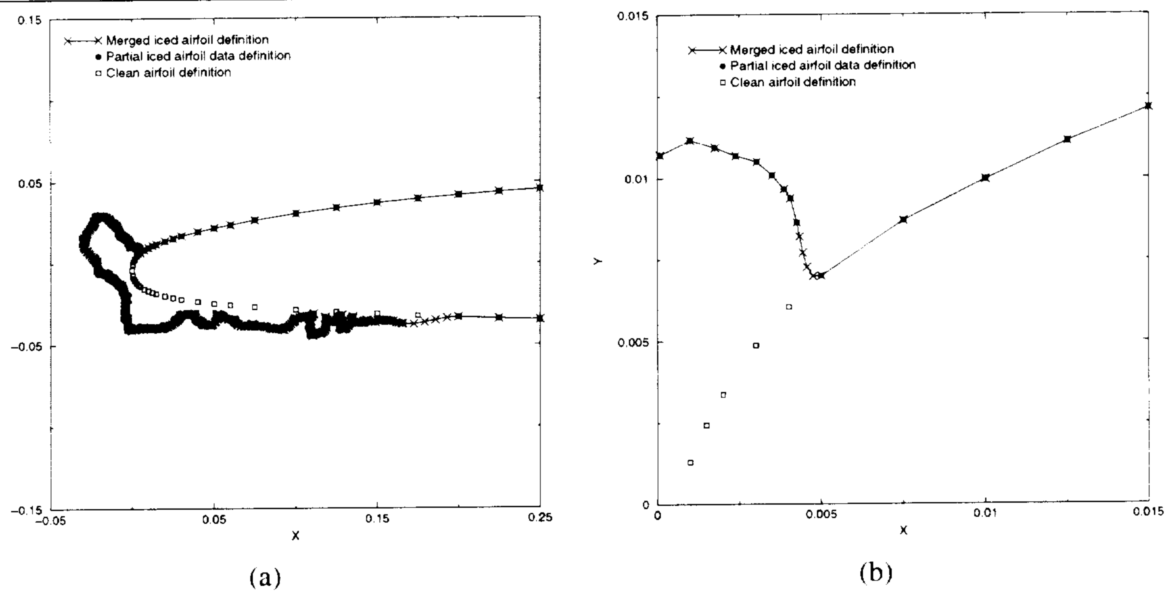


Figure 1: Demonstration of merging algorithm

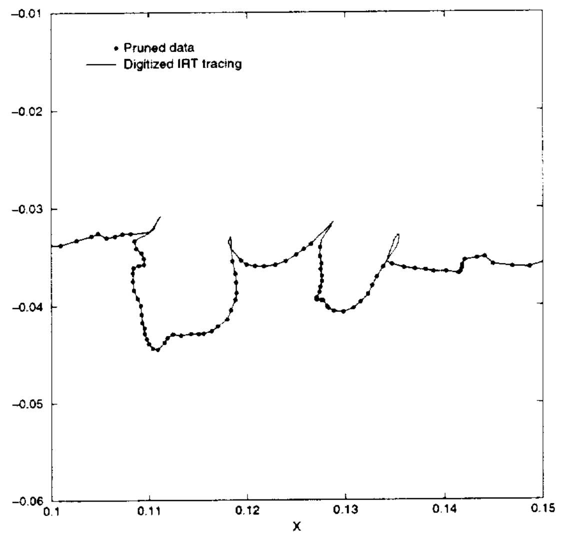


Figure 2: Demonstration of pruning algorithm

be defined such that the ice accretion is contained within the nose region. Since runback usually occurs only on the lower surface when the airfoil is at positive angle of attack, the upper surface split point is typically set significantly closer to the leading edge than the lower surface split point. The methods used to distribute points in each segment are described below.

3.4 Surface Point Distribution

The first step in distributing points on the airfoil surface is to determine the number of points to be used on each segment. The philosophy used here is to make the total number of points be a function of the length of the curve that describes the iced airfoil. In this way, a larger ice accretion is defined using more points. Currently, the only

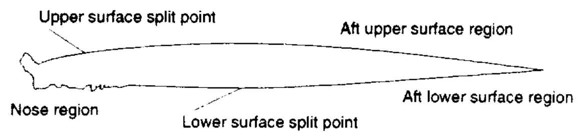


Figure 3: Nomenclature for NURBS description of iced airfoil

mechanism to account for a "jagged" surface in which the length of the curve does not change significantly is to use the **def(ine) surface** *rough* directive. This directive associates the file **dist_rough.ini** with *dist_parm_file*. The default values for the variables in the **DISTRIBUTION** namelist for smooth and rough surfaces are shown in Table 3.

The number of points on the iced airfoil surface is based on a function of the ratio of the iced airfoil surface length (s_{iced}) to the clean airfoil surface length (s_{clean})

$$\left(\frac{n_{iced}}{n_{clean}}\right) = \left(\frac{s_{iced}}{s_{clean}}\right)^{\gamma} \tag{3}$$

where

$$n\_clean = n\_points\_aft\_lower\_region\_nl + n\_points\_aft\_upper\_region\_nl + n\_points\_nose\_region \tag{4}$$

and γ is specified via the **DISTRIBUTION** namelist variable *rpower_nl*. To accommodate the method used to generate double-block grids, n_{iced} is automatically monitored to ensure that it is an odd number. The increase in points from n_{clean} to n_{iced} is assumed to occur in the iced region. The number of points in the aft lower and aft upper segments is defined via namelist input. It is assumed that the nose region is defined such that it is the only region in which ice accretion occurs. Therefore,

$$N = n_{iced} - n\_points\_aft\_lower\_region\_nl - n\_points\_aft\_upper\_region\_nl \tag{5}$$

where N is the resulting number of points in the nose region.

The distribution of points in the nose region is based on the equidistribution principle [16]. The approach used here is to distribute points on the NURBS description of the iced region of the airfoil, $\mathbf{P}_N(u)$, using a weight function based on surface curvature. For a distribution of N points along a curve of length s_{iced}, the location of the i^{th} point is computed using equidistribution from

$$\int_0^{s_i} w(s)\,ds = \frac{i-1}{N-1}\int_0^{s_{iced}} w(s)\,ds \tag{6}$$

where $w(s)$ is the weight function and s is the arc length along the curve [16]. The objective is to determine the value of the integrand s_i that satisfies the equality for each value of i. A direct result of using the equidistribution formulation is that in regions where the positive weight function is large, the mesh spacing is correspondingly small. Therefore, if the weight function is based on surface curvature, points will be clustered in regions of higher curvature leading to a better representation of the surface. For the problem at hand, the weight function is known explicitly in terms of the parametric arc length u. Therefore, using a change of variable, the equidistribution relation becomes

$$\int_0^{u_i} w(u)\left(\frac{ds}{du}\right)du = \frac{i-1}{N-1}\int_0^{1} w(u)\left(\frac{ds}{du}\right)du \tag{7}$$

where

$$\frac{ds}{du} = \sqrt{\mathbf{P}'_N(u)\cdot\mathbf{P}'_N(u)} \tag{8}$$

and $\mathbf{P}'_N(u)$ is computed using appropriate GGLib routines. The integral is evaluated by distributing *int_max_nl* (specified via the **DISTRIBUTION** namelist) equally spaced integration nodes on the NURBS representation $\mathbf{P}_N(u)$ in parametric space and performing a piecewise linear fit of the integrand. Using the integral formulation eliminates many of the problems associated with the finite-difference version of the equidistribution equation. In particular, no iteration is required to obtain the point distribution. Routines from GGLib are then used to evaluate the resulting (x, y) locations on $\mathbf{P}_N(u)$.

Using appropriate GGLib routines, the surface curvature is computed using

$$K(u) = \frac{\left|\left(\mathbf{P}'_N(u)\cdot\mathbf{P}'_N(u)\right)\mathbf{P}''_N(u) - \left(\mathbf{P}''_N(u)\cdot\mathbf{P}'_N(u)\right)\mathbf{P}'_N(u)\right|}{\left(\mathbf{P}'_N(u)\cdot\mathbf{P}'_N(u)\right)^2} \tag{9}$$

A clustering function is now defined at each of the integration nodes

$$c(u_j) = K(u_j)^{\left(1-\frac{1}{\alpha}\right)} \tag{10}$$

where $K(u_j)$ is the curvature and $\alpha \geq 1$ is a clustering parameter defined by the **DISTRIBUTION** namelist variable *cluster_nl*. The general effect of decreasing α is to increase the strength of the peaks of the normalized curvature, i.e., to increase the variation in curvature and increase the clustering in the point distribution.

The second step in formulating the weight function is to smooth the clustering function through repeated application of

$$\bar{c}(u_j) = \frac{1}{4}\left(c(u_{j-1}) + 2c(u_j) + c(u_{j+1})\right), \qquad j = 2, int\_max - 1 \tag{11}$$

It is important to note that the geometry is not being smoothed - only the clustering function used to define the weight function in the equidistribution scheme is being smoothed. The purpose of the smoothing is to reduce the variation of the surface point distribution in regions of rapid variation in curvature. Finally, the normalized weight function is defined in terms of the smoothed clustering function using

$$w(u) = \frac{\max\left(\bar{c}(u), \hat{c}_{min}\right)}{\bar{c}_{max}} \tag{12}$$

where $\hat{c}_{min} = \max\left(\bar{c}_{max}/\sigma, \bar{c}_{min}\right)$. $\hat{c}_{min}$ serves to keep the ratio of the maximum spacing to the minimum spacing in the nose region less than the tolerance σ which is specified via the **DISTRIBUTION** namelist variable *max_min_ratio_nl*.

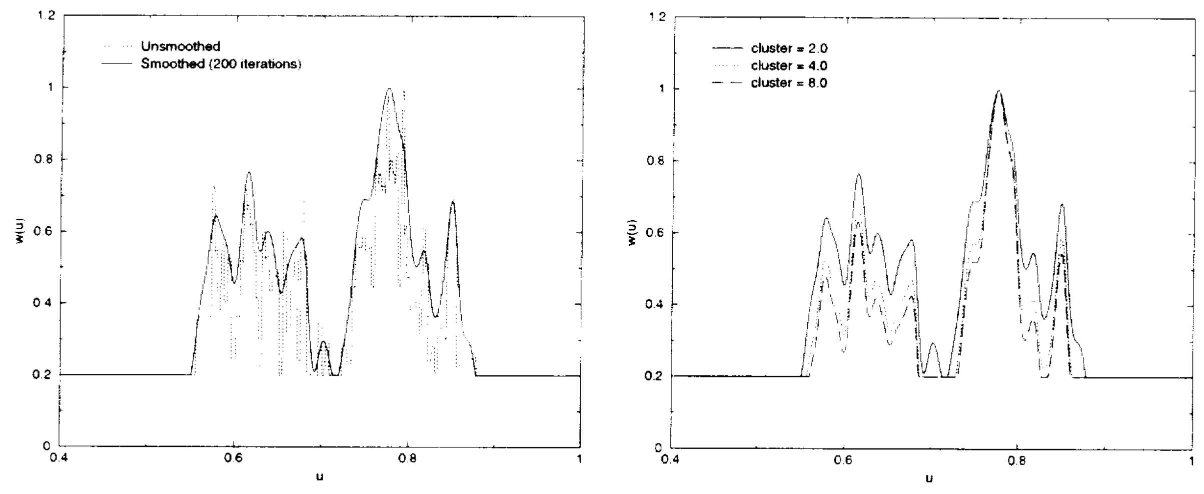
Points are distributed on the aft upper and aft lower surface segments, and respectively, using a hyperbolic tangent stretching function contained in GGLib. The spacing at the inboard end of each segment is set to match the spacing obtained from the nose region to ensure a smooth point distribution in the transition region. The spacing at the trailing edge is specified using the variable *delta_s_te_nl* from the **DISTRIBUTION** namelist.

3.5 Demonstration of Distribution Algorithm

It is now appropriate to present results for the point distribution algorithm as well as demonstrate the effects of various parameters that influence the surface distribution. Issues related to the fidelity of the geometric representation of the iced airfoil surface can easily be addressed by critical examination of several representative configurations. Ultimately, the issue of surface grid quality can be addressed only by performing numerical calculations and comparing the results with experimental data.

The first case considered here is a relatively smooth shape generated by LEWICE for the GLC305 airfoil section and is designated case 204LEW [10]. Figure 4.a shows a plot of the normalized weight function versus the parametric coordinate u in the iced region and demonstrates the effects of smoothing on the weight function. The parametric coordinate on this segment of the surface varies from a value of zero at the lower surface split point to a value of unity at the upper surface split point. Only a portion of the iced region of the airfoil is shown here. As can be seen from the figure, the general effect of smoothing is to reduce the rapid variation in the weight function. This ultimately results in a smoother point distribution on the surface as points are clustered in regions of high curvature rather than to discrete points of high curvature. This result suggests applying additional smoothing to rough or jagged ice shapes. This is the approach used in ICE_DIST_2D. A second distribution parameter file, **dist_rough.ini**, is used when *surface_type* is defined using **define surface** *rough*. Table 3 contains the default values for **DISTRIBUTION** namelist variables for both smooth and rough surfaces. Also shown in the figure is the effect of the variable *max_min_ratio_nl*. In this case, the ratio is set to 5.0 so that an artificial "floor" of 0.2 is placed on the normalized weight function.

Figure 4.b shows the effect of the parameter *cluster_nl* on the normalized weight function. The general effect of decreasing the parameter is the enhancement of peak values relative to the maximum peak. Given a fixed number of points, the net effect on the surface point distribution is the enhancement of clustering at the other peaks shown in the



(a) varying smoothing parameter (b) varying clustering parameter

Figure 4: Effect of smoothing and clustering parameters on normalized weight function

figure at the expense of clustering at the primary peak and other regions. In the extreme case, *cluster_nl=1*, the weight function is unity everywhere and equal spacing results.

Figure 5.a shows a plot of the surface distribution and the original iced airfoil data for case 204LEW [10]. As can be seen from the figure, the overall geometric fidelity of the surface is well maintained and points are clustered in regions of higher surface curvature. However, one point should be made regarding surface slope discontinuities. In the region where the ice horn intersects the upper surface of the airfoil, the surface representation is at least C^1 continuous (actually it is C^2 continuous) while the original data suggests only C^0 continuity. This condition will occur wherever the initial surface is only C^0 continuous. Note, however, that the curve does pass through the corner point. In effect, the surface is implicitly smoothed while maintaining the interpolating nature of the NURBS approximation. It should be noted that the straight-line segments in the original surface distribution near the right edge of the figure occur because the spacing between the points increases dramatically. The NURBS approximation in ICE_DIST_2D interpolates these points using a smooth curve rather than a line segment.

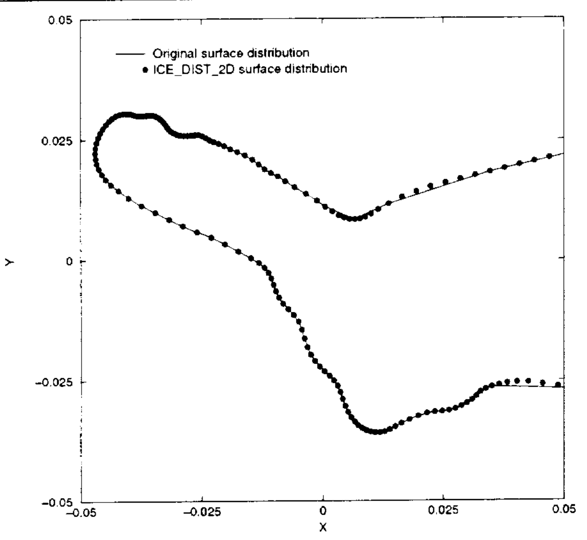
Figure 5.b shows a plot of the surface distribution and the original iced airfoil data for the case 904EXP obtained from IRT data [10]. This case is considerably more challenging. The surface distribution again shows point clustering in regions of higher curvature. However, the clustering is not as apparent as it is in Figure 5.a. This occurs because of the overall roughness of the ice shape. Essentially, there are fewer distinct features about which to cluster. Additionally, 904EXP has a significant, jagged accretion of ice on the lower surface. The lower surface accretion requires additional points to represent this portion of the surface and effectively removes points from the horn region.

It should be emphasized that the surface distribution is generated automatically. For most cases, no user intervention is required.

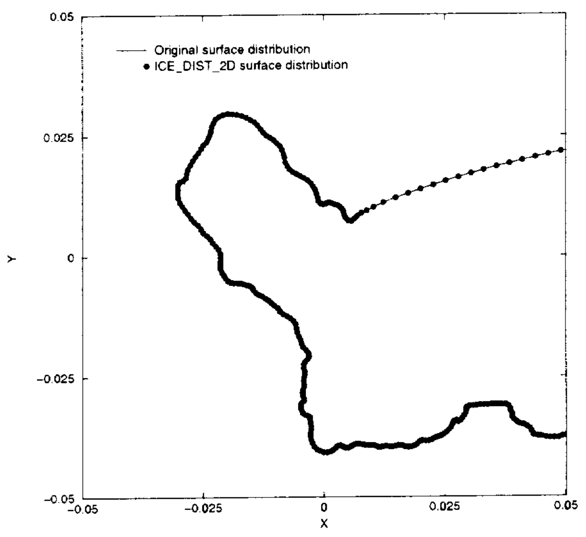
3.6 ICE_DIST_2D Output

The output from the ICE_DIST_2D module consists of a file containing the surface data and informational messages to *stdout*. The three airfoil segments are concatenated and output to a file in the FAST, multizone, formatted, two-dimensional, grid format [17]. The surface distribution may be viewed using the **display dist(ribution)** directive from ICEG2D (v2.0). The output file name is associated with *dist_file* with the default being *case_name.dst*. A sample of the information printed to stdout where *case_name* has been set to *623exp* and *surface_type* has been set to *rough* is shown below:

```
ICEG2D> gen dist
        Generating distribution using:
        Iced geometry file        - 623exp.dat
```



(a) 204LEW



(b) 904EXP

Figure 5: Automatic surface point distribution - iced region

```
Clean geometry file       - 623exp.cln
Parameter definition file - /ICEG2D/parameter_files/dist_rough.ini
Distribution output file  - 623exp.dst

Begin surface distribution calculation.

prune                     = 1.000000
order                     = 3
n_points_aft_lower_region = 61
n_points_nose_region      = 251
n_points_aft_upper_region = 101
lower_surface_split_point = 0.400000
upper_surface_split_point = 0.100000
delta_s_te                = 0.005000
cluster                   = 2.000000
rpower                    = 8.000000
max_min_ratio             = 5.000000
int_max                   = 4000
n_smooth                  = 800

*** WARNING *** Open trailing edge found for clean airfoil.
TE location set at average of upper and lower points.

*** WARNING *** Open trailing edge found for iced airfoil.
TE location set at average of upper and lower points.


Ratio = 1.053185
Number of points on airfoil = 623
```

```
        Approximate ds(max)/ds(min) = 5.521702

        Lower surface
           Trailing edge - ds(gen) = 0.005091, ds(spec) = 0.005000
           Split point   - ds(gen) = 0.003419, ds(spec) = 0.003348

        Upper surface
           Split point   - ds(gen) = 0.003384, ds(spec) = 0.003348
           Trailing edge - ds(gen) = 0.005044, ds(spec) = 0.005000

        Surface distribution generation complete!

ICEG2D>
```

The output consists of an inventory of relevant file associations followed by an echo of the namelist **DISTRIBUTION**. The user is then notified that an open trailing edge was detected and action was taken. The ratio s_{iced}/s_{clean} is then output along with the number of points on the airfoil surface n_{iced}. Diagnostic information follows showing an approximate maximum to minimum spacing ratio based on cord length (as opposed to arc length) in the nose region and a comparison of the specified and generated spacings at the end points of the aft upper segment and aft lower segment. Finally, a message is printed indicating successful completion of the surface definition.

The surface definition output file (associated with *dist_file*) is written in formatted, two-dimensional, multiblock FAST format. It can be plotted using any visualization software package that can read this format or the module ICE_PLOT_2D described in Section 7.

