Construct2D User Manual
Version 2.1

Daniel Prosser

November 24, 2014

Contents

0.1 Compiling Construct2D

The Construct2D executable needs to be built before it can be used. For convenience, executables are included for the x64 architecture for Linux and Windows. If you are running a different architecture or operating system, please compile the source code and then the binary can be added to the distribution of Construct2D. In any case, compiling the code is not difficult and is a good exercise to help you understand how the code and your computer work.

0.1.1 Installing the compiler

To compile the Construct2D on any platform, you will need a Fortran compiler. A good and free Fortran compiler is gfortran, the GNU Fortran compiler. On Linux, this should be very easy as gfortran is included in the repositories of most any distribution. For example, on Ubuntu you may install gfortran from the command line using the command:

```
sudo apt-get install gfortran
```

On Windows, the easiest way to get gfortran is to use the MinGW installer, called "mingw-get-inst-<VERSION>.exe," which can be found by searching for "mingw-get-inst" online (at the time of this writing, it is available at `http://sourceforge.net/projects/mingw/files/Installer/mingw-get-inst/`). Running the executable launches a graphical installer. When using the installer, be sure to select the Fortran compiler for installation, as shown in Fig. 1.

This will install the gfortran executable on your system. You may also install compilers for other languages if desired, but that is not necessary to compile Construct2D.

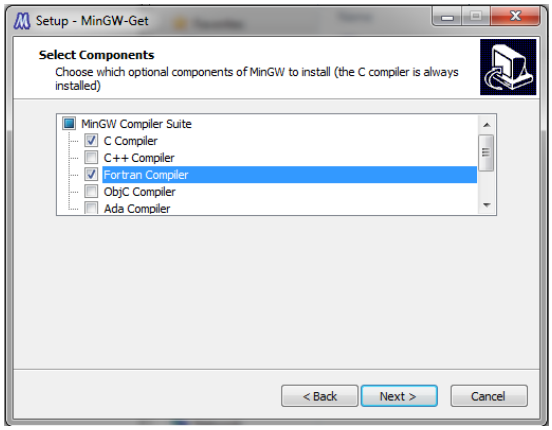


Figure 1: On Windows installation of MinGW, be sure to select the Fortran compiler.

After installing MinGW on Windows, it is important to add the executables to your Path. This is done by modifying the Path system environment variable in System Properties, which is shown in Fig. 2. This System Properties can be modified in the Control Panel, but getting to the system environment variable depends on the particular Windows version being used. Adding the executables directory (the \bin directory in the installation) to your system Path will allow them to be called from any other directory on your computer without having to copy them to that directory.

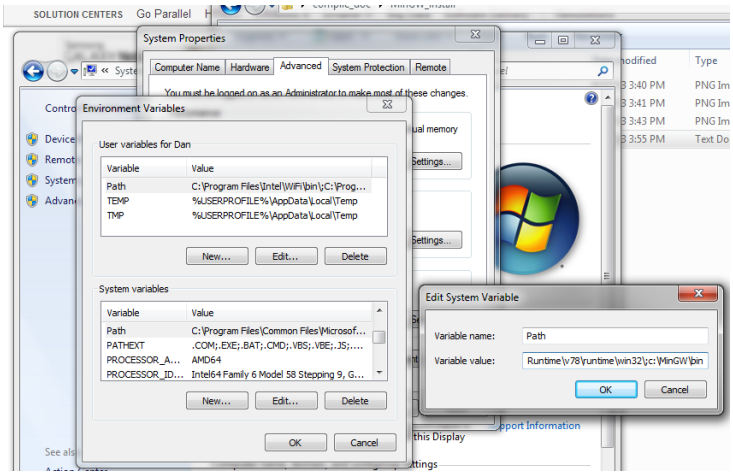


Figure 2: On Windows installation of MinGW, add the MinGW executables directory to your Path system environment variable.

On Mac OS X, gfortran binaries are available at `http://gcc.gnu.org/wiki/GFortranBinaries#MacOS`.

After installing gfortran, another valuable tool for compiling is a program called Make. Make allows instructions for compiling code to be contained in a file (called a "Makefile"). Make comes preinstalled on most (or all?) Linux distributions, so once again, if you are using Linux this task is easy. On Windows, there is an installer available at `gnuwin32.sourceforge.net/packages/make.htm` which will install Make on your system. After installing, the directory containing the executable should also be added to the Path.

On Mac OS X, Make is available by installing Xcode developer tools from the App Store. Installing should automatically place it in a directory within your path.

0.1.2 Compiling

For convenience, a Makefile is included to build Construct2D, which should be a fairly simple process. There are two Makefiles included, one for Linux and Mac and the other for Windows. They do the same thing but some of the command syntaxes are slightly different due to differences between the BASH shell (used on Linux and Mac OS X) and the DOS shell (used on Windows). The first thing you should do is open a command prompt and navigate to the Construct2D directory. On Windows, a very simple batch file is included that will automatically open a command prompt in the current directory when you double-click it.

