RANSFOIL User's Guide
Version 2.2

Hou Shuo

July 3, 2022

Contents

The licence of RANSFOIL is 3-clause BSD, please read the **COPYRIGHT** file in the root directory before using this program. This is the English version of RANSFOIL user's guide.

1 General Description

RANSFOIL is a console program to calculate airflow field around an isolated airfoil in low-speed, subsonic, transonic or supersonic regime by numerically solving the Reynolds averaged Navier-Stokes (RANS) equations. As postprocess results, the aerodynamic parameters of the airfoil, e.g., lift coefficient and drag coefficient, can be integrated from the airflow distribution near wall boundary. In addition, the coupling between airflow and heat transfer and the effect of surface roughness on near wall flow are considered by this program, it can be used for engineering calculation of the friction drag and heat transfer coefficients on airfoil surface. RANSFOIL reads coordinates data from a 1D *XYZ* file which describes the grid points or control points on airfoil as only input, then outputs grid and solution files in 2D *PLOT3D* format to record grid coordinates and airflow data and a report file to show aerodynamic parameters.

2 Installation

The RANSFOIL package uses GNU Autotools to deploy on Linux and Windows platforms.

On Linux systems, the familiar three shell commands (`./configure; make; make install`) should configure, build, and install this package. More-detailed generic instructions can be found in the **INSTALL** file, the following are specific options to this package.

This package uses GNU Libtool to build shared libraries. By default, the `make` command should generate a shared library named **libaero2d** and an executable file named **ransfoil** (depend on **libaero2d**). You can specify the `--disable-shared` option to `configure` command to turn off building shared library and the `--enable-static` option to build a static library instead

```
$./configure --disable-shared --enable-static
```

You can specify the `--enable-hyprelib` option to `configure` command to enable the LLNL's HYPRE library for solving sparse linear equations. If you decide to build with HYPRE library, you must have a binary version of HYPRE library installed and make sure that the `--with-hyprelib-dir=DIR`

option is used to assign a root installed directory `DIR` to the HYPRE library (This package can not search the location of HYPRE automatically in your file system)

```
$./configure --enable-hyprelib --with-hyprelib-dir=DIR
```

You can specify the `--enable-openmp` option to `configure` command to enable OpenMP parallelization. At present only the GNU OpenMP implementation is supported in this package.

```
$./configure --enable-openmp
```

If you want to use other OpenMP implementation, pass the appropriate `LDFLAGS` option to `configure` command.

You can pass `--enable-debug` and `--enable-profile` options to `configure` command to compile executable file suitable for debugging and performance test, respectively.

```
$./configure --enable-debug
$./configure --enable-profile
```

The compiler options specific to `gdb` and `gprof` are used by default.

Because the default compiler options `FCFLAGS` in this package are specially optimized for GNU gfortran compiler, if you want to use other Fortran compiler, pass the `FC=XXX` and `FCFLAGS=XXX` options to `configure` command to make sure the corresponding options are correctly used

```
$./configure FC=XXX FCFLAGS=XXX
```

Note that `ieee_arithmetic` module is used since the version of 2.2.1, the gcc and gfortran compiler version should be newer than 5.0 for correct compilation.

On Windows systems, you should have a MinGW+msys or a Cygwin environment. The build steps are the same as on Linux systems. A dynamic linking library (DLL) may be generated on Windows instead of shared library on Linux. The binary package on Windows released with the source package is configured with the default parameters, namely that no option is passed to the `configure` command. The following commands are used for building the binary package

```
$./configure
$make
```

On other Unix-like systems, the above build steps may also work, but the author did not test on these platforms.

3 Usage

RANSFOIL is a file driven command-line interface program. You can see the basic information of usage by typing `ransfoil` on terminal after the

executable file is built

```
$./ransfoil
usage:  ransfoil [options] [configuration file]
Options:
--stdin    interactive mode, read parameters from standard input
--script   batch mode, read parameters from a configuration file
--version  show the current version
-h, --help show this message
```

This program has two execution mode, an interactive mode and a batch mode.

You can use the `--stdin` option to access in interactive mode

```
$./ransfoil --stdin
```

Some control parameters should be input by the user, they are characters to select model and algorithm or numbers to set ambient conditions. There will be a hint message displayed on the terminal screen before you input corresponding parameters. For alphabetic parameters, some optional characters are enclosed in parentheses (split by slash) at the end of the hint message and you can select one of them to configure model or algorithm. For numerical parameters, the units are given at the end of the hint message (the hint message without unit indicates the numerical parameter is dimensionless). You can use any legal floating-point number format (without the units), e.g., 0.1 or 1e-1, to input numerical parameters. Any additional characters and numbers will be considered as illegal by the program.

