2017

Fellowship Symposium, 8 May 2017, Weber State University

# Implementation of OpenFOAM for Inviscid, Incompressible Aerodynamic Flows

Jackson T. Reid[*] and Douglas F. Hunsaker[†]

*Utah State University, Logan, Utah, 84321, USA*

## Abstract

**This paper is the description of the Utah State University AeroLab's Aerodynamic Center Analysis Tool (AeroCAT), which is an implementation of the OpenFOAM CFD toolbox. AeroCAT takes in a user input file, generates a mesh, and solves a steady, inviscid, incompressible flow, automatically repeating the process for a range of angles of attack. It then processes the results to predict the wing's span-wise locus of aerodynamic centers. The mesh generator used in this tool is GridX, developed by a former PhD student at USU, and the CFD solver is OpenFOAM.**

## Nomenclature

| | |
|---|---|
| $x$ | Axis from leading to trailing edge of the root airfoil |
| $y$ | Axis perpendicular to root chord line in the root airfoil plane |
| $z$ | Axis span-wise along a wing, perpendicular to both the $x$ and $y$ axes |
| $i$ | Cell index from wing root to tip |
| $j$ | Cell index around each cross-section |
| $k$ | Cell index radially from wing to boundary |
| $\alpha$ | Angle of Attack, deg or rad |
| $c_{ref}$ | Length of the root airfoil's chord |
| $\bar{x}_{ac}$ | $x$ location of the aerodynamic center |
| $\bar{y}_{ac}$ | $y$ location of the aerodynamic center |
| $C_{m,ac}$ | Coefficient of moment about the aerodynamic center |
| $C_N$ | Coefficient of the normal force ($y$ direction) |
| $C_A$ | Coefficient of the axial force ($x$ direction) |
| $C_m$ | Coefficient of moment about the leading edge |
| $C_{N,\alpha}$ | Derivative of $C_N$ with respect to $\alpha$ |
| $C_{A,\alpha}$ | Derivative of $C_A$ with respect to $\alpha$ |
| $C_{m,\alpha}$ | Derivative of $C_m$ with respect to $\alpha$ |
| $C_{N,\alpha,\alpha}$ | Derivative of $C_{N,\alpha}$ with respect to $\alpha$ |
| $C_{A,\alpha,\alpha}$ | Derivative of $C_{A,\alpha}$ with respect to $\alpha$ |
| $C_{m,\alpha,\alpha}$ | Derivative of $C_{m,\alpha}$ with respect to $\alpha$ |

[*]Graduate Research Assistant, Mechanical & Aerospace Engineering, 4130 Old Main Hill

[†]Assistant Professor, Mechanical & Aerospace Engineering, 4130 Old Main Hill

## I. Introduction

The AeroLab at Utah State University (USU) is dedicated to discovering and publishing fundamental aerodynamic principles. [1] As such, we actively work to develop open-source tools that will help us, and others, in aerodynamic design, analysis, and optimization. In order to perform this research and development, it was necessary to both have a means to gather data—to observe trends and create empirical equations—and have a manner in which to validate potential flow and other aerodynamics models developed by the lab. To this end, the Aerodynamic Center Analysis Tool (AeroCAT) was developed. Initially designed to run CFD simulation of steady, inviscid, incompressible flow over wings and airfoils, then calculating the span-wise locus of aerodynamic centers, AeroCAT is also capable of providing spanwise lift distribution data to validate our aerodynamic models against.

## II. Tool Overview

AeroCAT begins with a .json input file that contains mesh specifications, and simulation settings. Because this tool is meant to be user friendly and simple to run, not all possible settings are available in this input file, but rather are hard-coded into the source script. AeroCAT takes in the values from the input file and generates a mesh using GridX. Then, that mesh is used in an OpenFOAM simulation of the

[1]AeroLab Mission Statement as seen on aero.usu.edu

flow. The forces and moments on the airfoil or wing are recorded by OpenFOAM automatically to be used in post-processing routines. The process is then repeated for the next angle of attack, and continues throughout the range of angles of attack specified by the user.

A new mesh is generated for each angle of attack in the desired range because GridX uses a trailing edge streamline approximation to position the cells behind the aerodynamic surface. Once all angles of attack have been simulated and all the data collected, AeroCAT post-processes the collection, calculating the locus of aerodynamic centers and lift distributions, plotting them for easy visualization. The specific details of this process, as well as a description of all the setting options, is found in the sections hereafter.

AeroCAT has been developed using a Linux operating system and, as such, the following descriptions will be have only been tested using Linux. That being said, it is not expected that there be major issues trying to run AeroCAT on other operating systems.

## II.A. Installation and Input Files

The installation AeroCAT is performed in two stages. First OpenFOAM must be installed by following the instructions on its website. [2] The next step is to run the shell script "Compile.sh", which compiles the FORTRAN files needed for GridX and the post processing subroutines, creates the needed executables, and sets the necessary permissions for the Python scripts. The installation is then complete. [3]

AeroCAT should be installed in a location easily accessible to the user and run in a terminal with the OpenFOAM system variables set [1]. As it runs, AeroCAT creates a folder in the installation directory containing the mesh, configure files, and results of each project. OpenFOAM requires certain system variables to be set before runtime, and thus AeroCAT cannot be run without those variables.

When running AeroCAT, the user is first asked whether there is an existing input file to be used. If the user has an input file to use, it is specified. On the other hand, AeroCAT will walk the user through the creation of an input file that is then saved and can be reused or edited for future projects. An example of an AeroCAT input file is included in the Appendix.

AeroCAT uses a .json format for its input file so while the order of the elements is not important, in general all elements should be included in the input file to avoid runtime errors. The name of the input file (exluding the .json extension) becomes the project name off which all other files and folders are named. For example, if the input file were named "Test1.json", AeroCAT would create (or write over) a folder named "Test1", and, subsequently, save the OpenFOAM simulation files for an angle of attack $\alpha$ in a folder named "Test1_$\alpha$".

The specific elements of the AeroCAT input file will be described in more detail in their corresponding sections further down.

[2] As of 21 April 2017: www.openfoam.com/download

[3] To generate plots while running GridX as a stand alone program, PlotMTV must be installed as well.

## II.B. Mesh Generator

The mesh generator used by AeroCAT, GridX, was developed by Professor Warren Phillips and Nick Alley during Nick's PhD work here at Utah State. It is useful due to the range of airfoils and finite wings it is capable of generating, as well as the user's ability to adjust the cell spacing. The original version of GridX did not have compatibility with OpenFOAM, so part of the development of AeroCAT was the addition of functions that will produce the appropriate mesh and case files to run an OpenFOAM simulation.

### *II.B.1. Stand-Alone Walk-Through*

Using AeroCAT, GridX is run automatically from the main AeroCAT script. However, the following is a description of GridX being run as a stand-alone program. This will be of use to the reader, as it will demonstrate what is happening in the background of AeroCAT and provide insight as to how to modify the mesh settings in a more advanced way.

GridX is initiated by running the executable "Inner.out" in a directory that contains a GridX input file,"*.iGridX". GridX asks for the name of the input file, without extension, and then reads that file, creates an airfoil based on the settings read in, calculates the trailing edge streamline, and then displays the "MENU". This menu shows all the settings that are currently active in GridX, including: output file formats, flight state information, airfoil and wing data, and grid properties. These options can be modified by the user by entering the code displayed next to the option and then following the instructions displayed in the terminal.

Once all the options are set as desired, the code "gg" is entered to begin the grid generation. GridX starts by creating splines of the airfoil, trailing edge streamline, and outer boundary. These splines will be used by the generator as boundary points to smooth the mesh algebraically or by an elliptic partial differential equation. This smoothing ensures that the of a cell gradually grows or shrinks as it transitions to the next cell. In the case of a 3D simulation, GridX starts at the symmetry plane, which corresponds to the root airfoil of the wing, and generates that planar grid. For the 2D case, this is the end of the grid generation. In both the 3D and 2D cases at this time GridX asks the user if he or she would like to view a plot of the mesh, first of the boundary layer grid and then of the transition and full planar grids.

