Geompack++ File Formats for Regions and Meshes

Barry Joe
Zhou Computing Services Inc.
Partial address: Abbotsford, BC, Canada
E-mail: bjoe@shaw.ca
Web site: members.shaw.ca/bjoe

Technical Report ZCS2008-01
March 2008

Abstract

Geompack++ is a mesh generation package for performing various meshing operations on 2-D regions and meshes, 3-D regions and meshes, and surface meshes. The operations include generating a mesh given a region, generating a 3-D mesh given a surface mesh, improving a given mesh, and refining a given mesh. The regions and meshes are represented in the file formats described in this document. It is assumed that the reader is knowledgeable about geometric terms and NURBS curves and surfaces. A summary of minor extensions to the preceding Geompack90 file formats plus examples and figures are provided in the appendices.

1 Notation for file formats

There are seven types of file formats: 2-D region, 2-D curve, 2-D mesh, 3-D region, 3-D curve/surface, 3-D mesh, surface mesh. Each file could be in character (ASCII) or binary form. Character files are readable from C++, C, Fortran, and other programming languages. Binary files are readable from C++ and C but not from Fortran. Real numbers are stored as double precision floating point numbers. There is a 2-D curve file associated with each 2-D region or 2-D mesh file. There is a 3-D curve/surface file associated with each 3-D region, 3-D mesh, or surface mesh file. The reason for storing the curve/surface information in a separate file instead of appending it at the end of the region or mesh file is to avoid duplicating the same information in multiple region and mesh files.

Each file consists of a sequence of logical records. In the descriptions given in Sections 2 to 8, the fields of each logical record are put on a single line. For a character file, each logical record may be split into multiple physical records (lines), but must begin on a new line. For a binary file, there are no gaps between records so the file is just a sequence of integers and double precision floating point numbers.

The following notation is used in the descriptions of the file formats. Field names are given in *italic* font. If a field may have alternative names, the names are separated by the `|` symbol. Horizontal and vertical ellipsis are used for the missing fields of an array. In the case of a logical record of knots in the 2-D curve or 3-D curve/surface files, the record is omitted if the first index is greater than the last index. The `[` and `]n` delimiters, where `n` is the name of a field in typewriter font, are used to indicate that the sequence of logical records between the delimiters is repeated n consecutive times (with the same form, but different values for the fields). If n is zero, there are no occurrences of the logical records. The { and } delimiters are used to indicate that the sequence of records between the delimiters is one of several alternatives; the alternative forms of the records are separated by the word `or` in typewriter font. In a repetition group of n logical record sequences, the sequences are implicitly labelled (or indexed) from 1 to n. If x is a field in the ith sequence, then it is referenced as $x[i]$.

2 2-D region file format

---

nvc
[x y $vertinfo$]nvc
nvx
[$nodecode$ $icurv$ $ucurv$]nvx
nloop
[nv $regcode$ $inreg$
v_1 v_2 ... v_{nv}
$edginfo_1$ $edginfo_2$ ... $edginfo_{nv}$]nloop

The 2-D region file format consists of three groups of information: vertex coordinates, vertex extra-info records, and loops. The first logical record contains the positive integer field *nvc*, the number of vertex coordinates. The next *nvc* logical records each contain the x and y coordinates of a vertex and a vertex information field $vertinfo$. x and y are real numbers, and $vertinfo$ is an integer. $vertinfo \leq 0$ indicates that the vertex is not referenced. If $vertinfo > 0$, then $vertinfo = 20 \cdot xtraptr + vertyp$ encodes two fields. $xtraptr = 0$ if not used; else $xtraptr > 0$ is the label (or index) of a vertex extra-info record (see below). In the latter case, $nodecode[xtraptr]$, $icurv[xtraptr]$, and $ucurv[xtraptr]$ are extra fields associated with the vertex. *vertyp* indicates the type of the vertex (or midnode) as follows (some of the terms are defined later):

- 1 if an unconstrained vertex, e.g. in the interior of a subregion or on an unconstrained edge (cannot be on the region boundary or on a constrained curve)
- 2 if a constrained vertex, e.g. an endpoint of a labelled curve or on the interior of a labelled curve or an interior point that is to be a vertex of the mesh
- 3 if a vertex on the interior of a constrained curve (possibly an unlabelled line segment) that is free to move along the curve
- 4 if a vertex on the interior of a piecewise linear approximation of a constrained NURBS curve (so the projection of the vertex to the curve should eventually be done), that is free to move onto or along the curve
- 5 if a vertex on the interior of a constrained surface (possibly an unlabelled planar facet) that is free to move along the surface
- 6 if a vertex on the interior of a polygonal approximation of a constrained NURBS surface (so the projection of the vertex to the surface should eventually be done) that is free to move onto or along the surface
- 8 if a midnode at the midpoint of the straight edge joining corner nodes and the edge does not (topologically) lie on a constrained curve or constrained surface
- 9 if a midnode that is not at the midpoint of the straight edge joining corner nodes, and the curved edge joining the corner nodes is in the interior of the region and does not (topologically) lie on a constrained curve or constrained surface (e.g. the midnode is moved to improve the quality of quadratic elements)
- 10 if a midnode on the interior of a constrained curve (possibly an unlabelled line segment); for a line segment, the midnode is at the midpoint of the straight edge joining the corner nodes

- 11 if a midnode between the straight edge joining the corner nodes and the constrained NURBS curve that the edge topologically lies on (e.g. due to the projection to the curve causing an element with nonpositive shape measure)
- 12 if a midnode at the midpoint of the straight edge joining the corner nodes, and the edge topologically lies on a constrained NURBS curve so a projection to the curve is required (or moved back here due to a nonpositive shape measure)
- 13 if a midnode on the interior of a constrained surface (possibly an unlabelled planar facet); for a planar facet, the midnode is at the midpoint of the straight edge joining the corner nodes
- 14 if a midnode between the straight edge joining the corner nodes and the constrained NURBS surface that the edge topologically lies on (e.g. due to the projection to the surface causing an element with nonpositive shape measure)
- 15 if a midnode at the midpoint of the straight edge joining the corner nodes, and edge topologically lies on a constrained NURBS surface, so a projection to the surface is required (or moved back here due to a nonpositive shape measure)

Since the vertex types are used in all the file formats, they are all listed here. Types 5 and 6 are not used for 2-D regions and meshes. Midnodes are used only for meshes with quadratic elements.

For compatibility with the preceding Geompack90 file formats (see Appendix A), the following two vertex types are kept, but they are treated as equivalent to types 3 and 10, respectively:

- 7 if a vertex on the interior of a constrained interior space geometric line segment
- 16 if a midnode at the midpoint of the straight edge joining the corner nodes, and the edge lies on a constrained interior space geometric line segment

The next logical record contains the nonnegative integer field nvx, the number of vertex extra-info records. If $nvx > 0$, the next nvx logical records each contain the integer fields *nodecode*, *icurv* and real number field *ucurv*. *nodecode* is a node code (if positive), which can be used to specify a load or constraint at a node (or mapping to an "external" vertex label). If not used (and the record is used), *nodecode* is set to 0. *icurv* is the label of a curve (if positive), and $0.0 < ucurv < 1.0$ is the u parameter value of the point on (or topologically on) the curve. If not used, *icurv* is set to ≤ 0 and *ucurv* to outside interval $(0.0, 1.0)$, e.g. -1.0. NURBS curves are defined for $0 \leq u \leq 1$ (see the next section).

The latter two fields are used for all vertices or midnodes that (topologically) lie on the interior of a labelled curve (a curve that is defined in the curve file). For a labelled geometric line segment, the u parameter value is determined by the standard parameterization of the line segment with $0 \leq u \leq 1$. Depending on the *vertyp* value, *ucurv* is either the exact u value or an estimated u value. These three fields are stored separately from the vertex coordinates to save space (most vertices, e.g. interior mesh vertices, will not use these fields). *nodecode* is set to < 0 to indicate that none of the three fields is used (i.e. the record is not referenced by the *xtraptr* part of any $vertinfo$ field); in this case, the other two fields can have arbitrary values.