You can use the `--script` option to access in batch mode. For example, if you have a configuration file named **NACA0012.config**, you can use the following command to execute RANSFOIL

```
$./ransfoil --script NACA0012.config
```

The program will read all control parameters from the configuration file and then start iteration, the user doesn't need to interact until the program is terminated. The format of the configuration file is plain text that is similar to interactive mode. The difference is that in batch mode you should input all the control parameters in configuration file at first. The configuration file should be organized using an interlaced format, for example

---

-
NACA0012.xyz
-
N
-
incom
-

sa
-
...
-
100.0
-
.
.
.

The sequence of the control parameters must be same as that displayed in interactive mode, and the first line in this configuration file should be blank. You can input text in the blank lines as hint message and the program will ignore them. These is a sample configuration file named **ransfoil.configxyz** in this package as template for more detailed use.

You can use the `-h, --help` options or input the executable name alone to show some brief description and usage information

```
$./ransfoil --help
```

The option `--version` can be used to see the current version number of RANSFOIL

```
$./ransfoil --version
```

4 Input file

Before execution, the only input you should prepare is a data file which records the coordinates of grid points on airfoil. This file should be in 1D *XYZ* format (not the *xfoil* format) which divides the coordinates data into two segments: lower airfoil and upper airfoil. The low and upper segment must constitute a closed airfoil. Each segment should start with a number that describes the number of grid points on this part following X-Y coordinate pairs from the leading edge to the trailing edge, for example

---

number of grid points on lower airfoil (N)
X(1) Y(1)
X(2) Y(2)
. .
. .
X(N) Y(N)
number of grid points on upper airfoil (N)
X(1) Y(1)

X(2) Y(2)
. .
. .
X(N) Y(N)

The airfoil coordinates should be dimensionless with a chord length of unit 1. The coordinates data describes a grid line, and the final computational mesh is generated based on this grid line. The grid points will not be modified in the calculation process, and line segments are assumed between two grid points. You should make sure that the original data file provides enough information to describe the whole airfoil geometry. There is a sample *XYZ* file named **NACA0012.xyz** in this package as reference. At the beginning in interactive mode, or at the head of configuration file in batch mode, the user should provide the name of the *XYZ* file. The filename may include full path and the length should not beyond 64 characters.

In practical use, the *XYZ* file is usually generated using a CAD package and a mesher from original airfoil data. Since version 2.2, the only input can be another kind of data file which describes control points on airfoil. It may be the original data that the airfoil designers frequently use. The user can use control parameters 'Y' or 'N' to configure whether using the data file of control points as input. The format of coordinates to describe the control points is similar to a kind of Selig format (refer to UIUC Airfoil Data Site) which doesn't differentiate lower airfoil and upper airfoil. The coordinates are starting from trailing edge, along the upper surface to the leading edge and back around the lower surface to trailing edge, for example

---

number of control points on whole airfoil (N) (first upper from trailing to leading, then lower from leading to trailing)
X(1) Y(1)
X(2) Y(2)
. .
. .
X(N) Y(N)

Note that the number of control points on the whole airfoil should be given first in the input file, it can be much less than the number of grid points in the *XYZ* file. Like the format of *XYZ* file, the coordinates should be dimensionless scaled by chord length. The number of control points on upper airfoil may be different from the number of control points on lower airfoil, which depend on the geometry of the upper and lower part. The program can find the location of the leading edge automatically. The first and last control

points which represent trailing edge should be identical so that the curve they described is closed. The Selig format file can be used as the input of this program directly by replacing the airfoil name to the number of control points and making sure that a control point of trailing edge is defined. If the first and last control points don't coincide, this program can close the curve automatically by connecting these two points via a line segment. There are sample files of airfoil control points named **NACA001264.cpt** and **whitcomb.cpt** in this package as reference. The grid line will be generated by this program using a parametric cubic spline method based on these control points, which indicates that the grid line passes through all control points. The number of grid points on this grid line should be input by the user via a control parameter. No other CAD and mesh package is needed, RANSFOIL will do all the preprocess workflow.