Before compiling, it is recommended that you look at the Makefile. Open the Makefile for your system using your favorite text editor. You will notice on the first line the following:

```
FC=gfortran
```

This line specifies the Fortran compiler being used. If you installed gfortran previously, then leave this line alone. If you have a different Fortran compiler that you would rather use, you may enter it here in place of gfortran. The second line specifies compiler flags. By default, there is just one compiler flag:

```
FCFLAGS=-O3
```

which tells the compiler to use -O3 optimization to improve performance of the code. If you wish to change the optimization level or add other compiler flags, enter them on this line. Note that if you are using a different compiler you may have to add some flags and change the line below FCFLAGS; see the comments provided in the Makefiles. You are now ready to compile the code. It is possible to specify which Makefile to use when running the make command, but to simplify things it is recommended that you rename or copy the Makefile for your system and call it, simply 'Makefile.' Then, to compile and install the code, run the following commands from the command line:

```
make
make clean
```

The basic make command actually compiles the code, and the second command cleans up some object files that you don't need. As long as there were no errors after running the make command, the executable should now be compiled and ready to use.

For reference, if you prefer not to rename the Makefile as recommended above, you would then have to run the above commands as

```
make -f <file>
make -f <file> clean
```

where <file> is the Makefile for your system. As you may have guessed, the -f option to the make command allows you to specify a file name other than the default 'Makefile.'

0.2 Input File Format

The input file format used by Construct2D is the same as that used by XFoil. The first line of the file should be a label. The next lines list the x and y coordinates of the airfoil, starting with the top of the trailing edge and continuing counterclockwise to the bottom of the trailing edge (so the leading edge point will be somewhere near the middle of the list). Each x-y coordinate pair should be on its own line, with x and y separated by one or more spaces. For example, below is a portion of an input file for a NACA 0012 airfoil generated by XFoil.

```
NACA 0012
    1.000000      0.1260000E-02
   0.9916770      0.2421811E-02
   0.9803640      0.3982079E-02
    .              .
    .              .
    .              .
   0.0000000      0.00000000
    .              .
    .              .
    .              .
   0.9803640     -0.3982079E-02
   0.9916770     -0.2421811E-02
    1.000000     -0.1260000E-02
```

It is recommended that input airfoils have a chord length 1 and a leading edge at x = 0.0. However, Construct2D automatically translates and scales input airfoils such that they meet these criteria before generating a grid. More examples of input airfoil files are included in the sample_airfoils directory.

0.3 Running Construct2D

After compiling, the Construct2D executable will be placed in the Construct2D directory. On Linux and Mac OS X, the executable will be called 'construct2d,' and on Windows it will be called 'construct2d.exe.' You may run Construct2D by doing one of the following:

1. Opening a command line in the directory where the executable is contained and typing:

   ```
   ./construct2d
   ```

   on Linux and Mac OS X or

   ```
   construct2d.exe
   ```

   on Windows.

2. Opening a command line in the directory where the executable is contained and typing:

   ```
   ./construct2d <airfoil file>
   ```

   on Linux and Mac OS X or

   ```
   construct2d.exe <airfoil file>
   ```

   on Windows, where <airfoil file> is the file name of the input airfoil.

3. On Windows, double-clicking construct2d.exe (equivalent to option 1).

4. On Windows, dragging the airfoil file onto construct2d.exe (equivalent to option 2; basically means "open with construct2d.exe").

0.3.1 Program flow

When you first run the program, if you have not specified an input airfoil file as a command line option, you will prompted to enter it once the program opens. The prompt will look like this:

```
Enter name of airfoil to load (XFOIL labeled format)
 or QUIT to close the program:

Input >
```

You should enter the path of the airfoil file here. If it is in the same directory as the code, you may just enter the file name. You may also type 'QUIT' (or 'Quit' or 'quit' – any input command in this program may be entered in all caps, all lower case, or title case) to quit the program. After selecting the airfoil file, or if the file was specified as a command line option, some information will be printed about the airfoil file provided the file is read successfully. For example, in this case I have loaded an airfoil called naca0012.dat from the sample_airfoils directory. The following information is printed to the screen:

```
Successfully loaded airfoil file sample_airfoils/naca0012.dat
 Number of points:         160
 Setting default project name: sample_airfoils/naca0012
 Blunt trailing edge: O-grid topology is recommended.
```

Notice that the default project name is set based on the airfoil file name. You may want to change this, but how to change it will be explained later.