GridX then generates and connects planar grids span-wise along the $z$ direction. These grids are parallel (or at least very close) to one another until the endcap of the wing is reached. Sweep and dihedral in the wing is handled by a translation of these parallel grids in either the $y$ or $x$ directions. Once the endcap is reached, the planar grids begin to fold in half about the leading and trailing edge streamlines of the tip airfoil. The final planar grid is folded completely in half such that the points that make up the top and bottom halves are identical. The duplicated points along this plane are consolidated during the writing of the OpenFOAM mesh files.

A similar duplication of points occurs for the points that lay along of the trailing edge streamlines behind the wing. Here, in the "C" type grid that is the default in AeroCAT, the cells directly above and below the trailing edge streamline share a face that would normally be an external boundary face but, because of the folded nature of the C-grid, has become internal. This C-grid style of mesh has been visualized in Figure 1.

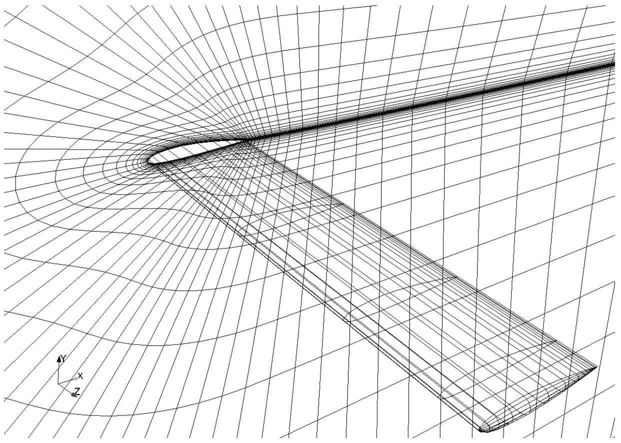

**Figure 1. Visualization of a "C-grid" mesh generated by GridX, showing the wing, endcap, and symmetry plane.**

Having generated the points of the mesh, GridX begins writing the output files specified by the GridX input file or in the "Menu". Depending on the input file settings, GridX may ask if the user would like to generate additional output files. After all output files have been generated, GridX displays the grid size in terms of points in the $i$, $j$, and $k$ directions, and the total number of points before any point consolidation. Finally, before the program ends, the user has one last opportunity to plot the finalized mesh.

### II.B.2. *Description of AeroCAT Input File Options for GridX*

As mentioned previously, the AeroCAT input file contains only a portion of the input settings available in GridX's input file (examples of each can be found in the Appendix). The AeroCAT input file options that correspond to the GridX portion of the program are:

- **"AR"**, Aspect Ratio of the 3D wing (3D grids with a symmetry plane result in half of a full wing of aspect ratio "AR").
- **"Chord"**, length of the root airfoil chord, in meters.
- **"Dihedral"**, angle of dihedral in degrees (a positive value results in dihedral, whereas a negative value results in anhedral).
- **"flap_yn"**, a "y" or "n" character denoting whether a flap deflection is to be generated (2D meshes only).
- **"flap_deg"**, degrees of flap deflection (if **flap_yn** is "y"). A positive value indicates downward deflection.
- **"flap_hinge"**, location of the flap's point of rotation (1 = Upper Surface; 2 = Lower Surface; 3 = Center)
- **"flap_len"**, length of the flap as a fraction of the chord length ('Flap-Length'/'Chord-Length').
- **"joukowski_yn"**, a "y" or "n" character denoting whether a Joukowski airfoil is to be generated (if "n", a NACA series airfoil is generated. See "NACA").
- **"jk_thick"**, maximum thickness of a Joukowski airfoil as a fraction of the chord length ('Max-Thickness'/'Chord-Length').
- **"jk_lift"**, the target coefficient of lift of a Joukowski airfoil at $\alpha = 0°$.
- **"Mesh_density"**, keywords "fine", "medium", or "coarse" that specify the density of the mesh. Where a "fine" mesh is initially generated, then every other internal node is removed for a "medium" mesh, and again every other internal node is removed for a "coarse" mesh, as specified.
- **"Mesh_dim"**, dimension of the mesh ("2"-D or "3"-D).
- **"NACA"**, specification of a four digit NACA series airfoil (XXXX).

- **"Semi_span"**, length of the 3D wing from centerline to tip along the $z$ axis, in meters.
- **"Sweep"**, angle of sweep in degrees (a positive value results in a swept back wing, whereas a negative value results in a wing with forward sweep).
- **"TR"**, Taper Ratio between the length of the root chord length and the tip chord length ("Tip-Chord"/"Root-Chord").
- **"Washout"**, angle of washout in degrees (a positive value results in a tip with lower $\alpha$ than the root, whereas a negative value results in a tip with a higher $\alpha$).
- **"chord_bl3ff"**, distance between the boundary layer cells and the outer boundary, in number of chord lengths.
- **"node_airfoil"**, number of nodes on the airfoil surface.
- **"node_behind"**, number of nodes located along the trailing edge streamline behind the wing.
- **"node_bl2ff"**, number of nodes located between the boundary layer and outer boundary above, below, and in front of the wing.
- **"node_boundary"**, number of nodes in the boundary layer.
- **"node_endcap"**, number of span-wise nodes on each half of the endcap ("node_endcap" on top, and "node_endcap" on bottom).
- **"thickness1_bl"**, thickness of the first boundary layer cell, as a fraction of the root chord length (measured perpendicular to the wing/airfoil surface).

The settings **"AoA_max"** and **"AoA_min"** in the AeroCAT input file dictate the range of angle of attack, in degrees, for which simulations will be run, so they also indirectly correspond to GridX input values.

There are many other settings and ways to customize a mesh using GridX, but all of the most necessary settings are available through the AeroCAT input file. If there is an occasion that calls for the modification of settings not found in that file (e.g. if cells are crossing and the right and left hand influence parameters need adjusting), the user may manually modify the default values found in the "AeroCAT_Main.py" script.

## II.C. OpenFOAM

As was done for GridX above, this section will be a discussion of OpenFOAM as a stand-alone program. OpenFOAM runs its simulations based of various files found in subdirectories of the main "case" directory, as shown in Figure 2. These files are to be discussed in the following sections in the context of their use in AeroCAT. In these descriptions, the headers of each file are not mentioned, but they are included in the example files found in the Appendix. More information can be found in the OpenFOAM users manual [1].

Running OpenFOAM is simple once it is properly installed and the input files have been created (see below). In a terminal with the OpenFOAM system variables set, navigate to the "case" directory and run the command "simpleFoam" (or whichever solver you choose). Alternatively, from any directory in a terminal, the command "simpleFoam -case <path-to-case-directory>" may be run.

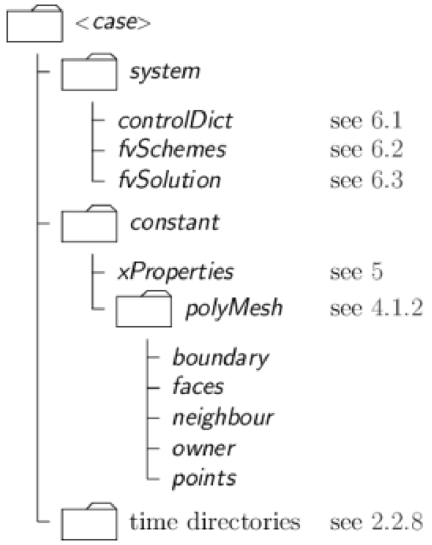

**Figure 2. The file structure of an OpenFOAM case [1].**

### *II.C.1. System Subdirectory*

This directory includes all the information about the solvers, tolerances, and parameters dealing with the execution of the case. The following is a description of the settings used in the cases run by AeroCAT (i.e. steady, incompressible, and inviscid). Examples of these files are found in the Appendix, but for a complete description of all parameter and solver options, search OpenFOAM's User Guide [1].

The **"controlDict"** file stores the information about start and stop time, output file frequency and formatting, and post-processing functions to be run during execution. The following is a list of the needed entries for this file and a brief description of each. An example "controlDict" file is located in the Appendix.