5 Output files

When RANSFOIL is executing, a text file named **iteration.txt** will be generated to record iteration information. After a few iteration steps, a grid data file named **Grid.xyz**, a solution functions file named **Solutions.dat** and a solution name file named **Solnames.nam** will be created. These three files are standard 2D *PLOT3D* files to record grid coordinates and airflow solutions. Concurrently, a binary file named **Autosave.dat** is generated to backup the numerical solutions of all transport equations automatically. In addition, the program will create a file **Wallsol.dat** to save near wall airflow data and a file **Aeroreport.txt** to report aerodynamic parameters. You can use visualization packages such as Paraview, Mayavi2 or Gnuplot to visualize these data files, but how to use these software packages is beyond the scope of current document. The user can check respective package documentation. For packages which use VTK as visualization engine and do not well support *PLOT3D* format, an extra file named **GridSol.vtk** which is written in legacy *vtk* format is generated as input dataset. When the program is normally terminated, the final results are saved in the last data files.

The format of almost all output files is ASCII (except **Autosave.dat**) for convenience of reading. The variables of airflow field included in the 2D *PLOT3D* files (**Grid.xyz**, **Solutions.dat**, **Solnames.nam**) are X-coordinates, Y-coordinates, X-velocity, Y-velocity, static pressure, static temperature, air density and some turbulence variables (depending on the turbulence model you select). The variables in legacy *vtk* file **GridSol.vtk** are identical to that in *PLOT3D* files. The variables of near wall airflow data in the file **Wallsol.dat** include X-coordinates, Y-coordinates and a kind of arc length

(origin at leading edge) on airfoil, and near wall pressure, near wall velocity, near wall temperature and near wall turbulence viscosity in wall-adjacent cells. In addition, as postprocess results, the wall shear stress (friction stress) in X, Y directions, surface heat transfer coefficient and dimensionless wall distance are also saved in this file. All aerodynamic parameters of the airfoil including lift coefficient, drag coefficient (induced by pressure), friction coefficient, pitching moment coefficient (taking at 1/4 chord) and the location of pressure center can be found in the report file **Aeroreport.txt**.

At last in interactive mode, or at the end of configuration file in batch mode, the user should input the directory name of output files. If you want to use current directory, just input '.'. This control parameter can be used for batch process. For example, if you want to run two cases on the same airfoil (each case corresponds to unique combination of control parameters), you can create two different configuration files and two different subdirectories, then set the name of each subdirectory in corresponding configuration file. The output files will be assigned to the subdirectory belong to themselves if you write a reliable shell script based on your platform.

6 Examples

The sample *XYZ* file **NACA0012.xyz** and configuration file **ransfoil.configxyz** can be used as an example of getting started. You just need to type

```
$./ransfoil --script ransfoil.configxyz
```

in root directory of this package after building the executable file. This program will access in batch mode and calculate airflow field and aerodynamic parameters around a NACA0012 airfoil at chord length of 0.5334m in low speed regime (velocity of free stream: 75m/s, angle of attack: 4deg) using the S-A turbulence model. All the foregoing output files will be generated for visualization and report.

The other sample coordinates file **NACA001264.cpt** and configuration file **ransfoil.configcpt** can be used as an example for the use of control points file (.cpt) as input. You just need to type

```
$./ransfoil --script ransfoil.configcpt
```

This program will access in batch mode and calculate airflow field and aerodynamic parameters around a NACA001264 airfoil at chord length of 0.5334m in low speed regime (velocity of free stream: 75m/s, angle of attack: 4deg) using the S-A turbulence model.

If you want to access in interactive mode, type

```
$./ransfoil --stdin
```

and input the control parameters according to the hint messages and the values in the sample configuration file.

You can also use other sample *XYZ* or *CPT* files in subdirectory **cases** (e.g., **whitcomb.xyz**) to test this program. Because the convergence and stability characteristics are different for the subsonic, transonic and supersonic cases, we should use more than one configuration files to make sure that the iteration is stable in many practical cases. For more details concerning the tips in stability of different flow regime, see Sec. 10. There is a subdirectory **scripts** in this package which stores a running script and some sample configuration files for a whitcomb airfoil. The *XYZ* file can be found in the sub-subdirectory **whitcomb1**, and the configuration files for different flow regimes are stored in corresponding sub-subdirectories. The configuration files provide the guidance for the combination of relaxation factors to guarantee the stability and efficiency of an iteration in different flow regime. Note that all the relaxation factors in the configuration files are for extreme cases (e.g., low-Reynolds mesh and high AoA, etc.), you can increase the value of relaxation factors according to your situation to improve the efficiency of iteration.