Construct2D is menu-driven. After loading the airfoil, the main menu is displayed, which looks like this:

```
Main program commands:

 SOPT  Change airfoil surface grid options
 VOPT  Change volume grid options
 OOPT  Grid output options
 LOAD  Load a new airfoil
 OPTW  Write current program options to a file
 GRID  Generate grid for currently loaded airfoil
 QUIT  Exit program

Command >
```

To enter a main program command, simply type the four-letter code listed in the main menu at the prompt and press enter. Again, any command may be entered in all caps, all lower case, or in title case. Each command is given a short description in the main menu for the code, but they are described in more detail here:

- SOPT: Surface grid options. Allows you to specify the number of grid points on the airfoil surface, grid point spacings on the airfoil surface, and other options to control the grid at the airfoil surface. Also includes options to help control the grid at the outer farfield boundary.
- VOPT: Volume grid options. Allows you to control volume grid parameters and other general grid options; for example, the project name may be changed within this submenu.
- OOPT: Grid output options. Allows you to specify whether to write a 2D-planar grid or a 3D grid created by extruding a number of planes in the third dimension.
- LOAD: Loads a new airfoil. You will be prompted to enter the file name.
- GRID: Generates the volume grid. Enter this command after the grid settings have been changed to your liking and you are ready to generate the grid.
- OPTW: Writes current options to a file, which is useful after the grid is generated if you want to save the options used for the program. Additionally, if this file is named 'grid_options.in,' it will be automatically read when Construct2D is started and used to set the default options. In this way, you may use this file to specify options instead of navigating through menus. This method is faster and especially useful if Construct2D is being run many times from a script.
- QUIT: Closes the program.

0.3.2 Program options

This section explains options contained in submenus including the SOPT, VOPT, and OOPT submenus.

SOPT

Each option in the SOPT submenu is described in the list below. As with any input command in Construct2D, commands may be entered in all caps, all lower case, or title case.

- NSRF: Number of points placed on the airfoil surface. For the O-grid topology, this is the same as the number of points in the i-direction, or $imax$. For the C-grid topology, this

is *imax* minus the number of points in the wake. XFoil's paneling subroutine is used in part to apply smooth spacing of the points over the surface and to control clustering at the leading edge and trailing edge.

- LESP: Point spacing at the leading edge. If the SMTH option is selected when generating a grid, Construct2D will minimize stretching of the surface points while honoring the leading edge and trailing edge point spacings set by the user.

- TESP: Point spacing at the trailing edge. If the SMTH option is selected when generating a grid, Construct2D will minimize stretching of the surface points while honoring the leading edge and trailing edge point spacings set by the user.

- RADI: Farfield radius. Note that airfoils are normalized to a chord length of 1, so the units of RADI are chord lengths. Also note that for the hyperbolic grid generator, the location of the outer boundary can only be approximately specified ahead of time. For the elliptic grid generator, it can be specified exactly.

- NWKE: For C-grids only, this is the number of (visible) points directly behind the airfoil in the wake region. Note that the actual number of points added is double this number, because the computational grid is cut along this line and so there are two identical sets of points added on top of each other, one belonging to the top side of the grid and the other belonging to the bottom side. A similar cut occurs behind the airfoil for the O-grid topology as well, but the number of points added at the cut for the O-grid is controlled by JMAX.

- QUIT: Returns to the main program menu.

Options specific to elliptic grid generation

- FDST: Controls clustering of grid points on the farfield for the O-grid topology. A value of 1 gives uniform spacing; a value less than 1 clusters more points near the top and bottom, and a value greater than 1 clusters more points near the front and back. This setting can be used to reduce grid skewness in some cases, but in general values close to 1 are recommended. Too low a value may actually send the code into an infinite loop (may occur if FDST is below about 0.4) when the grid is generated. So, if nothing is displayed for a few seconds after attempting to generate a grid with a low value of this parameter, kill the program and try again with a higher value.

- FWKL: For the C-grid topology, this is the ratio of the length of the "wake" section of the farfield relative to the wake region directly behind the airfoil. A value of 1 means that the length of this region is the same at the farfield as it is in the wake region directly behind

the airfoil. A value less than 1 means the wake region is shorter on the farfield. A value greater than 1 is not allowed (it will be automatically set back to 1 if a value greater than 1 is entered). Can be used to control skewness in the grid in some cases.

- FWKI: For the C-grid topology, this is the ratio of initial spacing of the "wake" section of the farfield, relative to the initial spacing in the wake region directly behind the airfoil. A value greater than 1 means that the initial spacing is greater than that directly behind the airfoil. A value less than 1 is not allowed. This parameter can also be used to control skewness in some cases.

VOPT

Each option in the VOPT submenu is described in the list below. As with any input command in Construct2D, commands may be entered in all caps, all lower case, or title case.

- NAME: Change project name. By default, this variable is set to the file prefix of the airfoil file that was loaded. Output files will be saved with this prefix as well. It is recommended not to include spaces in the name.