- **"application"**, the name of the solver to be used for the case. "simpleFoam" will be used for AeroCAT.
- **"startFrom"**, the time (or pseudo-time in steady flows) at which the solver should begin. The default is "startTime" (described below), though "firstTime" and "lastTime" are options. These two options are based on the time subdirectories found in the main "case" directory. "firstTime" begins the simulation from the information in the earliest time subdirectory, whereas "lastTime" starts the solver with the data in the last time subdirectory. Thus, to restart a solver from where it left off, the "lastTime" option is selected.
- **"startTime"**, the time at which the solver should begin if the "startFrom startTime" option is selected. The default for this field is "0".
- **"stopAt"**, the time at which the solver should end. The option used by AeroCAT here is "endTime". For more options see the User Guide [1].
- **"endTime"**, the time at which the solver should end when the option "stopAt stopTime" is selected.
- **"deltaT"**, the time step (or pseudo-time step) of the simulation.
- **"writeControl"**, the units assigned to "writeInterval" (described below), controlling the frequency at which data is written. The default is "timeStep", which sets the units to number of time steps (as defined by "deltaT"). Other options include "runTime", for units of seconds of simulated time, "cpuTime", for units of seconds of CPU time, and "clockTime", for units of seconds of real time.
- **"writeInterval"**, the frequency at which data is written to a new time subdirectory throughout the simulation. It has units that are defined by "writeControl".
- **"purgeWrite"**, the number of time subdirectories to remain in the "case" directory, excluding the starting time subdirectory. For example, if "purgeWrite" is equal to five, OpenFOAM will cyclically delete the oldest time subdirectory throughout the course of the simulation, maintaining the five most recent (and the initial) subdirectories. The default for "purgeWrite" is zero, meaning all time subdirectories are kept.
- **"writeFormat"**, the specification of "ascii" or "binary" format for the output files.
- **"writePrecision"**, the number of significant figures used in the data files.
- **"writeCompression"**, the "on" or "off" specification of whether the files are compressed to a ".zip" folder when written.
- **"timeFormat"**, the format for naming the time subdirectories. The options include: "fixed" (x.xxxxxx), "scientific" (x.xxxxxxexx), or "general" which changes between "fixed" and "general" depending on the scientific exponent of the time value and the value specified by "timePrecision".
- **"timePrecision"**, the maximum number of digits after the decimal point in time subdirectory names.
- **"runTimeModifiable"**, the switch that, if "true", tells OpenFOAM to re-read the input files at the beginning of each time-step, allowing for in-simulation modification of parameters and values.

After the above entries, there is an optional "functions" directory. In the example found in the Appendix there are two entries, "#includeFunc residuals" and "#includeFunc forceCoeffsIncompressible". These entries are run-time functions that calculate and output residual and force (lift, drag, and moment) coefficients, respectively, during the simulation. The result is a new subdirectory, "postProcessing", in the "case" directory that contains a data file for each function storing the progression of the desired values throughout the course of the simulation. There are many additional functions that may be included here, and the reader can find a list of these in OpenFOAM's User Guide [1].

The **"fvSchemes"** file houses the definitions for the numerical schemes to be used to calculate each part of the underlying equations. An example of this file may be found in the Appendix. Again, there are many options for each of these entries, but those displayed here are the ones that have been found to work best for "simpleFoam" in the applications of AeroCAT.

The "default" entry in each category is the setting used if nothing is set explicitly below it. In these schemes, "Gauss" refers to finite volume discretization of Gaussian integration. This discretization requires values be interpolated from the cell centers to the cell faces [1]. In the schemes used here, "linear" interpolation, or central differencing, is used almost exclusively.

- **"ddtSchemes"**, the calculation of derivatives with respect to time. Because this is a steady simulation, "steadyState" is used and time derivatives are set to zero.

- **"gradSchemes"**, the calculation of the gradient, "$\nabla$". "Gauss linear" is the default used here.

- **"divSchemes"**, the calculation of the divergence, "$\nabla\cdot$". "Gauss linear" is the default, but "bounded Gauss linearUpwind grad(U)" is specified for the "div(phi,U)" term ( $\nabla \cdot (\phi\mathbf{U}) = \nabla \cdot (\rho\mathbf{UU})$ ).
  In this scheme, the "bounded" refers to the treatment of the $\nabla \cdot \mathbf{U}$ term the material derivative. For incompressible flows, this term equals zero at convergence, but may not equal zero before convergence is reached. So, the "bounded" term in this scheme solves for the $\nabla\cdot\mathbf{U}$ term throughout the simulation.
  This scheme also uses "linearUpwind grad(U)" instead of "linear". The "linearUpwind" term calls for a weighted interpolation scheme that favors upwind values. The "linearUpwind" option requires that the discretization of the velocity gradient be specified, so the "grad(U)" term is referencing the discretization scheme for the velocity specified under "gradSchemes". (In this case it takes on the "default" "gradSchemes" value.) [1]

- **"laplacianSchemes"**, the calculation of the Laplacian, "$\nabla^2$". "Gauss linear corrected" is used for this scheme. Here, the "corrected" term is a "snGradScheme" scheme and will be described below.

- **"interpolationSchemes"**, the calculation of values at a cell face through interpolation of cell-centered values. The "linear" interpolation scheme is used.

- **"snGradSchemes"**, the calculation of the gradient, $\nabla$, normal to a cell face. Here a "corrected" scheme is used. This scheme adds a correction that takes into account any non-orthogonality of the cells in the mesh.

The **"fvSolution"** file specifies the solver type for each variable, as well as the error tolerances, number of corrector steps, and relaxation factors. An example of a "fvSolution" file is found in the Appendix, and a complete list of solvers and settings may again be found in the OpenFOAM User Guide [1].

The first dictionary of this file, "solvers", specifies the linear solver to be used to solve each discretized equation—in this case, "U" and "p"—as well as the absolute and relative tolerances of the solvers. AeroCAT uses the preconditioned conjugate gradient, "PCG", solver with a diagonal incomplete Cholesky, "DIC", preconditioner for the pressure, and a preconditioned bi-conjugate gradient, "PBiCG", solver with a diagonal incomplete LU, "DILU", preconditioner for the velocity. Two different solvers are used because of the symmetric and asymmetric nature of the pressure and velocity matrices, respectively.

For all solvers, "tolerance" specifies the absolute tolerance that the solver should use as an exit condition, and "relTol" dictates a relative tolerance based on the residual of the value when it first enters the solver. In the example file in the Appendix, the "relTol" values have been set to zero, meaning that the linear solvers for pressure and velocity will only exit once the absolute tolerance is achieved, and not because of a significant relative decrease. There is also a term, "nSweeps", that specifies the minimum amount of runs through a solver that should occur each time the solver is called. The example file has this term set to "2" for the velocity solver, and it is not defined for the pressure solver.

After the "solvers" section, the settings of the SIMPLE (or other) algorithm are specified and the "relaxationFactors" are set. The SIMPLE algorithm for incompressible flow requires "pRefCell" and "pRefValue" to define the location and value of the reference pressure. Because of the mesh structure of AeroCAT, the zeroth cell is along an outer boundary and thus is used as the location of the reference pressure. And, because it is only the change in pressure that is of interest, the reference value for the pressure is set to zero. The "residualControl" subsection is where the final convergence criteria is defined. There must be a value defined for "p" and "U" that defines when the simulation has converged.

The SIMPLE algorithm may be adjusted to assist convergence. One way is by turning on the "consistent" switch (as seen in the example file), which changes the algorithm to use the SIMPLEC pressure-velocity coupling method. There is also an option to run additional corrections steps on the SIMPLE algorithm to adjust for any non-orthogonality in the cells, specified by "nNonOrthogonalCorrentors". Finally, the "relaxationFactors" section of this file allows the user to set the the weighting that the SIMPLE algorithm puts on each new iteration versus the previous iteration. The syntax for setting these values is seen in the example file.

Finally, the system subdirectory has the **"force-**

**CoeffsIncompressible"** file (not shown in Figure 2) that contains the definitions of the lift and drag directions, the patches over which the forces should be calculated, and the reference values. As a note, some of the functions, like "residuals" mentioned above, do not require any additional information. But others, like "forceCoeffsIncompressible", require that additional information be stored in the files similar to this one in the "system" subdirectory. An example of the "forceCoeffsIncompressible" file is located in the Appendix.

### *II.C.2. Constant Subdirectory*

The "constant" subdirectory contains a subdirectory, "polyMesh", with the files defining the mesh, and parameter files describing the flow. For the "simpleFoam" solver, the files "transportProperties" and "turbulenceProperties" are required.

