AE 6042 A/Q Computational Fluid Dynamics

Assignment 4 (Due Wednesday, April 3, 2024)

This assignment will provide the foundational pre- and post-processing experience required to complete the final computer project. It will provide experience required to (1) preprocess a grid by augmenting it with the necessary "halo" (or "ghost") cells, (2) calculate the grid metrics for application of the finite-volume formulation (i.e., the projected cell face areas and cell volumes), and (3) implement the algorithmic steps necessary to develop a two-dimensional Euler solver.

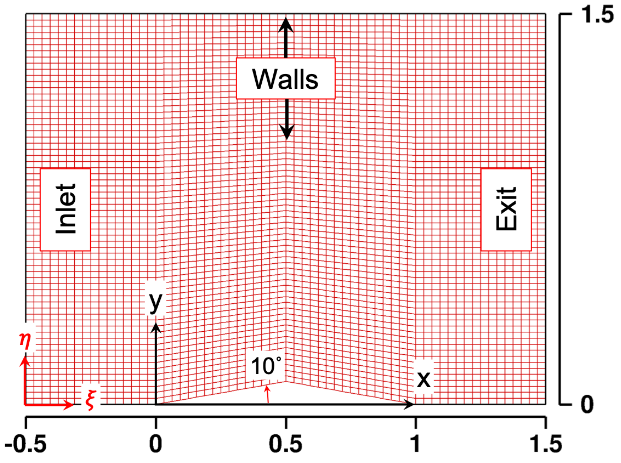

Figure 1. Computational domain and example grid for calculation of a supersonic flow over the top half of a diamond shaped airfoil in a channel.

Figure 1 shows a diagram of an example computational domain and grid for calculation of a supersonic flow over the top half of a diamond shaped airfoil. The leading edge of the airfoil is placed at $(x, y) = (0, 0)$, and trailing edge at $(x, y) = (1, 0)$. The airfoil is symmetric with respect to x and y. Its maximum half-thickness is located at $x = 0.5$ and is defined by two planar surfaces oriented at an angle of 10-degrees with respect to the leading and trailing edges (i.e. $y_{\text{half-thickness}} \approx 0.0882$). Grid points are described by (x, y) locations and (i, j) locations, where the i index follows the ξ direction and the j index follows the η direction. The primary grid has the integer dimensions $1 \le i \le nx$ and $1 \le j \le ny$.

The grid shown in Fig. 1 has $nx = 65$ points in the i direction and $ny = 49$ points in the j direction. The coordinate data for this grid is given in the file named "g65x49u.dat," which will provide the input data required to complete this assignment. For convenience, this file is in ASCII format so it can be opened and read directly by any computer architecture.[1] The first line of the file contains the grid dimensions (nx, ny). The remaining lines list the respective x, y coordinate pairs. Data from this file must be read in the order in which it was output, as illustrated in the following example code:

[1]Note that most production grids are stored as unformatted files that require less memory and can be read faster.

```
      program readgrid
      implicit none
      integer :: i,j,nx,ny
      real, dimension(:,:), allocatable :: x2d,y2d

      open (10,file='g65x49u.dat',status='unknown',form='formatted')
      read(10,*) nx,ny
      allocate(x2d(nx,ny),y2d(nx,ny))
      do j=1,ny
      do i=1,nx
         read(10,*) x2d(i,j), y2d(i,j)
      end do
      end do
      close(10)
      deallocate(x2d,y2d)

  999 format(1x,'ZONE',1x,'i=',i0,',',1x,'j=',i0)
 1000 format(e23.16,',',e23.16)

      stop
      end
```

The syntax used above corresponds to modern FORTRAN. However, this can easily be converted to any preferred coding language (e.g., C, C++, etc.). A compilable version of the program above is provided in the file "readgrid.f90" along with a simple Makefile that contains relevant options for various compilers. Also included are the equivalent codes written in C (see attached file "readgrid.c") and Python3 (see attached file "readgrid.py"). These programs will read, then output the grid into a different file to verify the operation was successful. Note that if you choose to use MATLAB the syntax you use will be very similar to C.

Using the grid data provided in the file "g65x49u.dat" and your preferred programming language, compiler, and plotting package(s), perform the following tasks:

1. Demonstrate that you can correctly read, then write out the grid, by providing a neatly formatted plot of the grid topology (similar to Fig. 1). The plot should include the coordinate axes, but you do not have to include the additional labels provided in Fig. 1.
2. Following the discussion related to the sketch of the finite-volume domain on page 7 of the Topic 17 notes, augment the primary grid given above with "halo-cells" by reflecting the adjacent interior coordinates across the respective grid boundaries. To accomplish this, you will want to dimension your grid arrays over the ranges x(0:nx+1,0:ny+1) and y(0:nx+1,0:ny+1). Recall that the primary grid occupies the ranges x(1:nx,1:ny) and y(1:nx,1:ny). Provide a plot of the augmented grid using the same format as was used for Task 1 above.
3. Using the methodology presented in the Topic 19 notes (e.g., starting on page 6), and the corresponding paper by Kordulla and Vinokur which has been uploaded to Canvas (*AIAA J*, **21**, 1983), derive analytic expressions for respective cell volumes and projected cell face areas for the two-dimensional cells of interest here. Note that these expressions will be simpler than those for a three-dimensional cell (see file "2D-FV-metrics.pdf" for the answer key to help guide you). Provide your derivations in a clear and systematic manner (i.e., you can simply use the results given in "2D-FV-metrics.pdf," but you must provide a concise write-up that shows you understand the exactly what each of these terms are).

4. Using the derivations from Task 3 and a format similar to that used for the plots presented for Tasks 1 and 2, provide a set of contour plots showing the variations in respective cell volumes and projected cell face areas. Note that the storage locations for the cell volumes correspond to the cell centers. The storage locations for the projected cell face areas correspond to respective cell faces (i.e., the face area centroids). Thus, to construct these plots it is convenient to define a set of cell-centered coordinates dimensioned consistently via the arrays xc(0:nx,0:ny) and yc(0:nx,0:ny). The cell volumes should then be plotted over the ranges (0:nx,0:ny). Similarly, cell faces perpendicular to the ξ directions should be plotted using coordinates over the range (1:nx,1:ny-1), and cell faces perpendicular to the η direction should be plotted using coordinates over the range (1:nx-1,1:ny).

Present your results in a short concisely written report that clearly outlines your approach and the programming language, compiler(s), and plotting package(s) used.

Each student needs to write their own code and include a printout of the code as an appendix to the report. The code should be clearly commented and be original work. Plagiarizing or using a code written by someone else will be considered a violation of the Georgia Tech Honor Code.