- JMAX: Number of points in the j-direction (the normal direction to the airfoil surface). More points reduces the error in the numerical solution, but it also increases the computation time both for the grid generator and the CFD solver.

- SLVR: Solver type; either hyperbolic or elliptic. Hyperbolic is generally recommended because it is orders of magnitude faster and tends to be able to generate grids with lower skew. In hyperbolic grid generation, the airfoil surface is marched outwards and the farfield boundary cannot be specified beforehand. In elliptic grid generation, both the airfoil surface and farfield boundary are specified ahead of time, an initial algebraic grid is generated, and then the elliptic solver smooths this grid iteratively. The iterative process is the reason for the increased computation time.

- TOPO: Grid topology; either O-grid or C-grid. The O-grid topology is the default for airfoils with a blunt trailing edge, and the C-grid topology is the default for airfoils with a sharp trailing edge. The topology is set automatically after the airfoil file is read. It is possible to change the topology, but typically it will be very hard to generate a good-quality grid with low skew when switching from the recommended topology.

- YPLS: Non-dimensional wall distance, y+, from the airfoil surface to the second grid node in the normal direction. For turbulent viscous computations, a y+ value less than 1 is recommended. For laminar viscous simulations it may be considerably greater, and for inviscid simulations it may be much greater as there is no boundary layer to resolve.

However, even in inviscid simulations the highest flow gradients generally occur near the airfoil surface, so the grid spacing should be significantly smaller here than near the farfield boundaries.

- RECD: Chord Reynolds number used to calculate y+. This is important for turbulent computations, where the y+ value based on this Reynolds number should be less than 1.
- QUIT: Returns to the main program menu.

Options specific to hyperbolic grid generation

- ALFA: Implicitness parameter for hyperbolic marching. Usually, a value of 1.0 gives good results. In cases where there are tight concave curves or corners, increasing this value can help prevent grid lines from crossing.
- EPSI: Implicit smoothing parameter. Can help smooth the grid and adds stability to the process. However, it is possible to set this value too high and cause instability as well.
- EPSE: Explicit smoothing parameter. Has a similar effect as EPSI, but is much more likely to cause instability. May also help with concave corners. In most cases, a value of 0.0 is best.
- FUNI: Uniformness of cell areas near the farfield. A value of 1.0 means completely uniform, and a value of 0.0 means the cells are clustered at the farfield similarly to how they are clustered near the airfoil surface. For O-grids, a value of 0.2 is recommended, and for C-grids it is usually best to keep it below 0.025. Setting it too high can negatively impact orthogonality and growth rates.
- ASMT: Number of cell area smoothing iterations for each level as the hyperbolic grid is marched outward. Smoothing iterations can help keep cell growth rates under control and to produce a smoother grid. However, it does increase computation time somewhat and using too many iterations may cause instability.

Options specific to elliptic grid generation

- STP1: Initial smoothing steps for grid generation. Initially, the grid is generated by connecting the airfoil surface and farfield boundaries with spline curves. The grid is then smoothed in two steps. First, smoothing is performed on a grid that is not clustered in the j-direction normal to the airfoil surface. The number of steps in this smoothing is controlled by STP1. After the initial smoothing, point spacings are applied in the j-direction based on JMAX and YPLS using interpolation, giving the proper clustering

in the j-direction. Finally, some more smoothing steps are performed, set by STP2, to remove kinks that may have been introduced by interpolation, and then the point spacings are reinforced again.

- STP2: Final smoothing steps for grid generation. Initially, the grid is generated by connecting the airfoils surface and farfield boundaries with spline curves. The grid is then smoothed in two steps. First, smoothing is performed on a grid that is not clustered in the j-direction normal to the airfoil surface. The number of steps in this smoothing is controlled by STP1. After the initial smoothing, point spacings are applied in the j-direction based on JMAX and YPLS using interpolation, giving the proper clustering in the j-direction. Finally, some more smoothing steps are performed, set by STP2, to remove kinks that may have been introduced by interpolation, and then the point spacings are reinforced again.

- NRMT: First point from the trailing edge on the top surface to start making sure the grid is normal to the airfoil surface. Generally, a grid that is normal to the surface is desirable, but sometimes enforcing this normal condition near the trailing edge results in higher skew elsewhere. Increasing this number may help reduce skewness.

- NRMB: First point from the trailing edge on the bottom surface to start making sure the grid is normal to the airfoil surface. Generally, a grid that is normal to the surface is desirable, but sometimes enforcing this normal condition near the trailing edge results in higher skew elsewhere. Increasing this number may help reduce skewness.

OOPT

Each option in the OOPT submenu is described in the list below. As with any input command in Construct2D, commands may be entered in all caps, all lower case, or title case.

- GDIM: Dimension of output grid. May be 2 or 3. A 2D grid is just a single plane. A 3D grid can be created by extruding this 2D plane in the third dimension with user-specified number of planes and spacing.