7 Solver Guide

This program is a 2D steady RANS solver specifically designed for aerodynamic calculation on an airfoil. A pressure-based finite volume method is used as the solving strategy and discretization scheme. A segregated algorithm is applied to treat the coupling between the velocity and pressure, which means the governing equations are solved sequentially. The discrete form of momentum equations are solved at first, then the continuity of equation is transformed into a pressure correction equation and solved, then the velocity is corrected by the pressure and the energy equation and turbulence transport equations are solved if necessary. Since the governing equations are nonlinear and coupled to one another, the above steps constitute an "iteration" and the final results always need many iterations which forms a loop of the steps. For compressible airflow, the state equation of ideal gas is used and the segregated algorithm is extended to couple velocity, pressure and density.

There is also a structured mesh generator in this program which can be considered as a pre-process of the airfoil coordinates. A single-block C-type structured mesh is generated by this mesher before the RANS solver works. A grid generation system based on hyperbolic partial differential equations (PDEs) is used in this mesher, in which the grid is generated by numerically

solving some hyperbolic PDEs, marching in the curvilinear direction normal to the airfoil in two dimensions. The orthogonality of mesh can be guaranteed using this hyperbolic generation system. The coordinates transformation is used to calculate the geometric parameters in the mesh and transform the physical domain into the computational domain. The RANS equations are discretized in this computational domain so that the derivatives of field variables and the coefficients of the algebraic equations are derived using this coordinates transformation, which can improve the accuracy of the solution, reduce the usage of memory and highlight the advantage of structured mesh.

There are many control parameters to configure the features in the RANS solver and hyperbolic mesh generator, which will be discussed in Sec. 8.

8 Model and algorithm selection

The control parameters concerning selection of aerodynamic model and numerical algorithm are characters constants. The following are details about each control parameter and a brief guide for selection.

The density of air modeled in this program may be constant or variable. The user can use control parameters 'incom' or 'com' to set the physical property of air, corresponding to incompressible or compressible. There are several turbulence models in this program. These turbulence models include standard $k-\epsilon$, Spalart-Allmaras (S-A) and Shear Stress Transport (SST) $k-\omega$ model. The user can select turbulence model using control parameters 'ke', 'sa' or 'sst', corresponding to $k-\epsilon$, S-A or SST turbulence model. The S-A and SST model are specifically designed for aerodynamic calculation, and generally S-A model costs less time than SST model. The $k-\epsilon$ model has more generality, but is less accurate in near wall region for an aerodynamic problem. You can also use control parameters 'inv' or 'lam' to turn off a turbulence model. If you set 'inv', the air is considered as inviscid, and the RANS equations degrade into Euler equations. In 'lam' case, the standard Navier-Stokes equations are used as all transport equations and the turbulence model is not needed. In turbulence problem, the near wall flow should be treated specifically in each turbulence model, there are two methods in this program to compute near wall flow: wall function method and low Reynolds number method. For S-A and SST turbulence model, both two methods can be set via control parameters 'wf' or 'lr', corresponding to wall function or low Reynolds number method. For $k-\epsilon$ model, only wall function method can be used, you should not set the control parameters concerning wall treatment in the configuration file in batch mode after you set 'ke' as the turbulence model.

The user can use control parameters 'Y' or 'N' to tell the program whether the energy equation is included in the transport equation sets or not. If you want to obtain heat transfer coefficient on airfoil surface (which is saved in **Wallsol.dat** data file) or you just set air property as compressible (input 'com'), you must turn on the energy equation (input 'Y'). When the energy equation is solved, the viscous heating terms may appear on the right side of the energy equation. If the energy equation is turned on, the user can use control parameters 'Y' or 'N' to tell the program whether the viscous heating terms are included in the energy equation. If you set 'Y' to include the additional heating terms, there are two situations which depend on the air property you configured before. If you set air property as 'incom', only viscous heating terms are included in the energy equation. If you set air property as 'com', A compressible heating term will be added in the energy equation with viscous heating terms. Different additional heating terms makes different results of surface heat transfer coefficient.