The file "transportProperties" contains the single line after its header:

```
nu      [0 2 -1 0 0 0 0]      1e-6;
```

Where "nu" is the kinematic viscosity, $\nu$, the bracketed values describe the units of $\nu$ in base units (i.e. $m^2 \cdot s^{-1}$), and the final number is the value of $\nu$. Because AeroCAT is not meant to model turbulent flow, the file "turbulenceProperties" is simple as well, containing only two lines after the header:

```
transportModel        Newtonian;
simulationType        laminar;
```

There are examples of both of these files in the Appendix to provide an example of the header sections.

The "polyMesh" subdirectory contains five files that define the mesh: "points", "faces", "owner", "neighbour", and "boundary". The Appendix contains an example of each of these files for a mesh that, for simplicity, has only one polyhedral cell (Figure 7). Of course, a more comprehensive description of the mesh format may be found in the User Guide [1].

OpenFOAM meshes are unique in the fact that they do not require a prescribed cell shape. Each cell is described by an arbitrary number of faces that are each defined from an arbitrary number of points. Each face is part of two cells (or a boundary), and thus the cells are strung together through the definition of which cell is the "owner" of a face and which is the "neighbour" (or "boundary"). All of this information is contained in the files mentioned above and described below.

The **"points"** file begins, after the header, with an integer announcing the number of points in the mesh. This is followed by a list of $x$, $y$, $z$ coordinate values enclosed in parenthesis. OpenFOAM assigns numbered IDs to these points based on the order they are found in this file, beginning with "0".

The **"faces"** file also begins with an integer describing the number of entries in the file. It is then followed by a list of cell face descriptions that begin with an integer telling the number of points used in the face description followed by the list of the point IDs that form the face. The order of the points is important, they must be ordered such that the face normal vector is pointing out of the cell—from the cell with the lower numbered ID, the "owner", into the cell with the higher ID number, the "neighbour". The normal is defined using the Right Hand Rule, such that if one were to be looking at the face from inside the owner cell, the points would be listed in the clockwise direction.

As mentioned above, the **"owner"** file contains a list of the owner cells. As with the other files in this subdirectory, this list begins by stating the number of entries within. This number should be the same as the number of faces that is declared in the "faces" file. Below that, the owner cell IDs are listed in the order that corresponds to the order of the faces in the "faces" file. In the example in the Appendix, there is only one cell, cell "0", and thus it owns all seven faces. So, the example "owner" file has a "7" followed by a list of seven "0"s. As with the points, the cell numbering begins with zero.

The **"neighbour"** file is similar to the "owner" file. It announces the number of entries, and then it contains the list of neighbour cells listed in the order corresponding to the order of the faces in the "faces" file. There is a subtle difference with the "neighbour" file, however. For faces that form a boundary, there is no neighbour cell, only an owner, so for these faces the "neighbour" file may contain either a "-1", or no entry at all. For the latter option, all boundary faces should be listed last in the "faces" file. Thus, while the "faces" and "owner" files would have the same number of entries, the "neighbour" file would contain that number minus the number of boundary faces. The practice of listing all the boundary faces last is common due to the way that boundaries are described in OpenFOAM, as is described in the following paragraph.

The **"boundary"** file contains the definitions of all the boundaries in the mesh. Below the header, the number of boundaries is stated followed by the list of boundary descriptions. Each boundary description begins with a name to identify the boundary, followed by the "type" of boundary, the number of faces that form that boundary ("nFaces"), and the ID of the face in the "faces" file where the list of that boundary's faces begins ("startFace"). It is necessary to list the boundary faces in the "faces" file such that all faces of a boundary are together. OpenFOAM recognizes a boundary by beginning at the face with ID "startFace" and counts "nFaces" down the "faces" list. This is why it is common to include all boundary faces at the end of the "faces" file.

The "boundary" file is not where boundary conditions are declared, but rather where boundary types are assigned. The airfoil or wing boundary is a "wall" type, meaning it should be treated as a solid wall. Both the front and back outer boundaries are "patch" type boundaries, meaning they are flow inlets or outlets. For three-dimensional simulation, the symmetry plane is declared a "symmetyPlane" type. In order to run a two-dimensional simulation in OpenFOAM, it is necessary to extrude the two-dimensional mesh to a thickness of one cell and assign the original plane and the new parallel plane to the "empty" type. OpenFOAM will then treat the simulation as a flow in two dimensions. Other boundary types not used by AeroCAT include "cyclic", for repeated geometries, and "wedge", for axi-symmetric flows [1].

As stated above, one of the goals of AeroCAT is the calculation of the locus of aerodynamic centers along the span of a wing. To achieve this, the ring of boundary faces at each span-wise location along the wing is defined as a separate boundary. This allows each ring to be specified individually by the function "forceCoeffsIncompressible", providing span-wise force distributions on the wing. This method of boundary definition can be seen in Figure 3.

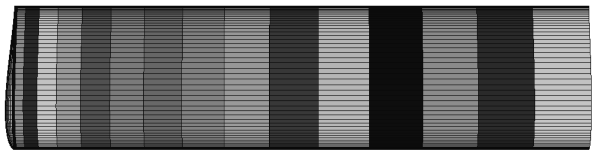

**Figure 3. A view from above of a rectangular 3D wing defined with each span-wise ring of boundary faces as a separate boundary. Each shade of gray represents a different boundary patch with the endcap on the left and the root on the right.**

### *II.C.3. Time Subdirectories*

The time subdirectories hold the start flow and boundary condition information for a case. They also store the data output by OpenFOAM throughout the simulation. For the cases run by AeroCAT, the initial conditions are saved in the "0" time directory, and the output results are stored in directories whose name corresponds to the time/iteration for which the data was calculated. For laminar "simpleFoam", the OpenFOAM solver used by AeroCAT, the files "p" and "U" need to be found in the "0" subdirectory.

The "U" and "p" files found in the "0" subdirectory initialize the velocity and pressure of the flow and assign conditions to each boundary. Following the header, the file begins with a "dimensions" description, which for velocity is "[0 1 -1 0 0 0 0]" ($m^1 \cdot s^{-1}$) and for pressure is "[0 2 -2 0 0 0 0]" ($m^2 \cdot s{-}2$). This is followed by the term "internalField", which is the description of what the flow inside the domain should initialize as. To improve convergence for the steady flow studied by AeroCAT, the entire internal flow is initialized to the freestream velocity and to reference pressure. This is done by the term "uniform" followed by the $x$, $y$, $z$ coordinate values of the desired velocity vector or by the scalar reference pressure value.

To set the boundary conditions in the "boundaryField" entry, a list of the boundary names, defined in the "boundary" file, is given with a "type" for each. In the AeroCAT implementation, the front and back boundaries are set to "freestream" for the velocity and "freestreamPressure" for the pressure. These two boundary conditions calculate the flux of the flow through the boundary and, if the flow is into the domain, the boundary holds a fixed velocity and pressure, but, if the flow is exiting the domain, the boundary enforces a zero-gradient condition for both velocity and pressure. For two-dimensional flow, AeroCAT sets the "type" for the airfoil to a "slip" condition to more accurately model inviscid flow and it sets the sides to an "empty" type, as discussed above. For three-dimensional flow, the wing and endcap are again set as "slip" boundaries and the symmetry plane is of type "symmetry". Examples of both the "U" and "p" files are located in the Appendix.

### *II.C.4. Description of AeroCAT Input File Options for OpenFOAM*

The AeroCAT input file options that correspond to the OpenFOAM portion of the program are:

- **"Convergence"**, the value to be used in the "residualControl" section of the "fvSolution" file. It is the convergence criteria for the solver.
- **"Sol_freq"**, the value to be used for the "writeInterval" term in the "controlDict" file.
- **"Velocity"**, the magnitude of the freestream velocity, in meters per second.

As with GridX, **"AoA_max"** and **"AoA_min"** in the AeroCAT input file dictate the range of angle of attack for which simulations will be run, so they indirectly correspond to OpenFOAM input values.

## II.D. Post-Processing

With the mesh generated and the simulation run, AeroCAT executes several post-processing routines to calculate the aerodynamic center of an airfoil—or locus of aerodynamic centers along a wing—and generate plots of the results. AeroCAT first reads the post-processing data files that result from the functions "residuals" and "forceCoeffsIncompressible", described previously. It compiles all this data into several condensed files that then are used for the aerodynamic center calculations.

