CHAPTER 5 File Formats

This chapter describes the input and output files listed in Section 3.1. Typical names are given below; however, these names can be changed by the user to pertain to certain cases.

```
I/O FILES
grid.bin
plot3dg.bin
plot3dq.bin
cfl3d.out
cfl3d.res
cfl3d.turres
cfl3d.blomax
cfl3d.out15
cfl3d.prout
cfl3d.out20
ovrlp.bin
patch.bin
restart.bin
```

5.1 Input Files

There are two input files that *must* be available to CFL3D whenever a case is submitted. These files are the grid file (unit 1) and the input file (unit 5) defining the case. The latter file is described in detail in Chapter 3. A restart file (unit 2) *must* be available for a restart case; however, for a start from "scratch", it is not needed. In addition, there are two files that *may* be needed at submission depending on the problem being solved. When overlapped grids are used, the interpolation and boundary data file (unit 21) must be available. When patching is utilized, the patched boundary data file (unit 22) must be available.

5.1.1 Grid File

grid.bin (unit 1)

The grid must be generated using grid-generation software (CFL3D does not generate grids). Take care when generating a grid. Grid spacings should vary smoothly; the grid should "look nice". (See the troubleshooting notes about grids in Chapter 10). Unless there is a good reason for not doing so, the grids, as well as all segments, should be multigridable (see Section 7.1.2 on page 127). Grid stretching should not be too severe if it can be avoided since it can degrade the accuracy and convergence of the code.

The grid file is in binary format. In the input file, the parameter **ngrid** is set as the total number of grids to be read in by CFL3D. The *sign* of **ngrid** indicates whether the grid will be read in CFL3D format or in PLOT3D format. (Note that TLNS3D[42] uses PLOT3D format.) CFL3D-type grids always have **alpha** measured in the $x - z$ plane, with z as the "up" direction (i.e. **alpha** = $90°$ would give a free-stream velocity vector in the positive z direction). PLOT3D-type grids may be utilized with either z being "up" (**alpha** measured in the $x - z$ plane) or y being "up" (**alpha** measured in the $x - y$ plane) depending on the value of **ialph** (see "LT3 - Flow Conditions" on page 19). See Appendix I for an illustrated definition of α. When PLOT3D-type grids are input, they are immediately converted internal to the code to CFL3D format. Thus the actual computations are *always* carried out internally as if z is "up".

If **ngrid** > 0, the following CFL3D grid format is used to read in the grid(s):

```
      do 10 n=1,ngrid
      read(1) jdim(n),kdim(n),idim(n)
      read(1) (((x(i,j,k,n),j=1,jdim(n)),k=1,kdim(n)),i=1,idim(n)),
     .        (((y(i,j,k,n),j=1,jdim(n)),k=1,kdim(n)),i=1,idim(n)),
     .        (((z(i,j,k,n),j=1,jdim(n)),k=1,kdim(n)),i=1,idim(n))
   10 continue
```

If **ngrid** < 0, the following PLOT3D grid format is used to read in the grid(s):

```
      read(1) ngrid
      read(1) (idim(n),jdim(n),kdim(n),n=1,ngrid)
      do 20 n=1,ngrid
      read(1) (((x(i,j,k,n),i=1,idim(n)),j=1,jdim(n)),k=1,kdim(n)),
     .        (((y(i,j,k,n),i=1,idim(n)),j=1,jdim(n)),k=1,kdim(n)),
     .        (((z(i,j,k,n),i=1,idim(n)),j=1,jdim(n)),k=1,kdim(n))
   20 continue
```

It does not matter which grid format is used. What *does* matter is the particular orientation of i, j, and k relative to the body and flow field, since the matrix inversions are done in a particular order. Experience has indicated that it is usually best (from speed of convergence perspective) to have the k direction be the primary viscous direction (i.e., the primary direction of the grid lines normal to the body). If bodies exist with more than one grid line orientation, then choose one of them to serve as the indicator of the "primary" viscous direction. Also, it is usually best if the j direction is taken as the streamwise direction and the i direction as the spanwise direction. Hence, for an existing grid with i is in the streamwise direction, j in the normal direction, and k in the spanwise direction, it might be a good idea to first swap the indices: $i \rightarrow j$, $j \rightarrow k$, and $k \rightarrow i$, to put them in CFL3D's "preferred" orientation before running. The new grid created can be either CFL3D-type or PLOT3D-type. Finally, note that it is not always necessary to follow this guideline; performance is case dependent. Just be sure to satisfy the right-hand rule at all times, as discussed in the next section.