There are some control parameters which define the numerical algorithm and schemes used by this program. The user can select SIMPLE or SIMPLEC algorithm to treat the coupling between velocity and pressure via control parameters 'SIMPLE' or 'SIMPLEC'. In most cases the SIMPLEC algorithm has faster convergence than SIMPLE algorithm. The user can use a few spatial schemes to discretize the transport equation. By setting control parameters '1upwind', '2upwind' or 'Quick', the first order upwind, second order upwind or QUICK schemes are applied in discretization process, corresponding to first order, second order or third order accuracy of convective terms in spatial dimensions. If you have set air property as compressible (input 'com'), the interpolation scheme of air density at the cell interface of control volume should be treated specifically. You can select center, first order upwind, second order upwind or QUICK schemes to interpolate density via control parameters 'center', '1upwind' '2upwind' or 'Quick'. To capture the shock wave accurately in highly compressible airflow, a TVD scheme is used as a selectable spatial scheme of convective terms which can diminish the oscillation near the shock wave. This TVD scheme can also be applied as a selectable interpolation scheme of air density. You can use the control parameters 'tvd' to select the TVD scheme for discretizing the convective terms or interpolating the air density.

In calculation process, The file **Autosave.dat** is generated periodically if the iteration step arrives a certain value. The numerical solutions of all flow field variables are saved in this file. You can use control parameters 'Y' or 'N' to tell the program whether using a data file for initialization. If you set 'Y' to initialize from a file, you can use **Autosave.dat** as the initializing file. You should make sure that the program was running at least once and

the file **Autosave.dat** is obtained, you should not modify the contents in this file. You also can use an external data file as the initializing file, in both cases you should input the name of the initializing file in interactive and batch mode. If you set 'N' to initialize by the program, there is another extra control parameter you should set. If you input 'Y' to initialize using the stagnation values, the flow variables will be set near stagnation (velocity small but not zero, pressure to total pressure of free stream, temperature to total temperature of free stream), which is more appropriate for supersonic airflow, otherwise (input 'N'), the flow field variables will be initialized using the values from ambient conditions.

9 Setting of ambient, mesh and iteration parameters

The control parameters concerning ambient conditions and iteration parameters are floating-point number constants. The typical ambient conditions in aerodynamic problem of an airfoil include velocity, static temperature and static pressure of free stream and angle of attack. The user can input the values of these ambient conditions via corresponding control parameters. If you want to use a turbulence model to simulate turbulent fluctuations, some ambient conditions which describe the fluctuating quantities in free stream should be configured. If you select S-A model (input 'sa'), a control parameter called turbulent viscosity ratio of the free stream should be set. The typical values of this parameter ranges from 1 to 100, corresponding to the interval from low turbulent viscosity to high turbulent viscosity. If you select $k-\epsilon$ or SST model (input 'ke' or 'sst'), an extra control parameters called turbulence intensity of free stream should be set. This parameter represents the intensity of fluctuating velocity in free stream with the typical values ranging from 0.01 to 0.1. Some other conditions should be set to describe the airfoil, including chord length, wall temperature (or wall heat flux) and equivalent sand grain roughness (set to 0 to model smooth surface) on airfoil surface. These ambient conditions are considered as boundary conditions of corresponding transport equations. For energy equation, you must input control parameter 'fixed' or 'flux' to configure the type of boundary condition at wall is whether fixed temperature or heat flux. The near wall temperature is affected obviously by the type of heat boundary condition.

Before iteration, this program automatically generates a 2D C-type structured mesh for discretization. The user should input the dimensionless near wall mesh spacing (the airfoil coordinates should be dimensionless in the 1D

XYZ file) because it determines the treatment of near wall flow and is important for obtaining accurate results. A brief guide concerning the selection of near wall mesh spacing will be discussed in Sec. 10. There are another two control parameters to adjust the grid points distribution normal to the airfoil surface, you can set the layers of near wall mesh which have the uniform near wall mesh spacing. In some turbulence model, the number of near wall mesh layers is a special request for obtaining better near-wall results. You also can input total number of grid points normal to the airfoil surface, of course the total number includes the layers of near wall mesh and should make sure that the mesh will be extended sufficiently far to the free stream region. Thus the normal total number should be much greater than the value of near-wall layers you set before. When using control points coordinates as input and the grid line to describe the airfoil should be generated first, there are another two control parameters to adjust the grid points distribution on the airfoil. One is dimensionless mesh spacing near leading edge, the other is dimensionless mesh spacing near trailing edge. Using these two control parameters with the control parameter of grid points number on airfoil, you can obtain a ideal grid line which considers both the whole airfoil geometric information and flow feature near it. Some default parameters may be modified if blunt trailing edge is detected, which mostly happen in cases of automatic close of airfoil profile.