### II.D.1. *Calculation of the Aerodynamic Center*

The aerodynamic center of an airfoil, or wing, is the point, or locus of points, at which the pitching moment remains constant with small variations of $\alpha$. The location of this point is valuable in aircraft stability analysis. The general equations used to calculate the aerodynamic center of an airfoil are [3]:

$$\bar{x}_{ac} = \frac{C_{A,\alpha}C_{m,\alpha,\alpha} - C_{m,\alpha}C_{A,\alpha,\alpha}}{C_{N,\alpha}C_{A,\alpha,\alpha} - C_{A,\alpha}C_{N,\alpha,\alpha}} \cdot c_{ref} \tag{1}$$

$$\bar{y}_{ac} = \frac{C_{N,\alpha}C_{m,\alpha,\alpha} - C_{m,\alpha}C_{N,\alpha,\alpha}}{C_{N,\alpha}C_{A,\alpha,\alpha} - C_{A,\alpha}C_{N,\alpha,\alpha}} \cdot c_{ref} \tag{2}$$

$$C_{m,ac} \cdot c_{ref} = C_m c_{ref} + \bar{x}_{ac}C_N - \bar{y}_{ac}C_A \tag{3}$$

The equations (1), (2), and (3) are functions of the axial and normal force coefficients, the coefficient of moment about the leading edge, and their derivatives with respect to $\alpha$. Because of this, AeroCAT sets "liftDir" and "dragDir" in the "forceCoeffsIncompressible" file to the normal and axial directions of the wing, (0, 1, 0) and (1, 0, 0) respectively, instead of the true lift and drag directions.

After running simulations through a range of $\alpha$, AeroCAT calculates the necessary derivatives and calculates the $x$ and $y$ coordinates of the aerodynamic center for each angle of attack, $\alpha$. Current research in the USU AeroLab is focused on the development of analytical equations that more accurately describe a lifting surface's lift and drag coefficients as a function of $\alpha$ [4]. Pending the conclusion of that research, AeroCAT will perform a Least-Squares Regression with the data points gathered from OpenFOAM, calculating coefficients that will fit the equations developed by the AeroLab to those data points. These fit equations will then be differentiated to obtain the first and second derivatives needed in Equations (1), (2), and (3).

Having completed the computations, plots are generated and data output files are created to provide a summary of the AeroCAT project. This summary includes plots of the lift, drag, and moment coefficients as a function of angle of attack, the $x$ and $y$ coordinates of the aerodynamic center for each angle of attack and for each span-wise cross-section, and the value of the residuals for each OpenFOAM case as a function of iteration (or pseudo time). The plotted data is also compiled and written to results data files for further review and processing by the user.

### II.D.2. *Description of AeroCAT Input File Options for Post-Processing*

The AeroCAT input file options that correspond to the post-processing portion of the program are:

- **"Aero_Center_yn"**, a "y" or "n" character denoting whether aerodynamic center calculations should be executed.
- **"AoA_max"**, the maximum angle of attack to be simulated, in degrees.
- **"AoA_min"**, the minimum angle of attack to be simulated, in degrees. [4]

The post-processing subroutines also utilize the input file options that describe the airfoil and wing geometry (e.g. "NACA", "Sweep", "Chord", etc.) to draw outlines of the wing or airfoil on certain plots.

## III. Validation

Using the GridX and OpenFOAM settings described in the previous sections, cases were run to determine the validity of the AeroCAT simulations for steady, incompressible, inviscid flow. A two-dimensional Joukowski airfoil was chosen as the base of comparison since it has an analytical solution for inviscid flow.

The first test was to verify that the lift, drag, and moment coefficients grid resolve to the correct values. To confirm this, cases where run at three angles of attack, each with a "course", "medium", and "fine" mesh. Because of the systematic removal of nodes to coarsen the mesh, a Richardson Extrapolation prediction was able to be calculated at each angle of attack as well [2]. In Figures 4, 5, and 6, it can be seen that OpenFOAM produces results that converge to the analytical solutions of the lift, drag, and moment coefficients as the grid refines. Thus, through proper selection of the mesh settings in GridX, appropriate results will be produced by OpenFOAM.

The next test on AeroCAT was to compare it's results to those of the commercial CFD package Star CCM+. OpenFOAM does not have an explicitly inviscid solver for incompressible flow, so inviscid flow needs to be simulated by manually adjusting the appropriate settings. First, from testing, it has been found that a Reynolds number of one million is the upper limit for which OpenFOAM will converge to

[4]AeroCAT is set to run simulations from "AoA_min" to "AoA_max" using a step size of one degree.

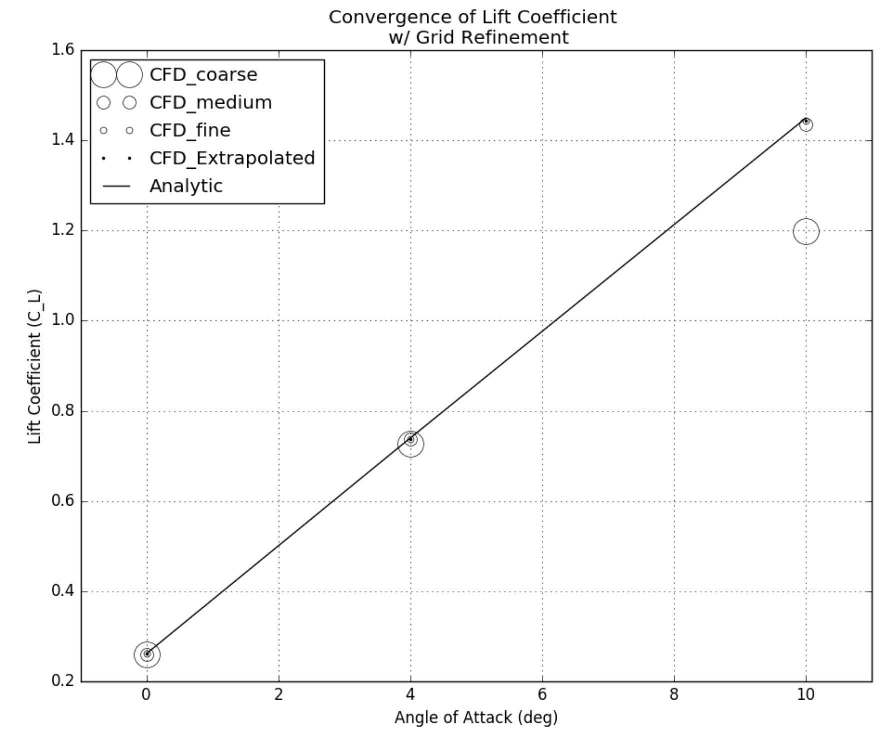

**Figure 4. Two-dimensional grid refinement convergence of the lift coefficient at $0^\circ$, $4^\circ$, and $10^\circ$ angle of attack, using OpenFOAM.**

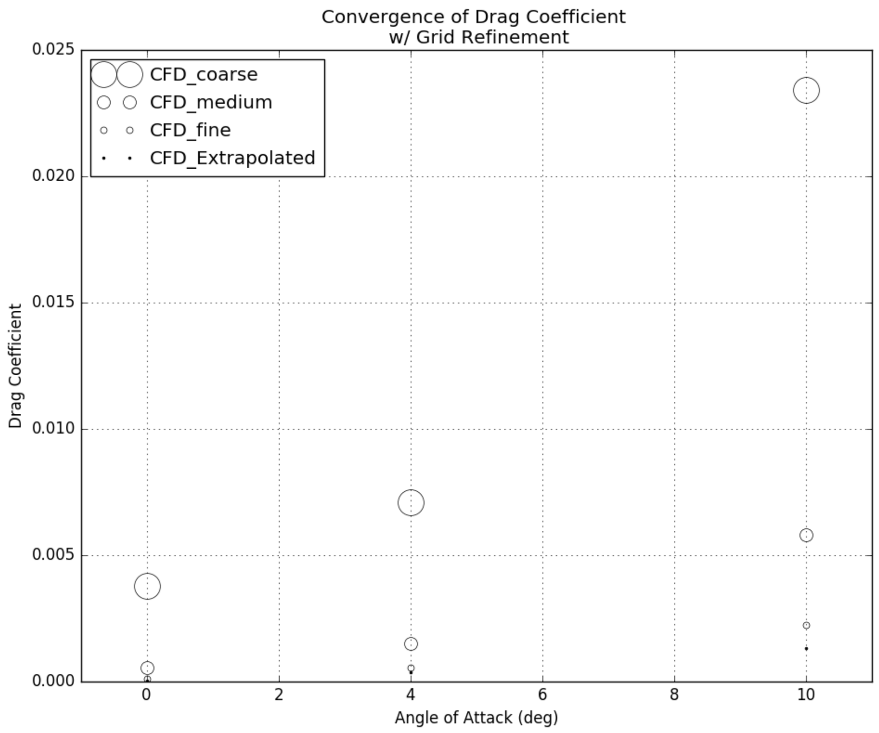

**Figure 5. Two-dimensional grid refinement convergence of the drag coefficient at $0^\circ$, $4^\circ$, and $10^\circ$ angle of attack, using OpenFOAM. Here the analytical solution is equal to zero, and thus it can be thought of as the horizontal axis.**