- NPLN: Number of planes to be created for a 3D output grid. Must be at least 1. If the user specifies one plane, a 2D grid will be written even if GDIM is 3.

- DPLN: Spacing between extruded planes for a 3D output grid. Cannot be 0.

- QUIT: Returns to the main program menu.

0.3.3 Generating a grid

After the options are set as desired, either by using the SOPT and VOPT menus or by using the grid_options.in file (see the entry called WOPT in section 0.3.1), or a combination of both, the grid may be generated. The grid is generated by entering the GRID command at the main menu prompt. Entering this command brings up the following options:

```
Generate grid for which airfoil?

  SMTH  Smoothed airfoil using current surface grid options
        (recommended)
  BUFF  Buffer airfoil defined directly by loaded geometry
        (nsrf will be set to number of points in buffer airfoil)
  QUIT  Leave menu without generating grid

Input >
```

It is recommended to use the SMTH option, which minimizes stretching of points on the surface while constraining the leading edge and trailing edge spacings to LESP and TESP, respectively, as specified in the SOPT menu. It will also use the specified number of surface points NSRF from the same menu. Airfoils with a gap between top and bottom surfaces are assumed to have a blunt trailing edge, which will be replaced by a small fillet to improve grid quality if the SMTH option is selected. The BUFF option uses the points in the airfoil file as the surface points in the grid. Note that the BUFF option does not honor the parameters NSRF, LESP, or TESP from the SOPT routine. This option also requires an airfoil with a closed trailing edge, as no gaps are allowed in the grid. The BUFF option should be used only if a very specific point distribution is desired.

After selecting a grid option, a number of things occur, depending on whether the hyperbolic grid generator is used or the elliptic grid generator. In hyperbolic grid generation, first the airfoil surface grid is generated. Next, this inner curve is marched outwards with the number of points and spacings controlled by JMAX, RADI, YPLS, and RECD. Marching continues until the specified number of levels specified by JMAX is created. Finally, point spacings are reapplied to the grid to ensure the correct value of YPLS is present everywhere. As the hyperbolic grid is being generated, the progress will be displayed, which will look something like this:

```
 Level (current, total):     72,   100

 Level (current, total):     73,   100
```

```
 Level (current, total):     74,   100

 Level (current, total):     75,   100

 Level (current, total):     76,   100

 Level (current, total):     77,   100

 Level (current, total):     78,   100
```

The hyperbolic grid generator is recommended over the elliptic generator because it is significantly faster and generally produces better grids. For example, using the hyperbolic grid generator it is usually not hard to achieve a maximum skew angle of less than 20° (and sometimes even less than 10°), but with the elliptic generator it is usually very difficult to keep the skew angle below 25° near the trailing edge of an O-grid. In some cases, it may be easier to get lower cell growth rates with the elliptic grid generator, but the difference is not as large as the difference in skewness, and proper surface spacings and optimal values of ASMT, FUNI, and EPSI can usually fix this difficulty. The hyperbolic generator is the program default.

In elliptic grid generation, the following steps are performed to generate a grid. First, the airfoil surface and farfield points are generated. Next, an initial algebraic grid is generated by connecting the airfoil and farfield with spline curves. Next, initial smoothing is performed on the algebraic grid using a poisson smoothing approach. During this step, information will be printed to the screen showing the current smoothing step number, the total number of smoothing steps, and information about how much the grid is changing from one step to the next (known as the *RMS residual*). After the initial smoothing steps are complete, you have the option to add more initial smoothing steps or continue on to final smoothing. The prompt will look like this:

```
 Iteration (current, total):    998,  1000
 RMS error:    8.20151447E-04

 Iteration (current, total):    999,  1000
 RMS error:    8.18914297E-04

 Iteration (current, total):   1000,  1000
 RMS error:    8.17679024E-04

Finished requested steps.  Perform more steps before
final smoothing (y/n)?
```

```
Input >
```

At this point, enter y or n (lower case or capital) for yes or no. If yes, you will be asked to enter the number of additional initial smoothing steps to perform. If no, proper point spacings will be applied based on JMAX and YPLS by interpolation, final smoothing will be performed, and then the spacings will finally be reapplied.

Whether hyperbolic or elliptic grid generation is used, once the grid is generated some information about the grid quality is computed and written to a file. A summary of this information is printed to the screen, as shown below:

```
                    Grid quality information
---------------------------------------------------------------
Max skew angle (deg):   30.51073
  x, y:    1.00219  -0.00439
  i, j:     247     30
Max growth in xi-direction:    0.23956
  x, y:    0.99915  -0.00160
  i, j:     247      1
Max growth in eta-direction:    0.11674
  x, y:    1.93130  -8.84446
  i, j:     194     99

Writing grid to file test.p3d ...

Writing grid quality information to file test_stats.p3d ...
```

If possible, you should shoot for less than 30° skew angle everywhere in the grid and growth rates less than 0.2. In this case, the elliptic grid generator was used and there are some problems with grid quality near the trailing edge. At this point, the grid should be examined using the Python postprocessing tool PostPycess, which is included with Construct2D, and changes should be made to the program options to improve the grid quality. More information about using PostPycess for examining grid quality is given in section 0.4.