Some iteration parameters should be set by the user including maximum iteration steps, minimum residual and relaxation factor. The maximum iteration steps and minimum residual determine the stop criterion of a practical iteration. If the current iteration steps are beyond maximum iteration steps or the current residual is smaller than minimum residual, the iteration will be terminated. You can simultaneously set numbers of relaxation factors for velocity, pressure, temperature and turbulence. Which relaxation factor you should set depends on the model you have selected (turbulence model, air property, etc.). The value of each relaxation factor should be greater than 0, and less than 1. The value of relaxation factor may affect the convergence rate, a larger relaxation factor can make faster convergence but less numerical stability, conversely an iteration with smaller relaxation factor is slower but more robust. You can also select type of linear solver for convective-diffusion equations using 'sor' or 'pbicg'. It is an advanced expert control parameter and in most cases, 'sor' is good. You can switch to 'pbicg' if the stability issue can not be solved, or speedup of openmp parallelization is not ideal.

10 Tips on Practical Use

The sample configuration files **ransfoil.configxyz** and **ransfoil.configcpt** does not include all control parameters which this program supports, the control parameters in it are for a special case. The appearance of some control parameters depend on other parameters which you just set. If you are not sure about the sequence or combination of the control parameters in your case, you can run this program in interactive mode at first. If the iteration information successfully appears, the control parameters you just input on terminal are the right combination.

The user can monitor this program by two ways in runtime. First, the mass residual (residual of continuity equation) will be printed on the terminal screen at every iteration step. Second, more detailed information including the residual of all transport equations will be saved in the plain text file **iteration.txt**.

The process of this program may be blocked or terminated exceptionally when the iteration diverges. The effective way to eliminate divergence is reducing the value of relaxation factor. When the file **Autosave.dat** is generated in an iteration with smaller relaxation factors, try to stop current iteration and run a new iteration with larger relaxation factors. To realize this, you should set 'Y' to initialize from a file and read initial data from **Autosave.dat**.

If you want to calculate airflow in transonic or supersonic regime, the high compressibility may result in divergence when you start an iteration directly by setting air property as compressible (input 'com'). In transonic regime, you can start an iteration with a smaller relaxation factor and incompressible air property (input 'incom'), after a few steps the file **Autosave.dat** may be generated. You can use this file as a starting point by setting 'Y' to initialize from this file, then run a new iteration with a larger relaxation factor and compressible air property (input 'com'). In the new iteration the convergence may be obtained. In supersonic regime, the flow feature is more different from subsonic regime, the method that using incompressible results as initial data may not be valid. You can initialize the flow variables to be near stagnation, it accords with the evolution of a supersonic flow. That means you should input 'N' to not initialize from a file and then input 'Y' to initialize using the stagnation values. There are sample configuration files in subdirectory **scripts** to guide you setting these control parameters.

The user can check the average value of y^+ (in S-A or SST turbulence model) or y^* (in $k-\epsilon$ model) in the report file **Aeroreport.txt**. This dimensionless number represents the distance from wall in turbulence viewpoint and is important for the selection of a reliable near wall mesh spacing

(proportional to y^+ or y^*). For S-A or SST model, if you set 'wf' (wall function) as wall treatment method, the value of y^+ can be in a large range from 1 to 300 (a kind of enhanced wall function is implemented in these models, for rough wall the value of y^+ can be larger than 300), conversely if you set 'lr' (low Reynolds number method) as wall treatment model, y^+ should be of order 1 ($y^+ \sim 1$, at viscous sublayer). For $k-\epsilon$ model, because only a standard wall function is implemented in the model, the value of y^* should be in the range from 30 to 300 ($30 < y^* < 300$, at log-law layer). Above means that you should run the program (with a certain near wall mesh spacing) at least once to obtain a **Aeroreport.txt** file, then check the value of y^+ or y^* in this file and modify the setting of near wall mesh spacing until the value is in the request range of each situation! Even so, there are some empirical criterions to set near wall mesh spacing concerning airflow around an airfoil. In subsonic regime, a typical value of dimensionless near wall mesh spacing in viscous sublayer is of order $10^{-6} \sim 10^{-5}$, a typical value in log-law layer is of order $10^{-4} \sim 10^{-3}$. But for transonic or supersonic regime, the value of near wall mesh spacing should be smaller than the typical values in subsonic regime because the thickness of boundary layer decreases with the increase of Reynolds number (free stream velocity).