4. Grid Generation Module - ICE_GRID_2D

Automated grid generation for complex configurations is an, as yet, unsolved challenge. Even grids for the topologically simple single-element iced airfoils are challenging because of concave regions occurring on the surface of the airfoil. These regions pose difficulties for algebraic or elliptic grid generation algorithms because of the difficulty in appropriately placing corresponding points on the outer boundary of the grid. One approach that obviates this problem is to use a marching technique to generate the grid. In this context, a marching scheme uses the surface distribution as the initial data curve and generates a grid layer by layer until a specified number of layers have been generated. Typically, the marching distance distribution, i.e., the thickness of each layer, is specified. Both parabolic [3, 4, 7–9] and hyperbolic [18] schemes have been used for marching grid generation algorithms. In this effort, a parabolic scheme was chosen because of the ability to generate solution adaptive grids [8, 9].

The module ICE_GRID_2D is written in FORTRAN90 with a C driver routine and utilizes dynamic memory allocation. ICEG2D (v2.0) passes the case name, an adaptation flag (associated with *adapt_type*), a HYBFL2D restart flag (associated with *reflag*), and six file names as command line arguments to ICE_GRID_2D: the airfoil surface definition file (associated with *dist_file*), the output file for the grid (associated with *grid_file*), the grid generation parameter file (associated with *grid_parm_file*), the block definition parameter file (associated with *block_parm_file*), and for solution adaptive grids, the initial grid file (associated with *input_grid_file*), and the weights file (associated with *weight_file*). These file names can be defined using the default associations in ICEG2D (v2.0) or by user specification. In the sections that follow, the input for the module is described along with the methodology used to automatically generate a single- or double-block, C-type structured grid and a generalized topology grid. The module ICE_GRID_2D is executed using the directive **gen(erate) grid**.

4.1 ICE_GRID_2D Input

The input to ICE_GRID_2D consists of a file containing the data for the namelist **GRID_PARAMETERS** (associated with *grid_parm*) and a file containing data for the namelists **NUMBER_OF_BLOCKS** and **BLOCK_PARAMETERS** (associated with *block_parm*). The airfoil surface definition file, associated with *dist_file*, is required to define the initial surface for the grid generation algorithm. The case name is used to generate an identifying banner in the output from the module to *stdout*. The variables in the various namelists and their default values are listed in Tables 4-7 below and are defined in the appropriate sections.

Table 4: Namelist **GRID_PARAMETERS** variables (*grid_parm*)

Namelist variable	Default Value
itMax_nl	20
nOrder_nl	2
omega_nl	1.0
addedDissipation_nl	1.0
iWake_nl	40
distWake_nl	15.0
convgTol_nl	1.0×10^{-7}

Table 5: Namelist **NUMBER_OF_BLOCKS** variables (*block_parm*)

Namelist variable	single_block.ini	double_block.ini
nBlocks_nl	1	2

4.2 Blocking Strategies

The ICEG2D (v2.0) framework allows for two different structured grid configurations: a single-block topology and a double-block topology. Here, the number of layers is defined so that $jMax = nLayers\_nl + 1$. The single-block

Table 6: Namelist **BLOCK_PARAMETERS** variables - single-block grid (*block_parm*)

Namelist variable	single_block.ini
distFirstPointLE_nl	1.0×10^{-6}
distFirstPointTE_nl	1.0×10^{-6}
distFirstPointDSB_nl	1.0×10^{-6}
distOuterBoundary_nl	15.0
xTransLower_nl	0.5
xTransUpper_nl	0.5
nLayers_nl	100
deleteNode_nl	.false.
geometricDeletionCriterion_nl	1.0
adaptiveDeletionCriterion_nl	0.025
insertNode_nl	.false.
geometricInsertionCriterion_nl	0.25
adaptiveInsertionCriterion_nl	0.75
adaptationScale_nl	2.5×10^{-3}

Table 7: Namelist **BLOCK_PARAMETERS** variables - double-block grid (*block_parm*)

Namelist variable	Block #1	Block #2
distFirstPointLE_nl	1.0×10^{-6}	computed
distFirstPointTE_nl	1.0×10^{-6}	computed
distFirstPointDSB_nl	1.0×10^{-6}	computed
distOuterBoundary_nl	1.0	14.0
xTransLower_nl	0.5	NA
xTransUpper_nl	0.5	NA
nLayers_nl	70	20
adaptationScale_nl	2.5×10^{-3}	NA

grid is generated directly using the techniques outlined below. The inner block of the double-block grid, designated as block number one, is treated in the same manner. The outer block grid generation requires the specification of an initial surface. The outer block is assumed to overlap the last two layers of the inner block. The outer-block initial surface is defined using a surface generated by taking the coordinates of the first and last points and every other point in between on the surface $j = jMax - 2$ on the inner grid. Recall that the number of points on a j=constant surface in the inner grid is forced to be odd, so this approach will always work.

In general, the file *block_parm_file* will consist of (*nBlocks_nl*+*1*) different namelists–a single occurrence of the namelist is **NUMBER_OF_BLOCKS** plus the *nBlocks_nl* occurrences of the **BLOCK_PARAMETERS** namelists–one for each block in the grid.

4.3 Wake Cut Definition

Because of improved resolution at the airfoil trailing edge, a C-type grid is used in all cases. Therefore, it is necessary to define a wake cut that extends from the trailing edge of the airfoil to the downstream boundary of the computational domain. Here, it is assumed that the cut is parallel to the x-axis. The distance to the downstream boundary is specified via the namelist **GRID_PARAMETERS** using the variable *distWake_nl*. The number of points defining the cut is given by *iWake_nl*. Note that the trailing edge point is considered a wake point so that the number of points downstream of the trailing edge is *iWake_nl-1*. Therefore, the total number of points on the $j = 1$ surface is given by

$$iMax = n_{iced} + 2 \times (iWake\_nl - 1) \quad . \tag{13}$$

The distribution of points on the cut is defined using a hyperbolic tangent stretching function [19]. The first point downstream of the trailing edge is located a distance equal to the distance from the trailing edge to the first point upstream of the trailing edge on the airfoil surface. This distance, the distance to the downstream boundary, and the number of points on the cut are sufficient to compute the distribution using the hyperbolic tangent function.

4.4 Parabolic Grid Generation

In the parabolic method [3, 4, 7–9], a reference grid is utilized to make the marching problem well posed. Starting from the initial data surface, two layers of a reference grid are generated. The Poisson grid generation equations are then applied to points on the intermediate surface of the reference grid, thereby smoothing the reference grid. The outer surface is updated after each iteration of the Poisson grid equations. Note that the outer surface of the reference grid is discarded once the desired number of smoothing iterations is complete. It should also be noted that the resulting smoothed grid still exhibits many of the characteristics of the reference grid.

As is customary [16], a transformation of the form

$$\begin{aligned} \xi &= \xi(x, y) \\ \eta &= \eta(x, y) \end{aligned} \tag{14}$$

is defined and the inverse transformation

$$\begin{aligned} x &= x(\xi, \eta) \\ y &= y(\xi, \eta) \end{aligned} \tag{15}$$

is assumed to exist. The initial data surface is assumed to be defined by a η=constant surface so that η is the marching direction.

4.4.1 Normal Distance Distribution

The normal distance distribution is generated using the hyperbolic tangent stretching function defined in [19]. The distance to the first point off the wall is specified along with the distance to the last point and the total number of points. The hyperbolic tangent function was selected because it was shown to induce the least overall truncation error of several stretching functions considered [19].

As implemented here, the distance to the first point off the wall is specified at three locations: the leading edge of the airfoil *distFirstPointLE_nl*, the trailing edge of the airfoil *distFirstPointTE_nl*, and the downstream boundary of the computational domain *distFirstPointDSB_nl*. Using these three values and the transition points *xTransLower_nl* and *xTransUpper_nl*, linear interpolation is used to define the distance to the first point for each point on the initial data surface. For the upper and lower surfaces

$$\delta_i^1 = \begin{cases} \delta^{LE} & , \quad x_{LE} \le x_i \le x_{TRANS} \\ \delta^{LE} + \left(\frac{x_i - x_{TRANS_{U,L}}}{x_{TE} - x_{TRANS_{U,L}}}\right) \times \left(\delta^{TE} - \delta^{LE}\right) & , \quad x_{TRANS} < x_i \le x_{TE} \\ \delta^{DSBE} + \left(\frac{x_i - x_{TE}}{x_{DSB} - x_{TE}}\right) \times \left(\delta^{DSB} - \delta^{TE}\right) & , \quad x_{TE} < x_i \le x_{DSB} \end{cases} \tag{16}$$

where δ_i^1 is the distance to the first point off the wall at x_i. These values are then smoothed to eliminate the propagation of slope discontinuities from the boundary into the interior of the grid. This approach is used to give flexibility regarding the normal mesh distribution at the leading edge, the trailing edge, and the downstream boundary. Then, the distance to the outer boundary *distOuterBoundary_nl* and number of points *jMax* are used to compute the normal distance distribution $\delta_i^j, j = 1 \ldots jMax$ at each point on the initial data surface.

In the case of the double-block configuration, the interior block is treated in an identical manner as above. The distance to the first point in the outer block is based on the spacing in the overlap region in the inner block to ensure a good match with the inner block grid.

4.4.2 *Reference Grid Definition*

The reference grid is generated algebraically to be locally orthogonal. The intermediate surface of the reference grid in layer j is defined using

$$\mathbf{r}_{i,1}^{j} = \mathbf{r}_{i,0}^{j} + \delta_i^j \mathbf{n}_{i,0}^{j} \tag{17}$$

where $\mathbf{r}_{i,0}^{j}$ is the position vector to the point located at i on the initial data surface (0 subscript), $\mathbf{r}_{i,1}^{j}$ is the position vector to the point i on the first surface of the reference grid, δ_i^j is the specified distance distribution, and $\mathbf{n}_{i,0}^{j}$ is the unit surface normal. The outer surface of the reference grid in layer j is generated using

$$\mathbf{r}_{i,2}^{j} = \mathbf{r}_{i,1}^{j} + \delta_i^{j+1} \mathbf{n}_{i,1}^{j} \ . \tag{18}$$

Although not discussed here, it is possible to adjust the reference grid as the grid marches away from the initial data surface to meet specified outer boundary points.

4.4.3 *Grid Smoothing*

The equations used to smooth the reference grid are the standard Poisson equations [16] used in grid generation with the assumption of grid orthogonality, i.e., g_{12} is zero:

$$g_{22}\left((1+\nu)\mathbf{r}_{\xi\xi} + \phi\mathbf{r}_{\xi}\right) + g_{11}\left(\mathbf{r}_{\eta\eta} + \phi\mathbf{r}_{\eta}\right) \tag{19}$$

where

$$\begin{aligned} g_{11} &= x_\xi^2 + y_\xi^2 \\ g_{22} &= x_\eta^2 + y_\eta^2 \end{aligned} \ . \tag{20}$$

The partial derivatives in the Poisson equation are approximated using second-order central differences. The resulting system of equations is solved using line relaxation.

The form of the control functions plays an important role in determining the quality of the mesh. The form of the control functions used here does not include curvature effects [20] and is given by

$$\begin{aligned} \phi &= -\frac{1}{2}\frac{1}{g_{11}}\frac{\partial g_{11}}{\partial \xi} \\ \psi &= -\frac{1}{2}\frac{1}{g_{22}}\frac{\partial g_{22}}{\partial \eta} \end{aligned} \ . \tag{21}$$

It should be noted that Equation 21 is equivalent to employing the "geometric" weight functions defined by

$$\begin{aligned} w_g^\xi &= \frac{1}{\sqrt{g_{11}}} \\ w_g^\eta &= \frac{1}{\sqrt{g_{22}}} \end{aligned} \tag{22}$$

with ϕ and ψ defined using

$$\begin{aligned} \phi &= \frac{1}{w_g^\xi}\frac{\partial w_g^\xi}{\partial \xi} \\ \psi &= \frac{1}{w_g^\eta}\frac{\partial w_g^\eta}{\partial \eta} \end{aligned} \ . \tag{23}$$

For structured grid generation, the control function ϕ is computed using the point distribution on the initial data line (g_{11} on the airfoil surface) and held constant along each ξ=constant line. ψ can be approximated using the normal distance distribution by noting that $g_{22} = \delta^2$.

On occasion, it has been found that adding dissipation to the Poisson smoothing equation is necessary, particularly when generating meshes in strongly nonconvex regions [3, 4]. This approach is similar in spirit to the approach taken by Chan and Steger [18] and is shown in detail for the ξ direction.

In regions where $g_{11} >> g_{22}$, the dominant term in the Poisson equation is the term containing the η derivatives. In nonconvex regions, the resulting lack of smoothing along η=constant lines can lead to mesh line crossing. The approach taken here is to include the additional smoothing term $(1+\nu)$ in the Poisson smoothing equation with

$$\nu = \sqrt{\frac{\max\left(g_{11}, g_{22}\right)}{g_{22}}} \times f\left(\theta_\xi\right) \tag{24}$$

where

$$f(\theta_\xi) = \begin{cases} 1 & , \quad 0 \le \theta_\xi < \frac{\pi}{2} \\ \sin\theta_\xi & , \quad \frac{\pi}{2} \le \theta_\xi < \pi \\ 0 & , \quad \pi \le \theta_\xi \end{cases} \tag{25}$$

and θ_ξ is the angle between $(\mathbf{r}_{i+1} - \mathbf{r}_i)$ and $(\mathbf{r}_{i-1} - \mathbf{r}_i)$. The amount of dissipation may be increased or decreased via the **GRID_PARAMETERS** namelist variable *addedDissipation_nl* that has a default value of unity. The parameter *convgTol_nl* specifies a criterion for the RMS of the point movement, below which no further smoothing is performed.

Recall the objective here is not to solve the elliptic grid generation equations but rather use them as an intelligent smoothing mechanism. In practice, it has been found that reducing the grid point movement induced by the smoothing by two orders of magnitude results in a grid that is sufficiently smooth. This value corresponding to the number of orders of magnitude reduction in the residual is specified via the **GRID_PARAMETERS** namelist variable *nOrder_nl*. Typically, this process requires the equivalent of fewer than ten iterations of an elliptic solver. Domains containing strongly non-convex regions require additional iteration. The maximum number of iterations is specified using *it-Max_nl* in the **GRID_PARAMETERS** namelist. It is also possible to adjust the relaxation factor *omega_nl*. However, it has been determined that the best results are typically obtained for the default value of unity.

4.5 Generalized Grid Generation

The grids generated using the method described here can be classified as generalized grids since they combine elements of both structured and unstructured topologies. They can perhaps best be categorized as semistructured. The grid is generated in structured layers starting at the initial data surface–typically the body. Each layer consists of two surfaces–the initial data surface for that layer and the surface generated during the grid generation process. Since the smoothing is applied to the structured layers as the grid is extruded, the parabolic marching algorithm described in Section 4.4 can be used.

A generalized grid with point insertion/deletion can be generated by defining the object **topo** to be *semi(structured)* and by setting the variables *delete_node_nl* and *insert_node_nl* in the namelist **BLOCK_PARAMETERS** to .true. It should be noted that it is possible to have deletion without insertion and vice versa, but, that to have either, **topo** must be set to *semi(structured)*.

4.5.1 Node Deletion/Insertion at Layer Interface

The deletion/insertion algorithm employed here operates at the interface between layers. Using criteria defined by the algorithm, a deletion/insertion stencil is created. This approach maintains the structured nature of the grid within layers so that the parabolic scheme remains applicable and produces ξ=constant grid lines that exist in one layer but do not continue into the adjacent layer. Traditionally, the resulting grid points have been called hanging nodes. The stencils are then used to define the initial data surface for the next layer and the process is repeated. When points are deleted, the geometric weight function w_g^ξ is recomputed using the local values of g_{11}. In this way, the proper spacing is maintained between grid points even when point deletion or insertion occurs. It should be noted that only grid lines normal to the extruded surface, in this case ξ lines, may be inserted. In this way, the structure of the grid is maintained within each layer.

The node deletion process is illustrated for a two-dimensional grid in Figure 6. The points 1, 2, and 3 form the initial data surface for layer k. Points 4, 5, and 6 are generated using the parabolic marching algorithm. Note that points 4, 5, and 6 are not collinear. Based on the deletion/insertion algorithm, it is determined that point 5 should not be used to generate the grid in the next layer $k + 1$. Therefore, the initial data surface for layer $k + 1$ contains the points 4 and 6 but not point 5. The marching algorithm is then used to generate the surface containing points 7 and 8. Note, however, that the cell in layer $k + 1$ is actually defined using points 4, 5, 6, 8, and 7 resulting in a five-sided cell even though point 5 was not used to compute points 7 and 8. The algorithms checks to ensure that the cells remain convex.

For the iced airfoil problem, there are four main advantages to using a grid topology of this type for near-body grids:

1. Unlike unstructured grid topologies using tetrahedral cells, it is possible to maintain large aspect ratio cells in viscous-dominated regions near no-slip boundaries resulting in accurate simulations in these regions and to easily transition to lower aspect ratio cells away from these regions.

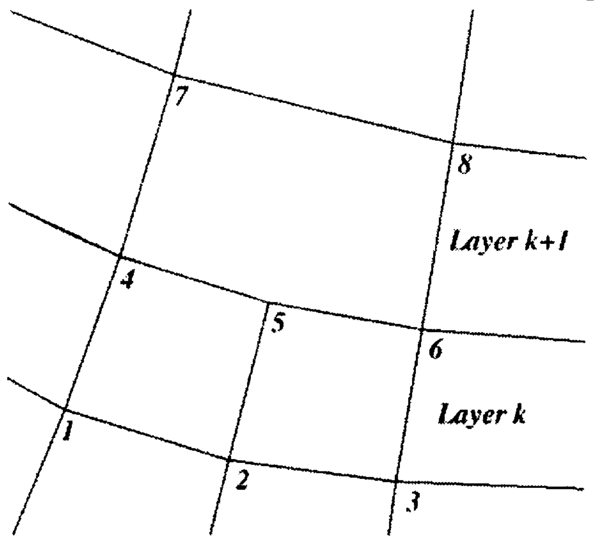


Figure 6: Semistructured grid topology

2. Higher-quality grids are obtained because lines can be inserted in regions of diverging grid lines, i.e., near convex corners.
3. Dense boundary point distributions may be used without the accompanying waste of grid points near the outer boundary that occurs with structured grid topologies. By using a line deletion/insertion algorithm based on cell aspect ratio, the points are deleted as the cell aspect ratio naturally decreases.
4. It is possible to develop a solution-adaptive grid strategy based on point redistribution and refinement. In addition to incorporating solution gradient information in the grid control functions, it is also possible to use solution information in the deletion/insertion algorithm.

4.5.2 Line Deletion/Insertion Strategy

The line deletion/insertion algorithm employed here is based solely on geometrical properties of the grid. There is great flexibility in the design of the deletion/insertion algorithm and the strategy discussed below is certainly not unique.

An ξ line is tagged for deletion if it meets a criterion based on a "slenderness" measure. This criterion is based on the observation that, in general, tall (low aspect ratio) cells do not benefit the computation. An ξ=constant line is marked for deletion if the value of the ratio of the η arc length to the ξ arc length is greater than a specified tolerance

$$\sqrt{\frac{g_{22}}{g_{11}}} > \alpha \ . \tag{26}$$

Here, α is given by *geometricDeletionCriterion_nl* defined in the namelist **BLOCK_PARAMETERS**.

Since the criteria described in Equation 26 is on a line-by-line basis without any information regarding adjacent lines, it is necessary to determine the deletion/insertion stencil incorporating the information from adjacent lines. The first step is tagging contiguous groups of ξ lines that are selected for deletion and identifying whether the groups contain an even number or an odd number of elements as described below:

- In the case of an odd number of contiguous ξ lines selected for deletion, the first line of the group, and every other succeeding line of the group, are deleted. The case of a single line selected for deletion is trivial.
- The case of an even number of contiguous ξ lines selected for deletion is treated using a delete two/insert one strategy. For each consecutive pair of lines tagged for deletion, both are deleted and replaced by the line defined by connecting the midpoints between the two lines tagged for deletion.

It should be noted that, in the case of line deletion in a convex region, it is possible that a nonconvex cell may be generated. This presents difficulties for the generation of cell areas and normals. Therefore, when line deletion occurs, a check is made to see if the local surface is convex. If so, the adjacent points are adjusted so that a convex cell is generated.

