FUN3D
Fully Unstructured Navier-Stokes

Current Release: **12.4-70371**
Site Last Updated: **Wed Jun 04 16:41:00 -0400 2014**

Previous (C2: Installation) | Up | Next (C4: Boundary Conditions)

Site Contents

Chapters

3. Grid Generation

3.1. 2D Grid Generation

Compiling the Codes

[NOTE: The 2D grid generation tools are not normally distributed with the FUN3D suite of source code. If you wish to obtain a copy, let us know.]

If you requested the 2D grid generation tools, you should have received a tar file from Langley containing scripts that wrap the AFLR2 grid generator itself, which you will have to obtain from the SimCenter at Mississippi State. We use these scripts to automate the grid generation process from xy-coordinates all the way to an output grid format that is compatible with FUN3D's preprocessor.

The first thing you will need to do is untar the AFLR2 package. Follow the directions within the distribution to install the basic grid generator on your machine. You may need to coordinate with your system administrator to get the files all into the appropriate directories on your system. Once complete, be sure the executables in the AFLR2 suite are available in your path.

Then untar the file containing the scripts you received from the FUN3D team. Edit the script *CompileCodes* and replace the compiler name and flags with whatever is relevant for your system. You may also need to change the path to your X11 libraries. Once done, execute this file to compile several small utility codes that the mesh generation script will call under the hood:

- minspac
- splineviewS
- splinetomarcumS

- coarsenS
- afltofunS
- aflfgridtofun
- maxmeshS
- realincS

minspac is a stand-alone code that can be used to estimate the minimum wall spacing parameter for viscous grids. Just enter *minspac* and the code will ask you for a Reynolds number then supply estimates of the initial wall spacing you should request from the grid generator.

splineviewS is another stand-alone code that you will execute as described below. It is used to spline an input geometry.

The remainder of the codes you compiled above are called internally by the *meshmaker* script you received in the tar file. You will not need to execute them explicitly yourself.