The next logical record contains the positive integer field *nloop*, the number of loops (boundary of simply-connected subregion or simple hole). The last *nloop* sequences of three logical records contain integer information for each loop. $nv \geq 2$ is the number of vertices in the loop; $nv = 2$ is allowed only if there is at least one curved (nonlinear) edge between the two vertices. If nonnegative, *regcode* is an integer region code associated with the subregion defined by the loop; it is 0 if the loop is the boundary of a hole or there is nothing special about the subregion, or positive to indicate a material type or some other property. *inreg* is 0 if the loop is an outer boundary, > 0 if the loop is the boundary of a hole, or < 0 if the loop is the boundary of a hole interface (where the "hole" is another subregion); so the sign of *inreg* indicates the loop type. The magnitude of *inreg* (if nonzero) is the label of the subregion (actually the boundary loop of the subregion) containing the hole or hole interface. In the case of nested hole interfaces, the label should be for

the smallest subregion containing the hole or hole interface. For the ith loop, $|inreg|$ may be any integer between 1 and *nloop*, except for i.

$v_1, v_2, \ldots, v_{nv}$ are the positive labels of vertices in CCW (counterclockwise) order on the loop, i.e. the subregion or hole is on the left as the loop is traversed. The starting vertex is arbitrary. The edges of the loop are implicitly represented by their two endpoints, i.e. there are (curved) edges joining vertices v_1 and v_2, v_2 and v_3, ..., v_{nv} and $v_{nv+1} = v_1$. A subregion can be simply-connected (holes attached to the outer boundary via cut interfaces are allowed; so are cracks), so the boundary loop may have duplicate vertices and duplicate edges with opposite orientation. A hole or hole interface must be simple so no duplicate vertices may appear in the boundary loop. The space inside a hole can be partially or totally occupied by one or more subregions. For example, a hole interface loop can be replaced by two identical loops, a hole loop and an outer boundary loop.

$edginfo_j = \pm(20 \cdot indcur + edgtyp)$ encodes information about the jth edge of the loop. $indcur > 0$ is the index of the labelled curve that the edge topologically lies on, or $indcur = 0$ in the case of an unlabelled line segment (e.g. introduced during the decomposition process). The sign is negative (positive) if the edge topologically lies on a labelled curve and the direction from v_j to v_{j+1} is opposite from (same as) the direction of the labelled curve. The sign does not matter in the case of $indcur = 0$. *edgtyp* indicates the type of the edge as follows:

- 1 if an unsplittable boundary edge (e.g. cannot be split into mesh subedges)
- 2 if a splittable boundary edge (on an outer region boundary or a hole)
- 3 if an unsplittable interior constrained edge (also used for a boundary edge on input if unknown whether boundary or interior)
- 4 if a splittable interior constrained edge (also used for a boundary edge on input if unknown whether boundary or interior)
- 5 if an interior unconstrained edge (the *regcode* value of the two subregions incident on the edge must be the same, and the edge cannot topologically lie on a labelled curve)

There are two special cases where *regcode* may take on a negative value; these special cases are not supported by all meshing operations. For example, they allow for the triangulation of isolated vertices or a general planar straight line graph (where chains of edges may lie in the interior of a subregion or be attached to a hole or hole interface), possibly with no embedded region, i.e. no outer boundary or hole loops. If there is no embedded region, then the region is the convex hull of the specified vertices. *regcode* may be:

- -1 for a list of isolated vertices (not on any edge) in compact form; the list of v_j may contain negated vertex labels to indicate the end of a range of vertices, e.g. 2 -5 10 20 -28 is short for the vertices with labels $2, \ldots, 5, 10, 20, \ldots, 28$; *inreg* is > 0 to indicate the subregion containing the isolated vertices (or arbitrary if there is no embedded region); all isolated vertices in the same subregion may be put in the same "loop" (if no embedded region, all isolated vertices may be put in the same "loop"); the logical record of *edginfo* values is omitted for this case.
- -2 for chains of isolated edges (not intersecting the interior of any edge) in compact form; the list of v_j may contain negated vertex labels to indicate the end of a chain of edges, e.g. 1 7 3 -11 17 -6 is short for the edges (1,7), (7,3), (3,11), (17,6); v_{nv} must be negative (a cycle can be specified by duplicating the first vertex of the cycle at the end); *inreg* is > 0 to indicate the subregion containing the isolated edges (or arbitrary if there is no embedded region); all isolated edges in the same subregion may be put in the same "loop" (if no embedded region, all isolated edges may be put in the same "loop"); the *edginfo* values corresponding to the negated vertex labels are arbitrary, since there is no edge to the next vertex.

3 2-D curve file format

---

ncurv
`[`
curvrep *bndcode*
{ *v*1 *v*2 } `or`
{ *v*1 *v*2 *v*3 } `or`
{ *curvtyp* *order* *hicpt* *dim*
$knot_{order}$ $knot_{order+1}$ ... $knot_{hicpt}$
$ctlpt_0$
$ctlpt_1$
⋮
$ctlpt_{hicpt}$ }
`]ncurv`

The 2-D curve file format consists of information on the boundary or interior curves of a 2-D region or 2-D mesh. Any curve that is listed in the curve file is considered to be *labelled*, i.e. it has a label or index based on its position in the file. All nonlinear curves must be labelled. A curve that is a straight line segment may be labelled or unlabelled. Line segments with no boundary condition code do not need to be labelled. All labelled curves are considered to be *constrained*, i.e. cannot be changed. But a constrained line segment does not need to be labelled.

There are currently three kinds of curve representations: *geometric line segment* represented by the vertex labels of its two endpoints, *geometric circular arc* represented by the vertex labels of its two endpoints plus a third point on interior of the arc, and NURBS curve. A line segment or circular arc could be represented by a NURBS curve instead of geometrically. A circular arc is considered to be a NURBS curve, even if it is represented using the geometric form (e.g. in the definition of the vertex types). A geometric circular arc will be internally converted to a standard NURBS curve with middle knot 0.5 and two rational quadratic Bézier pieces having end weights 1, in order to determine the u parameter values of points on the arc.

The first logical record contains the nonnegative integer field *ncurv*, the number of labelled curves. If there are no labelled curves, then the curve file can consist of a single record with the number 0.

Each labelled curve can be represented in one of three ways. The first logical record for each labelled curve consists of the integer fields *curvrep* and *bndcode*. *curvrep* is the curve representation kind; it is 1 for a geometric line segment, 2 for a geometric circular arc, or 3 for a NURBS curve. *bndcode* ≥ 0 is the boundary condition code; a 0 value means that no boundary condition is associated with the curve. Different positive *bndcode* values can be assigned for different types of boundary conditions (or *bndcode* may be used for mapping to an "external" curve label).

For the other records, the knots and control point coordinates are real numbers and the other fields are integers. For a geometric line segment (first alternative), v_1 and v_2 are the positive vertex labels of its endpoints (as given in the associated 2-D region or 2-D mesh file). For a geometric circular arc (second alternative), v_1 and v_3 are the vertex labels of its endpoints, and v_2 is the vertex label of a third (middle) point on the interior of the arc (the middle point is also considered to be a constrained vertex of the region or mesh).

For a NURBS curve (third alternative), the magnitude of *curvtyp* is 1 for a line segment, 2 for a circular arc of order 3, and 3 for any other type of NURBS curve. The sign of *curvtyp* is negative if the curve is in piecewise Bézier form (use positive sign if unknown). *order* is ≥ 2 for the order of the NURBS curve (the order is one greater than the degree of the polynomial in the spline curve). *hicpt* is the highest index or subscript of the control points (the low or starting index is 0); the number of control points is actually

$hicpt + 1$. dim is 2 or 3 for the dimension of the NURBS curve control points in homogeneous coordinates (i.e. 2 for nonrational or 3 for rational).

The knots with indices from $order$ to $hicpt$ must be in the interval $(0.0, 1.0)$, in nondecreasing order, and have multiplicity at most $order - 1$. The knots record is omitted if $order > hicpt$, which occurs if there are no interior knots. The end knots are $knot_0 = \ldots = knot_{order-1} = 0.0$ and $knot_{hicpt+1} = \ldots = knot_{hicpt+order} = 1.0$ so aren't explicitly stored in the file. The end knots of multiplicity $order$ mean that the NURBS curve interpolates its endpoints. A NURBS curve with non-multiple end knots or knots not in the interval $[0.0, 1.0]$ can always be transformed to the required form.