A practical iteration with a larger relaxation factor may also diverge if you set a relatively smaller near wall mesh spacing approaching viscous sublayer. Just reducing the relaxation factor or use the method described in the divergence problem caused by compressibility to achieve convergence.

At present the hyperbolic mesher in this program still can not treat the extremely concave corner effectively. Avoiding a blunt trailing edge on airfoil profile because the extension line and trailing edge may form an extremely concave corner. Concerning the selectable TVD spatial scheme (input 'tvd'), at present only the Van-Albada limiter is applied in this program.

If the OpenMP parallelization is enabled, before running RANSFOIL the user can set the environment variable `OMP_NUM_THREADS` to adjust the number of threads in runtime. Due to the problems of parallelization in ILU preconditioner, at present you should use low threads number (2-8 according to the physical cores of your CPUs) to run this program, i.e., set the value of `OMP_NUM_THREADS` to 2-8. The new version since 2.2.10 fixed the issues caused by ILU preconditioner parallelization, you can set `OMP_NUM_THREADS` to a high threads number, but the parallel performance may not be improved because ILU preconditioner parallelization may increase the iterative number of the linear solver obviously. There is an optimal threads number which is always the physical cores of your CPUs according to our simple test.

11 Library Usage

Except for the usage of RANSFOIL program to obtain output files, this package also provides Application Program Interface (API) in the shared library *libaero2d* for the developers. If you want to use the library interface, you should download the source package and make sure the source files are compiled and installed correctly. If the source package is installed successfully, the header file and library file will be included in the common system directory (e.g., `/usr/include, /usr/lib`).

The developers can use the functions in libaero2d to extend their own application program. The main data sets in library libaero2d (written by Fortran03) are included in a Fortran module named `aero2dcom`. Most subroutines in libaero2d make special operations on these public data sets by referring this module. One subroutine named `Aero2D` is located at root node of the call tree, and realize almost all the functions in libaero2d by calling other subroutines. The developers can call this root node subroutine using Fortran9x (90,95,2003) language to achieve main functions in libaero2d. One sample Fortran9x code block for calling subroutines in libaero2d is shown below:

```
...
use aero2dcom
integer printlevel
character(1) mode
character(64) scriptname
.
.
.
Call Allocarray(mode)
Call Aero2D(mode,printlevel,scriptname)
.
.
.
Call Deallocarray(mode)
...
```

The code at first line achieves the interface to all data sets in `aero2dcom` by referring this module, the codes at next three lines are the parameter declarations of subroutine `Aero2D`, the calling codes in the middle of the code block is a calling instance. First, you should allocate the dynamic arrays in module `aero2dcom` by calling subroutine `Allocarray` in library libaero2d. Then the root node subroutine `Aero2D` is called, certainly the three actual parameters passed to `Aero2D` should be assigned before calling. The parameters

in subroutine `Aero2D` are calling mode parameter (*mode*), print message level parameter (*printlevel*) and script file name parameter (*scriptname*). Where the *mode* and *scriptname* parameters belong to string variable, the *printlevel* parameter belongs to integer variable. At last, the subroutine `Deallocarray` which is also in libaero2d should be called to deallocate the dynamic arrays in module `aero2dcom`. Note that there is an actual parameter in subroutine `Allocarray` and `Deallocarray` which is identical to the calling mode parameter (*mode*). The three subroutines in this sample code block are commonest to realize a calling instance using libaero2d.