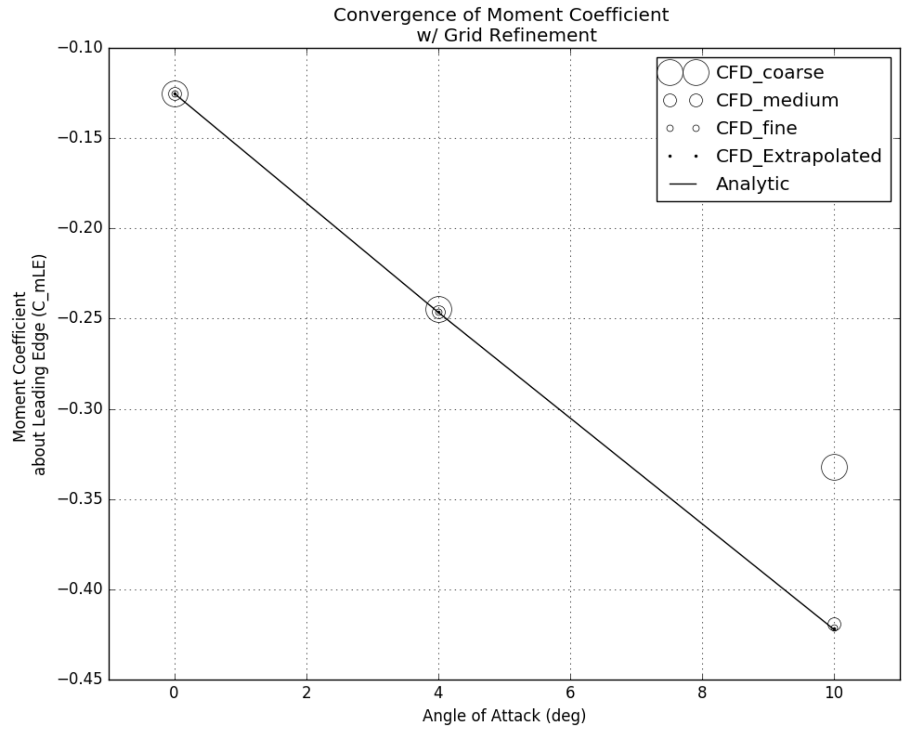

**Figure 6. Two-dimensional grid refinement convergence of the moment coefficient about the leading edge at $0°$, $4°$, and $10°$ angle of attack, using OpenFOAM.**

a steady-state answer for the aerodynamic flow AeroCAT simulates. Since a truly inviscid flow has a Reynolds number of infinity, this upper limit is maintained for the simulations. In addition, a "slip" condition is applied on the airfoil or wing, and "laminar" is specified as the "simulationType". The comparison of results with Star CCM+—which does has an explicit inviscid flow solver—was, then, a way to ensure that these settings were sufficient in approximating the inviscid flow AeroCAT was designed to model.

Table 1 shows the results of four methods used to calculate the lift and drag coefficients of a Joukowski airfoil at ten degrees angle of attack. Ten degrees angle of attack was used as the basis of comparison because viscous effects have a more obvious affect on the flow at higher angles, as opposed to angles closer to zero. The same "fine" grid was used for both OpenFOAM and Star CCM+. All results were compared to the analytical solution for the Joukowski airfoil, and OpenFOAM proved to accurately predict the lift and drag coefficients as well, and in fact better, than Star CCM+.

**Table 1. Comparison of Lift and Drag Coefficient Results for a Joukowski Airfoil (12% thickness and 0.261 target $C_L$) at $\alpha = 10°$.**

| Method | $C_L$(%Error) | $C_D$(%Error) |
|---|---|---|
| Analytical | 1.44916 | 0 |
| Vortex Panel | (+0.25%) | (±0.00%) |
| OpenFOAM | (–0.38%) | (+0.22%) |
| Star CCM+ | (–1.20%) | (+0.41%) |

## IV. Conclusion

The USU AeroLab's Aerodynamic Center Analysis Tool, AeroCAT, utilizes the meshing utility GridX and the CFD toolbox OpenFOAM to simulate steady, inviscid, incompressible flow over wings and airfoils, and post-processes the data to calculate the span-wise locus of aerodynamic centers. The results compiled by AeroCAT from the OpenFOAM simulations, as well as the post-processing calculations of the aerodynamic centers provides the user with access to the information necessary to better perform stability analysis and/or validate other analytic or numeric models. AeroCAT will continue to expand and develop, branching out to compressible and viscous flows, and gaining the functionality needed to model more complicated wing designs. ✈

## V. Appendix

Examples of the files needed, and used by, AeroCAT, GridX, and OpenFOAM.

### V.A. Example of an AeroCAT .json input file:

```
{
  "Aero_Center_yn": "y",
  "AR": 8.0,
  "AoA_max": 10,
  "AoA_min": −7,
  "Chord": 1.0,
  "Convergence": −9.0,
  "Dihedral": 0.0,
  "flap_yn": "n",
  "flap_deg": 10.0,
  "flap_hinge": 3,
  "flap_len": 0.2,
  "joukowski_yn": ''y'',
  "jk_thick": 0.12,
  "jk_lift": 0.26,
  "Mesh_density": "fine",
  "Mesh_dim": 1,
  "NACA": 4412,
  "Semi_span": 20,
  "Sol_freq": 100,
  "Sweep": 0.0,
  "TR": 1.0,
  "Velocity": 1.0,
  "Washout": 0.0,
  "chord_bl2ff": 10,
  "node_airfoil": 400,
  "node_behind": 90,
  "node_bl2ff": 90,
  "node_boundary": 10,
  "node_endcap": 15,
  "thickness1_bl": 5e−05
}
```