$ctlpt_j$ consists of dim coordinates. If $dim = 3$, then homogeneous coordinates are used with positive weights, i.e. $(w_j x_j, w_j y_j, w_j)$ are the coordinates where (x_j, y_j) are the coordinates of the control point and $w_j > 0$ is the weight. In the special case of circular arcs, the weights may be nonpositive; if $w_j = 0$, then $(x_j, y_j, 0)$ is a control point at infinity in the direction (x_j, y_j).

4 2-D mesh file format

---

nvc
`[` x y $vertinfo$ `]nvc`
nvx
`[` $nodecode$ $icurv$ $ucurv$ `]nvx`
$nodelem$ $nelem$
`[` v_0 v_1 ... $v_{|nodelem|-1}$ `]nelem`
`[` $regcode$ $edginfo_0$ $edginfo_1$ ... $edginfo_{vertelem-1}$ `]nelem`

The 2-D mesh file format consists of four groups of information: vertex coordinates, vertex extra-info records, element nodes, and element and edge information. In the last group, $vertelem$ is equal to either 3 or 4, depending on the value of $nodelem$.

The first logical record contains the positive integer field nvc, the number of vertex coordinates. The next nvc logical records each contain the x and y coordinates of a vertex (or midnode) and a vertex information field $vertinfo$. The next logical record contains the nonnegative integer field nvx, the number of vertex extra-info records. If $nvx > 0$, the next nvx logical records each contain the integer fields $nodecode$, $icurv$ and real number field $ucurv$. These fields are the same as in the 2-D region file format (see Section 2).

The next logical record contains the integer field $nodelem$ and positive integer field $nelem$, the number of elements. $|nodelem| \geq 3$ is the number of nodes per element: 3 for linear triangle, 4 for linear quadrilateral (or quadrilateral/triangle in a mixed mesh), 6 for quadratic triangle, or 8 for quadratic quadrilateral. $nodelem$ is negative (either -4 or -8) for a mixed mesh with at least one triangle.

The next $nelem$ logical records each contain the integer vertex labels v_j of the corner nodes (and midnodes) of a mesh element; the label is positive for reference to a vertex coordinate record. Let $vertelem =$ 3 or 4 be the maximum number of vertices per element ($|nodelem|$ for linear element, $|nodelem|/2$ for quadratic element). Then the first $vertelem$ nodes are vertices or corner nodes in CCW order, and if $vertelem < |nodelem|$, then the last $vertelem$ nodes are midnodes. $v_{j+vertelem}$ is the label of the midnode on the edge joining vertex v_j and its CCW successor. For a triangle in a mixed mesh, v_3 and v_7 are set to ≤ 0. Quadrilaterals must be simple and strictly convex (all four interior angles must be $> 0°$ and $< 180°$). In a quadratic mesh, midnode labels may be set to ≤ 0 to indicate that the midnode is at the midpoint of the straight edge joining the corner nodes and its coordinates are not explicitly stored.

The last $nelem$ logical records each contain $vertelem + 1$ integer fields for element and edge information. $regcode \geq 0$ is an integer region code associated with each element as described in Section 2. $edginfo_j = \pm(20 \cdot indcur + edgtyp)$ encodes information about the jth edge of element, as described in Section 2. The

jth edge is the one joining vertex v_j and its CCW successor. For a triangle in a mixed mesh, $edginfo_3$ is arbitrary, e.g. 0.

5 3-D region file format

---

nvc
[x y z $vertinfo$]nvc
nvx
[*nodecode* *icsur* *ucsur* *vcsur*]nvx
nloop
[nv $facinfo$ $infac$
v_1 v_2 ... v_{nv}
$edginfo_1$ $edginfo_2$ ... $edginfo_{nv}$]nloop
nshell
[nf *regcode* *inreg*
f_1 f_2 ... f_{nf}]nshell

The 3-D region file format consists of four groups of information: vertex coordinates, vertex extra-info records, loops, and shells. The first logical record contains the positive integer field *nvc*, the number of vertex coordinates. The next *nvc* logical records each contain the x, y, and z coordinates of a vertex and a vertex information field $vertinfo$. x, y, and z are real numbers, and $vertinfo$ is an integer. $vertinfo \leq 0$ indicates that the vertex is not referenced. If $vertinfo > 0$, then $vertinfo = 20 \cdot xtraptr + vertyp$ encodes two fields. $xtraptr = 0$ if not used; else $xtraptr > 0$ is the label (or index) of a vertex extra-info record (see below). In the latter case, $nodecode[xtraptr]$, $icsur[xtraptr]$, $ucsur[xtraptr]$, and $vcsur[xtraptr]$ are extra fields associated with the vertex. *vertyp* indicates the type of the vertex (or midnode) as described in Section 2.

The next logical record contains the nonnegative integer field nvx, the number of vertex extra-info records. If $nvx > 0$, the next nvx logical records each contain the integer fields *nodecode*, *icsur* and real number fields *ucsur*, *vcsur*. *nodecode* is a node code (if positive), which can be used to specify a load or constraint at a node (or mapping to an "external" vertex label). If not used (and the record is used), *nodecode* is set to 0. *icsur* is the label of a space curve or surface (if positive), $0.0 < ucsur < 1.0$ is the u parameter value of the point on (or topologically on) the curve or surface, and *vcsur* is either outside interval $(0.0, 1.0)$ for the curve or in interval $(0.0, 1.0)$ for the v parameter value of the point on (or topologically on) the surface. If not used, *icsur* is set to ≤ 0 and *ucsur*, *vcsur* to outside interval $(0.0, 1.0)$, e.g. -1.0. NURBS curves are defined for $0 \leq u \leq 1$, and NURBS surfaces for $0 \leq u \leq 1$ and $0 \leq v \leq 1$ (see the next section).

The latter three fields are used for all vertices or midnodes that (topologically) lie on the interior of a labelled curve or surface (a curve or surface that is defined in the curve/surface file). For a labelled geometric space line segment, the u parameter value is determined by the standard parameterization of the line segment with $0 \leq u \leq 1$. For a labelled geometric planar facet, the u and v parameter values are determined by the standard parameterization of the bounding rectangle with $0 \leq u \leq 1$ and $0 \leq v \leq 1$ (see the next section). Depending on the *vertyp* value, *ucsur* is either the exact u value or an estimated u value; similarly for *vcsur* (if for a surface). These four fields are stored separately from the vertex coordinates to save space (most vertices, e.g. interior mesh vertices, will not use these fields). *nodecode* is set to < 0 to indicate that none of the four fields is used (i.e. the record is not referenced by the *xtraptr* part of any $vertinfo$ field); in this case, the other three fields can have arbitrary values.

The next logical record contains the positive integer field *nloop*, the number of loops (boundary of simple face or simple face hole). A face on a NURBS surface does not have to be planar, but a face on a planar

facet should be planar. The next *nloop* sequences of three logical records contain integer information for each loop. $nv \geq 2$ is the number of vertices in the loop; $nv = 2$ is allowed only if there is at least one curved (nonlinear) edge between the two vertices. If nonnegative, $facinfo = 20 \cdot indsur + factyp$ encodes information about the face bounded by the loop, or $facinfo \geq 0$ is arbitrary if the loop bounds a face hole. $indsur > 0$ is the index of the labelled surface that the face topologically lies on (the face can be a portion of a trimmed NURBS surface patch), or 0 in the case of an unlabelled planar facet (e.g. introduced during the decomposition process). $factyp$ indicates the type of the face as follows:

- 1 if an unsplittable boundary face (e.g. cannot be split into subfaces); the face must be a triangle for a tetrahedral or triangular mesh, or a triangle or quadrilateral for a hexahedral or quadrilateral mesh
- 2 if a splittable boundary face (on an outer region boundary or an interior hole)
- 3 if an unsplittable interior constrained face
- 4 if a splittable interior constrained face
- 5 if an interior unconstrained face (the *regcode* value of the two subregions incident on the face must be the same, and the face cannot topologically lie on a labelled surface)

$infac$ is 0 if the loop is an outer boundary, > 0 if the loop is the boundary of a face hole, or < 0 if the loop is the boundary of a hole face (a face that is also the hole of another face); so the sign of $infac$ indicates the loop type. The magnitude of $infac$ (if nonzero) is the label of the face (actually the boundary loop of the face) containing the face hole or hole face. In the case of nested hole faces, the label should be for the smallest face containing the face hole or hole face. For the ith loop, $|infac|$ may be any integer between 1 and *nloop*, except for i. A hole face may alternatively be defined as two separate loops with identical vertices; one loop is the normal outer boundary of a face and the other is a face hole.