The criteria used to determine if an ξ=constant line should be inserted is based on the divergence of the η=constant faces of the cell. Unlike the deletion criteria, the insertion criteria apply to a cell face rather than a line. An ξ line is inserted on an edge provided

$$\frac{1}{\sqrt{g_{22}}}\frac{\partial}{\partial\eta}(\sqrt{g_{11}}) > \beta \tag{27}$$

where β is given by *geometricInsertionCriterion_nl* defined in the namelist **BLOCK_PARAMETERS**.

The line deletion/insertion scheme described above produces cell edges having one, two, or three segments. The cell faces can have four to seven edges. However, for viscous-type grids in particular, a majority of the cells (typically greater than 95 percent) are quadrilaterals. It should be noted that other deletion/insertion schemes will produce different cell topologies.

4.6 Solution Adaptive Grid Generation

One of the difficulties reported in [2] was that for airfoils with "horn" accretions, the separated flow downstream of the horn was not well predicted. Typical grids for this type configuration are shown in Figure 7. It was suggested that the quality of the grid downstream of the horn was responsible for the disagreement. To address this issue, as well as the overall issues of solution accuracy, solution adaptive gridding was included as an option in ICEG2D (v2.0) during Year 2.

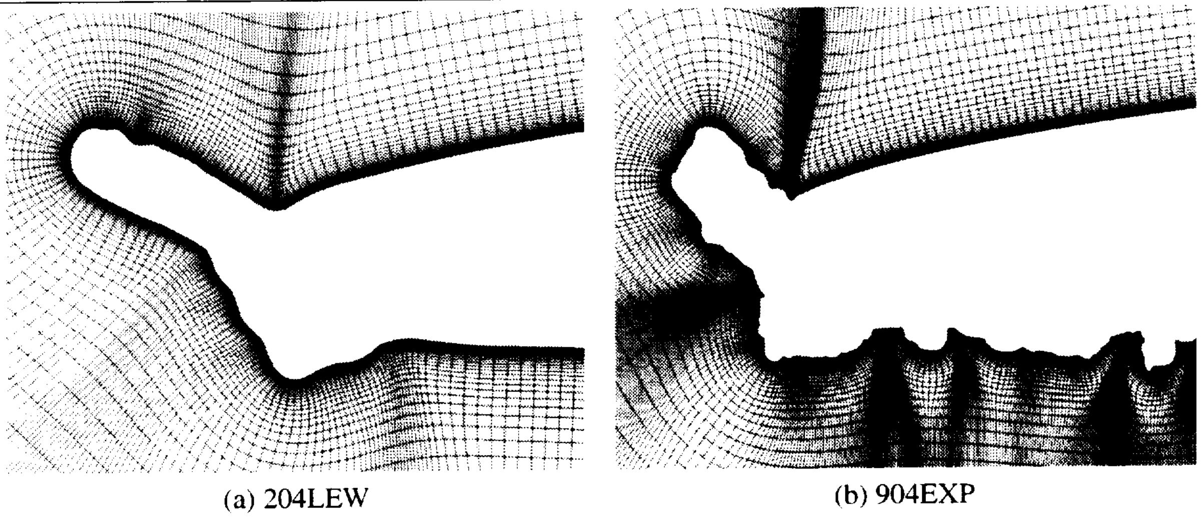


Figure 7: Typical automated iced airfoil grids - region near leading edge

The method employed here for redistribution-based, solution-adaptive structured grids is similar to the approach used in [8, 9, 21]. However, it has been specialized for low-speed flow fields. The method employed for generalized solution adaptive grids using redistribution and refinement is new and is reported here for the first time. However, at this time, it is recommended that only adaptation with redistribution be used. Many research issues relating to refinement of generalized grids remain unaddressed. In the sections that follow, the method used to define the weight function is discussed along with the specific techniques that were used to implement the algorithm.

4.6.1 Solution-Adaptive Weight Function Definition

The first step in generating any solution-adaptive grid is to define the sensors used to control the adaptivity. Since the objective of this effort is to improve the prediction of the separated region downstream of the horn, clustering the grid in the region where the shear layer is located would appear to be a prudent course of action. Separated shear layers are characterized by significant transverse velocity gradients while the transverse pressure gradient, typically smaller, is a function of the streamline curvature. Changes in the direction along the shear layer also tend to be less pronounced. Therefore, it would appear that a good choice of a variable to use in the sensor would be the velocity gradient. That is, a smaller mesh spacing is desired in the direction across the shear layer, i.e., the direction in which the velocity gradient is larger. This can be contrasted to [8, 9, 21] where functions employing pressure, density, and Mach number were used. Of course, the examples considered in those papers were for high speed flows where compressibility effects were important. It should be noted that de facto clustering already exists in the ξ direction because of point clustering on the airfoil surface in regions of high curvature. Typically, significant pressure changes occur in these regions.

For a general scalar f, the weight function is generated using the following procedure:

1. The gradient of f in physical (x, y) space ∇f is computed.
2. The components of ∇f are "clipped" based on a specified constant using

$$\nabla f_c = \left(\min\left(\frac{|f_x|}{f_{max}}, 1\right), \min\left(\frac{|f_y|}{f_{max}}, 1\right) \right) \quad . \tag{28}$$

   In this way, large local values are not allowed to "swamp" the field. The maximum gradient component is limited to a magnitude of $f_{max} = 0.2$.

3. The magnitude of the "clipped" gradient is evaluated using

$$|\nabla f_c| = \sqrt{f_{c_x}^2 + f_{c_y}^2} \tag{29}$$

4. The "clipped" gradient is then smoothed $\left(\overline{|\nabla f_c|}\right)$ and normalized to yield the solution-adaptive weight function

$$w_a = \frac{\overline{|\nabla f_c|}}{\max\left(\overline{|\nabla f_c|}\right)} \tag{30}$$

   where max indicates an extremum over the entire domain. As defined, the weight function is restricted to the range $0 \leq w_a \leq 1$, i.e., it is nonnegative and bounded by unity.

In this case, the scalar f is the velocity magnitude. The only user-specified variable in this formulation is the clipping value. We have found the value employed here to be effective for all problems considered to date.

4.6.2 Directional Weight Functions

As with any solution-adaptive grid, there is some desire to maintain the geometric characteristics of the initial grid. In most solution adaptive grid algorithms, this is accomplished at the level of the control function [21] using a linear weighting

$$\phi = \phi_g + \alpha\phi_a \quad . \tag{31}$$

However, this method seems to offer little control over the spacing due to adaptation relative to the spacing due to the initial grid. The solution to this problem is to consider combining the geometric weight functions (Equation 22) and the adaptive weight function (Equation 30) to obtain directional weight functions w^ξ and w^η and using

$$\begin{aligned} \phi &= \tfrac{1}{w^\xi}\tfrac{\partial w^\xi}{\partial \xi} \\ \psi &= \tfrac{1}{w^\eta}\tfrac{\partial w^\eta}{\partial \eta} \end{aligned} \quad . \tag{32}$$

The question, then, is how to combine $w_g^{\xi,\eta}$ with w_a. The approach employed here is to use a Boolean sum of the adaptive weight function with the normalized geometric weight functions as given by

$$w^{\xi,\eta} = \alpha w_a \oplus \left(\frac{w_g^{\xi,\eta}}{\max\left(w_g^{\xi,\eta}\right)} \right) \tag{33}$$

where $a \oplus b = a + b - ab$ with $0 \leq a,\ b \leq 1$ is the Boolean sum (note that $0 \leq a \oplus b \leq 1$) and $\max$ indicates the maximum value over the domain. The Boolean sum brings logical and/or effects into the evaluation of the weight function. Thus, the directional weight function defined by Equation 33 is nonnegative and is bounded by unity.

To understand the significance of α, consider the simple equidistribution equation

$$w \Delta s = C \tag{34}$$

where

$$w = \alpha w_a \oplus \left(\frac{w_g}{\max(w_g)} \right) \ . \tag{35}$$

For the purpose of this example, assume that $w_a << 1$ in the region where $w_g / \max(w_g) = 1$ and that Δs_g is the spacing in this region. Further assume that $w_a = 1$ in the region where $w_g / \max(w_g) << 1$ and that Δs_a is the spacing in this region. Note that since unity is the maximum value of the normalized weight function, Δs_g and Δs_a represent the minimum spacings due to the initial grid and solution adaptivity respectively. Now, substituting each of these conditions into Equation 34 yields

$$\left(\frac{w_g}{\max(w_g)} \right) \Delta s_g = \alpha w_a \Delta s_a \tag{36}$$

which becomes

$$\Delta s_g = \alpha \Delta s_a \tag{37}$$

or

$$\alpha = \frac{\Delta s_g}{\Delta s_a} \ . \tag{38}$$

Thus, α approximates the ratio of the minimum spacing due to geometry to the minimum spacing due to adaptivity. For example, if the first point of the wall has a spacing of 10^{-6}, $\alpha = 1$ implies a similar spacing in the region where the spacing dominated by adaptation is at its minimum value. However, if a spacing of 10^{-3} is desired, $\alpha = 10^{-3}$ is an appropriate value. The parameter α, with $0 \leq \alpha \leq 1$, defined by the variable *adaptationScale_nl* in the namelist **BLOCK_PARAMETERS**, controls the clustering due to solution adaptivity relative to the clustering due to the initial grid distribution. The default value, which has been found to work for all cases considered to date is 2.5×10^{-3}.

The module WEIGHT_GEN is written in FORTRAN90 with C driver routine. The weight function is generated using the directive **gen(erate) weights**. ICEG2D (v2.0) passes the grid topology and three file names to the module: the input grid (associated with *input_grid_file*), the input solution file (associated with *input_q_file*), and the output weights file (associated with *case_name.wt*). Currently, weights can be generated only from solutions generated on structured grids.

4.6.3 Solution-Adaptive Grid - Redistribution

The key to generating solution adaptive grids using a parabolic marching scheme is the procedure used to define the marching step distribution δ. Since the resulting smoothed grid shares many of the characteristics of the reference grid, the marching distribution used to define the reference grid must reflect the weight function distribution if a solution adaptive grid is to be generated successfully. A four-step procedure is employed here:

1. δ is defined based on the input parameters using a hyperbolic tangent distribution as before and a geometric weight function w_g is defined using
$$w_g^\eta = \frac{1}{\delta} \ . \tag{39}$$
2. The adaptive weight function is defined as a function of arc length along ξ=constant lines using the input grid, i.e., $w_a(s)$ from Equation 30.
3. The adaptive weight function is interpolated onto the hyperbolic tangent distribution and combined with the geometric weight function using
$$w = \alpha w_a \oplus \left(\frac{w_g}{\max(w_g^\eta)} \right) \tag{40}$$
where $\max$ indicates a maximum value on the given ξ line and α is the adaptation scale factor as discussed above.

4. An approach similar to that employed for the surface distribution (see Equation 6) is used to generate a new arc length distribution from

$$\int_0^{s_j} w(s)\,ds = \frac{j-1}{J-1}\int_0^{s_{OB}} w(s)\,ds \tag{41}$$

where J is the number of points and s_{OB} is the distance to the outer boundary. Now, using the distance distribution s_j, δ^j is computed using

$$\delta^j = s_{j+1} - s_j \tag{42}$$

Once the distance distribution is defined, the grid is generated as before with the control functions defined by Equation 32.

4.6.4 Solution-Adaptive Grid - Refinement

Since the criteria used for insertion and deletion of lines for generalized grid generation is arbitrary, information about the solution can be used to determine whether points are inserted or deleted. This criterion results in a grid generation algorithm that uses anisotropic refinement. In the particular grid generation algorithm employed here, only ξ lines can be inserted or deleted so that the structure within each layer may be maintained. Therefore, a directional weight function that recognizes variations in relevant flow field quantities in the xi direction is needed.

The basis of the grid refinement approach employed here is that if the magnitude of the directional weight function in the local ξ direction exceeds a user-specified parameter, that is,

$$w_r^\xi \geq \gamma_1 \tag{43}$$

where w_r^ξ is a directional refinement weight function (defined below) and γ_1 is defined by *adaptiveInsertionCriterion_nl* in the namelist **BLOCK_PARAMETERS** in the file associated with *block_parm_file*, insertion occurs. If this directional weight function is less than a user-specified parameter,

$$w_r^\xi \leq \gamma_2 \tag{44}$$

γ_2 is defined by *adaptiveDeletionCriterion_nl* in the namelist **BLOCK_PARAMETERS**, deletion occurs.

As in the case of geometric deletion or insertion, a stencil is defined indicating whether each point should continue in the next layer. In many instances, there will be conflicts between the stencil defined based on geometric considerations and the refinement stencils. Table 8, shown below, is used to resolve these differences.

Table 8: Stencil conflict resolution - Note: d indicates deletion, - indicates no action, and i indicates insertion

adaptive	geometric	result
d	d	d
d	-	d
d	i	-
-	d	d
-	-	-
-	i	i
i	d	-
i	-	i
i	i	i

The directional refinement function w_r^ξ for a general scalar f then defined following the procedure used for the solution adaptive weight function w_a:

1. The gradient of f in physical (x, y) space ∇f is computed.

2. Compute the magnitude of the gradient of f in the ξ direction using

$$|\nabla_\xi f| = |\nabla f \cdot \hat{\mathbf{e}}_\xi| \tag{45}$$

where $\hat{\mathbf{e}}_\xi$ is a unit vector in the direction normal to the local ξ=constant direction. Note that this is **not** the same as $\partial f / \partial \xi$.

3. $|\nabla_\xi f|$ is "clipped" based on a specified constant using

$$|\nabla_\xi f_c| = \min\left(\frac{|\nabla_\xi f|}{f_{max}}, 1\right) \tag{46}$$

In this way, large local values are not allowed to "swamp" the field. The maximum gradient is limited to a magnitude of $f_{max} = 0.2$.

4. The "clipped" gradient is then smoothed $\left(\overline{|\nabla_\xi f_c|}\right)$ and normalized to yield the directional refinement weight function

$$w_r^\xi = \frac{\alpha\overline{|\nabla_\xi f_c|}}{\alpha\overline{|\nabla_\xi f_c|} + \left(\frac{w_g^\xi}{\max(w_g^\xi)}\right)} \tag{47}$$

where max indicates an extremum over the entire domain. As defined, refinement will occur in regions where the grid spacing is already small. Further, the weight function is restricted to the range $0 \leq w_r^\xi \leq 1$, i.e., it is nonnegative and bounded by unity.

In this case, the scalar f is the velocity magnitude. The only user-specified variable in this formulation is the clipping value. We have found the value employed here to be effective for all problems considered to date.

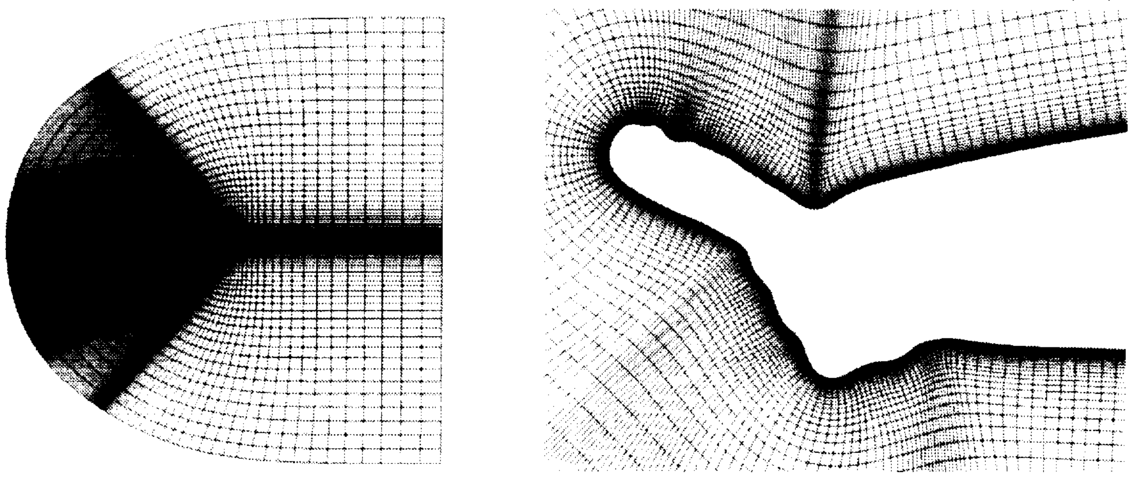
4.7 Demonstration of ICE_GRID_2D

ICEG2D (v2.0) was used to generate several representative grids. In all cases, the grids were generated using the surface definitions shown in the Section 3. **Again, it should be emphasized that both the single- and double-block grids are generated automatically.** For most cases, no user intervention is required.

Figure 8.a shows the full computational domain for a single-block grid generated around the 204EXP ice shape [10]. The grid dimensions are 517x101. Figure 8.b shows the region of the grid near the airfoil leading edge. As can be observed in the figures, the grid distribution is relatively smooth on the surface and in the interior of the domain. Further, the grids show the coalescence of lines emanating from concave regions of the airfoil surface that is characteristic of all marching methods. Figures 9.a and 9.b shows a double-block grid generated using the same surface distribution for the 204EXP ice shape and demonstrate the capability of ICEG2D (v2.0) to generate multi-block grids automatically. The dimensions of the inner and outer blocks are 517x71 and 259x21 respectively.

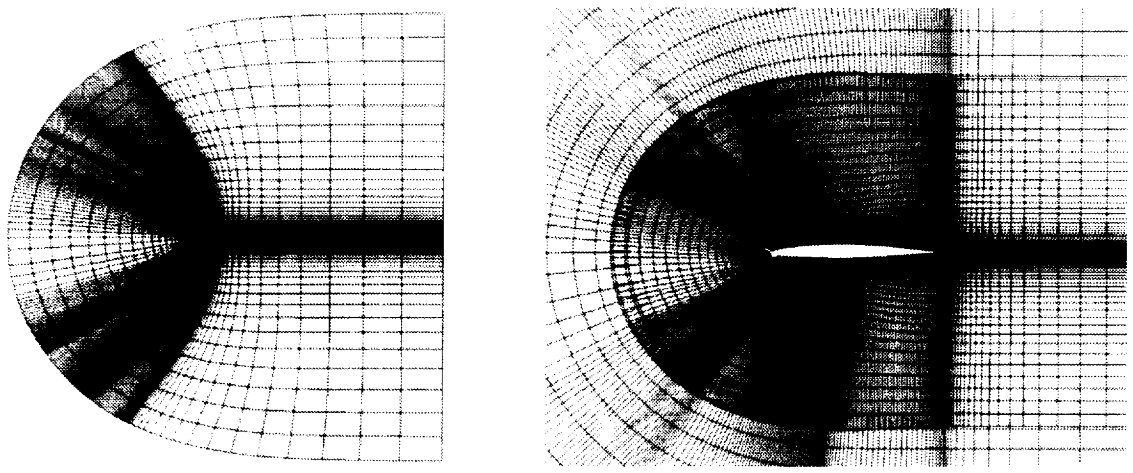
Figure 10.a shows a region of a single-block grid generated for the 904EXP ice shape [10]. The right edge of this figure corresponds to a distance of approximately 15 percent of the chord aft of the leading edge of the airfoil. Although this case in considerably more challenging due to the ice accretion on the lower surface, the grid appears to be of high quality. The coalescence of lines is much more evident than in Figures 8 and 9 because of the severity of the concave regions. Figure 10.b shows a close up of the grid near a strongly concave region of the ice. Again a smooth variation in the grid is evident in the figure. The grid dimensions in this case are 829x101. It should be noted that there is significant skewness that occurs in less than 0.1 percent of the cells. This skewness is due to the strongly concave regions on the airfoil surface and is unavoidable.