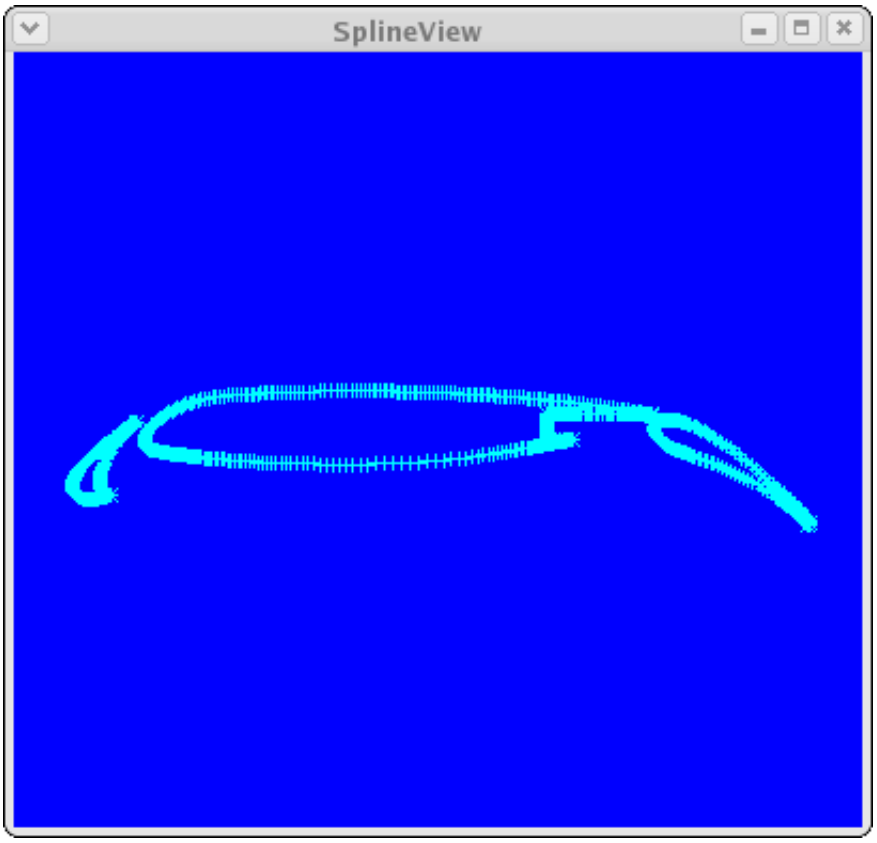
CONSTRUCTING A GRID

Splining Your Geometry

You will need your geometry description in x-y format. Edit the input file called *data*. This is the input for *splineviewS*, which will fit a spline to your input geometry.

Follow the format given in the example data file. Complete each section of the file for each curve and segment of your geometry. Note that your geometry should generally run in a clockwise fashion, so that the computational domain is kept to the “left”.

Directly below the x-y definition of each curve segment is a section of input including such parameters as number of points to be placed on the segment, etc. In general, you should only need to adjust the first two values. The first value specifies how many points you wish to be placed on the spline that is going to be fit to the current segment. This will more or less end up being the number of grid points on that segment. (The mesh generator may add a small number of points for a smoother distribution, but this is ball park what you will wind up with.) The second value in this set of data specifies the number of knot control points you would like along this segment. For each knot point you pick, you can specify a spacing parameter which will control clustering in the vicinity of the control point. Finally, the last set of lines for each data segment are the knot points you wish to use, followed by the spacing parameter for each.



When you think you have the input data file set up, execute the code *splineviewS*. If you set up the data file correctly, you will see your input geometry appear in red in an X-window. Click on the right mouse button, and you will see a blue curve appear. This is the spline that has been fit to your geometry, along with the point distribution you set up. Click the right mouse button once more, and the window will close, writing out the output file *spline.out*, which contains the spline information.

Tip: Concentrate on just getting your geometry read in correctly at first, ignoring the clustering. If there are problems with your data file, set the number of curves to 1, and get the first curve working correctly. You can even work one curve segment at a time. In this manner, you can systematically progress through your input geometry until *splineviewS* successfully reads the entire definition.

Tip: Once your entire geometry is being read in correctly, then go back and work on your clustering one curve or segment at a time.

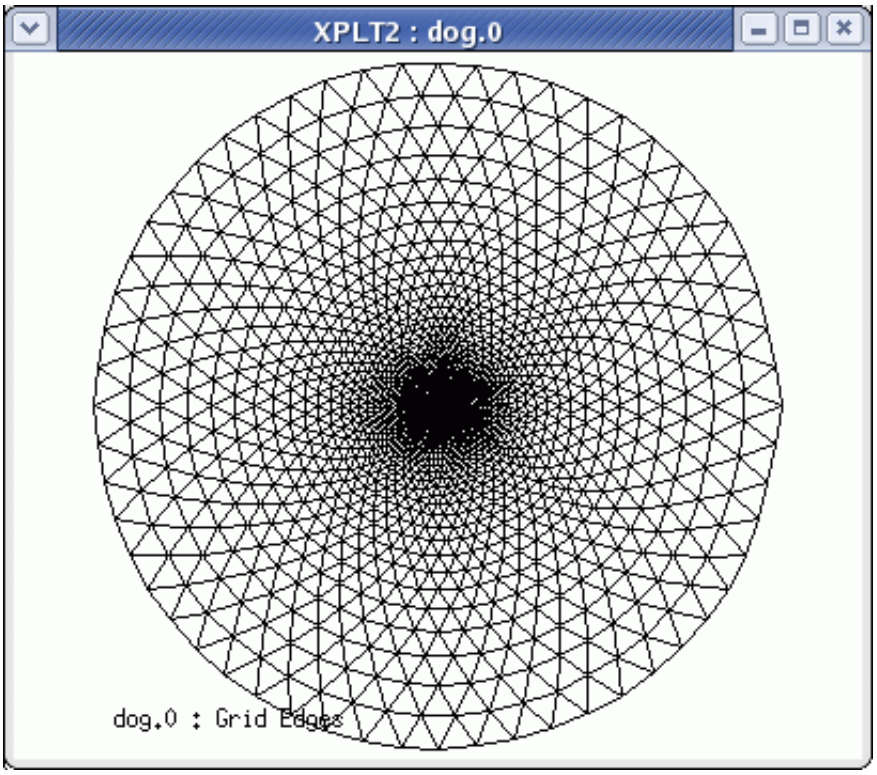
Tip: Within *splineviewS* , you may use the left button to translate the view, and the middle button to zoom in or out in the current view.