There are three calling modes in subroutine `Aero2D`: interactive mode, script mode and interface mode, which are realized by assigning 'I', 'S' and 'A' (acronyms of 'Input', 'Script' and 'API') to the calling mode parameter (*mode*). The calling process in interactive mode will be blocked for waiting the user to type corresponding control parameters (i.e., the control parameters of RANSFOIL program described in previous sections) from standard input. Script mode should read control parameters from a configuration file, if you want to call `Aero2D` via script mode, the third parameter *scriptname* of `Aero2D` can be used to pass a configuration file name, in other calling modes this parameter is never used, you can assign a null string to *scriptname*. The parameter *printlevel* represents the verbose level of the terminal printing information in calling process, there're two integer values which can be used: 0 and 1, corresponding to print simple information and print detailed iteration information, respectively. The above two modes are file based, libaero2d library should read airfoil coordinates from a file, and output files to save iteration information and calculation results. The data sets in module `aero2dcom` are not used by the caller directly, so that `aero2dcom` may not be referred in these calling modes, the first line in above sample codes can be removed. The interface mode is a non-file based calling mode provided by libaero2d, the `aero2dcom` module must be referred in this mode. Because the data sets in `aero2dcom` are all exposed to the developers, the user can configure libaero2d using these data in a programmable style. Calling `Aero2D` in interface mode will not output any files to save data, all data sets stay in the memory for the developers' next reference, unless the caller deallocate all the dynamic arrays in module `aero2dcom`. For example, the developer can assign the string variables to set the control parameters, and can choose reading airfoil coordinates from a programmable file or directly assign the one-dimensional floating-point arrays in module `aero2dcom` which represent the airfoil coordinates (the latter is very useful in optimizing of airfoil design). In addition, the variables which represent the aerodynamic parameters of an airfoil are all included in module `aero2dcom`, the developers can refer these variables to obtain final aerodynamic results after calling sub-

routine `Aero2D`. The developers also can refer the flow field variables which is realized by two-dimensional floating-point arrays in module `aero2dcom` to save the field data or plot them using a visualization engine. The detailed meaning of common variables in module `aero2dcom` are given in the comment section of corresponding source file **COM.f03**. Note that if you want to call `Aero2D` twice in succession, the control parameter which configures whether using a data file for initialization (variable name *Init* in module `aero2dcom`) may be set to 'A' (not 'Y' or 'N'). It is a special option to use the results of last iteration to initialize the flow field variables, even though the ambient conditions may be changed in the new iteration. There is a sample source file named **libcallexpf.f03** in root directory of source package, which shows how to call libaero2d using interface mode. The user can refer to the source file **console.f03** of RANSFOIL program (the RANSFOIL program is realized by calling libaero2d in interactive and script modes) for the examples to call libaero2d using the other two modes.

Thanks to the `ISO_C_BINDING` intrinsic module in Fortran 2003 standard, the developers can call libaero2d using C language since version 1.5. There is a C header file named **aero2dcom.h** in source package, which is the external declaration of common variables in module `aero2dcom` and common functions in libaero2d. The C developers can refer these variables or functions by including this header file at the header of their source file. Because the string and dynamic arrays in module `aero2dcom` are not interoperable with C, you can use the C partners of these variables which just add a 'c' prefix to the original variable name in Fortran. One sample C code block of calling libaero2d is shown below:

```
...
#include "aero2dcom.h"
...
int prlv;
char *mode,*scptn;
.
.
.
mode="C";
allocarray_(mode,strlen(mode));
aero2d_(mode,&prlv,scptn,strlen(mode),strlen(scptn));
.
.
.
deallocarray_(mode,strlen(mode));
...
```

Note that the calling mode parameter (*mode*) should be assigned to `"C"`, which is an additional interface mode of libaero2d specially designed for being called from C language. The underscore at the end of function names can be omitted if libaero2d is compiled using appropriate compiler option (e.g., option `-fno-underscoring` for gfortran compiler). If the formal parameter in a Fortran function is a string variable, the length of string should be passed to the corresponding C function at the end of the parameter list. Except that, the calling sequence is identical to that using Fortran9x language. There is a sample source file named **libcallexpc.c** in root directory of source package, which shows how to call libaero2d using C interface mode.

When you compile and link the Fortran9x source files which call the library libaero2d, assume that you use GNU gfortran compiler, you should add the `-I` option with the directory to module `aero2dcom` in compiling process, and the `-L` option with the directory to import library and `-laero2d` option in linking process, for example

```
$ gfortran ...  -I(directory to aero2dcom.mod) -L(directory
to libaero2d.a or libaero2d.dll.a) -laero2d -o xxx
```

The command to compile and link the C source file is similar:

```
$ gcc ...  -I(directory to aero2dcom.h) -L(directory to
libaero2d.a or libaero2d.dll.a) -laero2d -o xxx
```

12 Additional information

Please report bugs or errors to sdhzhs@gmail.com. For any comments or suggestions, please also email sdhzhs@gmail.com. :)