The next examples demonstrate the generalized grid generation capability of ICEG2D (v2.0). Figure 11 shows a single-block, generalized grid generated around the 204LEW ice shape [10]. The same surface distribution was employed as above with *deleteNode_nl* and *insertNode_nl* both set to *.true.* and the association **topo** defined to be *semi(structured)*. As can been seen from the figure, a smooth variation in cell size is obtained and the clustering of grid lines normally observed in marching grids is no longer apparent. This is because of the manner in which the deletion/insertion algorithm is defined, i.e., a cell is marked for deletion if it has an aspect ratio less than unity where aspect ratio is defined to be the ratio of the width to the height. The initial curve consisted of 517 points. Through deletion and insertion, a total of 45 points define the outer boundary. This results in a direct savings of approximately 35 percent in the number of nodes. It should be noted that more than 97 percent of the cells have



(a) full grid (b) region near airfoil leading edge

Figure 8: 204LEW iced airfoil - single-block grid

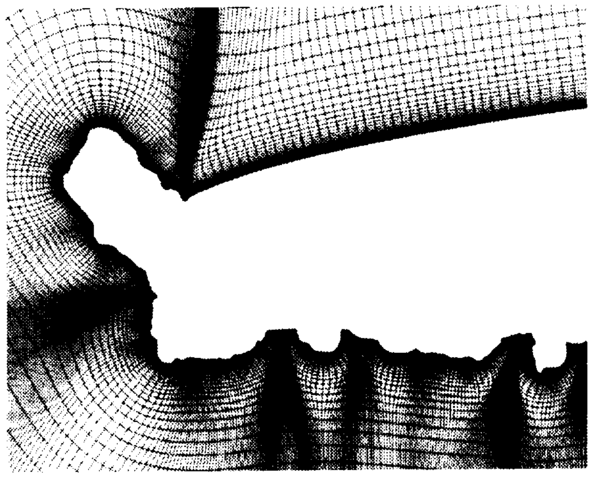


(a) full grid (b) region showing inner block

Figure 9: 204LEW iced airfoil- double-block grid

four edges. Figure 12 shows a single-block, generalized grid generated around the 904EXP ice shape [10]. Again, the same surface distribution was employed as above with *deleteNode_nl* and *insertNode_nl* both set to *.true.* and the association **topo** defined to be *semi(structured)*. The initial curve consisted of 829 points. Through deletion and insertion, a total of 44 points define the outer boundary. This results in a direct savings of approximately 43 percent in the number of nodes. In this case, because of the larger number of points in the ξ direction, i.e., the direction in which deletion occurs, a greater percentage in savings in the number of nodes occurs. Again, more than 97 percent of the cells have four edges.

The next demonstration is of the capability of ICEG2D (v2.0) to generate solution adaptive, structured grids through redistribution. Figures 13.a, 13.b, and 13.c show the initial grid, the weight function, and the resulting solution adaptive grid for the 623LEW airfoil [10] at an angle of attack of six degrees. The weight function reaches its maximum value (lighter colors) in the region containing the separated shear layer. The solution adaptive grid shows a distinct clustering of grid lines in the region where the weight function attains its maximum value.


(a) region near airfoil leading edge

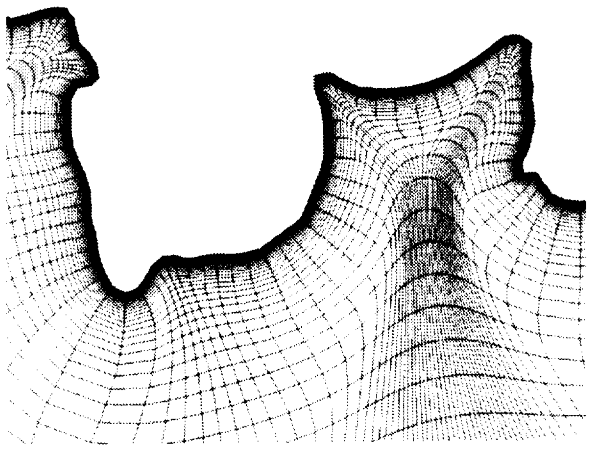
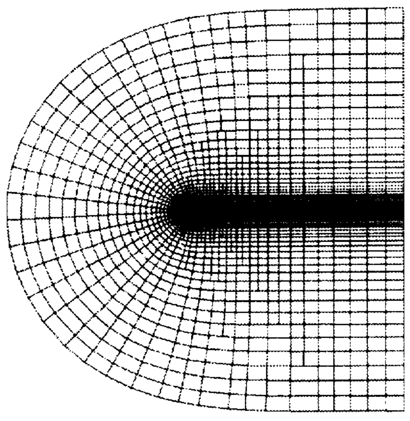

(b) detail of concave region

Figure 10: 904EXP iced airfoil - single-block grid


(a) full grid

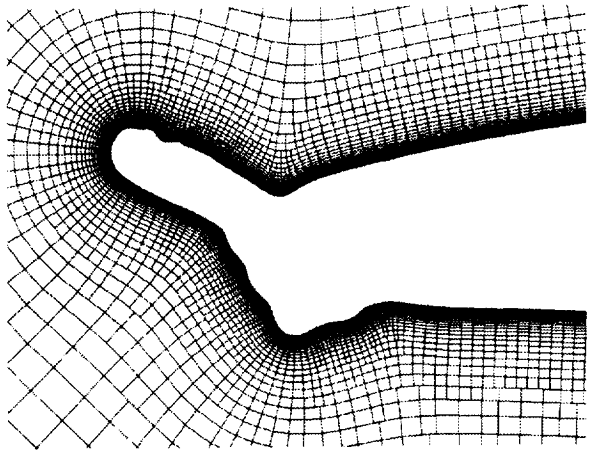

(b) region near airfoil leading edge

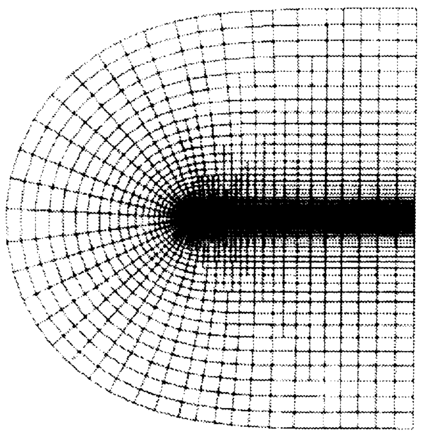
Figure 11: 204LEW iced airfoil - generalized grid

The final demonstration is for the solution adaptive, generalized grid with refinement and redistribution. Figures 14.a, 14.b, and 14.c show the initial grid, the weight function, and the resulting solution adaptive grid for the 623LEW airfoil, again at an angle of attack of six degrees. The weight function is the same as Figure 13.b. The quality of the grid in the region aft of the horn is increased dramatically through the use of the solution adaptive grid.

4.8 Output from ICE_GRID_2D

The output from the ICE_GRID_2D module consists of a file containing the grid and informational messages to *stdout*. The grid is output to a file in FAST, multizone, unformatted, two-dimensional, grid format [17]. The grid may be viewed using the **display grid** directive from ICEG2D (v2.0). The output file name is associated with *grid_file* with the default being *case_name.grd*. A sample of the information printed to *stdout* for a structured grid is shown below:

```
ICEG2D> gen grid
        Generating grid using:
        Distribution file     - 623exp.dst
        Grid topology         - structured
```


(a) full grid

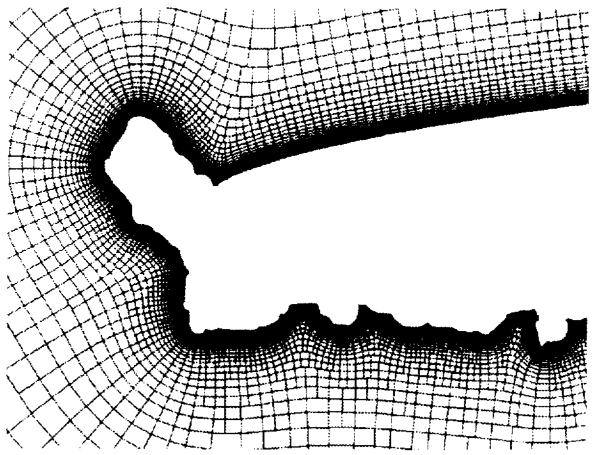

(b) region near airfoil leading edge

Figure 12: 904EXP iced airfoil - generalized grid

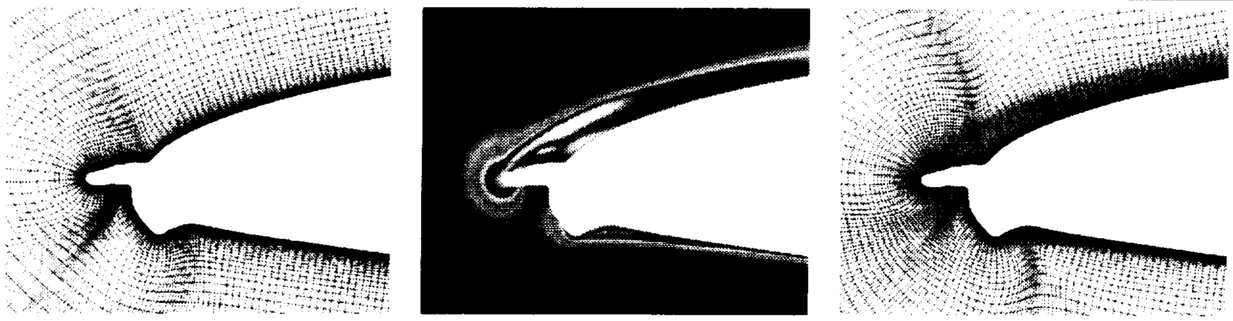

(a) initial grid (b) weight function (c) adaptive grid

Figure 13: 623LEW iced airfoil - solution adaptive grid

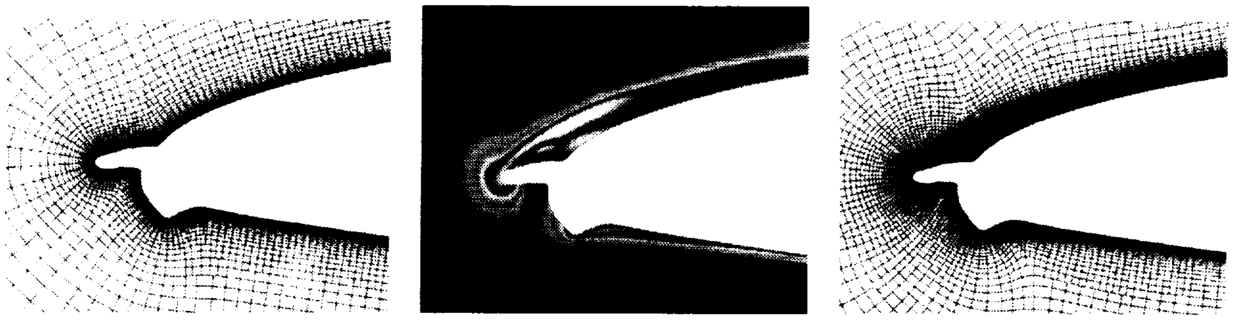

(a) initial grid (b) weight function (c) adaptive grid

Figure 14: 623LEW iced airfoil - solution adaptive generalized grid

```
Grid adaptivity       - none
Grid parameter   file - /ICEG2D/parameter_files/grid_gen.ini
Block definition file - /ICEG2D/parameter_files/single_block.ini
```

```
      Grid output file      - 623exp.grd


      Begin grid generation.


***** Grid Parameter Data *****

      itMax                    =  20
      nOrder                   =  2
      convgTol                 =  0.10000E-06
      omega                    =  1.00
      addedDissipation         =  1.00
      iWake                    =  40
      distWake                 =  15.00


***** Block Parameter Data *****

      nBlocks                  =  1

      BLOCK#1:
      distFirstPoint LE        =  0.100000E-05
      distFirstPoint TE        =  0.100000E-05
      distFirstPoint DSB       =  0.100000E-05
      xTransLower              =  0.500000E+00
      xTransUpper              =  0.500000E+00
      distOuterBoundary        =  15.00
      # of Layers              =  100
      structured grid          =  T
      redistribution allowed =  F


***** Information for grid: 623exp - Single block, C-type grid

      IMAX =   701, # of Layers =   100

      CONVERGENCE HISTORY

      ***** Layer #  1 *****
          IT         RMS      IRMAX       RMAX
           1    0.59573E-07   518    0.97969E-06

      ***** Layer #  2 *****
          IT         RMS      IRMAX       RMAX
           1    0.69448E-07   518    0.11419E-05

      ***** Layer #  3 *****
          IT         RMS      IRMAX       RMAX
           1    0.80949E-07   518    0.13306E-05

      .
      .
      .
```

```
        ***** Layer #100 *****
            IT          RMS       IRMAX       RMAX
             1     0.70572E-01   593     0.11666E+00
             2     0.34931E-01   593     0.57597E-01
             3     0.17485E-01   592     0.28797E-01
             4     0.88194E-02   591     0.14516E-01
             5     0.44739E-02   591     0.73607E-02
             6     0.22799E-02   590     0.37498E-02
             7     0.11664E-02   590     0.19177E-02
             8     0.59883E-03   589     0.98431E-03


        ***** Grid Quality Check - Block #1 *****

              All cells convex!

              Maximum deviation = 49.06 deg

              Deviation from orthgonality - angle distribution %

                     dev < 10.0 -   71.0818
              10.0 < dev < 20.0 -   26.8350
              20.0 < dev < 30.0 -    1.7432
              30.0 < dev < 40.0 -    0.2789
              40.0 < dev < 50.0 -    0.0611
                     dev > 50.0 -    0.0000


******* TRAP STATS FOR PID 93980 *********
     UNDERFLOW 1992
     OVERFLOW  0
     DIVZERO   0
     INVALID   0
     INT OVRFL 0

   bad sig count = 0
   bad code count = 0
******* END TRAP STATS FOR PID 93980 *****


        Grid generation complete!
```

An echo of the namelist **GRID_PARAMETERS** and an echo of the namelist **BLOCK_PARAMETERS** follow an inventory of the relevant associated files. The grid dimension is then followed by a grid informational line. The convergence history is output where the layer being smoothed is indicated, IT is the smoothing iteration number for that layer, RMS is the RMS in the change in point position for the layer, and IRMAX and RMAX are the location of the maximum error and the maximum error respectively. In the case of a double block grid, the **BLOCK_PARAMETERS** data and convergence history are repeated. Grid quality checks are made to ensure that a viable grid has been generated and are included in the output. Specifically, the convexity of each cell is checked to ensure that no grid line crossings occur. If a nonconvex cell is found, execution in the batch mode is terminated. Further, the deviation from orthogonality of the cell sides (ξ=constant lines) is evaluated to check for cell skewness. Currently, no action is taken except to indicate the minimum angle and display, using a list, the percentage of cell angles within certain ranges.

Below is sample output for a semistructured grid:

```
ICEG2D> gen grid
        Generating grid using:
        Distribution file      - 623exp.dst
        Grid topology          - semistructured
        Grid adaptivity        - none
        Grid parameter   file - /ICEG2D/parameter_files/grid_gen.ini
        Block definition file - /ICEG2D/parameter_files/single_block.ini
        Grid output file       - 623exp.grd


        Begin grid generation.


  ***** Grid Parameter Data *****

        itMax                    =  20
        nOrder                   =  2
        convgTol                 =  0.10000E-06
        omega                    =  1.00
        addedDissipation         =  1.00
        iWake                    =  40
        distWake                 =  15.00


  ***** Block Parameter Data *****

        nBlocks                  =  1

        BLOCK#1:
        distFirstPoint LE        =  0.100000E-05
        distFirstPoint TE        =  0.100000E-05
        distFirstPoint DSB       =  0.100000E-05
        xTransLower              =  0.500000E+00
        xTransUpper              =  0.500000E+00
        distOuterBoundary        =  15.00
        # of Layers              =  100
        structured grid          =  F
        redistribution allowed =  F
        refinement allowed       =  F
        deletion allowed         =  T
        deletion criterion       =  1.00
        insertion allowed        =  T
        insertion criterion      =  0.50


  ***** Information for grid: 623exp - Single block, C-type grid

        IMAX =   701, # of Layers =   100

        CONVERGENCE HISTORY

        ***** Layer #  1 *****
            IT         RMS      IRMAX       RMAX
             1    0.59573E-07   518    0.97969E-06
```

```
IMAX for current layer = 701
# nodes deleted =    0, # nodes inserted =    0
IMAX for next layer = 701

***** Layer #  2 *****
    IT         RMS      IRMAX       RMAX
     1    0.69448E-07  518    0.11419E-05
IMAX for current layer = 701
# nodes deleted =    0, # nodes inserted =    0
IMAX for next layer = 701

***** Layer #  3 *****
    IT         RMS      IRMAX       RMAX
     1    0.80949E-07  518    0.13306E-05
IMAX for current layer = 701
# nodes deleted =    0, # nodes inserted =    0
IMAX for next layer = 701

.
.
.

***** Layer #100 *****
    IT         RMS      IRMAX       RMAX
     1    0.26141E-01   22    0.55448E-01
     2    0.12190E-01   22    0.25526E-01
     3    0.57225E-02   22    0.11764E-01
     4    0.27031E-02   22    0.54248E-02
     5    0.12851E-02   22    0.25023E-02
     6    0.61510E-03   22    0.11545E-02
     7    0.29653E-03   22    0.53278E-03
     8    0.14399E-03   14    0.24770E-03


***** Cell Topology Statistics *****

Number of Nodes          =      42161
Number of Cells          =      41208
Number of Boundary Edges =        864

Percentage 3-sided cells =   0.000000
Percentage 4-sided cells =  97.558723
Percentage 5-sided cells =   2.356339
Percentage 6-sided cells =   0.082508
Percentage 7-sided cells =   0.002427


***** Grid Quality Check *****

      All cells convex!

      Maximum deviation = 49.06 deg

      Deviation from orthgonality - angle distribution %
```

```
                  dev < 10.0 -   90.0820
           10.0 < dev < 20.0 -    8.0166
           20.0 < dev < 30.0 -    1.5173
           30.0 < dev < 40.0 -    0.2967
           40.0 < dev < 50.0 -    0.0874
                  dev > 50.0 -    0.0000


******* TRAP STATS FOR PID 93984 *********
     UNDERFLOW 1952
     OVERFLOW  0
     DIVZERO   0
     INVALID   0
     INT OVRFL 0

   bad sig count = 0
   bad code count = 0
******* END TRAP STATS FOR PID 93984 *****


        Grid generation complete!
```

The primary differences are in the quantity of information that is output. Additional information regarding the geometric deletion/insertion criteria is output. Also, in each layer, information regarding the number of nodes inserted/deleted is output along with the number of points in the current and the number of points in the next layer. Cell topology statistics are output along with the grid quality measures. It should be noted that, for the generalized grid, a higher percentage of cells have less deviation from orthogonality than do cells for the structured grid. That is, a higher percentage of cells in the generalized grid are of higher quality, at least based on the quality measure of orthogonality. Also of interest is the fact that the percentage of cells in the lowest quality regime actually increases by approximately 40 percent. Because of the decrease in the number of cells in the generalized grid from 70000 to 41208, this percentage increase actually represents a net decrease of poor quality cells from 43 in the structured grid to 36 in the generalized grid. Since the skewness is due to the surface definition, the skewed cells are near the body and are not affected by the deletion/insertion algorithm as much as cells farther out in the field.

The structured grid output file (associated with *grid_file*) is written in formatted, two-dimensional, multiblock FAST format and can be plotted using any visualization software package that can read this format or the module ICE_PLOT_2D described in Section 7. The generalized grid output file can only be plotted using the module ICE_PLOT_2D.

5. NPARC Specific Modules - NPARC_INPUT, NPARC_INITIAL, and NPARC_RESTART

The modules NPARC_INITIAL and NPARC_INPUT generate the files necessary to execute nparc2ds. NPARC_RESTART interpolates a solution obtained on one grid onto another grid. NPARC_INPUT takes as input the problem definition information and grid information to generate an NPARC namelist input file with the internal association *case_name.inp*. NPARC_INITIAL takes problem definition information and the grid to generate an NPARC initial file with the internal association *case_name.ini*. Once these two files are defined, the ICEG2D (v2.0) directive **gen(erate) solution** may be used to start an NPARC solution run as a background process. The user is referred to [1] for information regarding NPARC output, etc. Both modules assume that a C-type grid is used with the positive direction of the streamwise index taken to be clockwise. NPARC_RESTART is used in cases where a solution adaptive grid has been generated and the computed solution needs to be interpolated onto the new, adaptive grid. This step improves the convergence of the solution.