Finally, the program notifies of output files. There should be two output files, both with the p3d extension, which specifies that they are in Plot3D format. In this case, the project name was set to 'test,' so the output files use this prefix. The first file is the actual grid, and the second file stores the grid quality information at every point in the grid. The grid file itself will be used as an input to the CFD solver.

0.4 Visualizing the Grid

A visualizer is included with Construct2D. The visualizer is written in Python and is called PostPycess. You will need Python in order to use it, and in addition the numpy and matplotlib Python libraries need to be installed. Python comes standard with most Linux distributions, and matplotlib and numpy are generally available from the repositories. On Windows and Mac, there is a free Python distribution which includes numpy and matplotlib, called Enthought Canopy (`https://www.enthought.com/store/` - the free version should be fine). If you are on Windows using the Enthought Canopy Python distribution, it is recommended that you change the properties of postpycess.py (right-click and select 'Properties') and check that it opens with Canopy.

On any machine, there are multiple ways to get Python. You can use the Enthought Canopy distribution for Windows or Mac OS X (or even Linux), or you can download the binaries or even build it from source. In any case, PostPycess is written for Python 2.x (not 3.x), and the 2.7.x branch is recommended. After generating a grid with Construct2D, make sure the .p3d files are in the same directory is PostPycess, then open a terminal in this directory and run PostPycess. On Linux or Mac OS X (if you just installed Python and the libraries), type the following:

```
./postpycess.py
```

If you are using the Enthought Canopy Python distribution, you can just open postpycess.py with Canopy by double-clicking it. When you do that, the working directory might not be the same as the directory in which you just double-clicked the file. In that case, you can change to the correct directory by right-clicking in the Python text area and selecting 'Change to Editor Directory,' as shown in Fig. 3.

Using Canopy tends to be slow compared to running from the command line. Therefore, it may be preferred to run from the command line instead even if you have Canopy installed. Canopy will give you the needed libraries, so there shouldn't be any problem doing it this way. To run the visualizer through the command line on Windows, as long as you have Python and the required libraries installed, open a command prompt and type the following to run PostPycess:

```
python.exe postpycess.py
```

Note that in order for the above command to work in Windows, you need to have Python and the Python libraries in your system Path. Installing the Canopy Python distribution should handle this for you automatically. When you run PostPycess, the following prompt appears initially:

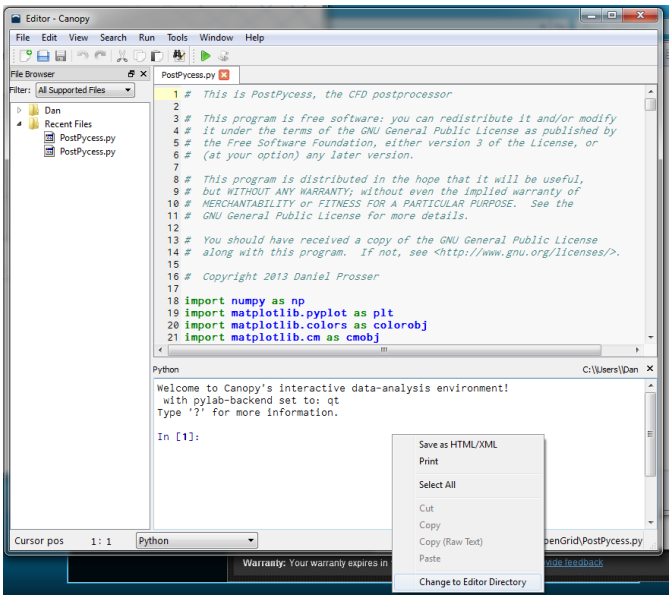


Figure 3: When running PostPycess through Canopy, set the correct working directory.

```
This is PostPycess, the CFD postprocessor
Version: 1.1

Enter project name:
```

You should enter the project name that was specified in the grid generator. In my case, it is 'test' (without the quotes). If the files are successfully read, PostPycess will say as much and then display the main menu.

```
Successfully read grid file test.p3d
Successfully read data file test_stats.p3d

Select plot type or other operation:

 1) Grid plot (grid only or colored by variable)
 2) Contour plot of variables on grid
 3) Line plot of variables on surface
 O) Change PostPycess options
 L) Load new grid and data
 Q) Quit PostPycess

Input:
```