For 2-D grids (**i2d**=1), **idim** must be 2 (i.e., the grid must be described by 2 repeated planes). We recommend making the grid-distance between the planes = 1, although any distance is okay provided that it is accounted for in the **sref** term in the input file.

5.1.2 The Right-Hand Rule

The right-hand rule *must* be obeyed by the grid coordinates (x, y, z) *and* the grid indices (i, j, k). Error messages and an immediate halt to execution will result if it is not. As a reminder of this important rule, consider the system of axes in Figure 5-1. Imagine pointing the right-hand index finger along the positive x axis. Then curl the rest of the right-hand fingers toward the positive y axis. The thumb, pointing straight out, is now in the positive z direction.

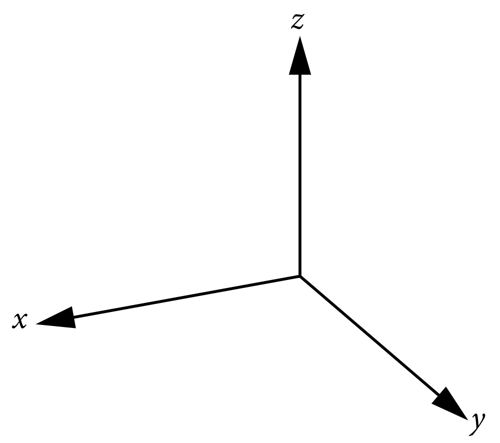

Figure 5-1. Three-dimensional system of axes showing positive x, y, and z directions.

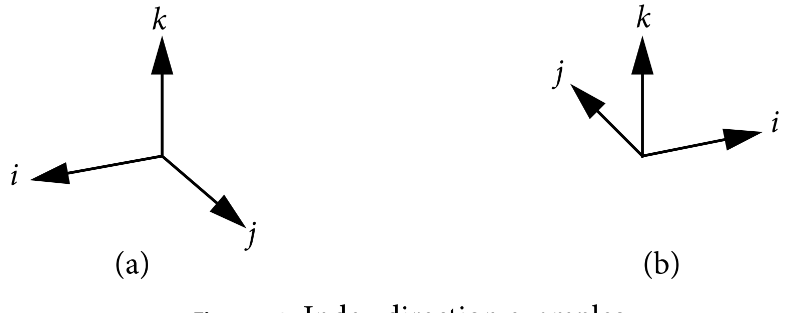

Figure 5-2. Index direction examples.

Keep in mind that the same rule applies to the indices i, j, k. Suppose the positive i, j, and k directions run along the positive x, y, and z axes, respectively, as drawn in Figure 5-1. This is the simplest case and is illustrated in Figure 5-2(a). Suppose it is necessary, however, for the positive i index direction to run in the opposite direction of positive x as drawn in Figure 5-2(b). (This means, for example, that if x varies from 0.0 to 1.0 and **idim** = 5, then at $i = 1$, $x = 1.0$ and at $i = 5$, $x = 0.0$.) One of the other index directions must also be changed to maintain the right-hand rule. In Figure 5-2(b), the j index direction has been changed.

Also, i, j, and k do not *have* to be in the x, y, and z directions, respectively. Any permutation is valid as long as the right-hand rule is upheld. (However, keep in mind the discussion above in Section 5.1.1 as well as *Note* (6) describing the **ivisc**(m) input on page 27 when setting the index directions.)

5.1.3 Restart File

restart.bin (unit 2)