NPARC_INPUT, NPARC_INITIAL, and NPARC_RESTART are written in FORTRAN90 with C driver routines. ICEG2D (v2.0) passes four file names as command line arguments to NPARC_INPUT: the NPARC information file, the grid file, the file containing the namelist **GRID_PARAMETERS**, and the output file for the NPARC namelist. Similarly, ICEG2D (v2.0) passes three file names as command line arguments to NPARC_INITIAL: the NPARC information file, the grid file, and the initial file. ICEG2D (v2.0) passes three file names as command line arguments to NPARC_RESTART: the NPARC restart file name, the new grid, and the new restart file name. In the sections that follow, the input and output for each module are described.

5.1 Input for NPARC_INPUT, NPARC_INITIAL, and NPARC_RESTART

Common to the input for both NPARC_INPUT and NPARC_INITIAL is the problem information file. The problem information file is internally associated with *info_file* and consists of five namelists: **TITLES**, **INPUTS**, **TURBIN**, **SEQDT**, and **BLOCK**. These namelists are shown in Tables 9-13 These namelists and their variables are identical to those defined in the NPARC User's Manual [1] with the exception of the variable *nclcd* in namelist **INPUTS**. This variable was added to specify the frequency of the lift, drag, and moment calculation added by Chung [10]. For definitions of the remaining variables, please refer to [1]. The module NPARC_INPUT is executed using the ICEG2D (v2.0) directive **gen(erate) input**. Similarly, the module NPARC_INITIAL is executed using the ICEG2D (v2.0) directive **gen(erate) initial**.

The file containing the grid is internally associated with *grid_file* and the file containing the namelist **GRID_PARAMETERS** is internally associated with *grid_parm_file*. The namelist **GRID_PARAMETERS** is needed because determination of the trailing edge point from the grid file requires the variable *iWake*.

5.2 Output from NPARC_INPUT

The output from NPARC_INPUT consists of a file containing five namelists: **TITLES**, **INPUTS**, **TURBIN**, **SEQDT**, and **BLOCK**. The namelist **INPUTS** is modified through the addition of *nclcd* (described previously) and *jtail1*, which corresponds to the index of the trailing edge point on the lower surface, and *jtail2*, which corresponds to the index of the trailing edge on the upper surface. Both were included as part of the lift, drag, and moment calculation [10]. The default output for NPARC_INPUT is internally associated with *case_name.inp* or an alternative name specified using the ICEG2D (v2.0) directive **def(ine) input** *input_file*. Output from NPARC_INPUT also includes information messages printed to *stdout*. Sample output is shown below:

```
ICEG2D> gen input
        Generating input file using:
        Info file           - NPARC_info.inf
        Grid file           - nlf0441.grd
        Grid parameter file - /ICEG2D/parameter_files/grid_gen.ini
        Input file          - nlf0441.inp

        Case Name:

        # Case: NLF-0441 Clean Airfoil
        # Run: 704, Pnt: 1816
        # Mach = 0.290, Re = 6.38x10e06, AOA = 3.2 deg (corrected)
# Single block grid
#

Grid file - nlf0441.grd
   Number of blocks =  1
      Block (1): IMAX, JMAX = 389, 101
Lower surface TE -  40
Upper surface TE - 350

NPARC input file generation complete!
```

The problem title and relevant information from the problem definition file follow the associated file inventory. Finally, grid information is followed by a message indicating successful generation of the input file.

Table 9: Namelist **TITLES** variables

Namelist variable	
title	5 lines of a title

5.3 Output from NPARC_INITIAL

The output from NPARC_INITIAL is an NPARC initial file (please refer to [1] for the specific format). The default internal file association is *case_name.ini* or an alternative name specified using the ICEG2D (v2.0) directive **def(ine) initial** *initial_file*. Output from NPARC_INITIAL also includes information messages printed to *stdout*. Sample output is shown below:

```
ICEG2D> gen initial
        Generating initial solution/grid file using:
        Problem info file - NPARC_info.inf
        Grid file         - nlf0441.grd
        Initial file      - nlf0441.ini

        Case Name:

        # Case: NLF-0441 Clean Airfoil
        # Run: 704, Pnt: 1816
        # Mach = 0.290, Re = 6.38x10e06, AOA = 3.2 deg (corrected)
        # Single block grid
        #

        Mach number      =  0.29
        Reynolds number =  0.63800E+07
        Angle of attack =  3.20

        Grid file - nlf0441.grd
           Number of blocks =  1
              Block (1): IMAX, JMAX = 389, 101

        NPARC initial file generation complete!
ICEG2D>
```

As usual, an inventory of the relevant file associations is printed first. This is followed by the title of the problem and information about the grid.

Table 10: Namelist **INPUTS** variables

Namelist variable	Default value
xmach	0.3
re	1.0×10^6
alpha	0.0
gamma	1.4
p0sd	0.71429
t0sd	1.0
pref	14.7
trefr	500.0
nc	0
nmax	100
dtcap	100.0
pcqmax	10.0
isolve	1
ivardt	2
irealt	0
realdt	5.0×10^{-6}
tstart	0.0
nsprt	10
np	0
iplot	1
ifxplt	0
ifxprt	0
l2plot	0
dis2	0.25
dis4	0.64
ispect	2
ifiltr	1
splend	0.5
imass	1
iaxisys	0
stopl2	1.0^{-12}
nskip	2
numdt	1
nclcd	100

5.4 Output from NPARC_RESTART

The output from NPARC_RESTART is an NPARC restart file (please refer to [1] for the specific format). The default internal file association is *case_name.rst* or an alternative name specified using the ICEG2D (v2.0) directive **def(ine) restart** *restart_file*. Output from NPARC_RESTART also includes information messages printed to *stdout*. Sample output is shown below:

```
ICEG2D> gen restart
        Generating restart solution/grid file using:
        Grid file          - 623exp.grd
```

Table 11: Namelist **TURBIN** variables

Namelist variable	Default value
imutur	2
nturb	2000
imutr2	15

Table 12: Namelist **SEQDT** variables

Namelist variable	Default values (assuming *numdt*=2)
dtseq(1)	1.0
iterdt(1)	2000
dtseq(2)	0.7
iterdt(2)	68000

Table 13: Namelist **BLOCK** variables

Namelist variable	Default values
invisc	1
lamin	1

```
Restart file      - fort.4
Initial file      - 623exp.ini

Output grid file - 623exp.grd
   Number of blocks =  1
      Block (1): IMAX, JMAX = 701, 101

 NC1 =     10

Input restart file - fort.4.old
   Number of blocks =  1
      Block (1): IMAX, JMAX = 701, 101

NPARC restart file generation complete!
```

As usual, an inventory of the relevant file associations is printed first followed by relevant informational messages.

On occasion, the interpolation routine in NPARC_RESTART will not be able to locate points in the old grid that surround the point in question in the new grid. This can occur when the new grid extends beyond the domain of the old grid. In this case, an informational message is printed and the value at the point in the old grid closest to the new grid point is used. Typically, this discrepancy is not important since it occurs near the outer boundaries where the variation in flow field is small.

6. HYBFL2D Specific Modules - HYBFL2D_INPUT

Because of problems associated with HYBFL2D, a three-dimensional version of the code is actually used here. A preprocessor converts the two-dimensional generalized grid into a three-dimensional grid by extrusion. The process occurs in a seamless manner that is transparent to the user. Efforts are currently under way to rectify the problems with the two-dimensional code. The turbulence model currently available in HYBFL2D is the Spalart-Allmaras model [22].

The module HYBFL2D_INPUT generates the input file needed to execute HYBFL2D. HYBFL2D_INPUT takes as input problem definition information and grid information to generate an HYBFL2D namelist input file with the internal association *case_name.inp*. Other preprocessing functions are performed upon execution of the the HYBFL2D solver. No explicit initial file need be defined.

Once the input file is defined, the ICEG2D (v2.0) directive **gen(erate) solution** may be used to execute a HYBFL2D solution as a background process. A user manual for HYBFL2D is being developed and will be distributed with future releases of the software.

HYBFL2D_INPUT is written in FORTRAN90 with C driver routines. ICEG2D (v2.0)passes two file names as command line arguments to HYBFL2D_INPUT: the HYBFL2D information file and the output file for the HYBFL2D namelist. In the sections that follow, the input and output for each module are described. Legacy preprocessing codes used by HYBFL2D are written in C, FORTRAN77, and FORTRAN90.

6.1 Input for HYBFL2D_INPUT

The HYBFL2D information file is internally associated with *info_file* and consists of one namelist **INPUTS** which is shown in Tables 14. The module HYBFL2D_INPUT is executed using the ICEG2D (v2.0) directive **gen(erate) input**.

Table 14: Namelist **INPUTS** variables for HYBFL2D

Namelist variable	Default value
machNumber	0.2
angleOfAttack	0.3
reynoldsNumber	1.0×10^6
CFLNumber	5.0
restart	'no'
numberOfIterations	10000
orderOfScheme	2
typeOfScheme	'i'
typeOfJacobian	'd'
typeOfSimulation	'n'
referenceTemperature	300.0
numberOfNewtonIterations	1
limiterThreshold	2.0
typeOfViscous	't'
typeOfLimiter	'b'
typeOfWeight	'l'
restartFrequency	100
timeStepFrequency	100

6.2 Output from HYBFL2D_INPUT

The output from HYBFL2D_INPUT consists of a file containing the namelist **INPUTS**. The default output for HYBFL2D_INPUT is internally associated with *case_name.inp* or an alternative name specified using the ICEG2D (v2.0) directive **def(ine) input** *input_file*. Output from HYBFL2D_INPUT also includes information messages printed to *stdout*. Sample output is shown below:

```
ICEG2D> gen input
        Generating input file using:
        Info file           - HYBFL2D_info.inf
        Input file          - 623exp.inp

        ***** HYBFL2D Input Data *****

              machNumber               =    0.28900
              angleOfAttack            =    9.20000
              referenceTemperature     =  302.60001
              CFLNumber                =    5.00000
              restart                  =    no
              numberOfIterations       =   20000
              orderOfScheme            =    2
              typeOfScheme             =    i
              numberOfNewtonIterations =    1
              typeOfJacobian           =    d
              typeOfSimulation         =    n
              limiterThreshold         =    2.00000
              typeOfViscous            =    t
              typeOfLimiter            =    b
              typeOfWeight             =    1
              reynoldsNumber           =    0.64000E+07
              restartFrequency         =    100
              timeStepFrequency        =    100


        HYBFL2D input file generation complete!
```

The problem title and relevant information from the problem definition file follow the associated file inventory. Finally, grid information is followed by a message indicating successful generation of the input file.

6.3 Output from HYBFL2D

Relevant output from the flow solver HYBFL2D is contained within the following files:

1. **hybrid.out** consists of a convergence history as well as lift, drag, and moment data.
2. **grid.dat** contains a triangulated grid written in unstructured, formatted FAST format that can be read by many visualization packages.
3. **soln.dat** contains the flow field solution written in unstructured, formatted FAST format that can be read by many visualization packages.
4. **Cp_dist.dat** contains pressure coefficient distribution on the airfoil surface.
5. **Function_401.dat** contains the turbulent viscosities computed for the Sparlart-Allmaras turbulence model.

Other files created during the solution process may be deleted.

7. Surface Distribution and Grid Plotting Module - ICE_PLOT_2D

The module ICE_PLOT_2D generates plots of the surface point distribution and the grid for both structured and generalized grid topologies. The module is executed by issuing the directive **display distribution** for the surface distribution and **display grid** for either of the grid topologies. ICE_PLOT_2D uses appropriate input routines based on the association *grid_topology*. ICE_PLOT_2D uses the X11 libraries and replaces calls to FAST in ICEG2D (v1.0) [2].

ICE_PLOT_2D has several interactive commands that are available to the user which are listed below:

- z (zoom) - Select a region to enlarge by creating a box with the mouse.
- t (toggle) - Toggle between the current view and the previous view.
- s (start) - Return to starting view.
- u (up), d (down) , l (left) , r (right) - Shift image up, down, left and right.
- q (quit) - Exit.

8. Evaluation of ICEG2D (v2.0)

While simple geometrical checks can be performed to ensure that all cell areas are positive, excessive skewness does not occur, etc., the usefulness of a grid for a specific calculation can ultimately be assessed only by the accuracy of results predicted using the grid. In this case, computations are made using NPARC and HYBFL2D for iced airfoil cases for which experimental data exists [10]. It is well understood and appreciated that many factors other than grid quality influence the numerical solution. It is not the intention here to perform an exhaustive validation of the NPARC and HYBFL2D codes and their various turbulence models but only to demonstrate that quality calculations can be made using the grids automatically generated using ICEG2D (v2.0).

8.1 Comparison of Computed Results with Experimental Data

The available experimental data is for the NLF0441 airfoil section. The data were obtained for the clean airfoil shape and several iced airfoil shapes. The Mach number for each case was approximately 0.3 and the Reynolds number was approximately 6.4×10^6. The range of angles of attack varied for each configuration but generally ranged from -4 degrees to just beyond stall. Force and moment data were obtained along with pressure coefficient data. In all cases below, the numerical solution was generated using the Mach and Reynolds numbers reported for each case. Additionally, the corrected angle of attack, as reported with the experimental data, was used. All force data were obtained by integration of the surface pressures. Default values of the various parameters were used except where noted.

8.2 The Clean Airfoil - NLF0441

The first case considered here is a clean NLF0441 airfoil. Figure 15 shows the single-block, 389 x 101 grid used for the calculations. As expected, points are clustered near the leading edge of the airfoil due to the surface curvature.

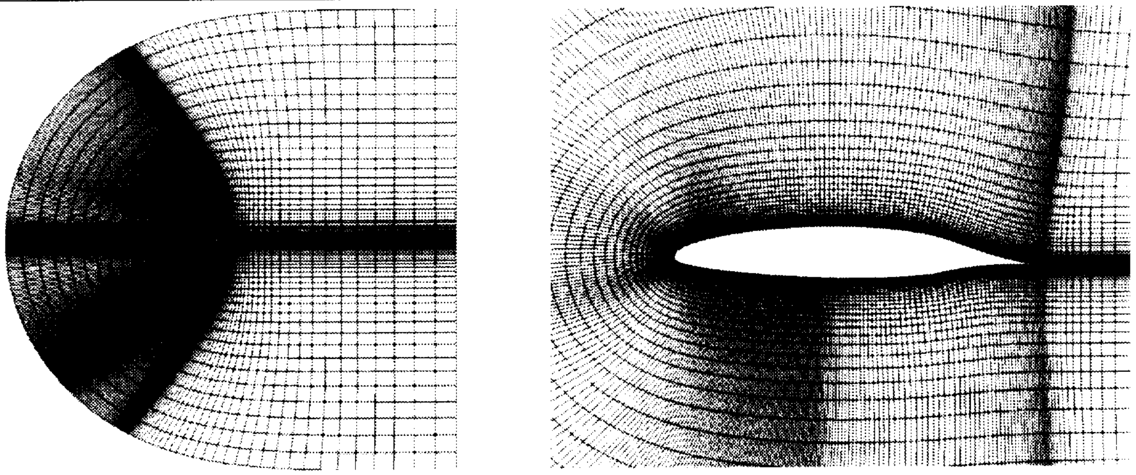
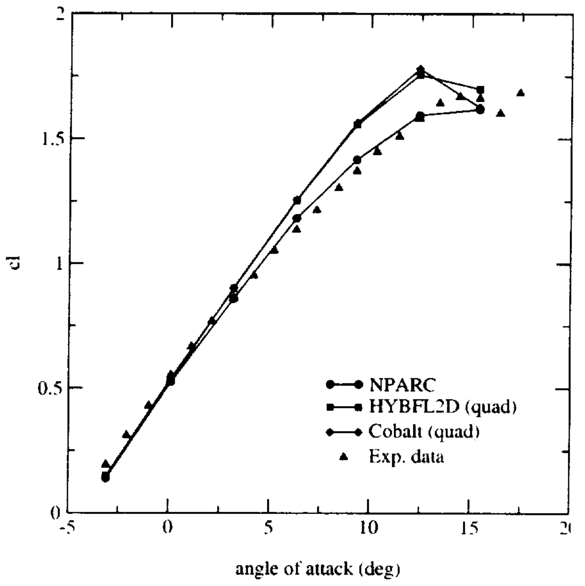

(a) full grid (b) region near airfoil

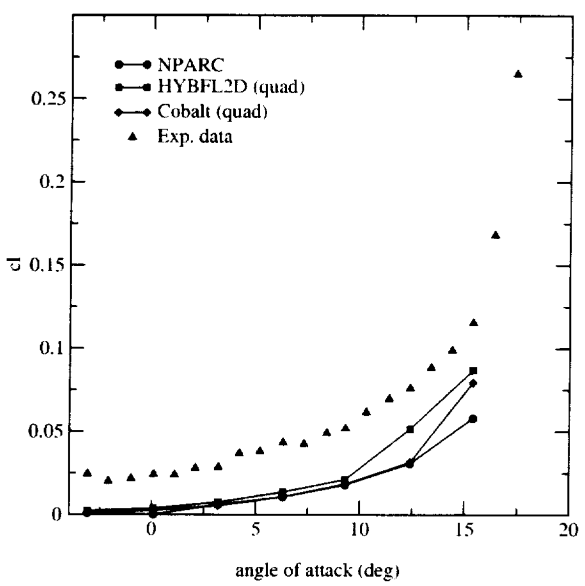
Figure 15: NLF0441 airfoil - structured grid

This structured, all quadrilateral grid was employed for calculations performed using NPARC, HYBFL2D, and the generalized grid flow solver $COBALT_{60}$ [23]. Figure 16 shows comparisons of the lift and drag coefficients predicted using the three codes with experimental data. With regard to the lift coefficient (Figure 16.a), the results predicted using NPARC show excellent agreement with the experimental data. The results predicted using HYBFL2D and $COBALT_{60}$ show good agreement with each other but agree less well with the NPARC predictions and the experimental data. In particular, the lift curve slope is predicted incorrectly and, correspondingly, the lift values are over-predicted at higher angles of attack. In all cases, however, the onset of stall is well predicted. The results for the drag coefficient (Figure 16.b) show relatively good agreement between the NPARC, HYBFL2D, and $COBALT_{60}$ predictions at lower angles of attack while the results diverge as angle of attack is increased.