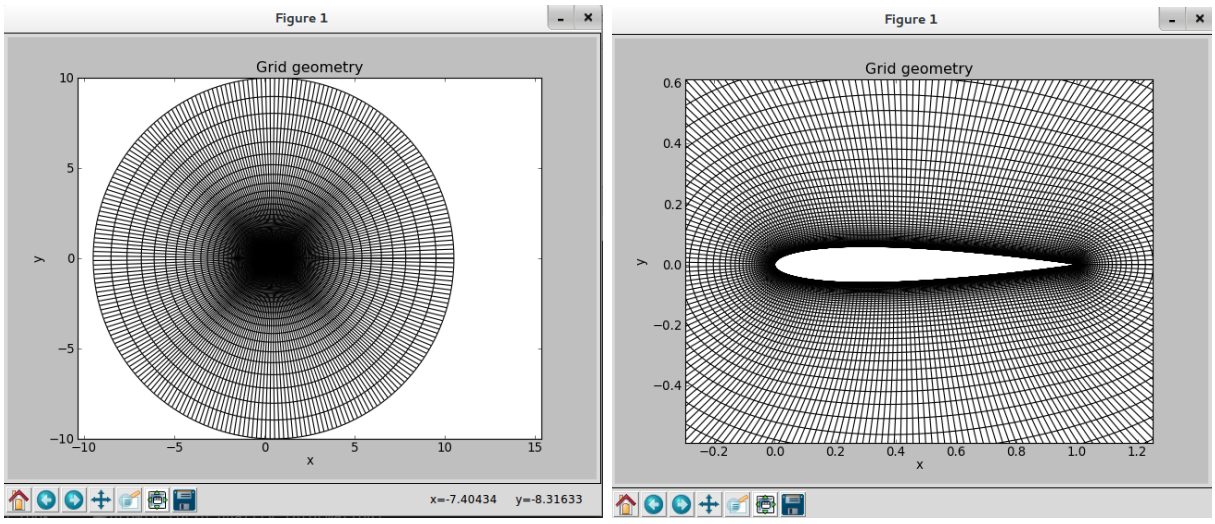
Unlike Construct2D, each entry in PostPycess is either a single number or letter, but letters may still be upper case or lower case. I want to plot the grid, so I will type '1' and press enter. Now there are more options:

```
Select plotting variable for grid plot:

 1) plain - show geometry only
 2) skew angle [Grid quality information]
 3) xi-growth [Grid quality information]
 4) eta-growth [Grid quality information]
 Q) go back to main menu

Input:
```

The options present different variables by which to color the grid. These variables are the ones that were stored in test_stats.p3d after running Construct2D. For now, I just want to look at the grid itself, so I will enter option 1 again. Now a window appears showing the grid. When it first appears, it looks as shown in Fig. 4(a). This view is completely zoomed out, allowing you to see the farfield boundaries and overall grid, but it doesn't let you see the details of the grid near the airfoil surface, which are most important. You can use the zoom and pan controls on the matplotlib window to get close to the airfoil, as shown in Fig. 4(b).

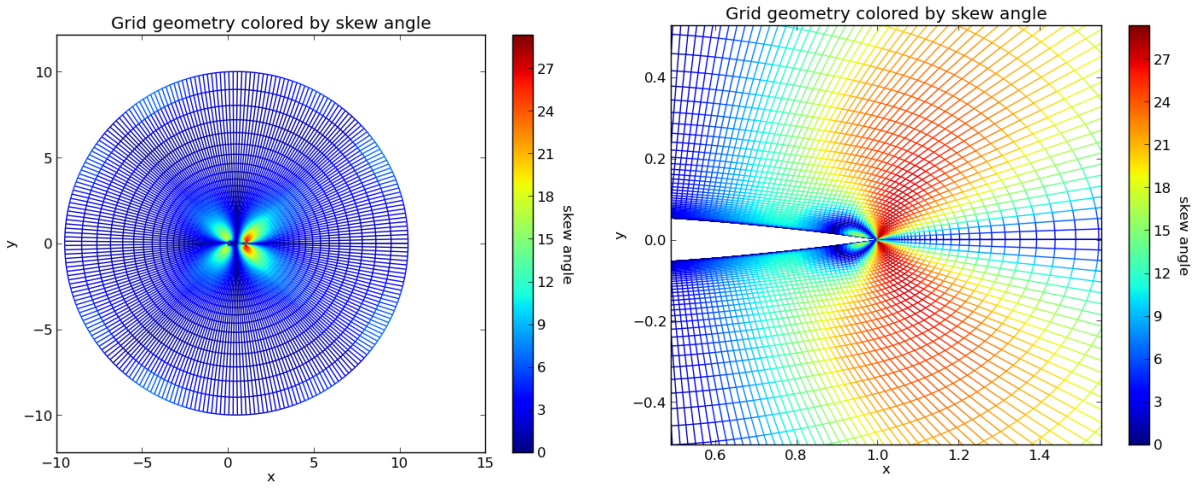


(a) Matplotlib window showing the grid

(b) Grid zoomed in near the airfoil

Figure 4: Plain grid plots using PostPycess

So, the plain grid looks pretty good, but now let's look at the grid quality information. Close the matplotlib window, and you will return to the grid plotting menu. This time, let's choose