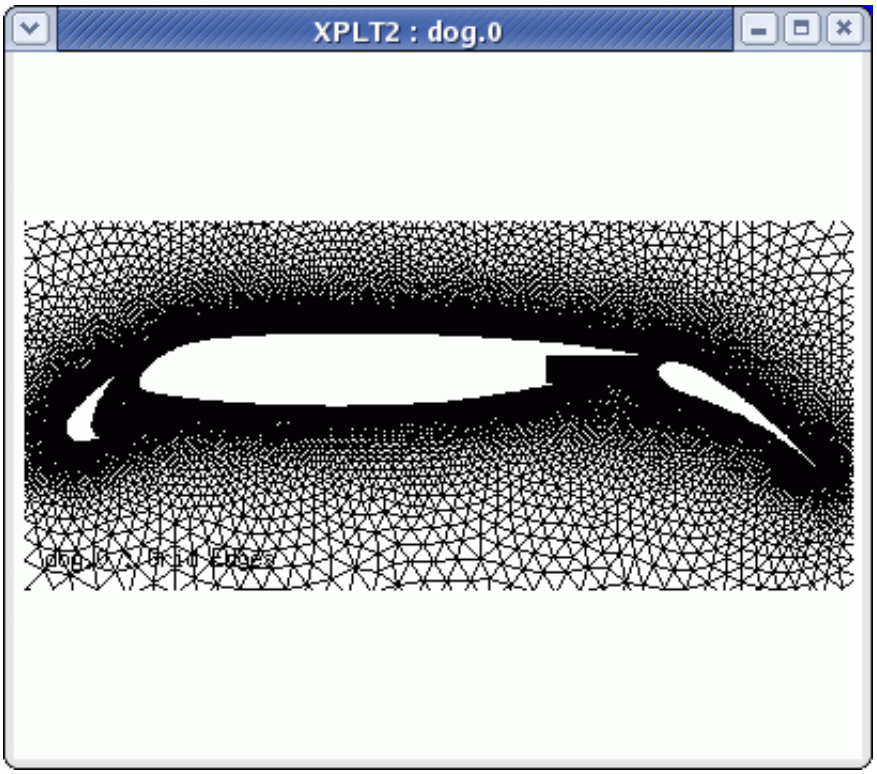
Tip: You may want to try to generate a grid using the default information in the *data* file before you change the input to suit your problem, just to get the hang of it.