The discrepancy in predicted lift coefficients can be explained by examining the surface pressure distributions predicted by each flow solver. Figure 17 shows plots of the surface pressure distribution predicted by NPARC and

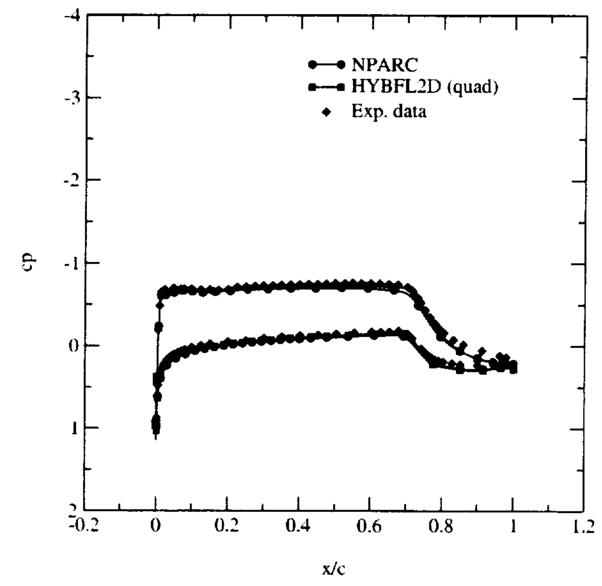


(a) lift coefficient

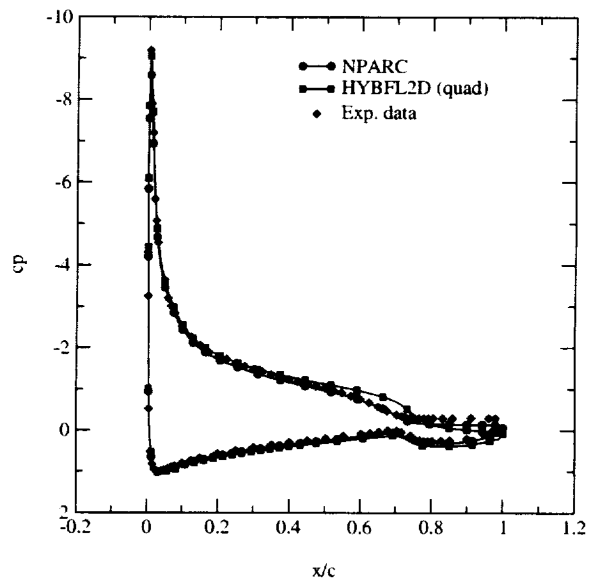


(b) drag coefficient

Figure 16: NLF0441 airfoil lift and drag - NPARC and HYBFL2D solutions



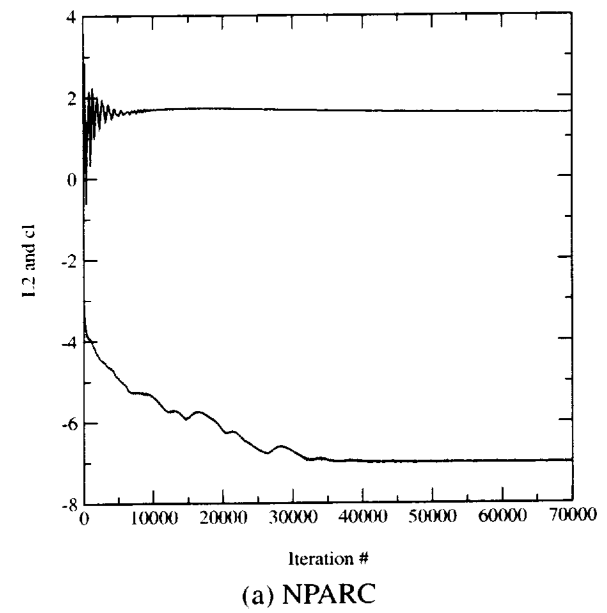
(a) $\alpha = 0°$



(b) $\alpha = 12°$

Figure 17: NLF0041 airfoil pressure coefficient distributions - NPARC and HYBFL2D solutions

there is very good agreement between the NPARC solution and the experimentally determined pressures. There is also very good agreement between the HYBFL2D predictions and the experimental data at lower angles of attack. At higher angles of attack, the HYBFL2D pressures displayed a notable "bulge" near the 60-70 percent chord region which explains the differences between the predicted lift values and experimental data. Finally, Figure 18 shows a



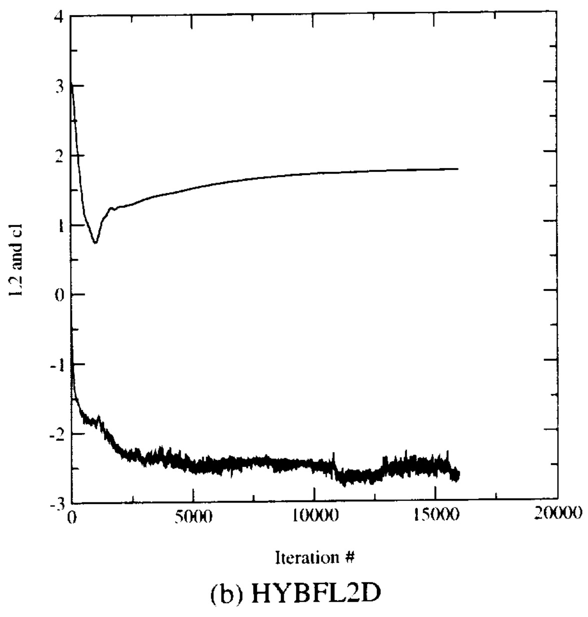


Figure 18: NLF0441 airfoil convergence histories - NPARC and HYBFL2D solutions

comparison of convergence histories for NPARC and HYBFL2D for the $\alpha = 2°$ case. The L_2 norm of the residual is shown along with the lift coefficient. Both are reasonably well behaved. Although not shown here, the HYBFL2D calculation was continued an additional 20000 iterations with no significant change in the data. Further, calculations performed by HYBFL2D using the generalized grid with geometric insertion and deletion showed no significant differences with the all quadrilateral grid calculations and are not included. These differences are currently under investigation.

It should be noted that no comparisons regarding relative efficiencies of the two codes should be made based on Figure 18. One iteration of HYBFL2D and one iteration of NPARC are, in no way, equivalent. Additionally, the calculation was not "optimized" in either case. A future task is to perform a comparison of the relative efficiencies of the two codes for iced airfoil calculations.

8.3 Case 623EXP - NLF0041 with Ice Accretion

We now consider a case with a significant ice accretion on the leading edge of the airfoil. Figure 19 shows the the 601 x 101 single-block structured grid automatically generated for the ice shape defined as case 623EXP. The grid shown in the figure was used for the fixed grid calculations. The directive **def surface** *rough* directive was used for the surface point distribution. This case is particularly challenging because of the ice growths on the lower surface of the airfoil that effectively remove points from the horn region. Additionally, there is a significant separated region aft of the horn.

Flow solutions were obtained for the range of angles of attack $-3° \leq \alpha \leq 9°$ in 3° increments. The flight conditions again correspond to the conditions at which experimental data has been obtained [10] at a Mach number of approximately 0.3 and a Reynolds number of 6.5×10^6.

8.3.1 *Structured Grid Solutions - Fixed and Solution-Adaptive Grids*

Fixed and solution adaptive grid solutions were obtained for structured grids using flow solver NPARC. The solution adaptive grid solutions were generated using one of the scripts discussed in Section 9. The solution was started on the automatically generated grids. The solution was allowed to proceed for 25000 iterations and a grid adaptation cycle was performed. This cycle included an interpolation of the solution to the new grid to improve convergence. Then, the solution was run 10000 iterations followed by another adaptation cycle. This step was then repeated yielding a total of three adaptation cycles. The solver was run for 25000 more iterations to complete the solution. Again, this is accomplished automatically using the ICEG2D (v2.0) default values. No attempt was made to optimize this sequence.

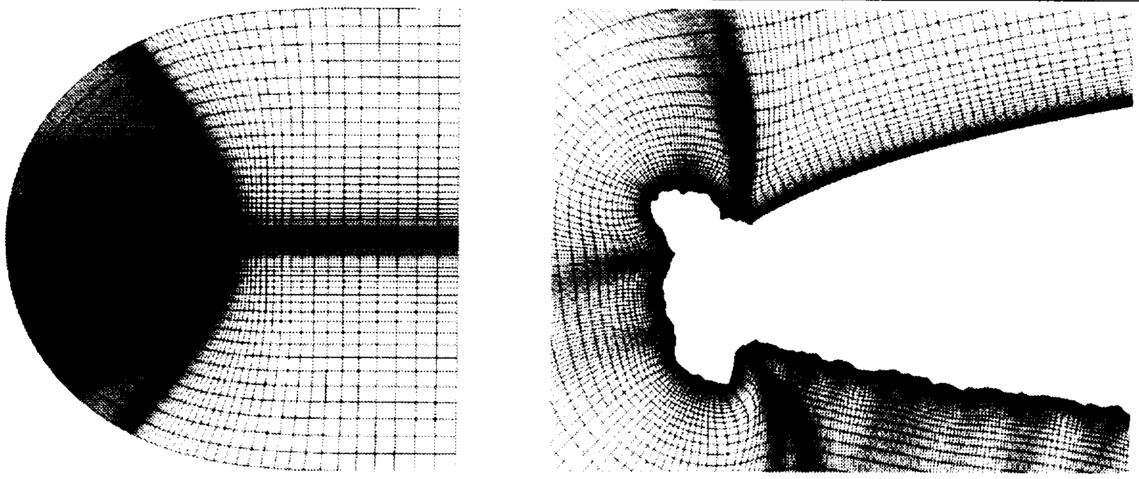


Figure 19: 623EXP iced airfoil - structured quadrilateral grid

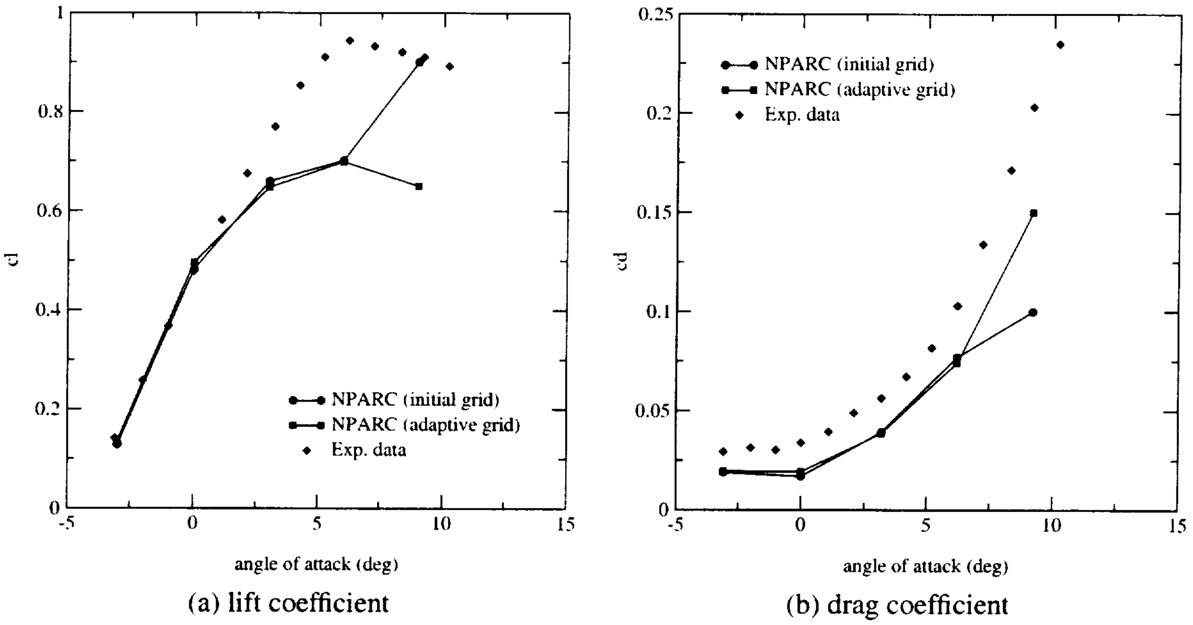


Figure 20: 623EXP iced airfoil lift and drag - initial and adaptive grids

Based on the results, the strategy employed here seems to be fairly conservative and a more aggressive approach may yield some benefits in terms of efficiency. It should be noted that the solution adaptive grid generation adds very little overhead to the calculation. The time needed to perform tasks associated with the adaptive grid generation process is measured in seconds while the flow solution is measure in tens of hours on a typical workstation.

Figure 20 shows a comparison between lift and drag coefficients predicted by NPARC using fixed and solution-adaptive grids and experimental data. In all cases, the agreement is good at lower angles of attack and degrades as angle of attack is increased. It should be noted that the apparent improved agreement between the fixed grid solution and the experimental data at $\alpha = 9°$ appears to be fortuitous. The solution obtained on the fixed grid at $\alpha = 9°$ is highly unsteady and the lift and drag values are estimates of time averages. Although not shown, the predicted pressure

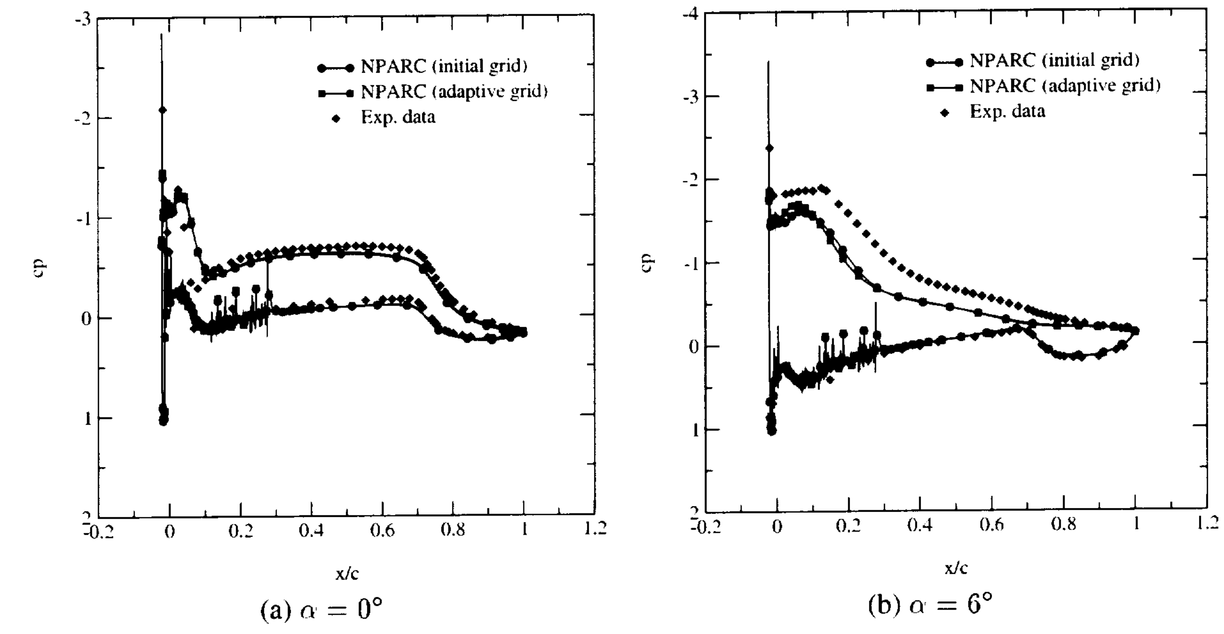


Figure 21: 623EXP iced airfoil pressure coefficient distribution - initial and adaptive grids

distribution bears little resemblance to the experimental data. On the other hand, the adaptive grid solution at $\alpha = 9°$ does exhibit a mild instability.

The lift discrepancy at higher angles of attack can easily be explained by examining the plot of the surface pressure coefficients shown in Figure 21 for $\alpha = 0$ and $\alpha = 6°$. Again, every fifth surface point is plotted for clarity. At the lower angle of attack (Figure 21.a), there is reasonable agreement between the predicted results and the experimental data except in the region just aft of the horn on the upper surface. At $\alpha = 6°$ (Figure 21.b), however, there is significant disagreement between the experimental data and the predicted results which accounts for the decrease in predicted lift coefficient. Unfortunately, virtually identical results were obtained for the fixed and solution adaptive grids.

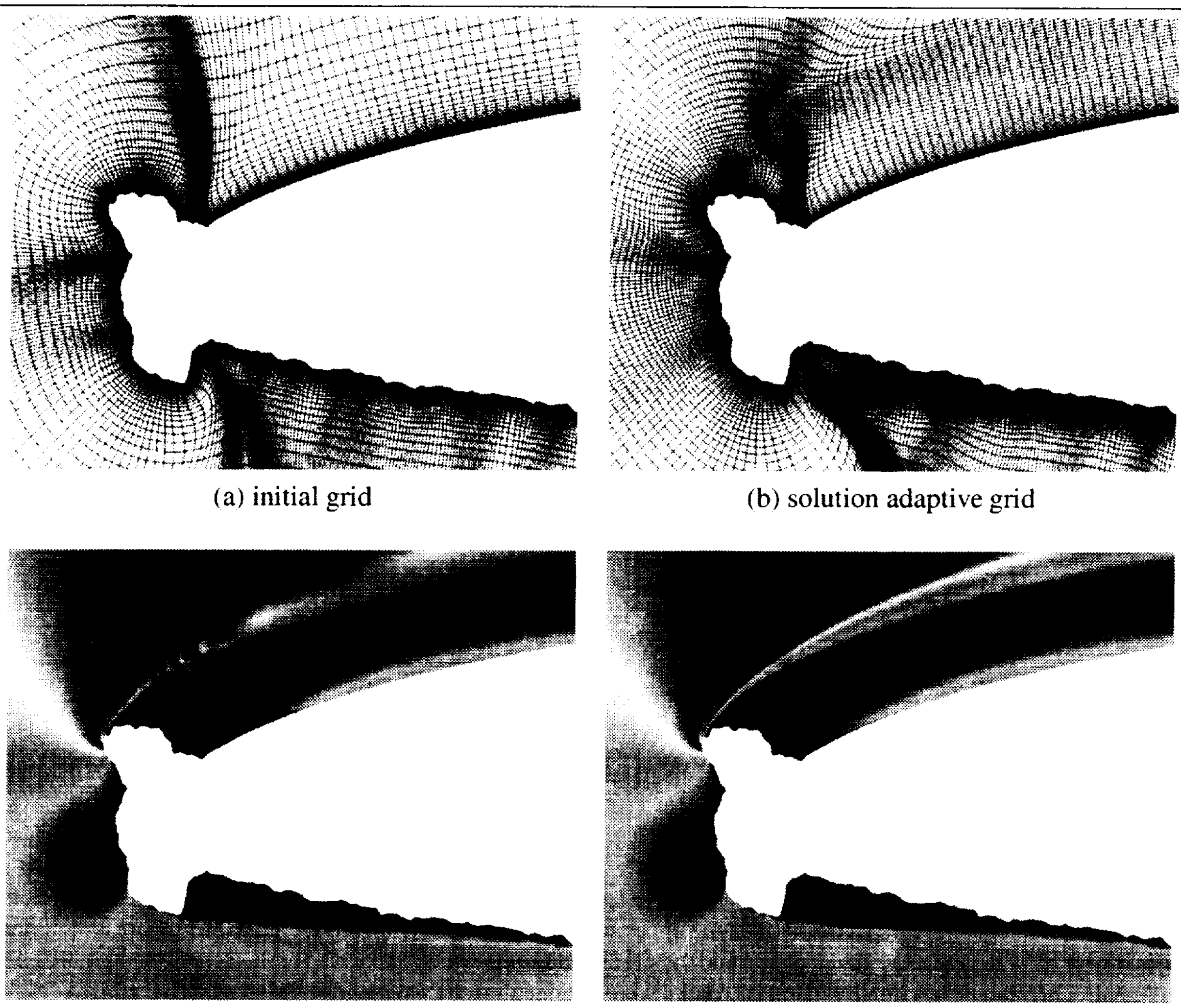
Figure 22 shows the fixed initial grid and the corresponding velocity magnitude contours along with the solution adaptive grid and its velocity magnitude contours for an angle of attack of $6°$. There is significant improvement in the quality of the velocity contours as evidenced by the smoother transition in the separated shear layer. The improvement is, of course, due to the improved point distribution in the region. However, the solution adaptive grid does not significantly influence the gross features of the flow and the surface pressure prediction and force predictions are not improved. No significant differences were observed using the one-equation turbulence model of Spalart and Allmaras [22] or the two-equation $\kappa - \epsilon$ model of Chien [24]. All calculations reported here assume a fully turbulent flow. It may be possible to specify an upper surface transition point and significantly improve the predictions [25].

Figure 23 shows a plot of the convergence histories for the fixed and solution adaptive grids for $\alpha = 6°$. In Figure 23.a, the affects of the adaptive grid cycling are evidenced by the residual spikes that occur at iterations 25000, 35000, and 45000. In each instance, the residual spike occurs because the interpolation is not "perfect." However, the residual drops rapidly and the calculation proceeds. As can be seen from Figure 23.b, the lift history is basically unaffected by the adaptation process.

In summary, the solution adaptive grid procedure has not been demonstrated to be effective in terms of eliminating the discrepancies in the predicted surface pressures and, correspondingly, the lift and drag. However, it does improve the quality of the solution in the field. Further, since the cost is relatively inexpensive with respect to the flow solver, it would seem to be prudent to use solution-adaptive grids for ice accretion calculations.