(a) Zoomed out view of the grid

(b) Grid zoomed in near the trailing edge

Figure 5: Grid plots colored by skew angle using PostPycess

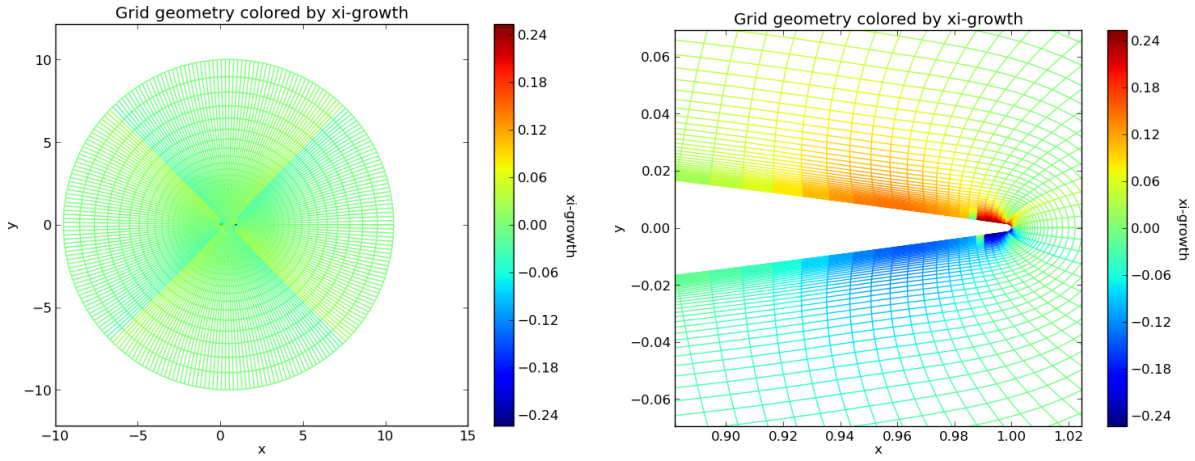
to color the grid by skew angle, or option 2. We see information about the min and max values reported:

```
Max skew angle:  29.3681936
Min skew angle:  0.0
```

In this case, the max skew angle is close to 30 degrees. That's not terrible, but it's not so good either. When we zoom in, as shown in Fig. 5(b), we see that the skew angle is worst near the trailing edge. In this case, the elliptic grid generator was used. Using the hyperbolic grid generator (the default) instead of the elliptic grid generator is usually the best way to reduce skewness. If the elliptic generator is used, or even with the hyperbolic generator, packing in some more points near the trailing edge will usually help to reduce the skewness there. This can be done using the TELE and TEPT settings in Construct2D.

We should also look at the growth rates. Let's close this plot and color by xi-growth (this is the growth rate in the i-direction, parallel to the airfoil surface). In this case, the min and max growth rates have a magnitude of about 0.25, or 25% maximum growth from one cell to another. This is really too high; we want this to stay below about 0.2 in general. If we zoom in, we see that the worst growth rates are also near the trailing edge, as shown in Fig. 6(b). The TELE, BUNC, and TEPT options can be adjusted in Construct2D to improve this problem (TELE is especially useful for this; increase the value to add more points near the trailing edge).

Instead of using the grid plots to visualize grid quality, we could have also used contour plots or surface plots (the latter if we were only interested in the grid quality at the airfoil surface). However, these types of plots are more useful for visualizing flow quantities, like pressure, velocity, etc. which would be computed by the flow solver. For visualizing grid quality,



(a) Zoomed out view of the grid

(b) Grid zoomed in near the trailing edge

Figure 6: Grid plots colored by xi-growth using PostPycess

the colored grid plots are usually best. The grid plots take longer to display and are less responsive, though, due to the number of lines that need to be drawn, so if you have a very fine grid you might consider using a contour plot instead.

Also note that there are a number of options you can change to modify how PostPycess displays your data. These are changed by entering 'O' at the main PostPycess menu. The options are shown below:

```
Select option to change:

 1) Color for plain plots (current = black)
 2) Colormap for colored grid and contour plots (current = jet)
 3) Number of levels in contour plots (current = 15)
 4) Top-surface variable color in line plot (current = red)
 5) Bottom-surface variable color in line plot (current = blue)
 Q) Go back to the main menu

Input:
```

These options should be fairly self-explanatory. Feel free to modify any of them to your liking. Finally, to close PostPycess, just enter 'Q' until you return to the main menu and then enter 'Q' again to close the program.