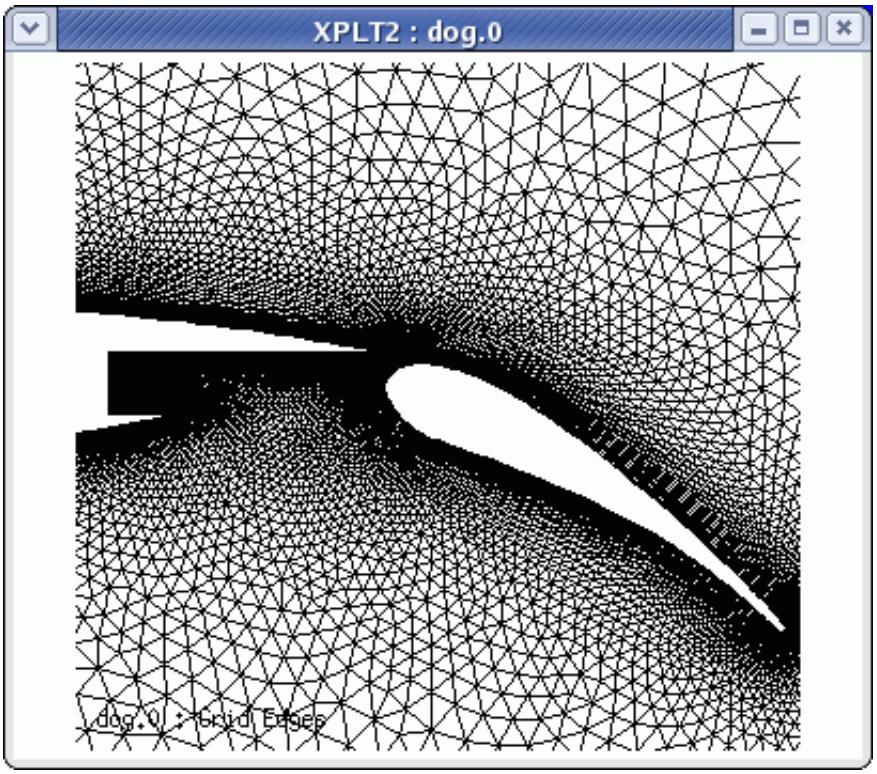
Generating a Grid

Edit the file *spline.out*, which came out of *splineviewS*. The first line contains the number of curves in the file. The second line states the number of segments in the first curve. The next line is the number of data points on the current segment, followed by a boundary condition flag for the segment. The values of the flag can be 0 (tangency), 1 (viscous), or 2 (farfield). These can be changed at this time if desired (they actually originate in the blocks of data in the *data* file for *splineviewS* , and can be set prior to running *splineviewS* if desired).

Now execute the script *meshmaker*. This script automates the grid generation process and hides the details from the user. The script will prompt you for the input spline file, which is spline.out. Enter 1 for the coarsen factor, and 1 for the number of mesh levels to generate. (The script is capable of generating a family of grids for multigrid, however we generally don't run multigrid.) The next input the script will ask for is a case name. Simply enter a one-word name such as "dog". All of the output files will then have the prefix "dog". If you have specified viscous boundaries in your spline input, the script will ask you for a wall spacing parameter. This is the distance from the surface to the first set of grid points in the boundary layer. A good estimate of the required spacing can be obtained from *minspac* , described above.







The mesh generator will then construct a grid, showing you the number of points and elements as it progresses. Upon completion, the script will ask you whether you would like to see the mesh plotted on the screen. If you answer yes, a plotting window will appear with your mesh displayed. Use the “z” key along with your mouse to zoom, “r” to reset the view, and “q” to quit when you are done. Upon exiting the graphics window, the script will complete. Your mesh is contained in the file *dog.0.faces*. This file can be read directly into Party using the corresponding main menu option for this format.

Tip: If *meshmaker* does not appear to execute correctly, you may have problems with executables in your path. To debug, simply edit the *meshmaker* script and put some write statements in to get a feel for what's going wrong. There isn't much to the script. Also, watch out for the command line options given to AFLR2 within the script. These have changed occasionally over the years – hopefully the options we have provided are consistent with the AFLR2 distribution you received, but we may have to work with you if you run into problems.

3.2. 3D Grid Generation

Three-dimensional grid generation is much more involved, and is left entirely up to the user. Our most common sources of 3D grids are VGRID (NASA Langley) and SolidMesh/AFLR3 (Mississippi State). We also prefer the GridEx package (NASA Langley), which serves as an interface for a variety of CAD packages and grid generators.

IMPORTANT: 3D grids should have the z-direction as "up" and the x-direction as "streamwise" If this is not the case, the specification of alpha (angle of attack) and beta (yaw) will not have the usual meanings.

Previous (C2: Installation) | Up | Next (C4: Boundary Conditions)

Today's NASA Official: Bil Kleb, a member of The FUN3D Development Team
Contact: FUN3D-support@lists.nasa.gov