8.3.2 Generalized Grid

We now turn our focus to HYBFL2D solutions computed for the iced airfoil 623EXP using generalized grids with point insertion/deletion based on geometrical criterion (Equations 26 and 27). Figure 24 shows four views of the generalized grid generated for the 623EXP configuration. The initial surface for this grid consisted of 701 points


(a) initial grid

(b) solution adaptive grid

(c) initial velocity contours

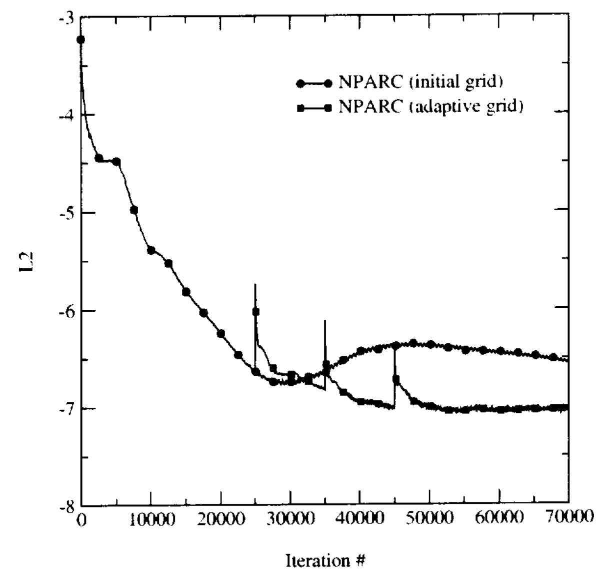
(d) solution adaptive velocity contours

Figure 22: 623EXP iced airfoil grid and velocity contours $\alpha = 6^\circ$ - initial and adaptive grids

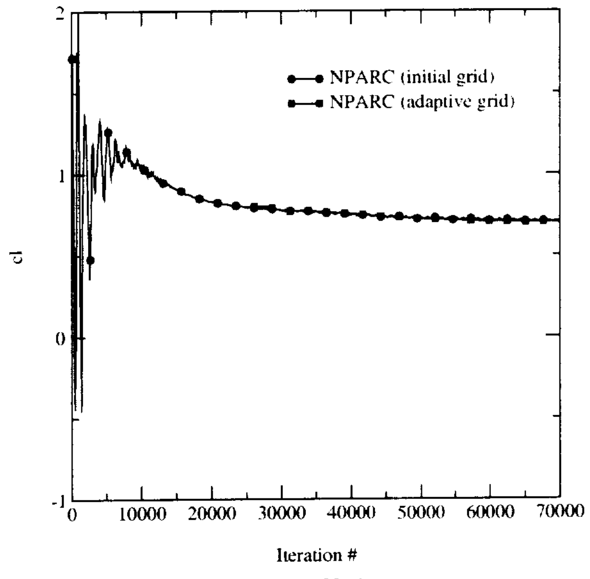
while the final surface consisted of 43 points. There were 41208 cells in this grid and more than 97.6 percent were quadrilaterals. Again, the grid shows a relatively smooth variation in cell size.

Figure 25 shows a comparison of the lift and drag coefficients predicted using HYBFL2D with the grid shown in Figure 24, HYBFL2D and NPARC using the structured quadrilateral grid shown in Figure 19, and experimental data. There are small differences between the results predicted using HYBFL2D with the two different grids. There are significant differences between the HYBFL2D results, the NPARC results, and the experimental data. In this case, HYBFL2D solution under-predicts the slope of the lift curve as well the lift values. The gross characteristics of the all three simulations are very similar and do not compare well with the experimental data. The drag predicted using HYBFL2D with either of the grids is remarkably similar to the experimental data at lower angles of attack. Given the discrepancies observed in the lift data, it seems likely that this agreement is fortuitous. It should be noted that results have not yet been obtained using the HYBFL2D for an angle of attack of 9°.

We next consider the surface pressure distribution. Figure 26 shows the surface pressure distribution for the two HYBFL2D solution, the NPARC solution, and experimental data at two angles of attack ($\alpha = 0^\circ$ for Figure 26.a and $\alpha = 6^\circ$ for Figure 26.b). There are few observable differences between the two HYBFL2D solutions. The HYBFL2D solutions show reasonable agreement with the NPARC solution except in the region just downstream of the horn. This discrepancy could well due to the overall coarseness of the grid in the region downstream of the horn (see Figure 24).



(a) L_2 norm of residual



(b) lift coefficient

Figure 23: 623EXP iced airfoil convergence histories $\alpha = 6°$ - initial and adaptive grids

The differences between predicted values are significantly less than the differences between the predicted values and the experimental data.

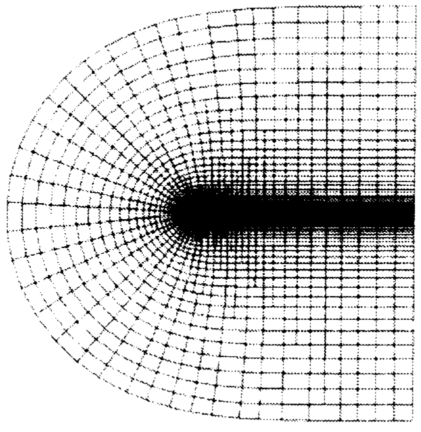
Figure 27 shows a comparison between the velocity magnitude contours predicted using HYBFL2D and NPARC for an angle of attack of 6°. The two main differences are that the shear layer is more diffuse for the solution obtained on the generalized grid and the region of higher velocity flow just aft of horn in HYBFL2D solutions. That the shear layer in more diffuse in the generalized grid solution comes a no surprise. Since there are fewer grid points in this region, it is expected that the shear layer would not be as well resolved. The region of accelerated flow in the HYBFL2D solution is not yet understood but does explain the differences observed in the surface pressure distribution in this region. Finally, Figure 28 shows the HYBFL2D convergence histories for the all quadrilateral grid and the generalize grid. Both are quite similar.

In general, results obtained using HYBFL2D with either a structured all-quadrilateral grid or a generalized grid with point deletion/insertion were very similar. This result indicates that the generalized grid approach has potential and should be investigated further.

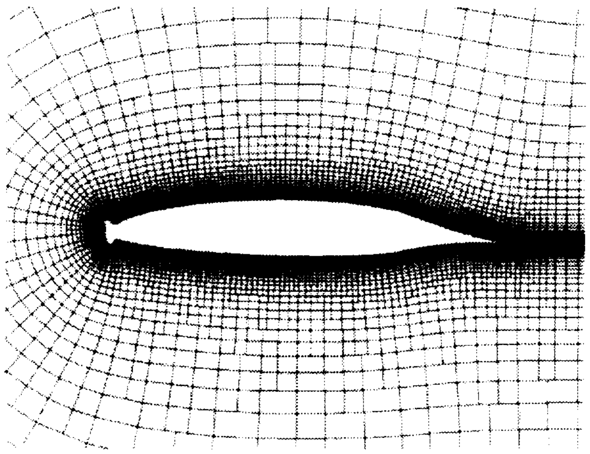
8.3.3 Solution-Adaptive Generalized Grid

To date, only limited results have been obtained for flow solutions on solution-adaptive generalized grids. This is due, in part, to the problems associated with the generalized flow solver. Also, the interpolation routine for the generalized grids has not yet been developed. Therefore, truly solution-adaptive generalized grids have not yet been generated. The results shown below were generated by taking the weight function generated for a structured grid solution at the given flight conditions. These weights were then used with the generalized grid generation algorithm to generate a generalized solution adaptive with point redistribution.

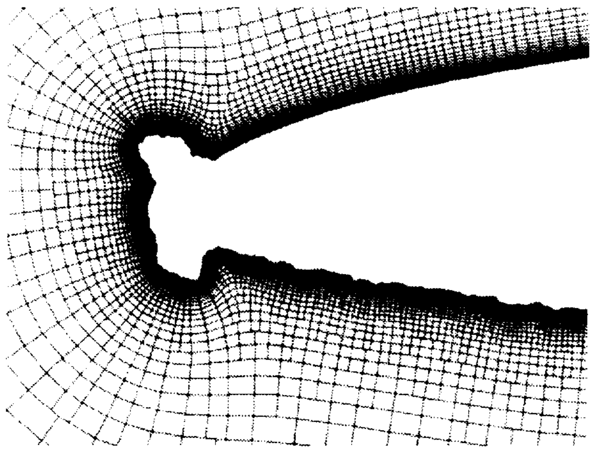
The results shown below are for the 623EXP at $\alpha = 0°$, a Mach number of approximately 0.3, and a Reynolds number of approximately 6.5×10^6. Although the lift and drag values are slightly different between the two cases, the differences are not particularly significant. Figure 29 shows the surface pressure distribution for the generalized grid and the solution adaptive generalized grids. As can be seen from the figure, there is a slight difference in surface pressures in the recirculating region downstream of the horn. This difference correlates with the results in Figure 30 which shows the fixed initial grid and its velocity magnitude contours along with the solution adaptive grid and its velocity magnitude contours for an angle of attack of 0°. There is a region of accelerated flow on the downstream side of the upper surface horn that appears in the adaptive grid flow solution that is accentuated relative to the generalized grid solution. Although not to the same degree as is evident in the structured grid case, there is significant improvement



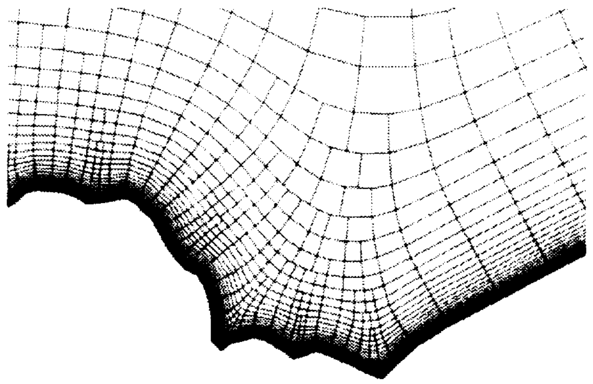
(a) full grid



(b) region near airfoil



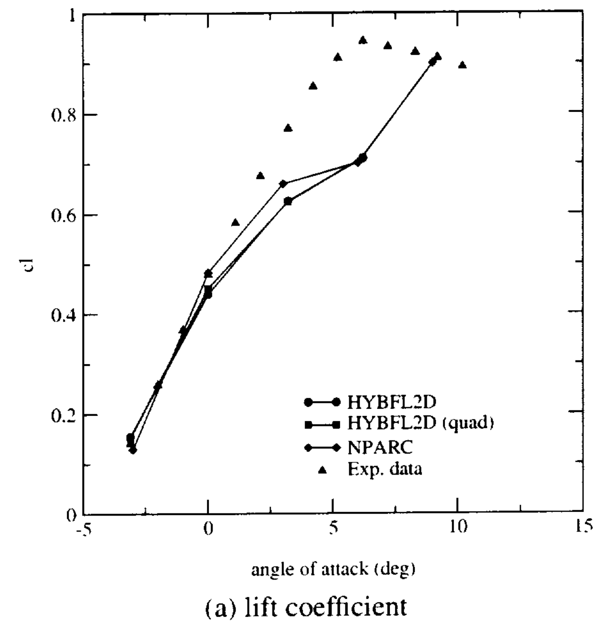
(c) region near airfoil leading edge



(d) detail of concave region

Figure 24: 623EXP iced airfoil - generalized grid

in the quality of the velocity contours as evidenced by the smoother transition in the separated shear layer. The improvement is, of course, due to the improved point distribution in the region. Again, the solution adaptive grid does not significantly influence the gross features of the flow and the surface pressure prediction and force predictions are not improved. The convergence histories for the two cases were similar and are not included.



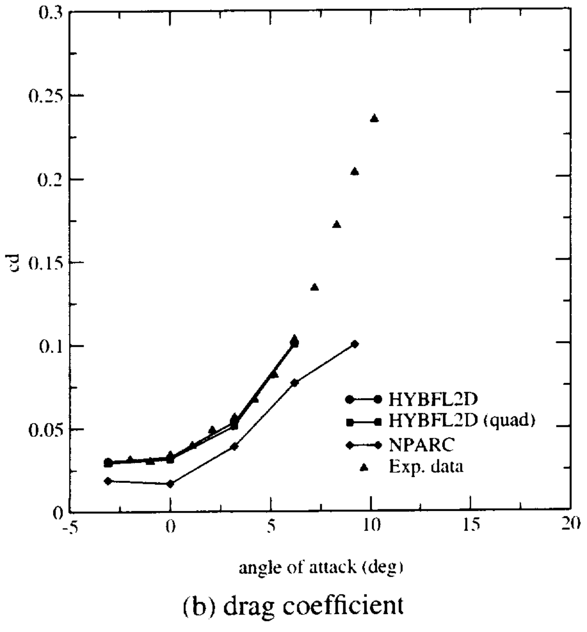
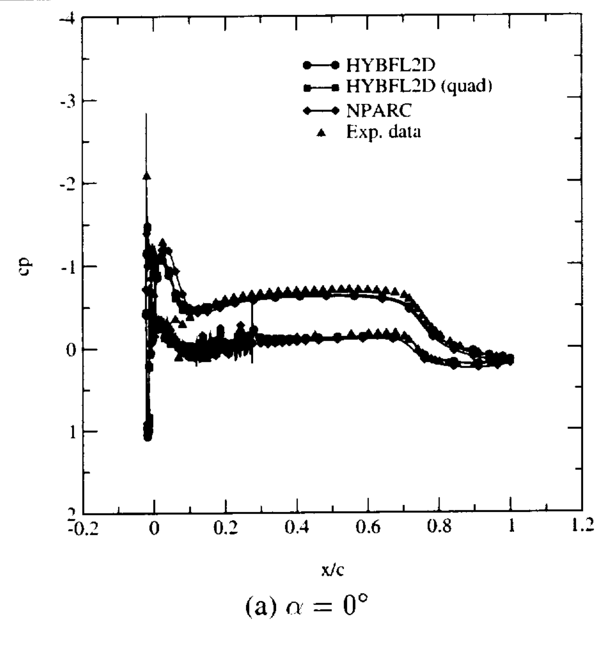


Figure 25: 623EXP iced airfoil lift and drag - generalized and quadrilateral grids



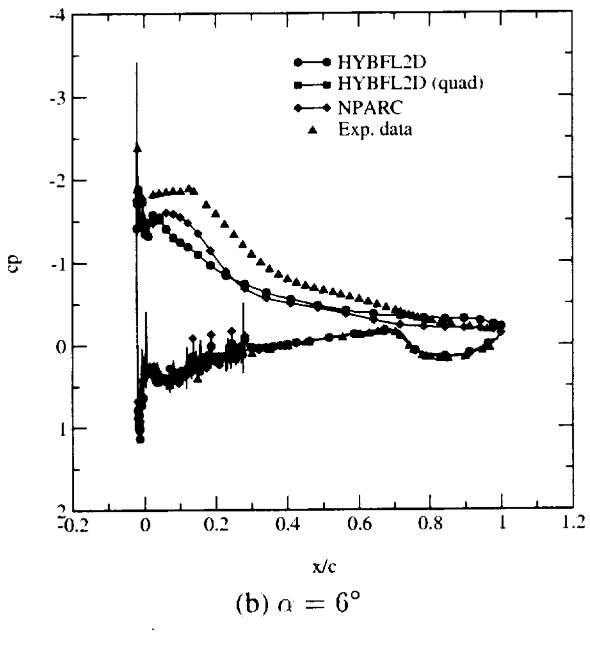
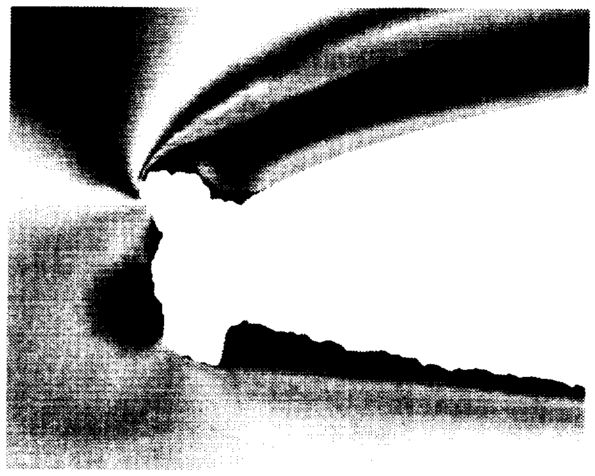


Figure 26: 623EXP iced airfoil pressure coefficient distribution - generalized and quadrilateral grids


(a) HYBFL2D (generalized grid) solution

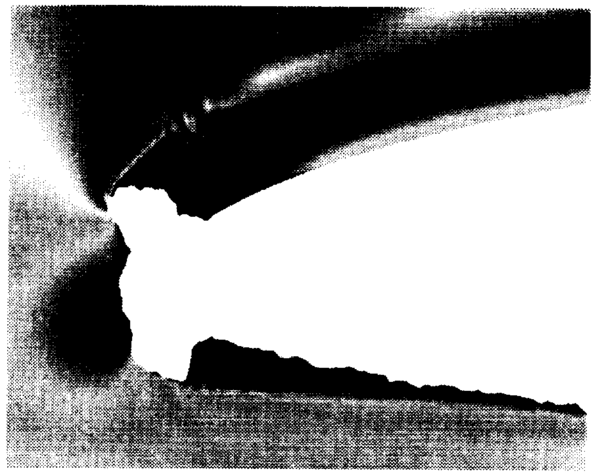
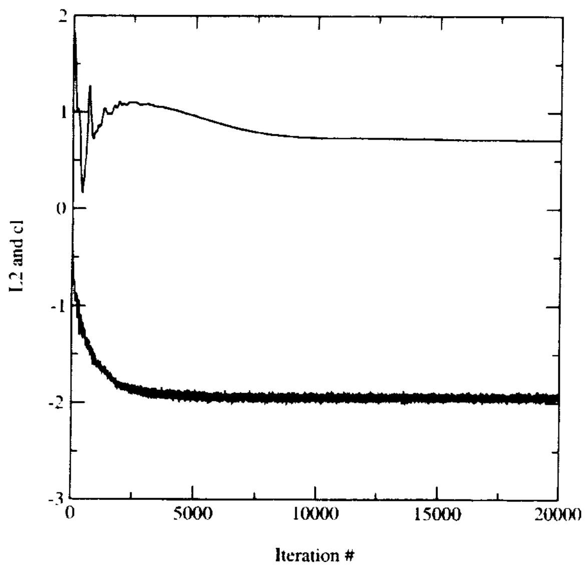
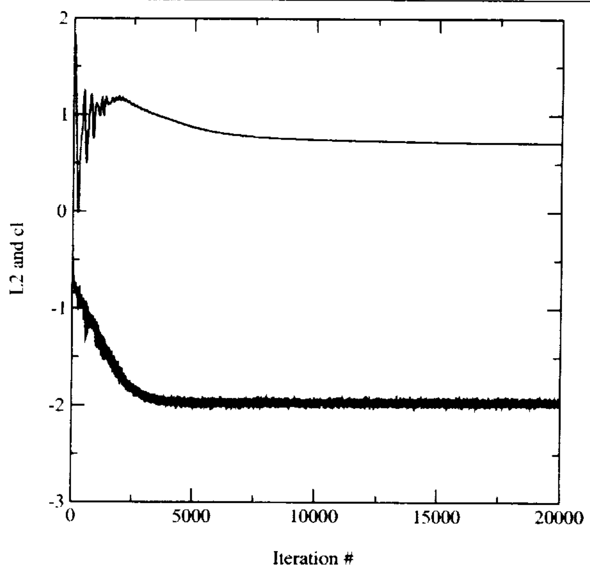

(b) NPARC solution

Figure 27: 623EXP iced airfoil $\alpha = 6°$ - HYBFL2D and NPARC velocity magnitude contours



(a) quadrilateral grid



(b) generalized grid

Figure 28: 623EXP iced airfoil convergence history $\alpha = 6°$ - quadrilateral and generalized grids