$v_1, v_2, \ldots, v_{nv}$ are the positive labels of vertices in circular order on the loop. The starting vertex is arbitrary. The edges of the loop are implicitly represented by their two endpoints, i.e. there are (curved) edges joining vertices v_1 and v_2, v_2 and v_3, ..., v_{nv} and $v_{nv+1} = v_1$. The face or area defined by each loop must be simple so no duplicate vertices may appear in the boundary loop. This means that the outer loops of faces are not allowed to have holes attached to cut interfaces (as in the 2-D case).

$edginfo = \pm(20 \cdot indcsur + edgtyp)$ encodes information about the jth edge of the loop. $indcsur > 0$ is the index of the labelled curve or surface that the edge topologically lies on, or $indcsur = 0$ in the case of an edge that does not topologically lie on a labelled curve or surface (e.g. an interior edge created during the decomposition process). In the case of a labelled curve, *indcsur* is the index of a space curve or a domain NURBS curve if the face determined by the loop topologically lies on a nonplanar NURBS surface, else *indcsur* is the index of a space curve on a planar facet. The sign is negative (positive) if the edge topologically lies on a labelled curve and the direction from v_j to v_{j+1} is opposite from (same as) the direction of the labelled curve. The sign does not matter for a surface or an unlabelled curve. *edgtyp* indicates the type of the edge as follows:

- 1 if an unsplittable edge topologically on a constrained curve (e.g. cannot be split into mesh subedges), where a constrained curve is on the boundary of a constrained (trimmed) surface or is isolated or dangling in the interior of a region
- 2 if a splittable edge topologically on a constrained curve [for the first 2 types, *indcsur* is either 0 or the index of a space or domain curve]
- 3 if an unsplittable constrained straight edge topologically on the interior of a constrained surface
- 4 if a splittable constrained straight edge topologically on the interior of a constrained surface
- 5 if a partially constrained straight edge topologically on the interior of a constrained surface, where the edge is restricted to topologically lie on the surface [for types 3 to 5, *indcsur* is either 0 or the index of a surface]

- 8 if an unconstrained straight edge in the interior of the region (e.g. lies on an unconstrained planar facet) [for this last type, *indcsur* is 0]

For compatibility with the preceding Geompack90 file formats (see Appendix A), the following two edge types are kept, but they are treated as equivalent to types 1 and 2, respectively:

- 6 if an unsplittable constrained straight edge on a constrained interior space geometric line segment (not topologically on a constrained curve or surface, with endpoints possibly constrained on the interior of a curve or surface), which can be an isolated edge or on the boundary of an unlabelled planar geometric facet or an interior face of type 5
- 7 if a splittable constrained straight edge on a constrained interior space geometric line segment [for types 6 and 7, *indcsur* is either 0 or the index of a space geometric line segment]

There are two special cases where $facinfo$ may take on a negative value; these special cases are not supported by all meshing operations. For example, they allow for the Delaunay triangulation of isolated vertices or the constrained triangulation of isolated vertices, isolated edges, and/or isolated faces. $facinfo$ may be:

- -1 for a list of isolated vertices (not on any edge or face) in compact form; the list of v_j may contain negated vertex labels to indicate the end of a range of vertices, e.g. 2 -5 10 20 -28 is short for the vertices with labels $2, \ldots, 5, 10, 20, \ldots, 28$; $infac$ is arbitrary and the logical record of $edginfo$ values is omitted for this case; all isolated vertices in the same subregion may be put in the same "loop".
- -2 for chains of isolated edges (not intersecting the interior of any edge or face) in compact form; the list of v_j may contain negated vertex labels to indicate the end of a chain of edges, e.g. 1 7 3 -11 17 -6 is short for the edges (1,7), (7,3), (3,11), (17,6); v_{nv} must be negative (a cycle can be specified by duplicating the first vertex of the cycle at end); $infac$ is arbitrary and the $edginfo$ values corresponding to the negated vertex labels are arbitrary, since there is no edge to next vertex; all isolated edges in the same subregion may be put in the same "loop".

The next logical record contains the positive integer field *nshell*, the number of shells (boundary of a subregion, interior space hole, or hole polyhedron). A shell must be the boundary of a simple topological polyhedron, with the addition of double-occurring cut faces allowed in the cases of a subregion or hole polyhedron. (A cut face or a group of adjacent cut faces can occur near a volume hole that goes through face holes at both ends.) The last *nshell* sequences of two logical records contain integer information for each shell. $nf \geq 2$ is the number of faces in the shell, e.g. a sphere must be split into at least two topological faces. *regcode* is a nonnegative region code associated with the subregion defined by the shell; it is 0 if the shell is boundary of interior hole or there is nothing special about the subregion, or positive to indicate a material type or some other property. *inreg* is 0 if the shell is an outer boundary, > 0 if the shell is the boundary of an interior hole, or < 0 if the shell is the boundary of a hole polyhedron (the "hole" is another subregion); so the sign of *inreg* indicates the shell type. A hole polyhedron may be adjacent to other hole polyhedra. The magnitude of *inreg* (if nonzero) is the label of the subregion (actually the boundary shell of the subregion) containing the interior hole or union of adjacent hole polyhedra. In the case of nested hole polyhedra, the label should be for the smallest subregion containing the interior hole or union of adjacent hole polyhedra. For the ith shell, $|inreg|$ may be any integer between 1 and *nshell*, except for i.

$f_1, f_2, \ldots, f_{nf}$ are the nonzero signed labels of faces in arbitrary order on the shell. f_j has a positive (negative) sign if face loop $|f_j|$ (with vertex order as provided in the loop record) is in CCW (CW) orientation when the subregion or interior hole is viewed from outside. This means that a face common to two subregions (not on the boundary of an interior hole or the union of adjacent hole polyhedra) must have opposite signs in the face lists for the two shells (or one shell in the case of a double-occurring cut face). Face holes or the "hole" part of hole faces are not put in any shell face lists (since it is known what faces they lie on). The space inside an interior hole can be partially or totally occupied by one or more subregions. For example,

a hole polyhedron that is not adjacent to other hole polyhedra can be replaced by two identical shells, an interior hole shell and an outer boundary shell.

A loop that is an isolated vertex or edge list is put into the shell face list of the smallest subregion (with nonpositive *inreg* value) containing the isolated vertices or edges. Either sign can be used for the signed face label. If the isolated vertices or edges lie outside all subregions, then the isolated vertex or edge list can be put into a separate special shell (with *inreg* value of 0) consisting of only isolated entities (vertices, edges, or faces).

Isolated or dangling faces are also allowed, where the face is not on the boundary of any subregion, may share an edge with a boundary face of a subregion, and may have an edge that is not shared with any other face. An isolated face must be specified as double-occurring in a shell face list (the signed face label occurs twice in list with opposite signs). It is either put into the shell face list of the smallest subregion (with nonpositive *inreg* value) containing the face or put into a separate isolated entity shell if it lies outside all subregions.

If all shells are isolated entity shells, then the region is either the convex hull of the specified vertices or a bounding polyhedron containing the vertices in its interior. Isolated entities are not supported by all meshing operations, e.g. if volume decompositition is required. If a region consists of subregion shells and isolated entity shells, then some meshing operations may ignore the latter shells.

6 3-D curve/surface file format

---

ncurv
`[`
curvrep bndcode|*curvnum*
{ *v*1 *v*2 } `or`
{ *v*1 *v*2 *v*3 } `or`
{ *curvtyp*|*surfnum order hicpt dim*
$knot_{order}$ $knot_{order+1}$ ... $knot_{hicpt}$
$ctlpt_0$
$ctlpt_1$
⋮
$ctlpt_{hicpt}$ }
`]ncurv`
nsurf
`[`
surfrep bndcode
{ $ctlpt_{00}$
$ctlpt_{10}$
$ctlpt_{01}$
$ctlpt_{11}$ } `or`
{ $ctlpt_{00}$
$ctlpt_{01}$
$ctlpt_{10}$
$ctlpt_{11}$ } `or`
{ *v*1 *v*2 *v*3 *v*4 *v*5 *v*6 } `or`
{ *surftyp degen order*1 *order*2 *hicpt*1 *hicpt*2 *dim*
$uknot_{order1}$ $uknot_{order1+1}$ ... $uknot_{hicpt1}$
$vknot_{order2}$ $vknot_{order2+1}$ ... $vknot_{hicpt2}$
$ctlpt_{00}$

$ctlpt_{10}$
$\vdots$
$ctlpt_{hicpt1,0}$
$ctlpt_{01}$
$\vdots$
$ctlpt_{hicpt1,hicpt2}$ } or
{ $surftyp$ $degen$ $order1$ $order2$ $hicpt1$ $hicpt2$ dim
$uknot_{order1}$ $uknot_{order1+1}$ ... $uknot_{hicpt1}$
$vknot_{order2}$ $vknot_{order2+1}$ ... $vknot_{hicpt2}$
$ctlpt_{00}$
$ctlpt_{01}$
$\vdots$
$ctlpt_{0,hicpt2}$
$ctlpt_{10}$
$\vdots$
$ctlpt_{hicpt1,hicpt2}$ }
]nsurf

The 3-D curve/surface file format consists of information on the curves and surfaces of a 3-D region or 3-D mesh or surface mesh, including planar domain NURBS curves. Any curve or surface that is listed in the curve/surface file is considered to be *labelled*, i.e. it has a label or index based on its position in the curve or surface portion of the file. All nonlinear curves and nonplanar surfaces must be labelled. A labelled space curve can be on the boundary of a (trimmed) surface patch or isolated or dangling in the interior of a 3-D region. A space curve that is a straight line segment may be labelled or unlabelled. A surface that is a planar facet may be labelled or unlabelled. Line segments or planar facets with no boundary condition code do not need to be labelled. All labelled curves and surfaces are considered to be *constrained*. But a constrained line segment or constrained planar facet does not need to be labelled.

There are currently three kinds of curve representations: *geometric line segment* represented by the vertex labels of its two endpoints, *geometric circular arc* represented by the vertex labels of its two endpoints plus a third point on interior of arc, and NURBS curve. A line segment or circular arc could be represented by a NURBS curve instead of geometrically. A circular arc is considered to be a NURBS curve, even if it is represented using the geometric form (e.g. in the definition of the vertex types). A geometric circular arc will be internally converted to a standard NURBS curve with middle knot 0.5 and two rational quadratic Bézier pieces having end weights 1, in order to determine the u parameter values of points on the arc.

There are currently three kinds of surface representations: *geometric planar facet* represented by the four corners of a bounding rectangle, *geometric right circular cylinder rectangular patch* represented by six vertex labels for the endpoints and middle point of two circular arcs, and trimmed NURBS surface. A planar facet or right circular cylinder rectangular patch could be represented by a NURBS surface instead of geometrically. A right circular cylinder rectangular patch is considered to be a NURBS surface, even if it is represented using the geometric form. A geometric right circular cylinder rectangular patch will be internally converted to a standard NURBS surface with middle u or v knot 0.5 and two rational Bézier pieces of degree (2,1) or (1,2) having corner weights 1, in order to determine the u and v parameter values of points on the patch.

A domain curve $(u(t), v(t))$, $0 \le t \le 1$, is a trimming curve for a nonplanar NURBS surface $S(u, v)$, $0 \le u, v \le 1$ such that the space curve $C(t)$, $0 \le t \le 1$, is approximately curve $S(u(t), v(t))$, $0 \le t \le 1$, i.e. $C(t) \approx S(u(t), v(t))$ for $0 \le t \le 1$. The identity mapping between the parameter t of the domain and space curves means that a space curve and any corresponding domain curves must start and end at

corresponding "points". Since intersection curves are usually only approximations, $C(t)$ is not identical to $S(u(t), v(t))$ in general. Every space curve bounding a nonplanar trimmed NURBS surface must be labelled and have a corresponding planar domain NURBS curve.

The first logical record contains the nonnegative integer field $ncurv$, the number of labelled curves (including planar domain NURBS curves). After the logical records for curves, there is a logical record that contains the the nonnegative integer field $nsurf$, the number of labelled surfaces. The file format ends with logical records for surfaces. If there are no labelled curves or surfaces, then the curve/surface file can consist of two records, each with the number 0.

Each labelled curve can be represented in one of three ways. The first logical record for each labelled curve consists of the integer fields *curvrep* and *bndcode*. In the case of planar domain NURBS curves, *bndcode* is alternatively named as *curvnum*. *curvrep* is the curve representation kind; it is 1 for a geometric line segment, 2 for a geometric circular arc, or 3 for a space NURBS curve or planar domain NURBS curve. All domain curves must be of the third kind and must be used for nonplanar surfaces; domain curves are not used for planar facets.

If nonnegative, *bndcode* is the boundary condition code for a space curve; a 0 value means that no boundary condition is associated with the curve. Different positive *bndcode* values can be assigned for different types of boundary conditions (or *bndcode* may be used for mapping to an "external" curve label). Negative *bndcode* values are used for planar domain NURBS curve; in this case, $-bndcode$ (or $-curvnum$) is the label of the space curve corresponding to the domain curve (the space and domain curve should have the same direction and identity mapping between their parameter).

For the other records, the knots and control point coordinates are real numbers and the other fields are integers. For a geometric line segment (first alternative), v_1 and v_2 are the positive vertex labels of its endpoints (as given in the associated 3-D region or 3-D mesh or surface mesh file). For a geometric circular arc (second alternative), v_1 and v_3 are the vertex labels of its endpoints, and v_2 is the vertex label of a third (middle) point on the interior of the arc (the middle point is also considered to be a constrained vertex of the region or mesh).

For a NURBS curve (third alternative), most of the fields have the same meaning as in Section 3. The differences are as follows. In addition to space NURBS curves, all planar domain curves are NURBS curves. For a domain NURBS curve, the field *curvtyp* is alternatively named as $surfnum$. $surfnum$ is the positive label of the nonplanar NURBS surface that the domain curve is defined for. The field *dim* is 2 or 3 for planar domain curves, or 3 or 4 for space curves, where the smaller (larger) dimension value is used for the nonrational (rational) case.

Each labelled surface can be represented in one of three ways. The first logical record for each labelled surface consists of the integer fields $surfrep$ and *bndcode*. $surfrep$ is the surface representation kind; it is ± 1 for a geometric planar facet, ± 2 for a geometric right circular cylinder rectangular patch, or ± 3 for a trimmed NURBS surface. $bndcode \geq 0$ is the boundary condition code; a 0 value means that no boundary condition is associated with the surface. Different positive *bndcode* values can be assigned for different types of boundary conditions (or *bndcode* may be used for mapping to an "external" surface label).

For the other records, the knots and control point coordinates are real numbers and the other fields are integers. For a geometric planar facet (first or second alternative), $ctlpt_{00}$, $ctlpt_{10}$, $ctlpt_{01}$, and $ctlpt_{11}$ are the "control" or corner points (of dimension 3) of a rectangle in the plane of the facet that bounds the vertices of the facet; $ctlpt_{00}$ and $ctlpt_{11}$ are opposite corners of the rectangle, like for a degree (1,1) B-spline surface. The $surfrep$ value is 1 if the control points are listed by columns (index 10 before 01) or -1 if the control points are listed by rows (index 01 before 10).

For a geometric right circular cylinder rectangular patch (third alternative), v_1, v_2, v_3 and v_4, v_5, v_6 define circular arcs, where v_1 and v_3 are the positive vertex labels of the endpoints of the first arc and v_2 is the vertex label of a third (middle) point on the interior of the first arc (that is also considered to be a constrained vertex of the region or mesh), and similarly for the second arc. Also, v_1v_4, v_2v_5, and v_3v_6 are line segments that are each parallel to the axis of the cylinder. Any non-rectangular patch on a right circular cylinder must be represented in NURBS form. The $surfrep$ value is 2 or -2 if the internal NURBS representation is defined by subpatches of degree (2,1) or (1,2), respectively. For $surfrep = 2$, the circular

arc is specified by the u parameter and the line segment by the v parameter. For $surfrep = -2$, the roles of u and v are reversed. The sign of $surfrep$ affects the control point coordinates of the domain curves for the nonplanar patch.

For a trimmed NURBS surface (fourth or fifth alternative), the magnitude of $surftyp$ is 1 for a planar facet, 2 for a right circular cylinder rectangular patch of degree (2,1) or (1,2), 3 for a circular cone patch, 4 for a sphere patch, and 5 for any other type of NURBS surface. The sign of $surftyp$ is negative if the u knots are in piecewise Bézier form (use positive sign if unknown). If $|surftyp|$ is 1, then the surface is a planar NURBS surface so domain curves are not used for the surface. $degen$ is a nonnegative integer used to indicate if any of the four boundary curves of the "untrimmed" NURBS surface are degenerate. The rightmost bit is 1 iff the $u = 0$ curve is degenerate. The second rightmost bit is 1 iff the $u = 1$ curve is degenerate. The third rightmost bit is 1 iff the $v = 0$ curve is degenerate. The fourth rightmost bit is 1 iff the $v = 1$ curve is degenerate. The possible values for the righmost four bits are 0, 1, 2, 3, 4, 8, 12. The value is set to ≥ 16 if the v knots are in piecewise Bézier form (use value ≤ 15 if unknown).

$order1$ and $order2$ are the order of B-splines in the two dimensions, with $order1 \geq 2$ and $order2 \geq 2$ (the order is one greater than the degree). $hicpt1$ and $hicpt2$ are the highest index or subscript of the control points in the first or second dimension (u or v), respectively. The number of control points is $(hicpt1+1)(hicpt2+1)$. dim is 3 or 4 for the dimension of the NURBS surface control points in homogeneous coordinates (i.e. 3 for nonrational or 4 for rational).

The $uknot$ and $vknot$ sequences are similar to the knot sequence for the curve case; there is one sequence for each of the two dimensions. The control points each have dim coordinates. As in the curve case, the weights should be positive and homogeneous coordinates used if $dim = 4$, except that nonpositive weights may be used in the special case of a right circular cylinder rectangular patch. The $surfrep$ value is 3 if the control points are listed by columns (index 10 after 00, i.e. vary in u before v) or -3 if the control points are listed by rows (index 01 after 00). In the case of a planar facet, the four corner control points of the NURBS surface should form a strictly convex planar quadrilateral that bounds the vertices of the facet.

7 3-D mesh file format

---

nvc
`[` x y z $vertinfo$ `]nvc`
nvx
`[` $nodecode$ $icsur$ $ucsur$ $vcsur$ `]nvx`
$nodelem$ $nelem$
`[` v_0 v_1 ... $v_{|nodelem|-1}$ `]nelem`
`[` $regcode$ $facinfo_0$ $facinfo_1$ ... $facinfo_{facelem-1}$ `]nelem`
$ncedg$
`[` $edgv_0$ $edgv_1$ $edginfo$ `]ncedg`

The 3-D mesh file format consists of five groups of information: vertex coordinates, vertex extra-info records, element nodes, element and face information, and constrained edges. In the fourth group, $facelem$ is equal to either 4 or 6, depending on the value of $nodelem$.

The first logical record contains the positive integer field nvc, the number of vertex coordinates. The next nvc logical records each contain the x, y, and z coordinates of a vertex (or midnode) and a vertex information field $vertinfo$. The next logical record contains the nonnegative integer field nvx, the number of vertex extra-info records. If $nvx > 0$, the next nvx logical records each contain the integer fields $nodecode$, $icsur$ and real number fields $ucsur$, $vcsur$. These fields are the same as in the 3-D region file format (see Section 5).

The next logical record contains the integer field *nodelem* and positive integer field *nelem*, the number of elements. $|nodelem| \geq 4$ is the number of nodes per element: 4 for linear tetrahedron, 8 for linear hexahedron (or hexahedron/wedge/pyramid/tetrahedron in a mixed mesh), 10 for quadratic tetrahedron, or 20 for quadratic hexahedron. *nodelem* is negative (either -8 or -20) for a mixed mesh with at least one non-hexahedral element.

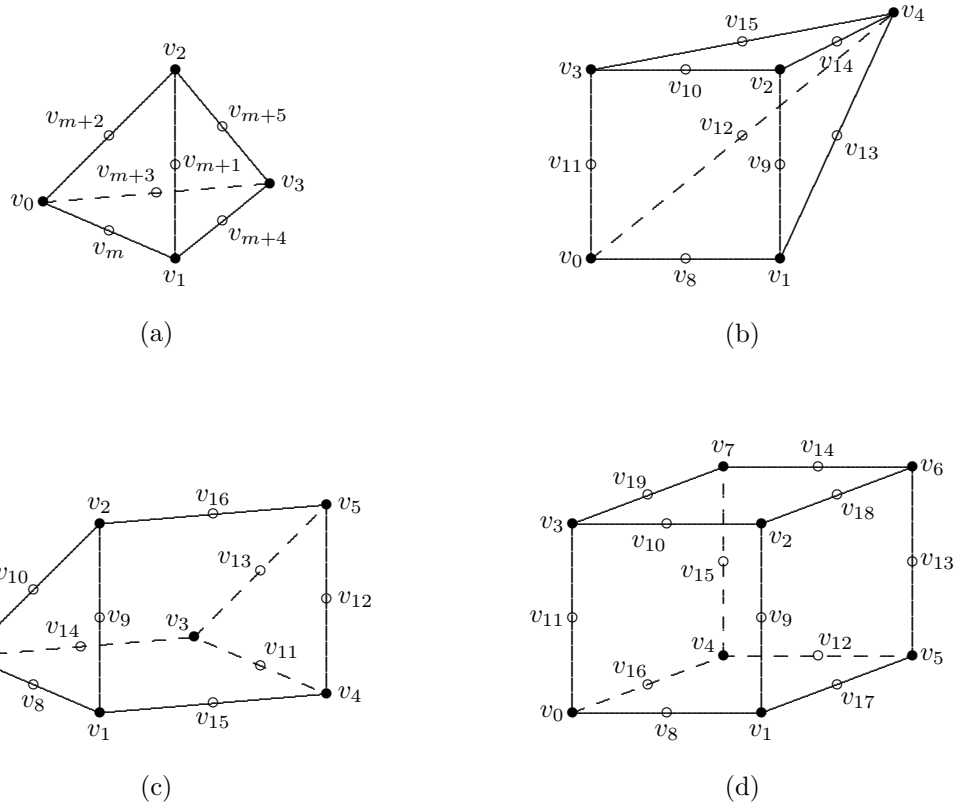

Figure 7.1: Illustration of node ordering for tetrahedron, pyramid, wedge, and hexahedron. In (a), $m = 4$ for a tetrahedral mesh, or $m = 8$ for a mixed hexahedral mesh.

The next *nelem* logical records each contain the integer vertex labels v_j of the corner nodes (and midnodes) of a mesh element; the label is positive for reference to a vertex coordinate record. Let $facelem = 4$ or 6 be the maximum number of faces per element (4 if *nodelem* is 4 or 10, 6 otherwise). Let $m = 4$ or 8 for a tetrahedral or mixed hexahedral mesh, respectively. Then v_0 to v_{m-1} are used for vertices or corner nodes, and in the quadratic case, v_m to $v_{|nodelem|-1}$ are used for midnodes. The ordering of vertices and midnodes (in the quadratic case) for each type of element is as shown in Figure 7.1. For the tetrahedron and wedge, $v_0v_1v_2$ is a triangular face with vertices in CCW order when viewed from outside the element. For the pyramid and hexahedron, $v_0v_1v_2v_3$ is a quadrilateral face with vertices in CCW order when viewed from outside the element. For a tetrahedron in a mixed mesh, v_4, v_5, v_6 and v_{14} to v_{19} are set to ≤ 0 and v_7 is set to -4. For a pyramid in a mixed mesh, v_5, v_6 and v_{16} to v_{19} are set to ≤ 0 and v_7 is set to -5. For a wedge in a mixed mesh, v_6 and v_{17} to v_{19} are set to ≤ 0 and v_7 is set to -6. Note that v_7 is positive for a hexahedron and the negated number of vertices in the element for a tetrahedron, pyramid, or wedge. In a quadratic mesh, midnode labels may be set to ≤ 0 to indicate that the midnode is at the midpoint of the

straight edge joining the corner nodes and its coordinates are not explicitly stored.

The next *nelem* logical records each contain $facelem + 1$ integer fields for element and face information. $regcode \geq 0$ is an integer region code associated with each element as described in Section 5. $facinfo_j = 20 \cdot indsur + factyp$ encodes information about the jth face of element, as described in Section 5. For a non-hexahedral element in a mixed mesh, the one or two unneeded $facinfo_j$ at the end of the record are set to an arbitrary value, e.g. 0. For a tetrahedron, the jth face is the one opposite vertex v_j, $0 \leq j \leq 3$. For a pyramid, the 0th to 4th faces are the ones with vertex subscripts 041, 142, 243, 340, 0123, respectively (the last face is the quadrilateral). For a wedge, the 0th to 4th faces are the ones with vertex subscripts 0341, 1452, 2530, 012, 354, respectively. For a hexahedron, the 0th to 5th faces are the ones with vertex subscripts 0451, 1562, 2673, 3740, 0123, 4765, respectively.

The next logical record contains the positive integer field *ncedg*, the number of constrained or partially constrained mesh edges (whose *edgtyp* value is not 8). The last *ncedg* logical records each contain three integer fields. $edgv_0$ and $edgv_1$ are the positive labels of the vertices of the edge. $edginfo = \pm(20 \cdot indcsur + edgtyp)$ encodes information about the edge, as described in Section 5. The *edgtyp* part of $edginfo$ should not be 8.

8 Surface mesh file format

---

nvc
`[` *x y z vertinfo* `]nvc`
nvx
`[` *nodecode icsur ucsur vcsur* `]nvx`
nodelem nelem
`[` v_0 v_1 ... $v_{|nodelem|-101}$ `]nelem`
`[` *facinfo* $edginfo_0$ $edginfo_1$ ... $edginfo_{vertelem-1}$ `]nelem`

The surface mesh file format consists of four groups of information: vertex coordinates, vertex extra-info records, element nodes, and element and edge information. In the last group, *vertelem* is equal to either 3 or 4, depending on the value of *nodelem*.

The first logical record contains the positive integer field *nvc*, the number of vertex coordinates. The next *nvc* logical records each contain the x, y, and z coordinates of a vertex (or midnode) and a vertex information field *vertinfo*. The next logical record contains the nonnegative integer field *nvx*, the number of vertex extra-info records. If $nvx > 0$, the next *nvx* logical records each contain the integer fields *nodecode*, *icsur* and real number fields *ucsur*, *vcsur*. These fields are the same as in the 3-D region file format (see Section 5).

The next logical record contains the integer field *nodelem* and positive integer field *nelem*, the number of elements. $|nodelem| - 100 \geq 3$ is the number of nodes per element: 3 for linear triangle, 4 for linear quadrilateral (or quadrilateral/triangle in a mixed mesh), 6 for quadratic triangle, or 8 for quadratic quadrilateral. In other words, $|nodelem|$ is either 103, 104, 106, or 108. *nodelem* is negative (either -104 or -108) for a mixed mesh with at least one triangle. The magnitude of *nodelem* is incremented by 100 to distinguish it from the values for 3-D meshes.

The next *nelem* logical records each contain the integer vertex labels v_j of the corner nodes (and midnodes) of a mesh element; the label is positive for reference to a vertex coordinate record. Let $vertelem =$ 3 or 4 be the maximum number of vertices per element ($|nodelem|-100$ for linear element, $(|nodelem|-100)/2$ for quadratic element). Then the first *vertelem* nodes are vertices or corner nodes in circular order, and if $vertelem < |nodelem|$, then the last *vertelem* nodes are midnodes. $v_{j+vertelem}$ is the label of the midnode on the edge joining vertex v_j and its circular successor. For a triangle in a mixed mesh, v_3 and v_7 are set to

≤ 0. Quadrilaterals may be nonplanar, but the normal vectors of the four triangles (with same orientation) formed from three quadrilateral vertices must satisfy the property that the dot product of any pair of normal vectors is positive (i.e. the angle between any two normal vectors is $< 90°$). For a planar quadrilateral, this property implies that the quadrilateral must be simple and strictly convex (all four interior angles are $> 0°$ and $< 180°$). In a quadratic mesh, midnode labels may be set to ≤ 0 to indicate that the midnode is at the midpoint of the straight edge joining the corner nodes and its coordinates are not explicitly stored.

The last *nelem* logical records each contain $vertelem + 1$ integer fields for element and edge information. $facinfo = 20 \cdot indsur + factyp$ encodes information about the element, as described in Section 5. $edginfo_j = \pm(20 \cdot indcsur + edgtyp)$ encodes information about the jth edge of element, as described in Section 5. The jth edge is the one joining vertex v_j and its circular successor. For a triangle in a mixed mesh, $edginfo_3$ is arbitrary, e.g. 0.

A Extensions to Geompack90 file formats

Geompack90 existed from June 1999 to June 2003. Geompack++ was initially released in March 2002. The Geompack++ file formats are similar to the preceding Geompack90 file formats, with the following minor extensions and different usages:

(a) For Geompack90, a constrained space curve must be on the boundary of a constrained (trimmed) surface. For Geompack++, a constrained space curve may also be a constrained space geometric line segment or other constrained space curve that is isolated or dangling in the interior of a 3-D region. As a result, vertex types 7 and 16 are treated as equivalent to 3 and 10, respectively, and 3-D edge types 6 and 7 are treated as equivalent to 1 and 2, respectively.

(b) For Geompack90, the *indcsur* part of the $edginfo$ field of 3-D region loops is the index of a domain NURBS curve if the edge topologically lies on a space curve and the face determined by the loop topologically lies on a nonplanar NURBS surface. For Geompack++, *indcsur* may be the index of a space curve or a domain NURBS curve in this case.

(c) For Geompack90, all faces of interior space holes of 3-D regions must be boundary faces of the region. For Geompack++, these faces may be interior faces of the region, since the space inside an interior hole of a subregion can be partially or totally occupied by other subregions.

(d) For Geompack++, $surfrep$ values of -1, -2, and -3 have been added to the 3-D curve/surface file format to allow control points to be listed by rows as well as columns, and to allow a geometric right circular cylinder rectangular patch to have degree (1,2) as well as (2,1), i.e. the circular arc may be specified by the v parameter or u parameter.

(e) The *inreg* field is used differently for lists of isolated vertices or edges in 2-D regions. For Geompack90, *inreg* is arbitrary for isolated vertices but is used similarly to hole loops for isolated edges. For Geompack++, *inreg* is used similarly to hole loops for both isolated vertices and isolated edges, i.e. $inreg > 0$ to indicate the subregion containing the isolated entities in its interior. If there is no embedded region (i.e. all loops are lists of isolated vertices or edges, and the region is the convex hull of the specified vertices), *inreg* is arbitrary, e.g. 0.

(f) The placement of some isolated faces in shell face lists of 3-D regions is different. For Geompack90, a dangling face outside all subregions but sharing an edge with a subregion boundary face can be put in the subregion shell or in a separate shell. For Geompack++, any isolated face outside all subregions must be put in a separate isolated entity shell.

B Examples of region, curve/surface, and mesh files

In this appendix, region, curve/surface, and mesh files in the Geompack++ zip file (containing documents and samples) are referenced. The vertices, loops, and curves of 2-D region file `region2d/cirnurb.rg2` and 2-D curve file `region2d/cirnurb.cs2` are illustrated in Figure B.1.