The restart file is in binary format. It must be available at the beginning of the program execution for a restart of a previous solution (**irest** > 0). *The restart file is overwritten during program execution; therefore, before resubmitting the case, be sure to save any restart files that may be needed later!* (In the following, `nblk` is the number of blocks. Also, note that in versions of CFL3D prior to 2001-2002, `irghost` and `itime2read` both always = 0, but in later versions the defaults are 1):

```
      do nb=1,nblk
      read(3) titlw,xmachw,jdim,kdim,idim,alphw,reuew,ntr,time
      if (nb.eq.1) then
         read(3)  (rms(n),     n=1,ntr),(clw(n),     n=1,ntr),
     .            (cdw(n),     n=1,ntr),(cdpw(n),    n=1,ntr),
     .            (cdvw(n),    n=1,ntr),(cxw(n),     n=1,ntr),
     .            (cyw(n),     n=1,ntr),(czw(n),     n=1,ntr),
     .            (cmxw(n),    n=1,ntr),(cmyw(n),    n=1,ntr),
     .            (cmzw(n),    n=1,ntr),(fmdotw(n),  n=1,ntr),
     .            (cftmomw(n), n=1,ntr),(cftpw(n),   n=1,ntr),
     .            (cftvw(n),   n=1,ntr),(cfttotw(n), n=1,ntr)
      end if
      read(3) ((((q(j,k,i,l,nb),j=1,jdim-1),k=1,kdim-1),
     .               i=1,idim-1),l=1,5)
      if (irghost .ne. 0)
     .   read(3)  ((((qi0(j,k,l,m),j=1,jdim-1),k=1,kdim-1),l=1,5),m=1,4),
     .            ((((qj0(k,i,l,m),k=1,kdim-1),i=1,idim-1),l=1,5),m=1,4),
     .            ((((qk0(j,i,l,m),j=1,jdim-1),i=1,idim-1),l=1,5),m=1,4)
      read(3) iv1,iv2,iv3
      if (nb.eq.1) then
         read(3)  (rmstr1(n),n=1,ntr),(rmstr2(n),n=1,ntr),
     .            (nneg1(n), n=1,ntr),(nneg2(n), n=1,ntr)
      end if
      if (iv1.ge.2 .or. iv2.ge.2 .or. iv3.ge.2) then
         read(3) (((vist3d(j,k,i,nb),j=1,jdim-1),k=1,kdim-1),i=1,idim-1)
         if (irghost .ne. 0)
     .     read(3) ((((vi0(j,k,l,m),j=1,jdim),k=1,kdim),l=1,1),m=1,4),
     .             ((((vj0(k,i,l,m),k=1,kdim),i=1,idim-1),l=1,1),m=1,4),
     .             ((((vk0(j,i,l,m),j=1,jdim),i=1,idim-1),l=1,1),m=1,4)
      end if
      if (iv1.ge.4 .or. iv2.ge.4 .or. iv3.ge.4) then
         read(3) ((((tursav(j,k,i,m,nb),j=1,jdim-1),k=1,kdim-1),
     .                i=1,idim-1),m=1,2)
         if (irghost .ne. 0)
     .     read(3) ((((ti0(j,k,l,m),j=1,jdim),k=1,kdim),l=1,2),m=1,4),
     .             ((((tj0(k,i,l,m),k=1,kdim),i=1,idim-1),l=1,2),m=1,4),
     .             ((((tk0(j,i,l,m),j=1,jdim),i=1,idim-1),l=1,2),m=1,4)
         read(3) (((smin(j,k,i,nb),j=1,jdim-1),k=1,kdim-1),i=1,idim-1)
         if (iv1.eq.4 .or. iv2.eq.4 .or. iv3.eq.4) then
         read(3)(((xjb(j,k,i,nb),j=1,jdim-1),k=1,kdim-1),i=1,idim-1)
         read(3)(((xkb(j,k,i,nb),j=1,jdim-1),k=1,kdim-1),i=1,idim-1)
```

```
          read(3)(((blnum(j,k,i,nb),j=1,jdim-1),k=1,kdim-1),i=1,idim-1)
          end if
          if (iv1.eq.8 .or. iv2.eq.8 .or. iv3.eq.8 .or. iv1.eq.9 .or.
     .        iv2.eq.9 .or. iv3.eq.9 .or. iv1.eq.13.or. iv2.eq.13.or.
     .        iv3.eq.13.or. iv1.eq.14.or. iv2.eq.14.or. iv3.eq.14) then
          read(3) (((cmuv(j,k,i,nb),j=1,jdim-1),k=1,kdim-1),i=1,idim-1)
          end if
       end if
       end do
       if (dt.gt.0) then
         read(3) iflagg
         if (iflagg.eq.1 .or. iflagg.eq.3) then
           do nb=1,nblk
              read(3) jdim,kdim,idim
              read(3) ((((qc0(j,k,i,l,nb),j=1,jdim-1),k=1,kdim-1),
     .                     i=1,idim-1),l=1,5)
              if (itime2read .ne. 0) then
                read(3) dt
                read(3) ((((tursav2(j,k,i,l),j=1,jdim-1),k=1,kdim-1),
     .                    i=1,idim-1),l=1,2)
              end if
           end do
         end if
         if (iflagg.eq.2 .or. iflagg .eq.3) then
           do nb=1,nblk
              read(3) iunst
              if (iunst .ne. 0) then
                read(3) jdim,kdim,idim
                read(3) itrans,rfreqt,xorig,yorig,zorig,xorig0,yorig0,zorig0,
     .                   utrans,vtrans,wtrans,dxmx,dymx,dzmx,itransmc,rfreqtmc,
     .                   xorigmc,yorigmc,zorigmc,xorig0mc,yorig0mc,zorig0mc,
     .                   utransmc,vtransmc,wtransmc,xmc,ymc,zmc,dxmxmc,dymxmc,
     .                   dzmxmc,irotat,rfreqr,thetax,thetay,thetaz,omegax,
     .                   omegay,omegaz,dthxmx,dthymx,dthzmx,irotatmc,rfreqrmc,
     .                   thetaxmc,thetaymc,thetazmc,omegaxmc,omegaymc,
     .                   omegazmc,dthxmxmc,dthymxmc,dthzmxmc,time2,time2mc,dt
              end if
           end do
         end if
       end if
```

Where	*Indicates*
`titlw`	a real array of length 20 describing the case
`xmachw`	Mach number
`jdim,kdim,idim`	grid dimensions
`alphw`	angle of attack
`reuew`	Reynolds number
`ntr`	number of iterations
`time`	time (for time-accurate computations)
`rms`	residual
`clw`	lift coefficient
`cdw`	drag coefficient
`cyw`	force coefficient in y direction
`cmyw`	moment coefficient about y
`cdpw`	pressure drag coefficient
`cdvw`	viscous drag coefficient
`cxw`	force coefficient in x direction
`czw`	force coefficient in z direction
`cmxw`	moment coefficient about x

`cmzw`	moment coefficient about z
`fmdotw`	mass flow through control surface
`cftmomw`	magnitude of resultant momentum force on control surface
`cftpw`	magnitude of resultant pressure force on control surface
`cftvw`	magnitude of resultant viscous force on control surface
`cfttotw`	magnitude of resultant (pressure + momentum + viscous) force on control surface
`q`	nondimensional primitive variables; $\mathbf{q} = [\rho, u, v, w, p]$
`qi0, qj0, qk0`	$\mathbf{q}$ boundary condition values
`iv1,iv2,iv3`	**ivisc**(1), **ivisc**(2), and **ivisc**(3)
`rmstr1`	residual for first turbulence model equation (if used)
`rmstr2`	residual for second turbulence model equation (if used)
`nneg1`	number of points at which first turbulence model variable needed to be limited
`nneg2`	number of points at which second turbulence model variable needed to be limited
`vist3d`	nondimensional turbulent eddy viscosity μ_T
`tursav`	nondimensional turbulence quantities for field-equation turbulence models (varies with model); `tursav`(2) is also used to store nearest wall i location for the Baldwin-Barth turbulence model
`smin`	minimum distance function for field-equation turbulence models
`xjb`	nearest wall j location for Baldwin-Barth turbulence model
`xkb`	nearest wall k location for Baldwin-Barth turbulence model
`blnum`	nearest wall block number for Baldwin-Barth turbulence model
`cmuv`	variable C_μ coefficient for Girimaji turbulence model
`qc0`	$\mathbf{q}$ values at previous time step
`dt`	time step
`tursav2`	turbulence quantities at previous time step
`iflagg`	parameter describing temporal order and whether grid is moving or stationary

= 0	not second order, stationary grid
= 1	second order, stationary grid
= 2	not second order, moving grid
= 3	second order, moving grid

The dynamic grid parameters in the restart file are not described here.

5.1.4 Grid-Overlapping and Patching Data Files

ovrlp.bin (unit 21)

The `ovrlp.bin` file is a binary file, which is only used for grid-overlapping cases. It is always listed at the top of the input deck. The file contains the interpolation stencils for overlapped grids and is created by running MaGGiE.

patch.bin (unit 22)

The `patch.bin` file is a binary file, which is only used for grid-patching cases. It is also always listed at the top of the input deck. The file contains the interpolation stencils for patched grids and is created by running ronnie.

5.2 Output Files

CFL3D produces several output files. The following sections describe the contents and formats of these files.

5.2.1 PLOT3D *Files*

The PLOT3D files are binary files (or ascii, with the modification to the code as described in the footnote at the beginning of Chapter 3) which are input to flow visualization programs like PLOT3D or FAST. Partial or entire flow-field regions may be specified by the user in the input file. The parameter **nplot3d** specifies the number of regions to be written to the files and "LT28 - PLOT3D Output Specifications" on page 39 indicates the desired index ranges. Data may also be extracted from these files for post-processing programs written by the user. In order to read these files for such a purpose, use the formats below. The number of grids to be read is indicated by `nplots` and `id`, `jd`, `kd` are the number of points to be read on each grid. Note that these files can be specified as grid-point data files (**iptype** = 0) or cell-center data files (**iptype** = 1).

plot3dg.bin (unit 3)

For 3-d grids,

```
   write(3) nplots
   write(3)(id(n),jd(n),kd(n),n=1,nplots)
   do n=1,nplots
   write(3) (((x(j,k,i,n),i=1,id(n)),j=1,jd(n)),k=1,kd(n)),
  .         (((y(j,k,i,n),i=1,id(n)),j=1,jd(n)),k=1,kd(n)),
  .         (((z(j,k,i,n),i=1,id(n)),j=1,jd(n)),k=1,kd(n)),
  .         (((iblank(j,k,i,n),i=1,id(n)),j=1,jd(n)),k=1,kd(n))
   end do
```

For 2-d grids,

```
   write(3) nplots
   write(3)(jd(n),kd(n),n=1,nplots)
   do n=1,nplots
   write(3) ((x(j,k,n),j=1,jd(n)),k=1,kd(n)),
  .         ((z(j,k,n),j=1,jd(n)),k=1,kd(n)),
  .         ((iblank(j,k,n),j=1,jd(n)),k=1,kd(n))
   end do
```

plot3dq.bin (unit 4)

The flow-field data file has the following formats. For 3-d grids,

```
write(4) nplots
write(4)(id(n),jd(n),kd(n),n=1,nplots)
do n=1,nplots
write(4) xmach,alpha,reue,time
write(4) ((((q(j,k,i,m,n),i=1,id(n)),j=1,jd(n)),k=1,kd(n)),m=1,5)
end do
```

For 2-d flow-field data files,

```
write(4) nplots
write(4)(jd(n),kd(n),n=1,nplots)
do n=1,nplots
write(4) xmach,alpha,reue,time
write(4) (((q(j,k,m,n),j=1,jd(n)),k=1,kd(n)),m=1,4)
end do
```

Note that five flow-field variables are output for 3-d grids, whereas four variables are output for 2-d grids. Also, note that `q` here is the array of conserved variables **Q**.

When using **iptype** = 0, be aware that the solution on block edges and corners may be inaccurate. This is because CFL3D solves for cell-center values. Values at grid points must be reconstructed from cell-center data and the reconstruction process at edges/corners is not unique.

When **iptype** = 2, CFL3D outputs turbulence quantity information rather than the conserved variables **Q**. See the note in Section 3.28 on page 39.

When **iptype** > 2, a PLOT3D function file is output in place of the **Q** file. The function file format for 3-d grids is

```
write(4) nplots
write(4)(id(n),jd(n),kd(n),1,n=1,nplots)
do n=1,nplots
write(4) (((f(j,k,i,n),i=1,id(n)),j=1,jd(n)),k=1,kd(n))
end do
```

For 2-d function files,

```
write(4) nplots
write(4)(jd(n),kd(n),1,n=1,nplots)
do n=1,nplots
write(4) ((f(j,k,n),j=1,jd(n)),k=1,kd(n))
end do
```

Note that function files are currently written for a single function only. See Section 3.28 on page 39 for information on what functions are currently available.

5.2.2 Ascii Format Files

cfl3d.out (unit 11)

The `cfl3d.out` file is the primary output file. It echoes the input file with the appropriate bookkeeping changes, such as updates in block numbers when coarser grids are generated internally. The file then gives a "blow-by-blow" message about what steps have been performed: the grid is read in, the metrics are computed, etc. until finally the other output files are written. Another important function of this file is to print any warnings and error messages. For example, it will reveal any mismatches in one-to-one block connections, signaling a problem with the grid or input file. It is a good idea to run a new job for a few iterations only and then to study this file to make sure everything looks as expected.

cfl3d.res (unit 12)

When **ihstry** = 0, the `cfl3d.res` file contains the residual, lift, drag, side force, and moment histories. When **ihstry** > 0, the residual, mass flow, pressure force, viscous force, and momentum force are given instead. This type of file has been used for all of the "iteration history" plots presented in this manual. The following sample format shows a portion of the `cfl3d.res` file (when **ihstry** = 0) for the flat plate test case discussed in Section 9.1.5 on page 173. When sub-iterations are used in a time-accurate computation, only the final residuals and forces from the last sub-iteration of each time step are output to unit 12. (For the sub-iteration residual history file, see page 76.)

```
   turbulent flat plate (plate from j=17-65, prior to 17 is symmetry)
Mach=  0.2000E+00, alpha=  0.0000E+00, ReUe=  0.6000E+07
Final res=  0.3623E-09
Final cl,cd,cy,cmy= -0.1869E-02  0.2827E-02  0.0000E+00  0.3174E-03
  800 it    log(res)        cl            cd            cy            cmy
      1  -0.15700E+02   0.25376E-12   0.67093E+00   0.00000E+00  -0.12688E-12
      2  -0.59009E+01   0.13542E-01   0.40383E-01   0.00000E+00  -0.72299E-02
             .
             .
             .
    799  -0.94402E+01  -0.18662E-02   0.28275E-02   0.00000E+00   0.31613E-03
    800  -0.94409E+01  -0.18695E-02   0.28273E-02   0.00000E+00   0.31742E-03
```

Here, `res` is the L-2 norm of the residual for the equation for density.

cfl3d.turres (unit 13)

The `cfl3d.turres` file contains the residual history for the one or two-equation turbulence models (**ivisc** > 3). The sample below shows part of the `cfl3d.turres` file for the flat plate test case discussed in Section 9.1.5 on page 173.The `log(turres1)` quantity is the log of the residual for all one-equation models or the log of the residual for the ω or ε equation for two-equation models. `Log(turres2)` is the log of the residual for the k equation for two-equation models and is not used for one-equation models. `Nneg1` and `nneg2` indicate how many field points require limiting (to avoid going negative) during that iteration. They should be zero or, at most, a small fraction of the total number of points for a converged solution. If they are large, they indicate a problem with convergence. (They may be large during start-up, as in this example, but should always go to zero or small values as the solution progresses.) When sub-iterations are used in a time-accurate computation, only the final residuals from the *last* sub-iteration of each time step are output.

```
    turbulent flat plate (plate from j=17-65, prior to 17 is symmetry)
 Mach=  0.2000E+00, alpha=  0.0000E+00, ReUe=  0.6000E+07
 Final turres1=  0.5279E-09
 Final turres2=  0.5186E-09
   800 it  log(turres1)  log(turres2)  nneg1  nneg2
        1  -0.31193E+01  -0.43162E+01      0     92
        2  -0.34342E+01  -0.53959E+01      0   2637
                 .
                 .
                 .
      799  -0.92770E+01  -0.92830E+01      0      0
      800  -0.92774E+01  -0.92852E+01      0      0
```

cfl3d.blomax (unit 14)

The `cfl3d.blomax` file provides information about the Baldwin-Lomax turbulence model (**ivisc** = 2).

cfl3d.out15 (unit 15)

The `cfl3d.out15` file is a secondary output file. It provides more in depth information about the progression of a case. It is primarily used by the programmer for debugging purposes.

cfl3d.prout (unit 17)

The `cfl3d.prout` file contains the output as specified in the input file with the **nprint** parameter and "LT30 - Print Out Specifications" on page 41. Note that this file can be written for values at the grid points (**iptype** = 0) or cell-centers (**iptype** = 1). Keep in mind that this file can grow quite large rather quickly, so limit the regions specified for output. The `cfl3d.prout` file will also contain the control surface data if **ncs** > 0 (see Section 3.31 on page 42). (The sample below is from the flat plate test case discussed in Section 9.1.5 on page 173.) Note that `dn`, `Cf`, `Ch`, and `yplus` are only output when **iptype** = 0.

```
    turbulent flat plate (plate from j=17-65, prior to 17 is symmetry)
      Mach     alpha      beta      ReUe   Tinf,dR      time
   0.20000   0.00000   0.00000 0.600E+07 460.00000   0.00000


 BLOCK   1     IDIM,JDIM,KDIM=    2   65   97

   I  J   K     X          Y          Z        U/Uinf     V/Vinf     W/Winf     P/
Pinf     T/Tinf      MACH          cp    tur. vis.
    1   1   1 -.3333E+00 0.0000E+00 0.0000E+00 0.9987E+00 0.0000E+00 -.7200E-07
0.1000E+01 0.1000E+01 0.1997E+00 0.5992E-02 0.2054E-02
    1   2   1 -.3125E+00 0.0000E+00 0.0000E+00 0.9989E+00 0.0000E+00 0.1323E-21
0.1000E+01 0.1000E+01 0.1998E+00 0.6012E-02 0.1379E-02
                  .
                  .
                  .
    1  64   1 0.9792E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
0.9999E+00 0.1008E+01 0.0000E+00 -.3123E-02 0.1966E-09
    1  65   1 0.1000E+01 0.0000E+00 0.0000E+00 0.3675E-02 0.0000E+00 0.5962E-08
0.9999E+00 0.1008E+01 0.7320E-03 -.3607E-02 0.1963E-09

   I  J   K      X            Y            Z            dn         P/Pinf       T/
Tinf         Cf           Ch         yplus
```

```
    1   2   1 -0.31250E+00  0.00000E+00  0.00000E+00  0.10000E-05  0.10002E+01
0.10000E+01  0.00000E+00  0.00000E+00  0.00000E+00
    1   3   1 -0.29167E+00  0.00000E+00  0.00000E+00  0.10000E-05  0.10002E+01
0.10000E+01  0.00000E+00  0.00000E+00  0.00000E+00
                  .
                  .
                  .
    1  63   1  0.95833E+00  0.00000E+00  0.00000E+00  0.10000E-05  0.99994E+00
0.10080E+01  0.24677E-02  0.33657E-02  0.20858E+00
    1  64   1  0.97916E+00  0.00000E+00  0.00000E+00  0.10000E-05  0.99991E+00
0.10080E+01  0.24660E-02  0.24780E-02  0.20851E+00


 BLOCK  1     IDIM,JDIM,KDIM=    2   65   97

   I  J  K     X          Y          Z        U/Uinf     V/Vinf     W/Winf     P/
Pinf     T/Tinf      MACH          cp    tur. vis.
    1  49   1 0.6667E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
0.1000E+01 0.1008E+01 0.0000E+00 0.1029E-02 0.2188E-09
    1  49   2 0.6667E+00 0.0000E+00 0.1000E-05 0.8330E-02 0.0000E+00 -.1105E-06
0.1000E+01 0.1008E+01 0.1659E-02 0.1029E-02 0.2112E-07
                  .
                  .
                  .
    1  49  96 0.6667E+00 0.0000E+00 0.9594E+00 0.9987E+00 0.0000E+00 0.2794E-03
0.1000E+01 0.1000E+01 0.1997E+00 0.2587E-02 0.7274E-02
    1  49  97 0.6667E+00 0.0000E+00 0.9837E+00 0.9987E+00 0.0000E+00 0.2672E-03
0.1000E+01 0.1000E+01 0.1997E+00 0.2609E-02 0.7274E-02

   I  J  K      X            Y            Z            dn         P/Pinf       T/
Tinf        Cf           Ch         yplus
    1  49   1  0.66667E+00  0.00000E+00  0.00000E+00  0.10000E-05  0.10000E+01
0.10080E+01  0.25533E-02  0.16414E-02  0.21218E+00
    1  49  97  0.66667E+00  0.00000E+00  0.98367E+00  0.24255E-01  0.10001E+01
0.10000E+01  0.23455E-13  0.60677E-09  0.15760E-01
```

A sample of the type of control surface information that can be expected when **ncs** > 0 is shown here (this is no longer for the flat plate test case):

```
 Control Surface  1
 ===================

 Grid  1 (Block  1)  i = 25, 25  j =  1, 33  k =  1, 25  iwall =   0  Normal =   1

 x-area =   0.2687E+02 y-area =   0.1842E+00 z-area =  -0.1842E+00 total-area =
0.2687E+02

 Mass averaged properties
 P/Pinf      =   0.9446E+00         Pt/Pinf =   0.1022E+01
 T/Tinf      =   0.9840E+00         Tt/Tinf =   0.1007E+01
 Mach number =   0.3367E+00

 Mass flow / (rhoinf*vinf*(L_R)**2) =    0.42300E+02

                     x-force       y-force       z-force   resultant-force lift-
force
   drag-force
 Pressure force    -0.52122E+02  -0.23196E+01   0.23205E+01   0.52226E+02
0.23205E+01  -0.52122E+02
 Thrust force       0.12872E+03   0.23886E+02  -0.23848E+02   0.13307E+03  -
0.23848E+02   0.12872E+03
 Total force        0.76598E+02   0.21566E+02  -0.21527E+02   0.82437E+02  -
0.21527E+02   0.76598E+02
```

This information is printed at the end of the `cfl3d.prout` file. ***NOTE: the following definitions are used in the output (∞ represents the reference values - see note on p. 3):

mass flow rate through specified area: $\dot{m} = \dfrac{\tilde{\dot{m}}}{\tilde{\rho}_\infty |\tilde{\mathbf{V}}|_\infty \tilde{L}_R^2}$ ($\tilde{\dot{m}}$ in slug/s or kg/s, etc.)

distance to 1st gridpoint above wall: $y^+ = \tilde{n}\sqrt{\tilde{\tau}_w/\tilde{\rho}_w}(\tilde{\rho}_w/\tilde{\mu}_w)$ ref. White[45], eq. 7-123

pressure coefficient: $c_p = \dfrac{2(\tilde{p}_w - \tilde{p}_\infty)}{\tilde{\rho}_\infty |\tilde{\mathbf{V}}|_\infty^2}$ ref. White, eq. 1-3

skin friction coefficient: $c_f = \dfrac{2\tilde{\tau}_w}{\tilde{\rho}_\infty |\tilde{\mathbf{V}}|_\infty^2} = \dfrac{2\tilde{\mu}_w(\partial|\tilde{\mathbf{V}}|/\partial\tilde{n})_w}{\tilde{\rho}_\infty |\tilde{\mathbf{V}}|_\infty^2}$ ref. White, eq. 3-181

heat transfer coefficient: $c_h = \dfrac{\tilde{k}(\partial\tilde{T}/\partial\tilde{n})_w}{\tilde{\rho}_\infty |\tilde{\mathbf{V}}|_\infty \tilde{C}_p(\tilde{T}_w - \tilde{T}_{t,\infty})}$ ref. White, eq. 7-138

c_h + indicates heat flux *toward* wall

Currently, c_f is given as positive when u (the x-component of velocity) at the first cell center above the wall is positive; c_f is negative when u is negative. In the definition of c_h, the Prandtl number $Pr \equiv \tilde{\mu}\tilde{C}_p/\tilde{k}$ is taken to be constant = 0.72. (Also note that in the definition of c_h a reference total temperature $\tilde{T}_{t,\infty}$ is used in CFL3D rather than a local wall total temperature $\tilde{T}_{t,w}$ as defined in White[45].)

cfl3d.out20 (unit 20)

The `cfl3d.out20` file is an auxiliary output file providing information when **iunst** > 0. It provides the unsteady pressures from forced oscillations. This file is primarily used by the programmer for debugging purposes. To output this file, hard-wire `irite = 1` in subroutine `resp` in `rhs.f`.

cfl3d.subit_res (unit 23)

The `cfl3d.subit_res` file is similar to the `cfl3d.res` file, except that it outputs residuals for all sub-iterations when time-accurate computations with sub-iterations are performed. This file is hard-wired with this name and is *not* in the file list at the top of the input file. However, whereas unit 12 outputs only $R(\mathbf{q}^m)$ for the equation for density (in Equation (B-12) or Equation (B-22) of Appendix B), unit 23 outputs the entire right-hand side of Equation (B-12) or Equation (B-22) for the equation for density. Therefore, the residuals in the two files do not match.

cfl3d.subit_turres (unit 24)

The `cfl3d.subit_turres` file is similar to the `cfl3d.turres` file, except that it outputs field-equation turbulence model residuals for all sub-iterations when time-accurate cases with sub-iterations are run. This file is hard-wired with this name and is *not* in the file list at the top of the input file.

cfl3d.dynamic_patch (unit 25)

The `cfl3d.dynamic_patch` file is output when using the moving grid option with dynamic patching. This file is hard-wired with this name and is *not* in the file list at the top of the input file.

5.2.3 PLOT3D Function Files

If desired, CFL3D can output PLOT3D function files (for 3-d cases) with surface grid, y^+, eddy viscosity, and normal spacing at the first grid point above all viscous walls. These files are:

surf_xyz.fmt (unit 28)

surf_y+.fmt (unit 29)

surf_vist.fmt (unit 30)

surf_dn.fmt (unit 31)

For 2-d cases, a single file is output:

surf_y+_2d.fmt (unit 28)

This file is in column format, containing all of the above information, with headers between different segments. By default, this option to write these files is currently turned *off*. To activate this option, the user must change `ifunc` from 0 to 1 in subroutine `yplusout` in `lbcx.f`.