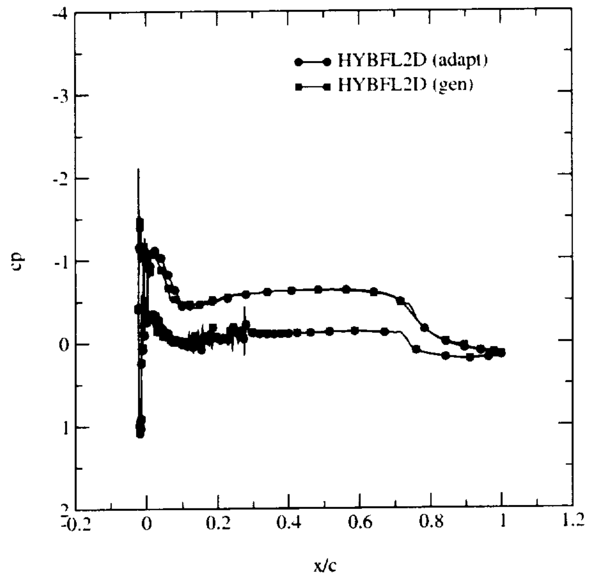
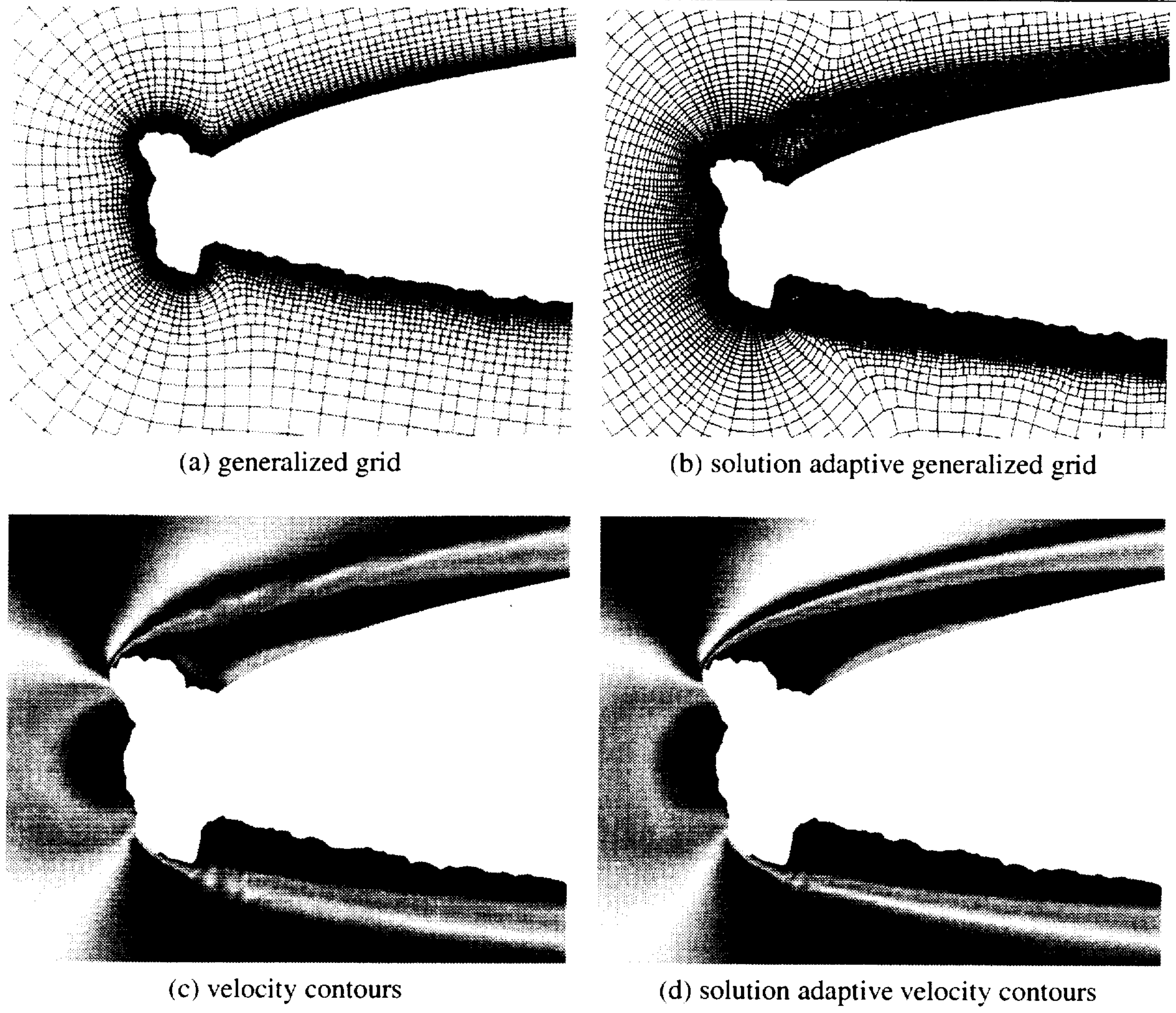


Figure 29: 623EXP iced airfoil surface pressure distribution $\alpha = 0°$ - generalized and adaptive generalized grids


(a) generalized grid

(b) solution adaptive generalized grid

(c) velocity contours

(d) solution adaptive velocity contours

Figure 30: 623EXP iced airfoil grid and velocity contours $\alpha = 0°$ - generalized and adaptive grids

9. Example Cases

Included in this section are several example directive streams illustrating the use of ICEG2D (v2.0). These input streams and the data needed to run these cases are included in the ICEG2D (v2.0) distribution in the directory $ICEG2D_DIR/examples.

The first case considered is one in which we want to generate a single-block grid for a LEWICE-generated ice shape 204LEW [10]. The data for this case is contained in the directory $ICEG2D_DIR/examples/example_1. The ice shape is smooth and is defined in the ASCII file **newdata.dat**. The clean airfoil shape is defined in the ASCII file **newdata.cln**. In this instance, the file extensions are appropriate for the data types for a case name of *newdata*. There is also an NPARC information file named **nparc.inf** in the local directory. The ICEG2D (v2.0) directive stream for an interactive session is listed below:

```
ICEG2D> def case newdata
ICEG2D> gen dist
ICEG2D> gen grid
ICEG2D> display grid
ICEG2D> def info nparc.inf
ICEG2D> gen input
ICEG2D> gen initial
ICEG2D> gen solution
ICEG2D> exit
```

In this case, the default parameter file associations are used. Also, note that the minimal versions of the directives are used, i.e., **def** instead of **define**. The initial file is named **newdata.ini**, the input file is named **newdata.inp**, etc.

Consider the same data as the previous example only this time using a generalized grid. The ICEG2D (v2.0) directive stream for an interactive session is listed below:

```
ICEG2D> def case newdata
ICEG2D> gen dist
ICEG2D> def topo semi
ICEG2D> gen grid
ICEG2D> display grid
ICEG2D> def info hybfl.inf
ICEG2D> gen input
ICEG2D> gen solution
ICEG2D> exit
```

Here, the only differences are the addition of the **def topo** *semi(structured)* and the new information file **hybfl.inf**. This automatically ensures that correct procedures will be followed when generating and displaying the grid, an appropriate HYBFL2D input will be generated, and HYBFL2D will be executed.

Now consider a case that we want to execute in batch mode with an experimental ice shape defined in the ASCII file **904iced.af** and a clean airfoil defined in the file **glc305.af**. These data files and the ASCII driver file **driver.inp** are located in the directory $ICEG2D_DIR/examples/example_2. The NPARC information file is **input.inf**. We want to use a modified DISTRIBUTION namelist in the file **dist_local.ini**. We also want to generate a double-block grid using the default double-block parameter file **double_block.ini**. The directive stream contained in **driver.inp** is listed below:

```
def mode batch
def case 904iced
def iced_geom 904iced.af
def clean_geom glc305.af
def dist_parm dist_local.ini
gen dist
def block_parm double
gen grid
def info input.inf
```

```
gen input
gen initial
# gen solution
exit
```

The ICEG2D (v2.0) script is executed using "ICEG2D < driver.inp > output." In this case, the input file is named **904iced.inp**. If user-defined blocking parameters are to be used and are defined in the local file **block_def.inp**, the directive **def block_parm** *double* should be replaced with **def block_parm** *block_def.inp* Note that the **gen solution** directive has been commented out.

We now consider a case starting from an intermediate step. Here, we assume a single-block grid file exists that is in the FAST formatted, two-dimensional, multizone, grid format and is named **grid.xy**. The NPARC information file is located in **prob.inp**. Both files are located in the directory $ICEG2D_DIR/examples/example_3. The directive stream for an interactive session is listed below:

```
ICEG2D> def case problem1
ICEG2D> def grid grid.xy
ICEG2D> def info prob.inp
ICEG2D> show
ICEG2D> gen input
ICEG2D> gen initial
ICEG2D> gen solution
ICEG2D> exit
```

Although the file associations required by the modules ICE_DIST_2D and ICE_GRID_2D are not needed, we still need a case name for the default file associations for the NPARC info file, the initial file, etc. Note that the **show** directive lists all the current file associations and is useful for checking to make sure everything has been defined as intended.

Finally, a directive stream is included to illustrate how ICEG2D (v2.0) was used to generate solution adaptive, structured grids for an iced airfoil configuration. The files for this case are contained in $ICEG2D_DIR/examples/example_4. As noted in Section 8, flow solutions were obtained using a sequence of 25000 iterations, a grid adaptation cycle, 10000 iterations, a grid adaptation cycle, 10000 iterations, a grid adaptation cycle, and 25000 iterations. Below is the script used to perform this calculation. In this case, the iced and clean airfoil data are contained in **623exp.dat** and **623exp.cln**, respectively. Since this is an experimental ice shape, *surface_type* is defined to be *rough*. The surface distribution is computed and then a structured grid is generated followed by the generation of the NPARC input and initial files. The solution is then generated. In this script, the **sys** directive is used repeatedly to execute the Unix command "mv" so that various NPARC-generated ASCII files can be saved and have later data appended to them. *adaptation_type* is set to *redist* followed by the generation of the weight function. The grid is generated based on these weight functions and the solution interpolated onto this new grid and the process is repeated. The directive stream to perform the solution adaptive grid calculation is given by:

```
def mode batch
def case 623exp
#
# Initial distribution, grid generation and solution
#
def surface rough
gen dist
gen grid
def info NPARC_info_start.inf
gen input
gen initial
gen solution
#
# Rename files to save data
#
sys "mv fort.61 lifthistory.plt"
sys "mv fort.62 draghistory.plt"
sys "mv fort.63 momenthistory.plt"
sys "mv fort.21 l2history.plt"
#
# First adaptation step
#
def adapt redist
gen weight
gen grid
def info NPARC_info_adapt.inf
gen input
gen restart
gen solution
#
# Rename files and concatenate with old files
#
sys "cat lifthistory.plt fort.61 > temp"
sys "mv temp lifthistory.plt"
sys "cat draghistory.plt fort.62 > temp"
sys "mv temp draghistory.plt"
sys "cat momenthistory.plt fort.63 > temp"
sys "mv temp momenthistory.plt"
sys "cat l2history.plt fort.21 > temp"
sys "mv temp l2history.plt"
#
# Second adaptation step
#
def adapt redist
gen weight
gen grid
def info NPARC_info_adapt.inf
gen input
gen restart
gen solution
#
# Rename files and concatenate with old files
#
sys "cat lifthistory.plt fort.61 > temp"
sys "mv temp lifthistory.plt"
sys "cat draghistory.plt fort.62 > temp"
sys "mv temp draghistory.plt"
sys "cat momenthistory.plt fort.63 > temp"
sys "mv temp momenthistory.plt"
sys "cat l2history.plt fort.21 > temp"
sys "mv temp l2history.plt"
#
# Third and final adaptation step
#
def adapt redist
gen weight
gen grid
def info NPARC_info_final.inf
gen input
gen restart
gen solution
#
# Rename files and concatenate with old files
#
sys "cat lifthistory.plt fort.61 > temp"
sys "mv temp lifthistory.plt"
sys "cat draghistory.plt fort.62 > temp"
sys "mv temp draghistory.plt"
sys "cat momenthistory.plt fort.63 > temp"
sys "mv temp momenthistory.plt"
sys "cat l2history.plt fort.21 > temp"
sys "mv temp l2history.plt"
#
exit
```

Notice that several NPARC information files are used in this script.

10. Summary

An integrated geometry/grid/simulation software package, ICEG2D, was developed to automate computational fluid dynamics (CFD) simulations for single- and multi-element airfoils with ice accretion. The current version, ICEG2D (v2.0), was designed to automatically perform four primary functions: 1) generate a grid-ready surface definition based on the geometrical characteristics of the iced airfoil surface, 2) generate high-quality structured and generalized grids starting from a defined surface definition, 3) generate the input and restart files needed to run the structured grid CFD solver NPARC or the generalized grid CFD solver HYBFL2D, and 4) using the flow solutions, generate solution-adaptive grids. ICEG2D (v2.0) can be operated in either a batch mode using a script file or in an interactive mode by entering directives from a command line within a Unix shell. ICEG2D (v2.0) consists of several modules that are integrated within a framework of associations that relate external data to internal file names.

During the second year of a proposed three-year effort, the capability to generate solution adaptive grids was implemented. Although the solution adaptive grids did not successfully address the problem for which they were intended, the solution adaptive grids do improve the resolution of separated shear layers. The additional cost is negligible in comparison with the flow field solution. The capabilities to generate semistructured, generalized grids [3, 4] and compute flow solutions on these grids utilizing the flow solver HYBFL2D [5] were also incorporated into the software. Because of problems associated with HYBFL2D, a three-dimensional version of the code is actually used here. A preprocessor converts the two-dimensional generalized grid into a three-dimensional grid by extrusion. The process occurs in a seamless manner that is transparent to the user. Efforts are currently under way to rectify the problems with the two-dimensional code. These difficulties slowed the evaluation of the generalized grid techniques. Therefore, the results reported here should be considered to be preliminary, at best. However, results obtained using HYBFL2D with either a structured all-quadrilateral grid or a generalized grid with point deletion/insertion were very similar. This result indicates that the generalized grid approach has potential and should be investigated further. The potential for improved grid quality, at least in terms of grid skewness, also bolsters this conclusion.

ICEG2D (v2.0) has been tested by application to numerous iced airfoil configurations. The geometry module of ICEG2D (v2.0) was evaluated based on its ability to distribute points on the surface of the iced airfoil while maintaining geometric fidelity. Additionally, the geometry module has the ability to merge incomplete iced airfoil descriptions with clean airfoil shapes and to prune ice shapes with loops and twists. The grid generation module was evaluated based on two criteria. The first, a necessary condition, is based on geometric characteristics of the criteria, i.e., cell area positivity and no excessive skewness. The second condition is based on the accuracy of solutions computed using the automatically generated grids. Solutions for all clean airfoil cases agreed well with the experimental data. Iced airfoil solutions, on the other hand, exhibited discrepancies with experimental data for cases in which there was significant flow separation. The cause of these discrepancies have not been determined conclusively.

References

[1] Nparc user's guide version 3.0, 1996.

[2] D. S. Thompson and B. K. Soni. ICEG2D—An Integrated Software Package for Automated Prediction of Flow Fields for Single-Element Airfoils with Ice Accretion. Technical Report CR-2000-209914, NASA, February 2000.

[3] D. S. Thompson and B. K. Soni. Generation of Quad- and Hex-Dominant, Semistructured Meshes Using an Advancing Layer Scheme. In *Proceedings of the 8th International Meshing Roundtable*, October 1999.

[4] D. S. Thompson and B. K. Soni. Semistructured Grid Generation in Three-Dimensions using a Parabolic Marching Scheme. In *AIAA Paper 2000-1004, AIAA 38th Aerospace Sciences Meeting*, Reno, NV, January 2000.

[5] R. P. Koomullil and B. K. Soni. Generalized Grid Techniques in Computational Field Simulation. In M. Cross, P.R. Eiseman, J. Hauser, B. K. Soni, and J. F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Simulations*, pages 521–531. International Society for Grid Generation, Mississippi State, MS, 1998.

[6] G. B. Hong. GGLib: A Geometry & Grid Generation Library. Technical report, MSU/NSF Engineering Research Center, May 1999.

[7] S. Nakumura. Noniterative Grid Generation Using Parabolic Partial Differential Equations. In J. F. Thompson, editor, *Numerical Grid Generation*. Elsevier Science Publishing Company, Inc., New York, 1982.

[8] R. W. Noack and D. A. Anderson. Solution-Adaptive Grid Generation Using Parabolic Differential Equations. *AIAA J.*, 28:1016–1023, 1990.

[9] R. W. Noack and I. H. Parpia. Solution-Adaptive Parabolic Grid Generation in Two and Three Dimensions. In A. S.-Arcilla J. Hauser P.R. Eiseman and J. F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*. Elsevier Science Publishing Company, Inc., New York, 1991.

[10] J. K. Chung. private communication, 1999.

[11] W. B. Wright. A Summary of Validation Results for LEWICE 2.0. In *AIAA Paper 99-0249, AIAA 37th Aerospace Sciences Meeting*, Reno, NV, January 1999.

[12] H. E. Addy, M. G. Potapczuk, and D. W. Sheldon. Modern Airfoil Ice Accretions. Technical Report TM-1997-107423, NASA, March 1997.

[13] J. Chung, Y. Choo, A. Reehorst, D. W. Sheldon, and J. Slater. Numerical Study of Aircraft Controllability under Certain Icing Conditions. In *AIAA Paper 99-0375, AIAA 37th Aerospace Sciences Meeting*, Reno, NV, January 1999.

[14] Y. Choo, J. Slater, T. Henderson, C. Bidwell, D. Braun, and J. Chung. User Manual for Beta Version of Turbo-GRD - A Software System for Interactive Two-Dimensional Boundary/Field Grid Generation, Modification, and Refinement. Technical Report TM-1998-206632, NASA, October 1998.

[15] L. Piegl and W. Tiller. *The NURBS Book*. Monographs in Visual Communications. Springer, New York, 2nd edition, 1997.

[16] J. F. Thompson, Z. U. A. Warsi, and W. Mastin. *Numerical Grid Generation*. Elsevier Science Publishing Company, New York, 1985.

[17] *FAST User Guide,< http : //www.nas.nasa.gov/Software/FAST/ >*, 1999.

[18] W. M. Chan and J. L. Steger. Enhancements of a three-dimensional grid generation scheme. *App.Math. Comp.*, 51:181–205, 1992.

[19] G. O. Roberts. Computational Meshes for Boundary Layer Problems. *Lect. Notes Phys.*, 8:171–177, 1971.

[20] B. K. Soni. Elliptic Mesh Generation Systems: Control Functions Revisited. *App. Math. Comp.*, 59:151–164, 1993.

[21] B. K. Soni, R. P. Koomullil, D. S. Thompson, and H. Thornburg. Solution Adaptive Grid Strategies Based on Point Redistribution. *Comp. Methods Appl. Mech. Engrg*, 189:1183–1204, 2000.

[22] P. R. Spalart and S. R. Allmaras. A One-Equation Turbulence Model for Aerodynamic Flows. In *AIAA Paper 92-0439, AIAA 30th Aerospace Sciences Meeting*, Reno, NV, January 1992.

[23] R. F. Tomaro, W. Z. Strang, and L. N. Sankar. An Implicit Algorithm for Solving Time-Dependent Flows on Unstructured Grids. In *AIAA Paper 97-0333, AIAA 35th Aerospace Sciences Meeting*, Reno, NV, January 1997.

[24] K.-Y. Chien. Predictions of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model. *AIAA J.*, 20:33–38, January 1982.

[25] M. G. Potapczuk. private communication, 2000.

REPORT DOCUMENTATION PAGE		*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2001	**3. REPORT TYPE AND DATES COVERED** Final Contractor Report

4. TITLE AND SUBTITLE

ICEG2D (v2.0)—An Integrated Software Package for Automated Prediction of Flow Fields for Single-Element Airfoils With Ice Accretion

5. FUNDING NUMBERS

WU-711-20-23-00
NAG3-2235

6. AUTHOR(S)

David S. Thompson and Bharat K. Soni

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Mississippi State University
P.O. Box 9627
Mississippi State, Mississippi 39762

8. PERFORMING ORGANIZATION REPORT NUMBER

E-12818

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2001-210965

11. SUPPLEMENTARY NOTES

Project Manager, Yung K. Choo, Tubomachinery and Propulsion Systems Division, NASA Glenn Research Center, organization code 5840, 216–433–5868.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 02 Distribution: Nonstandard

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

An integrated geometry/grid/simulation software package, ICEG2D, is being developed to automate computational fluid dynamics (CFD) simulations for single- and multi-element airfoils with ice accretions. The current version, ICEG2D (v2.0), was designed to automatically perform four primary functions: (1) generate a grid-ready surface definition based on the geometrical characteristics of the iced airfoil surface, (2) generate high-quality structured and generalized grids starting from a defined surface definition, (3) generate the input and restart files needed to run the structured grid CFD solver NPARC or the generalized grid CFD solver HYBFL2D, and (4) using the flow solutions, generate solution-adaptive grids. ICEG2D (v2.0) can be operated in either a batch mode using a script file or in an interactive mode by entering directives from a command line within a Unix shell. This report summarizes activities completed in the first two years of a three-year research and development program to address automation issues related to CFD simulations for airfoils with ice accretions. As well as describing the technology employed in the software, this document serves as a users manual providing installation and operating instructions. An evaluation of the software is also presented.

14. SUBJECT TERMS

Aircraft ice; Grid generation; Aerodynamics

15. NUMBER OF PAGES 70

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT**