FLITE SYSTEM

Version 4

User Manual

O. Hassan , **K. Morgan** and **N.P. Weatherill**

Swansea University

Singleton Park

Swansea SA2 8PP, U. K.

June 9, 2008

Contents

Chapter 1

Mesh Generation Modules

This manual describes the modules of the **FLITE** system, and the associated data files, dealing with the tasks of discretizing the three-dimensional computational domain into a mesh of tetrahedral elements: these are the modules ST and VT. A sketch of the organization of the mesh generation modules and data files of the **FLITE** system is depicted in Figure 1.1[1]

1.1 Module ST

The module ST is the surface triangulator. Its function is to produce a triangulation of the boundary of the computational domain. The input data required is the geometrical description of the boundary and the spatial distribution of the mesh parameters through the computational domain. This triangulation will constitute the initial data for the 3-D mesh generation process carried out by the module VT.

During a run, the module writes all the geometrical information being generated into an unformatted re-start file (`case.RST`). This file is updated following the successful discretization of each curve and surface component. The re-start file is automatically deleted upon completion of the surface triangulation procedure. Whenever the run is interrupted, the program, if required, can recover this information in the next run by reading this re-start file. This file is not described in this manual.

[1]The files enclosed in dashed boxes are created by the modules but not described in this manual.

User-generated files **FLITE** system modules **FLITE**-generated files

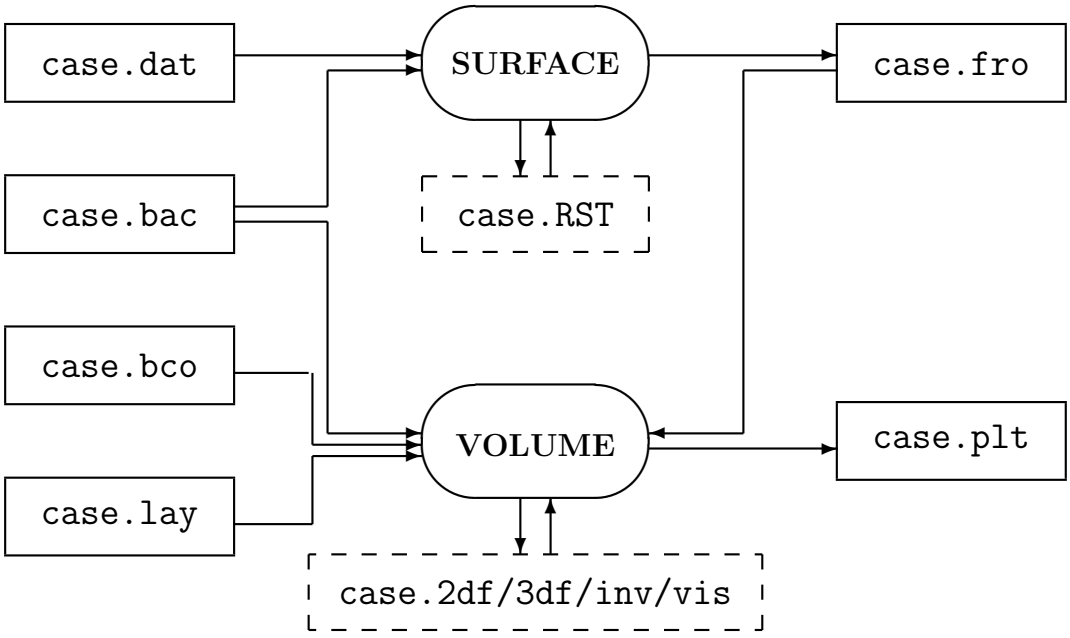


Figure 1.1: Mesh generation modules of the **FLITE** system.

1.1.1 Input data

The information required by the module *ST* is contained in two files: the *geometry definition* file (`case.dat`) and the *background mesh* file (`case.bac`).

Geometry definition file: `case.dat`

This file contains the geometrical description of the boundary of the three-dimensional computational domain. This consists of:

- *curve components*: which are represented by means of an ordered set of nu points. The curve component is the curvature continuous composite cubic spline which is interpolated through these points.
- *surface components*: which are represented by means of a rectangular network of points. In this network, it is possible to define two parametric coordinates (v, w). The surface will have nv points in the v-direction and nw points in the w-direction. The first nv points in the sequence form the first curve in the v-direction (the curve $w = 0$), the second group of nv points forms the curve $w = 1$, etc., up to nw groups, with the last group representing the curve $w = nw - 1$. The normal to the surface is defined as the vector product of the tangent vectors in the v-direction and the w-direction at the point of the surface.
- *surface region connectivity*: The region to be triangulated on a surface component is defined by a closed loop of oriented curve components.

The file `case.dat` is FORMATTED and the information about curve and surface components contained in the file can be displayed, on a computer running the graphic module **PSUE**.

Figure 1.2 presents a synoptic description of the geometry mesh file.

The data is read in *free format* in the following manner[2]:

Here we have followed the default FORTRAN convention for defining the variable types. Variables starting with the letters `[A-H,O-Z]` are `real` and the those with the letters `[I-N]` are `integer`. Variables of type `character` are used in the formatted files of the **FLITE** system in order to facilitate the identification of the different sets of variables. They always appear as a line of text that is read by a statement such as "`read(inp,*)`". A synoptic description of the geometry definition data file is displayed in Figure 1.2. In these synoptic

[2]This code illustrates the way the data is read. It is not the actual code used in the modules.

descriptions of the **FLITE** system files, lines of text in the data appear as "`Line of Text`". Variables of the `logical` type will be explicitly indentified.

```
c
c *** opens geometry definition data file
c
      open(inp,file='case.dat',form='formatted')
c
c *** reads number of segments and surfaces
c
      read(inp,*)
      read(inp,*) ncv,nsf
c
c *** curve components
c
      read(inp,*)
      do 200 is=1,ncv
      read(inp,*) js,itcv
      read(inp,*) nu
      do 100 ip=1,nu
      read(inp,*) (xcv(i),i=1,3)
  100 continue
  200 continue
c
c *** surface components
c
      read(inp,*)
      do 500 is=1,nsf
      read(inp,*) js,itsf
      read(inp,*) nv,nw
      do 300 ip=1,nv*nw
      read(inp,*) (xsf(i),i=1,3)
  300 continue
c
c *** region connectivity information
c
      read(inp,*)
      read(inp,*) ncurve, nregion
      read(inp,*)
      do 600 is=1,ncurve
      read(inp,*) js,ls,nn
  600 continue
      read(inp,*)
      do 700 is=1,nregion
      read(inp,*) ireg,isf,nn,idm
      read(inp,*) lcv
      read(inp,*) (icv(i),i=1,lcv)
```

```
  700 continue
c
c *** closes geometry definition data file
c
      close(inp)
```

Background mesh file: `case.bac`

This file contains the spatial distribution of mesh parameters specified by means of:

- *background mesh*: a mesh of tetrahedral elements in which the mesh parameters are specified as nodal values.
- *distribution of sources*: the spacing—equal in all directions— at a point is defined as an exponential function of the distance to point, line and triangle sources.

Figure 1.3 presents a synoptic description of the background mesh file.

The data is read in *free format* in the following manner:

```
c
c *** opens background mesh data file
c
      open(inp,file='case.bac',form='formatted')
c
c *** reads the background mesh
c
      read(inp,*)
      read(inp,*) npbg,nebg,n1,n2,n3
c
c *** nodal coordinates and mesh parameters
c
      do 100 ip=1,npbg
      read(inp,*) jp,(xbg(i),i=1,3),(dbg(i),i=1,12)
  100 continue
c
c *** element connectivities
c
      do 200 ie = 1,nebg
      read(inp,*) je,(kel(i),i=1,4)
  200 continue
c
c *** point sources
```

`Line of Text`

$ncv \quad nsf$

`Line of Text`

$$(icv)\begin{cases} icv\ itcv \\ nu \\ \quad x_1^{(1)}\ x_2^{(1)}\ x_3^{(1)} \\ \quad \vdots \\ \quad x_1^{(i)}\ x_2^{(i)}\ x_3^{(i)} \\ \quad \vdots \\ x_1^{(nu)}\ x_2^{(nu)}\ x_3^{(nu)} \end{cases} icv = 1, \ldots, ncv$$

`Line of Text`

$$(isf)\begin{cases} isf\ itsf \\ nv\ nw \\ \quad x_1^{(1)}\ x_2^{(1)}\ x_3^{(1)} \\ \quad \vdots \\ \quad x_1^{(nv)}\ x_2^{(nv)}\ x_3^{(nv)} \\ x_1^{(nv+1)}\ x_2^{(nv+1)}\ x_3^{(nv+1)} \\ \quad \vdots \\ x_1^{(nv*nw)}\ x_2^{(nv*nw)}\ x_3^{(nv*nw)} \end{cases} isf = 1, \ldots, nsf$$

`Line of Text`

$ncr \quad nrg$

`Line of Text`

$$(icv) jcv\ itcv \Big\} icv = 1, \ldots, ncr$$

`Line of Text`

$$(irg)\begin{cases} isf\ nn\ idm \\ lcv \\ kv^{(1)}\ \ldots\ kv^{(i)}\ \ldots\ kv^{(lcv)} \end{cases} irg = 1, \ldots, nrg$$

Geometry Definition

- ncv: No of curve components
- nsf: No of surface components

○ **Curve Components**

- icv: Curve component number
- $itcv$: Curve type (always 1)
- nu: No of points defining the curve
- $x_1^{(i)}$, $x_2^{(i)}$, $x_3^{(i)}$: Coordinates of the points defining the curve component

○ **Surface Components**

- isf: Surface component number
- $itsf$: Surface type ($+/-2$). The $-$ sign changes the original orientation of the surface
- nv: No of points in the v-direction
- nw: No of points in the w-direction
- $x_1^{(i)}$, $x_2^{(i)}$, $x_3^{(i)}$: Coordinates of the points defining the surface component

Regions Definition

- ncr: Total No of curves
- nrg: Total No of regions

○ **Curve information**

- icv: Curve component number
- jcv: Curve support number
- $itcv$: Curve type, always (1)

○ **Surface Region Connectivity**

- irg: Surface component number
- isf: Surface region number
- nn: Surface type always (1)
- idm: The domain number which contains the surface, if negative then quadrilateral elements will be generated on that surface
- lcv: No of curve components forming the boundary of the region irg
- $kv^{(i)}$: Curve component number

Figure 1.2: The geometry definition file `case.dat`

```
c
      read(inp,*)
      do 300 ip=1,n1
      read(inp,*)
      read(inp,*) (ab1(i),i=1,6)
  300 continue
c
c *** line sources
c
      read(inp,*)
      do 400 ip=1,n2
      read(inp,*)
      read(inp,*) (ab1(i),i=1,6)
      read(inp,*) (ab2(i),i=1,6)
  400 continue
c
c *** triangle sources
c
      read(inp,*)
      do 500 ip=1,n3
      read(inp,*)
      read(inp,*) (ab1(i),i=1,6)
      read(inp,*) (ab2(i),i=1,6)
      read(inp,*) (ab3(i),i=1,6)
  500 continue
c
c *** closes background mesh data file
c
      close(inp)
```

1.1.2 Running the module

The notation used for describing the interactive input required from the user is as follows:

- Messages *prompted by the modules* are written in `typewriter` font.
- Input data *supplied by the user* is displayed in *slanted* font.

For instance, the module *ST* outputs prompts of the form[3]:

[3]Some of the messages appearing on the screen when running the modules are not included and the format of the prompts may differ slightly from the ones described in this manual

`Line of Text`

$np \quad ne \quad nps \quad nls \quad nts$

$$\left.\begin{array}{cccc} & \vdots & & \\ jp \quad x_1 & x_2 & x_3 & \\ \alpha_{11} & \alpha_{12} & \alpha_{13} & \delta_1 \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \delta_2 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \delta_3 \\ & \vdots & & \end{array}\right\} ip = 1, \ldots, np$$

$$\left.\begin{array}{c} \vdots \\ je \quad kp1 \quad kp2 \quad kp3 \quad kp4 \\ \vdots \end{array}\right\} ie = 1, \ldots, ne$$

`Line of Text`
`Line of Text`

$$(ips)\left\{\begin{array}{l} \vdots \\ \texttt{Line of Text} \\ x_1\ x_2\ x_3\ \delta\ S_c\ D \\ \vdots \end{array}\right\} ips = 1, \ldots, nps$$

`Line of Text`

$$(ils)\left\{\begin{array}{l} \vdots \\ \texttt{Line of Text} \\ x_1^{(1)}\ x_2^{(1)}\ x_3^{(1)}\ \delta^{(1)}\ S_c^{(1)}\ D^{(1)} \\ x_1^{(2)}\ x_2^{(2)}\ x_3^{(2)}\ \delta^{(2)}\ S_c^{(2)}\ D^{(2)} \\ \vdots \end{array}\right\} ils = 1, \ldots, nls$$

`Line of Text`

$$(its)\left\{\begin{array}{l} \vdots \\ \texttt{Line of Text} \\ x_1^{(1)}\ x_2^{(1)}\ x_3^{(1)}\ \delta^{(1)}\ S_c^{(1)}\ D^{(1)} \\ x_1^{(2)}\ x_2^{(2)}\ x_3^{(2)}\ \delta^{(2)}\ S_c^{(2)}\ D^{(2)} \\ x_1^{(3)}\ x_2^{(3)}\ x_3^{(3)}\ \delta^{(3)}\ S_c^{(3)}\ D^{(3)} \\ \vdots \end{array}\right\} its = 1, \ldots, nts$$

Background Mesh

- np: No of background mesh nodes
- ne: No of background mesh elements
- nps: No of point sources
- nls: No of line sources
- nts: No of triangle sources

○ **Nodal Values**

- jp: node number ($= ip$)
- x_1, x_2, x_3: 3-D coordinates of the node
- α_{ij}; $j = 1, 2, 3$: cartesian components of the vector $\vec{\alpha}_i$
- δ_i; $i = 1, 2, 3$: spacing in the direction $\vec{\alpha}_i$

○ **Element Connectivities**

- je: Tetrahedral element number ($= ie$)
- $kp1$, $kp2$, $kp3$, $kp4$: No of the nodes forming the tetrahedral element je

Source Distribution

○ **Point sources**

- ips: Point source number
- x_1, x_2, x_3: Cartesian coordinates of the point source
- δ: Spacing at the point
- S_c: Radius of the sphere where spacing is constant ($= \delta$)
- D: Distance from the source at which the spacing is $2 \times \delta$

○ **Triangle sources**

- ils: Line source number

The rest of the notation is the same as in the previous set. Now the indexes (1) and (2) refer to the first and second points defining the line source

○ **Line sources**

- its: Triangle source number

Same notation. Now the indexes (1), (2) and (3) refer to the first, second and third points which define the triangle source

Figure 1.3: The background mesh file `case.bac`

```
Enter problem name: case
Restarting ? (y/n): n
Estimate Number of generated elements ? (y/n): n
Smoothing type 0,1,2 - Default(1) : 1
```

If the answer to the second prompt is "*y*"[4] the module will read the file `case.RST` containing restart information generated in a previous run of the module.

If the answer to the third prompt is "*y*" the module will estimate the number of expected elements on every curve and every surface. At the end of this process the following prompt will appear:

```
do you want to fix number of elements ? (y/n): y
```

if the answer to the last prompt is "*y*" the module will ask for the number of desired elements to be generated. The module will use the enter number to scale the spacing defined in the background mesh and output the file `case_n.bac` which contains the new spacing. This file should be used in instead of `case.bac` when running the volume generator module *VT*

Smoothing in the parametric plane is the default option, however, smoothing type 2 perform the smoothing on the physical surface.

Remark: It is important to employ the *same* input files `case.dat` and `case.bac` when re-starting the module to ensure compatibility with the file `case.RST` because the module *does not* check this.

During execution, the code will write, in the standard output, information about the current status of the generation process in the curve and surface components: points generated in a curve, nodes and elements generated in a surface, etc.

In the absence of any errors, the module *ST* will produce a triangulation of the surface in the form of the FORMATTED file `case.fro`. The file contents are described in the next section.

1.2 Module *VT*

The module *VT* is the 3-D volume mesh generator. It performs the discretization of the computational domain into a tetrahedral mesh. The input data for

[4] To be more precise, the answer to the prompt is to input "*y*" followed by `Enter` or `RET`.

this module is the surface triangulation of the boundary created by the module *ST* and the *same* spatial distribution of mesh parameters as used by that module. The output data is the hybrid or tetrahedral mesh file `case.plt`.

1.2.1 Input data

The information required by the module *VT* is contained in four files: the *background mesh* file (`case.bac`), the *surface triangulation* file (`case.fro`), the *boundary condition* file (`case.gco`), *boundary layer* file (`case.lay`). The contents of the file `case.bac` have already been described in the previous section.

Surface triangulation file: `case.fro`

This data file is read by the modules *VT_3* . Its contents are:

- *triangular mesh*: the cartesian coordinates of the nodes and the element connectivites describing the triangular surface mesh in the three-dimensional space.
- *local coordinates*: the parametric coordinate u of the nodes generated in the curve components, and the parametric coordinates (v, w) of the nodes generated in the surface components.
- *interpolated geometry*: the coefficients defining the interpolated composite cubic curve and surface components.

Figure 1.5 presents a synoptic description of the surface triangulation file.

The data is read in *free format* in the following manner:

```
c
c *** opens surface triangulation data file
c
      open(inp,file='case.fro',form='formatted')
c
c *** reads the triangulation definition
c
      read(inp,*) ne,np,idum,idum,ncv,nsf,nshet,nwire
c
c *** coordinates of the nodes
c
      do 100 ip=1,npst
      read(inp,*) jp,(xst(i),i=1,3)
```

```
  100 continue
c
c *** connectivities of the triangular faces
c
      do 200 ie=1,nest
      read(inp,*) je,(kst(i),i=1,4)
  200 continue
c
c *** parametric coordinate u of nodes in the curves
c
      do 300 ic=1,ncv
      read(inp,*) jc,npcv
      read(inp,*) (kncv(i),xu(i),i=1,npcv)
  300 continue
c
c *** parametric coordinates (v,w) of nodes in the surfaces
c
      do 400 is=1,nsf
      read(inp,20) js,npsf
      read(inp,70) (knsf(i),xv(i),xw(i),i=1,npsf)
  400 continue
c
c *** reads curve geometry definition coefficients
c
      do 600 is=1,ncv
      read(inp,*) js,nu
      do 500 ip=1,nu
      read(inp,*) (rcv(i),i=1,6)
  500 continue
  600 continue
c
c *** reads surface geometry definition coefficients
c
      do 800 is = 1,nsf
      read(inp,*) js,nv,nw
      do 700 ip=1,nv*nw
      read(inp,*) (rsf(i),i=1,12)
  700 continue
  800 continue
c
c *** closes surface triangulation file
c
      close(inp)
```

$$np \quad ne \quad id1 \quad id2 \quad ncv \quad nsf \quad nshet \quad nwire$$

Surface Triangulation

- np: No of surface mesh nodes
- ne: No of triangular elements
- $id1$, $id2$: Two "dummy" numbers
- ncv: No of curve components
- nsf: No of surface components
- $nshet$: No of sheet surfaces
- $nwire$: No of wire segments

$$\left.\begin{array}{c} \vdots \\ jp\ x_1\ x_2\ x_3 \\ \vdots \end{array}\right\} ip = 1, \ldots, np$$

○ **Nodal Values**

- jp: Node number ($= ip$)
- x_1, x_2, x_3: 3-D coordinates of the node

$$\left.\begin{array}{ccccc} & & \vdots & & \\ je & kp1 & kp2 & kp3 & ksf \\ & & \vdots & & \end{array}\right\} ie = 1, \ldots, ne$$

○ **Element Connectivities**

- je: Triangular element number ($= ie$)
- $kp1$, $kp2$, $kp3$: No of the nodes forming the element je
- ksf: No of the surface component to which the element belongs

Local Coordinates

○ **Curve Components**

$$(icv)\left\{\begin{array}{cc} \vdots & \\ npc & \\ knc^{(1)} & u^{(1)} \\ \vdots & \\ knc^{(i)} & u^{(i)} \\ \vdots & \\ knc^{(npc)} & u^{(npc)} \\ \vdots & \end{array}\right\} icv = 1, \ldots, ncv$$

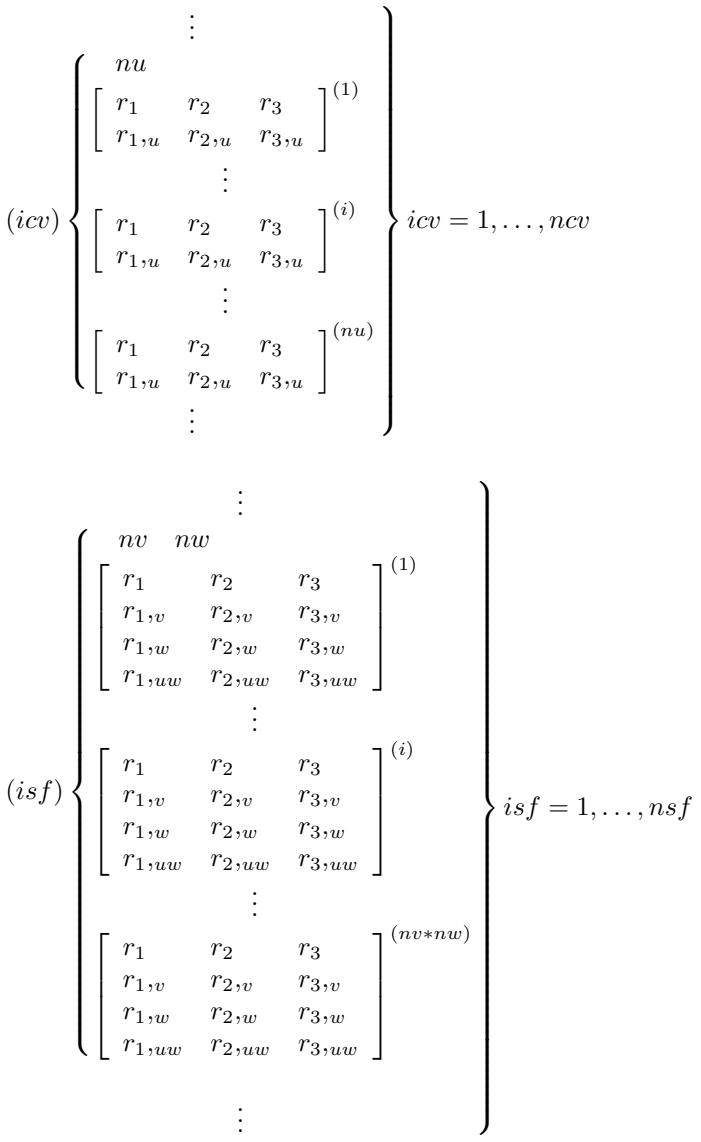
- icv: Curve component number
- npc: No of nodes in the mesh generated along the curve component icv
- $knc^{(i)}$: Global number in the mesh of the i-th generated node in this curve
- $u^{(i)}$: Parametric coordinate of the i-th generated node in this curve

○ **Surface Components**

$$(isf)\left\{\begin{array}{ccc} \vdots & & \\ nps & & \\ kns^{(1)} & v^{(1)} & w^{(1)} \\ & \vdots & \\ kns^{(i)} & v^{(i)} & w^{(i)} \\ & \vdots & \\ kns^{(nps)} & v^{(nps)} & w^{(nps)} \\ \vdots & & \end{array}\right\} isf = 1, \ldots, nsf$$

- isf: Surface component number
- nps: No of nodes in the mesh generated on the surface component isf
- $kns^{(i)}$: Global number in the mesh of the i-th generated node in this surface
- $v^{(i)}$, $w^{(i)}$: Coordinates in the parameter plane of the i-th generated node in this surface

Figure 1.4: The surface triangulation file `case.fro`



Interpolated Geometry

○ Curve Components

- icv: Curve component number
- nu: No of points defining the curve
- $[r_1\ r_2\ r_3]^{(i)}$: 3-D cartesian coordinates of the points i; $i = 1, \ldots, nu$ defining the curve component
- $[r_{1,u}\ r_{2,u}\ r_{3,u}]^{(i)}$: Components of the tangent vector at point i in the interpolated composite cubic spline defining the curve component

○ Surface Components

- isf: Curve component number
- nv: No of points in the v-direction
- nw: No of points in the w-direction
- $[r_1\ r_2\ r_3]^{(i)}$: 3-D cartesian coordinates of the points defining the surface component
- $[r_{1,v}\ r_{2,v}\ r_{3,v}]^{(i)}$: Components of the tangent vector in the v-direction at point i
- $[r_{1,w}\ r_{2,w}\ r_{3,w}]^{(i)}$: Components of the tangent vector in the w-direction at point i
- $[r_{1,vw}\ r_{2,vw}\ r_{3,vw}]^{(i)}$: Components of the twist vector at point i

Figure 1.5: The surface triangulation file `case.fro` (cont.)

Boundary condition file: `case.gco`

This file contains the flags associated with the boundary conditions to be used by the viscous volume generator. For each surface one marker is required. The flag is equal to 1 if the boundary layers are to be generated orthogonal to the surface, 2 if the surface mesh is to be re–generated due to the generation of viscous layers on other surfaces and 3 for unchanged surfaces. If a quadrilateral mesh was generated on the surface and hexahedral elements are to be generated in the boundary layers, then a flag of -1 should be used.

Similarly, for each intersection segment, two flags are used. The first flag indicates if the segment is a trailing edge. While the second flag identifies the segments on which smoothing of the normals during the generation, is to be restricted. However, if the surface file is not generated using the Flite surface generator, the intersection curve information can be omitted.

Figure 2.2 presents a synoptic description of the boundary condition file.

The data is read in *free format* in the following manner:

```
c
c *** opens boundary conditions data file
c
      open(inp,file='case.bco',form='formatted')
c
c *** reads number of surface and curve components
c
      read(inp,*)
      read(inp,*) nsf, ncv
c
c *** reads surface boundary condition flags
c
      read(inp,*)
      do 100 isf = 1,nsf
      read(inp,*) jsf,ktsf
  100 continue
c
c *** reads curve boundary condition flags
c
      read(inp,*)
      do 200 icv = 1,ncv
      read(inp,*) jcv,ktcv,kscv
  200 continue
c
c *** closes boundary conditions data file
c
      close(inp)
```

`Line of Text`

$nsf \quad ncv$

`Line of Text`

$knsf^{(1)} \quad ktsf^{(1)}$

$\vdots$

$knsf^{(i)} \quad ktsf^{(i)}$

$\vdots$

$knsf^{(nsf)} \quad ktsf^{(nsf)}$

`Line of Text`

$kncv^{(1)} \quad ktcv^{(1)} \quad kscv^{(1)}$

$\vdots$

$kncv^{(i)} \quad ktcv^{(i)} \quad kscv^{(i)}$

$\vdots$

$kncv^{(ncv)} \quad ktcv^{(ncv)} \quad kscv^{(ncv)}$

Boundary Condition Flags

- nsf: No of surface components
- ncv: No of curve components

○ **Surface Components**

- $knsf^{(i)}$: Surface component number ($= i$)
- $ktsf^{(i)}$: Surface component boundary type:

  1 ..... generate viscous layers
  2 ..... re–generate surface mesh
  3 ..... no change to surface mesh

- $ktsf^{(i)}$: Surface element type:

  \+ ..... triangles
  \- ..... quadrilaterals

○ **Curve Components**

- $kncv^{(i)}$: Curve component number ($= i$)
- $ktsf^{(i)}$: Curve component boundary type:

  0 ..... curve is not on trailing edges
  1 ..... curve is on trailing edges

- $ktsf^{(i)}$: Curve smoothing type:

  0 ..... smooth using points on surfaces
  1 ..... smooth using points on the same curve

Figure 1.6: The boundary conditions file `case.gco`

Boundary layer file: `case.lay`

This file is a user generated file and it contains:

- the number of viscous layers
- the height of each layer

1.2.2 Running the module

When running the module VT, the user is prompted with:

```
Enter problem name :
Enter Type of Mesh: 1-Invisicd , 2-Viscous:
Enter Stage: 1-First start , 2-Resume , 3-Complete: (Only for type 2)
Enter normal computation: 1-Geometry , 2-Triangulation ,
                3-Triangulation+edges from geometry: (Only for type 2)
Enter 1-All Tetrahedral , 2-Hybrid: (Only for type 2)
Enter Alpha:
Enter 1-recover first , 2-recover last:
Enter number of Cosmetics Loops:
Enter Maximum Swapping Angle:
Enter Maximum Collapsing Angle:
Enter Number of Smoothing Loops:
Do you want to check the Surface triangulation? (y/n):
```

If the Flite geometry information is not available, option 2 should be used to compute the normal at each surface point. The coefficient alpha determines the uniformity of the generated elements. A value in the range of 0.8 to 1.2 will produce uniform elements. The recommended value is one. Lower values will generate more points in the domain. In each cosmetics loop three procedures take place; Swapping of edges, collapsing of edges and smoothing of the coordinates. The recommended value for the swapping angle is 30 degrees, above such a value longer execution time will be noticed. For edge collapse, an angle between 10 and 15 degrees should be used, a larger value will result in a reduction in the resolution. Finally 3-5 smoothing loops are normally used in three dimensions. When mesh type Viscous is selected the code will produce two files at the end of the generation of the viscous layers. The file `case.vis` contains the mesh in the viscous region and the file `case.3df` contains the triangulation of the domain which has to be filled with isotropic elements. These two files can be used to resume the generation of the isotropic region if required. The

file `case.inv` is generated at the end of the generation of the isotropic region and can be used with the previous two files to complete the mesh if required. During execution, every time a certain number of elements has been generated the code will write, in the standard output, information concerning the current status of the generation process.

When the generation process is completed the code will write the resulting tetrahedral mesh in the file `case.plt`.

Mesh display file: `case.plt`

This file is UNFORMATTED and contains the geometrical description of the generated mesh and the boundary surface triangulation.

The data is read in the following manner:

```
c
c *** opens mesh display data file
c
      open(inp,file='case_n.plt',form='unformatted')
c
c *** reads the following numbers
c     ne   No of Number of elements
c     np   No of nodes
c     nbf  No of boundary faces
c     ihex No of hexahedral elements
c     ipri No of prismatic elements
c     ipyr No of pyramid elements
c     itet No of tetrahedral elements
c     nfaceqNo of quadrilateral boundary faces
c     nfacetNo of triangular boundary faces
c
c
      read(inp) ne, np, nbf, ihex, ipri, ipyr, itet, nfaceq, nfacet
c
c *** element-to-nodes connectivities
c
      read(inp) ((iel(in,ie),in=1,8),ie=1,ihex)
      read(inp) ((iel(in,ie),in=1,6),ie=ihex+1,ihex+ipri)
      read(inp) ((iel(in,ie),in=1,5),ie=ihex+ipri+1,ihex+ipri+ipyr)
      read(inp) ((iel(in,ie),in=1,4),ie=ihex+ipri+ipry+1,ne )
c
c *** node coordinates
c
      read(inp) ((coor(in,ip),in=1,3),ip=1,np)
c
```

```
c *** boundary surface triangulation
c     iface(1:4,it)  face-to-nodes connectivities
c     iface(5,it)    surface component number for face it
c
      read(inp) ((iface(in,it),in=1,5),it=1,nbf)
c
c *** closes display mesh data file
c
      close(inp)
```

In the case of all tetrahedra mesh, the first 3 records of the connectivity areas are omitted.

Chapter 2

Flow Solution Modules

This chapter describes the modules of the **FLITE** system, which undertake the steady–state flow simulation, together with their associated data files. These modules are: **Gen3d** and **Mgns3d**. A sketch of the organization of the flow solution modules and data files of the **FLITE** system is depicted in Figure 2.1

2.1 Module Gen3d

The module **gen3d** is a flow solver preprocessor. It transforms the element–based description of the hybrid volume mesh into the side–based data structure employed by the module **Mgns3d**. In addition, it processes all the information required for applying the flow boundary conditions on the surface triangulation of the boundary of the computational domain . It also performs the partitioning of the mesh sides and the boundary faces which is required for parallel processing.

The input data required is the three-dimensional volume mesh, generated by the module **VT**, and the description of the boundary conditions which are to be applied to the discretized curve and surface components which make up the computational domain.

2.1.1 Input data

The information required by the module **Gen3d** is contained in three files: the *Hybrid mesh* file (`case.plt`), the *boundary conditions* file (`case.bco`) and the *turbulence triggering information* file (`case.wid`).

User-generated files **FLITE** system modules **FLITE**-generated files

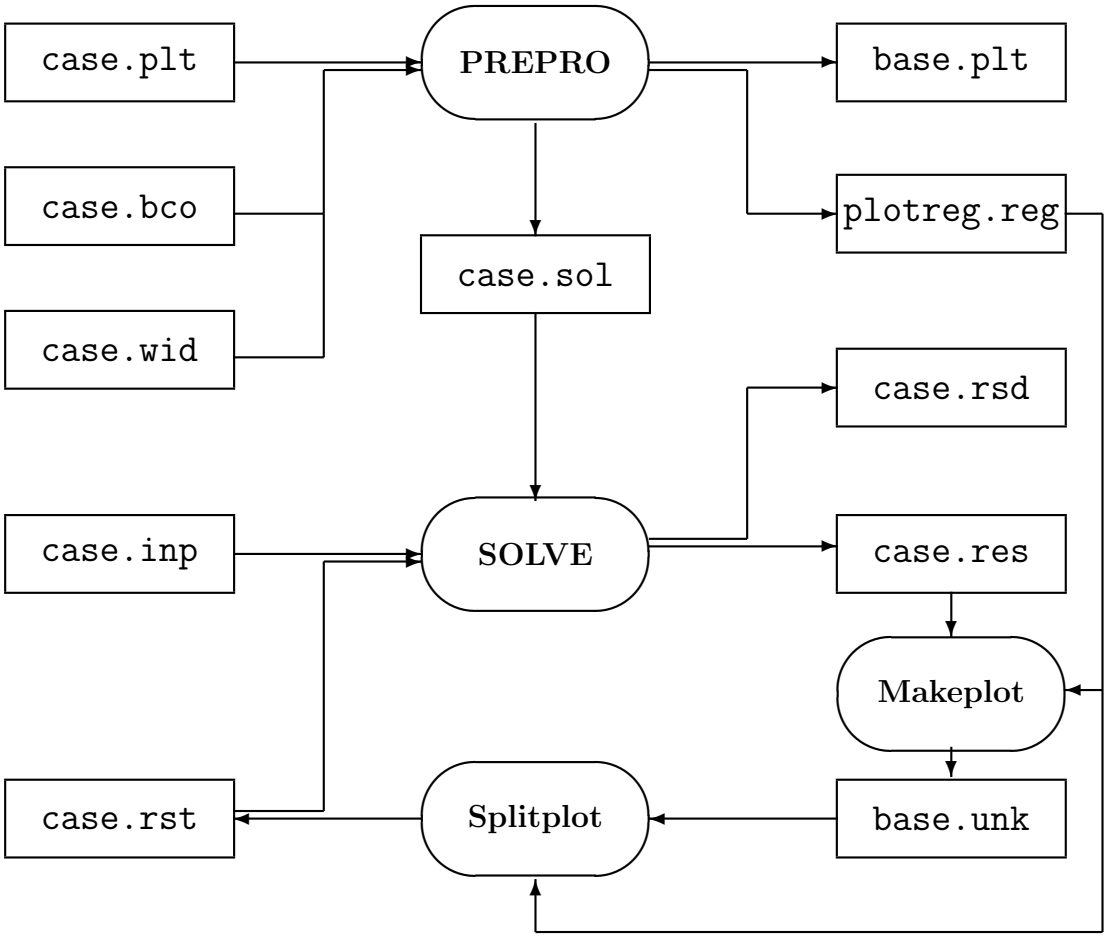


Figure 2.1: Flow solution modules of the **FLITE** system.

The file `case.plt` has been described in the previous chapter.

Boundary conditions file: `case.bco`

This file contains the flags associated with the boundary conditions to be applied by the module **SOLVE** to boundary curve and surface components.

The boundary condition flags employed in the **FLITE** system are the following:

- *surface components*: These define the type of boundary conditions to be applied to a certain surface component. The current implementation allows:

Flag	Surface type
1	wall
2	symmetry
3–4	far field
5–6	engine inlet
7–8	engine outlet

- *curve components*: These identify points in the triangulation, which lie on wall surface components, where the normal to the surface is not defined, such as wing trailing edges. We will refer to these as *singular points*. At these points, no wall boundary correction is applied. The flags employed in the description of the curve types are:

Flag	Singular point ?		
	first node	**interior** nodes	**end** node
0	no	no	no
1	yes	yes	yes
2	yes	no	yes
3	yes	no	no
4	no	no	yes

The data is read in *free format* in the following manner:

```
c
c *** opens boundary conditions data file
c
      open(inp,file='case.bco',form='formatted')
```

```
c
c *** reads number of surface and curve components
c
      read(inp,*)
      read(inp,*) nsf, ncv
c
c *** reads surface boundary condition flags
c
      read(inp,*)
      do 100 isf = 1,nsf
      read(inp,*) jsf,ktsf
  100 continue
c
c *** reads curve boundary condition flags
c
      read(inp,*)
      do 200 icv = 1,ncv
      read(inp,*) jcv,ktcv
  200 continue
c
c *** closes boundary conditions data file
c
      close(inp)
```

Figure 2.2 displays a description of the boundary conditions file.

turbulence triggering information: `case.wid`

This file contains the coordinates of the lines along which the triggering will take place. Also included are the radius from the line where surface point will be considered and the thickness from the wall inside which the distance to the point has to accurately found. The file is a FORMATTED Fortran NAMELIST. Figure 2.3 presents a synoptic description of the file.

`Line of Text`

$nsf \quad ncv$

`Line of Text`

$knsf^{(1)} \quad ktsf^{(1)}$

$\vdots \quad \vdots$

$knsf^{(i)} \quad ktsf^{(i)}$

$\vdots \quad \vdots$

$knsf^{(nsf)} \quad ktsf^{(nsf)}$

`Line of Text`

$kncv^{(1)} \quad ktcv^{(1)}$

$\vdots \quad \vdots$

$kncv^{(i)} \quad ktcv^{(i)}$

$\vdots \quad \vdots$

$kncv^{(ncv)} \quad ktcv^{(ncv)}$

Boundary Condition Flags

- nsf: No of surface components
- ncv: No of curve components

◦ **Surface Components**

- $knsf^{(i)}$: Surface component number ($= i$)
- $ktsf^{(i)}$: Surface component boundary type:

  1 ..... wall
  2 ..... symmetry
  3–4 ..... far field
  5–6 ..... engine inlet
  7–8 ..... engine outlet

◦ **Curve Components**

- $kncv^{(i)}$: Curve component number ($= i$)
- $ktsf^{(i)}$: Curve component boundary type. It depends on the number of singular nodes in the curve:

  0 ..... none is singular
  1 ..... all are singular
  2 ..... first and last are singular
  3 ..... only first is singular
  4 ..... only last is singular

Figure 2.2: The boundary conditions file `case.bco`

```
&control
 numberOftripLine = 0
 tripLineCoordinates(:, 1) = 0.0
 tripLineCoordinates(:, 2) = 0.0
 tripLineCoordinates(:, 3) = 0.0
 tripLineCoordinates(:, 4) = 0.0
 tripLineCoordinates(:, 5) = 0.0
 tripLineCoordinates(:, 6) = 0.0
 tripRadius = 1.0
 wallDistanceThickness = 0
&
```

Triggering Parameters

- `numberOfTripLine`: Total number of lines along which tripping will take place
- `tripLineCoordinates(:,1:3` : The coordinates of the starting point of the each line
- `tripLineCoordinates(:,4:6` : The coordinates of the end point of the each line
- `tripRadius`: Triggering will be evaluated for surface points which fall within this radius from the line

Figure 2.3: The preprocessor triggering file `case.wid`

2.1.2 Running the module

The module **Gen3d** outputs a prompts of the form:

```
Enter problem name: case
1:  Inviscid , 2:  Viscous:
Number of grids to generate:
Is mesh hybrid?  (T/F):
Number of parallel domains:
```

The module then runs without any user intervention. The module will produce a file for each processor which contains the side–based data structure, the communication information and the boundary conditions required by the module **Mgns3d** in the form of the UNFORMATTED file `case.sol_N`, where **N** is the processor number.

The module also produces the UNFORMATTED file `base.plt` that combines, in a single file, the information required in the visualization module **PSUE**. The module also output the formatted file`plotreg.reg` which contains the renumbering information between the global mesh and the partitioned mesh. This file is required to recombine the the multiple results file into one file suitable the visualization module **PSUE** using the module **Makeplot**. This file is also required to split the single result file into **N** partitioned restart files using the module **Splitplot**

2.2 Module Mgns3d

The module **Mgns3d** is the unstructured Navier–Stokes flow solver. It performs the numerical computation of steady–state solutions of the transient form of the Navier–Stokes equations of compressible viscous flow. The input data for this module is the solver data produced by the module **Gen3d** and the user–specified flow conditions, such as free–stream Mach number, angle of attack, etc., and the algorithmic options, such as number of timesteps, dissipation coefficients, CFL number, etc.

2.2.1 Input data

The information required by the module **Mgns3d** is contained in two files: the *solver data* file (`case.sol`) and the *solver control* file (`case.inp`).

Solver data file: `case.sol`

This data file is read by the module **Mgns3d**. Its contents are:

- *side–based data structure*: the coordinates of the nodes in the mesh, the side–to–nodes and the boundary edge–to–nodes connectivities, and the weights for sides and boundary edges.
- *boundary conditions*: the boundary condition flags for boundary edges and boundary nodes, the values of the surface normal at the boundary nodes.
- *multigrid data*: the nodes and weights required to perform the restriction operator and the prolongation operator.
- *Communication data*: the list of nodes to be communicated between this processor and all the other processor.

Solver control file: `case.inp`

This data file is read by the module **Mgns3d**. It contains a set of flow conditions and algorithmic constants for the flow solver. The flow solver contains built–in defaults that are overwritten by the values specified in `case.inp`. This file is not, therefore, required to provide the values of all the constants, but only those which differ from the defaults.

The file `case.inp` is a FORMATTED Fortran NAMELIST. Figure 2.4 presents a synoptic description of the control file. The name and type of the variables specified in this description are the *actual* those employed in the code and the values assigned to the variables are the default values used in the code.

&control

numberOfMGIterations = 100
numberOfSGIterations = 0
numberOfCFLIncrements = 0
writeToFileInterval = 25
MachNumber = 0.5
alpha = 0.0
beta = 0.0
CFLNumber = 1.0
ReynoldsNumber = $1.0E06$
viscosityScheme = 1
inflowField(1, 1) = 1.0
inflowField(2, 1) = 1.0
inflowField(3, 1) = 0.0
inflowField(4, 1) = 0.0
inflowField(5, 1) = 1.0
inflowField(:, 2) = 0.0
inflowTemperature = 528.0
wallsAreIsentropic = $.true.$
wallTemperature = 528.0
enginesFrontMassFlow(2) = 0.0
enginesRearParameters(:, 2) = 0.0
engineBCRelaxation = 1.0
numberOfEORelaxationSteps = 0.0
dissipationScheme = 2
coarseGridDissipationScheme = 2
secondOrderDissipationFactor = 0.4
fourthOrderDissipationFactor = 0.2
coarseGridDissipationFactor = 0.75
HartensCorrectionFactor = 0.0
useMatrixDissipation = $.false.$

&

Solver Parameters

- `numberOfMGIterations`: Total number of multigrid cycle. Negative number indicates the order of convergence required.
- `numberOfSGIterations`: Total number of single grid iteration before the start of the multigrid
- `numberOfCFLIncrements`: Number of cycles to increase the CFLnumber from zero to the specified value
- `writeToFileInterval`: No of cycles for writting the unknowns file `case.res`
- `MachNumber`: Free-stream Mach number
- `alpha`: Free-stream angle of attack
- `beta(1:2)`: Free-stream yaw angle
- `CFLNumber`: Value of the CFL number
- `ReynoldsNumber`: The Reynolds number, zero value indicate inviscid calculation
- `viscosityScheme`: 1 FV calculation, 2- compact stencil calculation
- `inflowField`: The value of the unknown at the inflow
- `inflowTemperature`: The prescribed temperature at the far field
- `wallsAreIsentropic`: .false. value indicates isothermal wall
- `wallTemperature`: The prescribed temperature at the wall for Isothermal wall
- `enginesFrontMassFlow`: Total mass flow through an engine inlet
- `enginesRearParameters(1:6,1:2)`: Values of density, three components of the velocity, energy and pressure at the exit of an engine outlet
- `engineBCRelaxation`: Value less than 1 will result in the application of the engine boundary condition gradually
- `numberOfEORelaxationSteps`: number of cycle used to introduce the engine outflow conditions
- `dissipationScheme`: 1- Linearly preserving dissipation, 2- JST dissipation, 3- First order dissipation
- `coarseGridDissipationScheme`: 1- Linearly preserving dissipation, 2- JST dissipation, 3- First order dissipation
- `secondOrderDissipationFactor`: Second-order differences dissipation coefficient
- `fourthOrderDissipationFactor`: fourth-order differences dissipation coefficient
- `coarseGridDissipationFactor`: Second-order differences dissipation coefficient for the coarse grids
- `coarseGridDissipationFactor`: Second-order differences dissipation coefficient for the coarse grids
- `HartensCorrectionFactor`: Coefficient of Harten's eigenvalue correction for Roe's matrix

Figure 2.4: The solver control file `case.inp`

residualSmoothingFactor = 0.0
multigridScheme = 3
numberOfRelaxationSteps = 1
numberOfGridsToUse = 1
numberOfRSSteps = 0
numberOfPSSteps = 0
multigridBCRelaxation = 1.0
turbulenceModel = 0
numberOfTurbulenceGrids = 1
turbulentCFLNumber = 1.0
tripFactor = 1.0
triggerRadius = 9999999.9
turbulencetriggerValue = 0
referenceArea = 1.0
useTimeResidual = $.false.$
numberOfProcesses = 1
dataDirectory = $'..'$

&end

Solver Parameters

- `residualSmoothingFactor`: Residual smoothing coefficient
- `multigridScheme`: 1- V cycle, 2- W cycle, 3- growing V, 4- decaying V, 5- weighted W cycle
- `numberOfRelaxationSteps`: Number of iteration on the finest grid in the multigrid cycle
- `numberOfGridsToUse`: Number of grid to employ in the multigrid cycle
- `numberOfRSSteps`: Number of residual smoothing loop to perform on the coarse grids during the restriction operator.
- `numberOfPSSteps`: Number of residual smoothing loop to perform on the coarse grids during the prolongation operator.
- `multigridBCRelaxation`: Value less than 1 will result in the application of the boundary condition gradually on the coarse grids.
- `turbulenceModel`: 0- Laminar, 1- One equation model.
- `numberOfTurbulenceGrids`: number of grid to use for the turbulence model
- `tripFactor`: Factor to apply the tripping gradually.
- `turbulentCFLNumber`: CFL value used to update the tubulence equation.
- `triggerRadius`: factor to control the effect of the triggering.
- `turbulencetriggerValue`: The intial value applied to the triggering term.
- `referenceArea`: Wall area to scale the value of lift and drag.
- `useTimeResidual`: .True. value will output CPU time against residual, .False. value will output number of cycle against residual.
- `numberOfProcesses`: The number of CPU to be used, should be the same as the number of partitions.
- `dataDirectory`: The path to the directory where the data files are stored.

Figure 2.5: The solver control file `case.inp`

2.2.2 Running the module

The module **Mgns3d** prompts the user with:

```
Enter control filename: case.inp
Enter computation filename: case.sol
Enter startup filename: case.rst
Enter result filename: case.res
Enter residual filename: case.rsd
```

After this, the code will write, in the standard output, a list of the specified control parameters and flow conditions and the values of the L–2 norm of the residuals of the conservative variables at each timestep.

At prescribed timestep intervals the module **Mgns3d** will write the file `case.res`, which will contain the current nodal values of the primitive variables (density, velocity and energy).

The file `case.res` can be used with the file `plotreg.reg`, created by the preprocessor module **Gen3d**, to create a single result file, `base.unk` using the module **Makeplot**. This file can be then used with file `base.plt` to postprocess the solution using the visualization module **PSUE**.

The module can start either from free–stream conditions or from a previously computed solution. In the first case, a blank response to the startup filename prompt should be given whereas in the second case the file `case.rst` must exist. The file `case.rst` can be created by re–splitting the combined result file `base.unk` using the module **Splitplot** and the file `plotreg.reg`.

In every run of the module, a file `case.rsd` is created that contains the history of convergence of the current run in terms of the values of the L–2 norm of the residuals. The format of the file is very simple and is not described here. If a long computation is performed by making several runs of the module, and if a record of the full convergence history is required, it is necessary to rename the current `case.rsd`. If this is not done, this file will be overwritten in the next run.