### V.B. Example of an GridX .iGridX input file:

```
Align all character variables with the far right margin

<===Program Defaults===>
n |Output formatted Plot3D grid?
n |Output SU2 grid?
y |Adjust grid sizes to accommodate three-level multigridability?
n |Enable FIDAP output?
n |Output CFL3D input files?
n |Output SU2 configure file?
y |Output OpenFOAM mesh files?
C |2-D Grid Type (O,C)
2 |Grid dimension default (2 or 3)
n |Use 2D grid to generate 3D infinite wing for SU2?
2 |Grid generator default
| 1=> Alley Algebraic.
| 2=> Phillips-Alley Algebraic.
| 3=> Elliptic PDE (Alley base).
| 4=> Elliptic PDE (Phillips-Alley base).
n |Do you want to include a flap deflection?
n |Do you want to see an airfoil plot?
n |Do you want to see the streamlines?
n |Do you want to generate an airfoil template?
n |Do you want to include a coanda port?
n |Do you want to add a Coanda flap?
n |Do you want to add a dual-radius flap?
n |Do you want to add a jet flap?

<===Freestream/Coanda Jet Properties===>
3 |Input/Output units flag (1=English-inches, 2=English-feet, 3=SI-meters)
10.0 |Altitude (ft, ft, m)
0.0 |Angle of Attack (deg)
0 |Sideslip angle, 3D-full grids only (deg)
0 |Freestream Reynolds Number (enter 0 to specify using velocity)
1.0 |Freestream Velocity (m/s)
1.225 |Density of freestream (kg/m**3)
1.81d-5 |Viscosity of freestream (N*s/m**2)
518.670 |Temperature of freestream (deg R)
159.422 |Gas constant of air (J/kg*deg R)
1.4 |Specific heat ratio for air at freestream temp (gamma)

<===User Defined Inputs===>
<–Reference Length—>
1.000d0 |Airfoil/Mean Chord Length (meters)

<–Airfoil Generator–>
200 |Points on generated airfoil |<-Set this value to a negative
5 |Airfoil selection default |number if airfoil is "specified
| 1=> NACA 4-digit series airfoil |by data" and you wish to use
| 2=> USU 12-digit series airfoil |the data points in the data file
| 3=> Airfoil specified by data |
| 4=> USU DBF2001 series airfoil |
| 5=> Joukowski airfoil |
| 6=> Joukowski cylinder/ellipse |
4412 |Default NACA airfoil
303012-1124.13 |Default USU airofil
NACA65A008.dat |Airfoil data filename (max 40 characters!!!)
0.12 |Default max thickness/chord for Joukowski airfoil/cylinder
0.261 |Default design lift coefficient for Joukowski airfoil
0. |Default percent circular-arc camber for Joukowski ellipse
10.0 |Flap angle (degrees)
0.20 |Flap length (l/c)
3 |Flap hinge location
| 1=> Upper Surface
| 2=> Lower Surface
| 3=> Surface Centered
1 |Close airfoil trailing edge (0=disable, 1=enable)
25 |Distance to offset TE center point when closing open TE (%TE thickeness)

<—Wing Generator—>
1 |Wing generation flag
| 1=> Specify RA,RT,Sweep,Twist
| 2=> Define wing geometry using data file
8.0 |Wing aspect ratio
1.0 |Wing taper ratio
0.0 |Wing dihedral (degrees)
0.0 |Wing sweep (degrees)
0.0 |Wing total twist-positive washout (degrees)
20 |Number of semi-span nodal points
wing.dat |Wing data filename (max 40 characters!!!)
15 |Number of endcap nodal points (0 for matched spacing)
2 |Endcap type
| 1=> Flat endcap
| 2=> Rounded endcap
| 3=> Specified by data
1 |Grid symmetry
| 1=> Symmetric centerline boundary
| 2=> Full grid

<——Airfoil——->
200 |Points on the airfoil ( points>=50 )
1.05d0 |Clustering parameter 1<beta<inf. As beta->1, clustering
| will occur near airfoil's LE and TE (BetaAF)
| Set to negative value to specify LE spacing
0.075 | Leading edge node spacing in % of local chord (dLE)
| Recommend 0.1 or less [used only when BetaAF<0]

<—TE Streamline—->
90 |Number of points on the TE streamline
y |Trailing Edge stagnation streamline generator
| y=> Generated using panel code
| n=> Generated algebraically
1.05d0 |TE streamline node clustering parameter
0.001d0 |Initial step size for TE streamline generator
1.1d0 |TE streamline generator growth factor
1.0000 |Trailing edge stagnation point location (x/c)
| [For Joukowski cylinder/ellipse only]

<—Boundary Layer—>
1 |Nodes within Boundary Layer (JB)
5.d-4 |Thickness of first cell (db/c)
1.25d0 |Growth Factor (BetaBL)
0.001 |Left-hand influence length->distance from node in the neg-
```

```
ative
| i-direction that is used to define the surface normal (dZL/c)
0.001 |Right-hand influence length->distance from node in the pos-
tive
| i-direction that is used to define the surface normal (dZR/c)
|Boundary Layer influence lengths are equal to the jet size (tj/c)
| for O-grids containing Coanda jets (dZL=dZR=tj)
|If boundary layer nodes are crossing, reduce JB,db,or BetaBL,
| or increase dZL and dZR. Increasing dZL and dZR might cause
| boundary-layer cells to become skewed. Reducing dZL and dZR
| will force the boundary to be orthogonal to the surface but
| increases the chance of crossed nodes.
n |Expand trailing boundary layer?
5.0e-3 |Thickness of the j=1 boundary layer cells at i=1 and i=IM
(dEBL)
1.1d0 |Growth factor of the boundary layer cells at i=1 and i=IM
(BetaEBL)
15 |Number of nodes past trailing edge where expansion begins
(EBLs)

<—–Inner Grid/Outer Boundary—–>
90 |Nodes between the outer boundary
| and the boundary layer (JIG)
10 |Number of chord lengths to outer boundary (I_chord)
0. |Angle that outer boundary O-grid joint line is rotated from
|horizontal (degrees)
1.25d0 |Transition growth factor for Phillips-Alley method (Tgf>0)
| Larger values of Tgf will increase the occurrence of folded nodes
| but will cause the grid to be more orthogonal to the surface
| The "transition" region is where the grid changes direction
| from orthogonal to the airfoil surface to a straight line
| toward the outer boundary.
0.001d0 |Clustering parameter 0<beta<inf, as beta->inf
| boundary node clustering will occur near the
| leading edge of the boundary
| NOTE: For Joukowski Cylinders, 1<beta<inf, as beta->1
| boundary node clustering will occur near the ends
| (i=1 and IM) of the boundary (BetaOB)
1 |match_IG=1 will match the first row width of the algebraic
| inner grid to the last row in the boundary layer
1.06 |Clustering parameter 1<beta<inf, as beta->1 algebraic inner
node
| clustering will occur near the airfoil: for match_IG=0 only (Be-
taIG)

<—-Elliptic PDE—->
0.20d0 |Forcing Parameter (FPO>0) as FPO->0 orthogonality is
forced.
0.20d0 |Forcing Parameter (FPC>0) as FPC->0 clustering near
airfoil is forced.
0.1d0 |Relaxation factor (lambda)
0.08d0 |Relaxation factor (wp)
0.08d0 |Relaxation factor (wq)
1.d-3 |Convergance Criteria (total x-y error < conv)
100 |Maximum number of allowable iterations

<—-Output Files—->
jk_0 |Output filename w/o extension (leave blank to use .GEN
filename)
.FDNEUT ==>FIDAP grid output
.p3D ==>Formatted Plot3D grid output
.unf ==>Unformatted Plot3D grid output
.oGridX ==>Name of elliptic-inner-grid file that can be used in
OuterGrid.exe
.inp ==>CFL3D input file

<——CFL3D/SU2/OpenFOAM Inputs——>
5. |Steady CFL Number (Default=5)
-99. |Moment center in X direction,positive back (Set to -99 to
compute at root 1/4-chord) (m)
0. |Moment center in Y direction,positive up (m)
0. |Moment center in Z direction,positive out left wing (m)
-99. |Reference area (Set to -99 to for grid dimensions)
-99. |Lateral reference length (Set to -99 to for grid dimensions)
-99. |Longitudinal reference length (Set to -99 to for grid dimen-
sions)
1000 |Number of fine-grid iterations
1000 |Number of medium-grid iterations
1000 |Number of coarse-grid iterations
1000 |Number of coarsest-grid iterations
fine |SU2/OpenFOAM mesh density (fine, medium, coarse)
-6 |SU2/OpenFOAM residual convergence criteria
10 |Frequency of SU2/OpenFOAM solution files

<——Coanda Jet Grid——>
0.85d0 |Coanda port location from leading edge (x/c)
0.00125d0 |Coanda port thickness (t/c)
0.95 |Airfoil chord length with dual-radius flap
0.4 |Blowing momentum coefficient (Cmu)
518.670 |Jet stagnation temperature (deg R)
20 |Nodes within Jet Layer (JJ)
1.d-5 |Thickness of first cell (dj/c)
0 |Nodes aft of coanda port (0 for auto spacing)
1 |match_J=1 will match the node clustering on the
| airfoil aft of the jet to the clustering in
| front the jet
2.55d0 |Clustering parameter 1<beta<inf, as beta->1
| clustering will occur near the jet and the
| TE: for match_J=0 only (BetaJ)

<——Jet Flap Grid——>
1.d-5 |Thickness of first cell (djf/c)
| NOTE: It is recommended that the thickness of the first
| cell in the boundary layer, "db", and "djf" be equal
100 |Number of nodes along fore-lower airfoil boundary, between
leading edge and upper duct inlet (Iafl)
160 |Number of nodes along mid-lower airfoil boundary, between
upper and lower duct inlets (Iaml)
160 |Number of nodes along aft-lower airfoil boundary, between
traling edge of lower jet flap and lower duct inlet (Iaal)
-20 |Number of nodes along upper surface of airfoil, between lead-
ing edge and trailing edge of upper jet flap (Iatu)
| Set to a negative value to equal the total number of nodes along
the lower surface
15. |Thrust vector angle, positive down (degrees)
0.63042 |Axial location of upper jet flap hinge (xjfu/c)
0.63042 |Axial location of lower jet flap hinge (xjfl/c)
0.00800 |Jet flap airfoil wall thickness (twj/c)
0.07948 |Ejector (exit) nozzle height (tej/c)
0.02323 |Inner nozzle throat height (ttj/c)
0.03843 |Inner nozzle total height, includes inner nozzle wall thick-
ness (tnj/c)
100 |Number of axial nodes within nozzle (Inoz)
40 |Number of radial nodes in the inner nozzle throat (Jint)
20 |Number of radial nodes in the inner nozzle walls (Jinw)
0.01412 |Upper bypass inlet height (tubi/c)
0.01782 |Upper bypass duct height (tubd/c)
100 |Number of axial nodes within the upper bypass duct (Iubd)
40 |Number of radial nodes within the upper bypass duct (Jubd)
0.01225 |Lower bypass inlet height (tlbi/c)
0.01552 |Lower bypass duct height (tlbd/c)
60 |Number of axial nodes within the lower bypass duct (Ilbd)
40 |Number of radial nodes within the lower bypass duct (Jlbd)

<—-OuterGrid Generator—-> (For 3-D Joukowski cylinders only)
40 |Number of chords from airfoil to outer boudary (I_ochord)
16 |Number of additional wake nodes (jadd)
16 |Number of additional radial nodes (kadd)
```

## V.C. Example of "controlDict"

```
FoamFile
{
    version       2.0;
    format        ascii;
    class         dictionary;
    location      "system";
    object        controlDict;
}

application       simpleFoam;

startFrom         startTime;

startTime         0;

stopAt            endTime;

endTime           300;
```

```
deltaT            0.1;

writeControl      runTime;

writeInterval     10;

purgeWrite        0;

writeFormat       ascii;

writePrecision    6;

writeCompression  off;

timeFormat        general;

timePrecision     6;

runTimeModifiable true;

functions
{
#includeFunc      residuals
#includeFunc      forceCoeffsIncompressible
}
```

### V.D. Example of "fvSchemes"

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}

ddtSchemes
{
 default    steadyState;
}

gradSchemes
{
 default    Gauss linear;
}

divSchemes
{
 default    Gauss linear;
 div(phi,U)    bounded Gauss linearUpwind grad(U);
}

laplacianSchemes
{
 default    Gauss linear corrected;
}

interpolationSchemes
{
 default    linear;
}

snGradSchemes
{
 default    corrected;
}
```

### V.E. Example of "fvSolution"

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC
        tolerance       1e-06;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU
        nSweeps         2;
        tolerance       1e-08;
        relTol          0;
    }

}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;

    consistent yes;

    residualControl
    {
        p               1e-6;
        U               1e-6;
    }
}

relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        U               0.7;
    }
}
```

### V.F. Example of "forceCoeffsIncompressible"

```
patches     (airfoil);

magUInf     1;
lRef        1;
Aref        .1;

liftDir     (0 1 0);
dragDir     (1 0 0);
```

```
CofR          (0 0 0);
pitchAxis     (0 0 −1);

#includeEtc "caseDicts/postProcessing/...
                    forces/forceCoeffs.cfg"
```

## V.G. Example of "transportProperties"

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}

transportModel  Newtonian;

nu              [0 2 −1 0 0 0 0] 1e−6;
```

## V.H. Example of "turbulenceProperties"

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}

simulationType  laminar;
```

## V.I. Example of the Files in the "polyMesh" Subdirectory

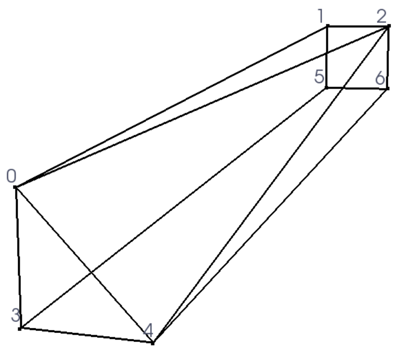

**Figure 7. A figure of the example single-cell mesh generated by the files "points", "faces", "owner", "neighbour", and "boundary" found in the Appendix.**

### *V.I.1. Example of "points"*

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       vectorField;
    location    "constant/polyMesh";
    object      points;
}

7
(
 ( 1     0       −4      )
 (0.9    0.01    −4      )
 (0.9    0.01    −4.01   )
 ( 1     −0.01   −4      )
 ( 1     −0.01   −4.01   )
 (0.9    0       −4      )
 (0.9    0       −4.01   )
)
```

### *V.I.2. Example of "faces"*

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       faceList;
    location    "constant/polyMesh";
    object      points;
}

7
(
 3( 1  0  2 )
 3( 2  0  4 )
 3( 4  6  2 )
 4( 1  2  6  5 )
 4( 5  6  4  3 )
 4( 1  5  3  0 )
 3( 0  3  4 )
)
```

### *V.I.3. Example of "owner"*

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       labelList;
    location    "constant/polyMesh";
    object      points;
}

7
(
  0
  0
  0
  0
  0
  0
  0
)
```

### *V.I.4. Example of "neighbour"*

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       labelList;
    location    "constant/polyMesh";
    object      points;
}

7
(
  −1
  −1
  −1
  −1
  −1
  −1
  −1
)
```

### *V.I.5. Example of "boundary"*

```
FoamFile
{
    version       2.0;
    format        ascii;
    class         polyBoundaryMesh;
    location      "constant/polyMesh";
    object        points;
}

6
(
    top
    {
         type               empty;
         nFaces             1;
         startFace          0;
    }

    right
    {
         type               wall;
         nFaces             2;
         startFace          1;
    }

    front
    {
         type               patch;
         nFaces             1;
         startFace          3;
    }

    bottom
    {
         type               empty;
         nFaces             1;
         startFace          4;
    }

    left
    {
         type               symmetryPlane;
         nFaces             1;
         startFace          5;
    }

    back
    {
         type               patch;
         nFaces             1;
         startFace          6;
    }
)
```

## V.J. Example of "U" from Time Subdirectory "0"

```
FoamFile
{
    version       2.0;
    format        ascii;
    class         volVectorField;
    object        U;
}

dimensions        [0 1 −1 0 0 0 0];

internalField     uniform (1 0 0);

boundaryField
{
    back
    {
        type               freestream;
        freestreamValue    uniform (1 0 0);
    }

    front
    {
        type               freestream;
        freestreamValue    uniform (1 0 0);
    }

    airfoil
    {
        type               slip;
    }

    sides
    {
        type               empty;
    }
}
```

## V.K. Example of "p" from Time Subdirectory "0"

```
FoamFile
{
    version       2.0;
    format        ascii;
    class         volScalarField;
    object        p;
}

dimensions        [0 2 −2 0 0 0 0];

internalField     uniform 0;

boundaryField
{
    back
    {
        type               freestreamPressure;
    }

    front
    {
        type               freestreamPressure;
    }

    airfoil
    {
        type            zeroGradient;
    }

    sides
    {
        type            empty;
    }
}
```

## Acknowledgments

The author is extremely grateful to his fellow students in the USU AeroLab for sharing their knowledge and suggestions.

## References

[1] OpenFOAM Ltd. User guide. *http://www.openfoam.com/documentation/user-guide/*, April 2017.

[2] Warren F. Phillips. Minimizing induced drag with wing twist, computational-fluid-dynamics validation. *Journal of Aircraft*, 43(2), March-April 2006.

[3] Warren F. Phillips. *Mechanics of Flight.* John Wiley & Sons, Inc., 2nd edition, 2010.

[4] Orrin D. Pope. The aerodynamic center of inviscid airfoils. In *Utah NASA Space Grant Consortium Fellowship Symposium*, 2017.