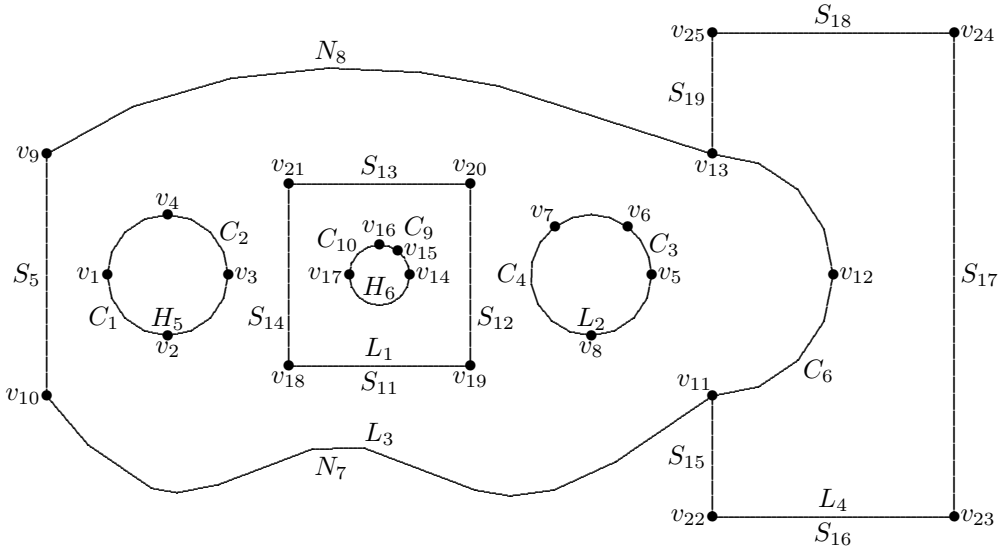

Figure B.1: Illustration of 2-D region and 2-D curve files. Vertices are v_1 to v_{25}.
Loops are L_1, L_2, L_3, L_4, H_5, H_6 (latter two are loops of holes).
Curves are line segments S_j, circular arcs C_j, NURBS curves N_j, where $1 \leq j \leq 19$.

The vertices and quadrilateral elements of 2-D mesh file `mesh2d/semicirc.mh2` and 2-D curve file `region2d/semicirc.cs2` are illustrated in Figure B.2.

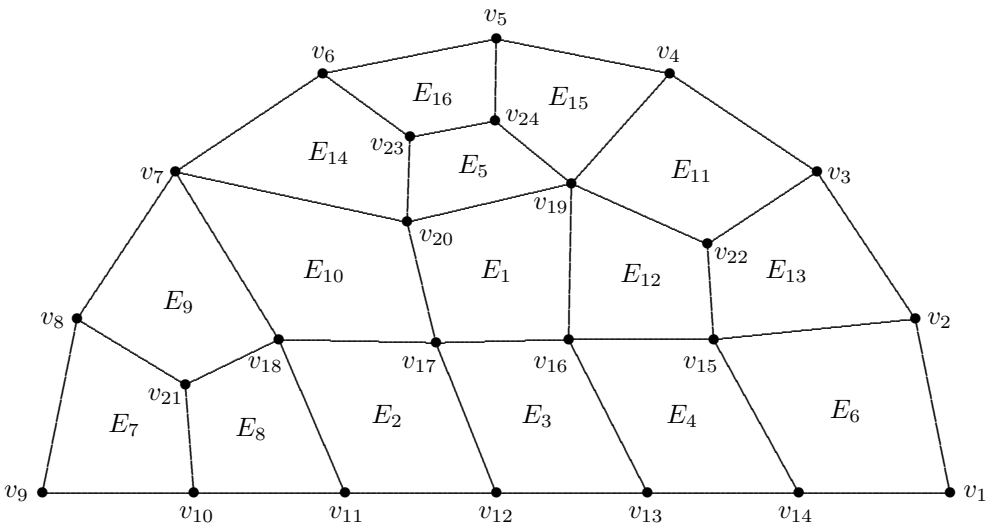

Figure B.2: Illustration of 2-D mesh file. Vertices are v_1 to v_{24}.
Quadrilateral elements are E_1 to E_{16}.

The vertices, loops, space curves, and surfaces of 3-D region file `region3d/cylnurb.rg3` and 3-D curve/surface file `region3d/cylnurb.cs3` are illustrated in Figure B.3. The domain curves (line segments for this example) have labels from 19 to 30 in the curve/surface file. Domain curves 19 to 22, 23 to 26, and 27 to 30 are for surface 6 (left half of circular cylinder), surface 7 (right half of circular cylinder), and surface 8 (top NURBS surface), respectively.

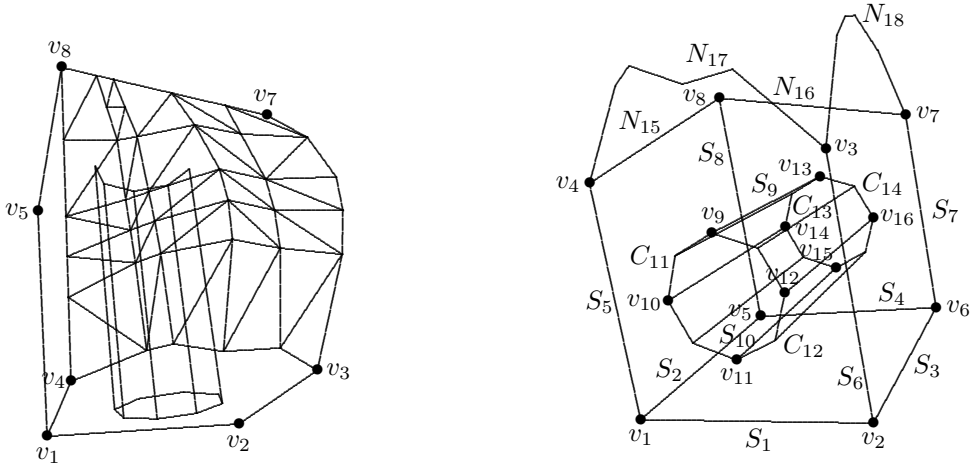

Figure B.3: Illustration of 3-D region and 3-D curve/surface files, with two views. Vertices are v_1 to v_{16}. Loops are L_1 for bottom face, L_2 for left face, L_3 for right face, L_4 for front face (outer part), L_5 for back face (outer part), L_6 for left half of cylinder, L_7 for right half of cylinder, L_8 for top surface, H_9 for hole of front face, H_{10} for hole of back face. Space curves are line segments S_j, circular arcs C_j, NURBS curves N_j, where $1 \leq j \leq 18$. Surfaces with labels from 1 to 8 are, respectively, bottom planar facet, left planar facet, right planar facet, front planar facet, back planar facet, left half of circular cylinder, right half of circular cylinder, top NURBS surface. The NURBS surface is approximated by triangular facets in left view. The cylinder hole goes through front and back facets as seen in right view.

The vertices and hexahedral elements of 3-D mesh file `mesh3d/polyh2.mh3` and 3-D curve/surface file `region3d/polyh2.cs3` are illustrated in Figure B.4.

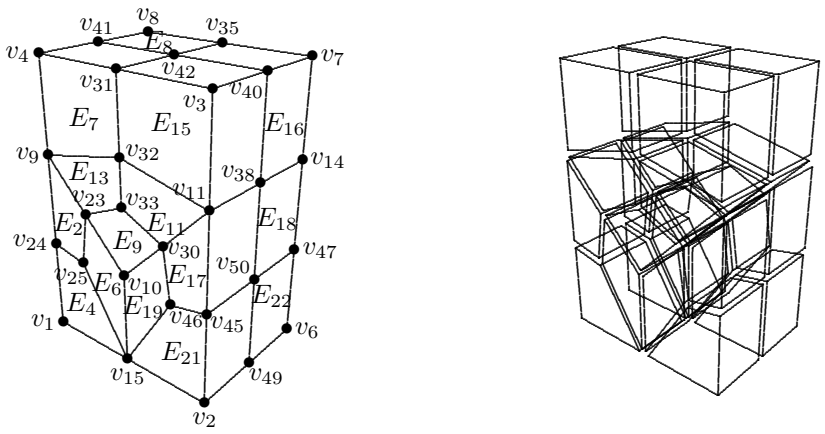

Figure B.4: Illustration of 3-D mesh file, with two views. Left view shows part of boundary of hexahedral mesh. Right view shows shrunken elements. Vertices are v_1 to v_{51} (not all visible). Hexahedral elements are E_1 to E_{22}. Hidden elements are E_1 (behind E_2), E_3 (behind E_4), E_5 (behind E_6), E_{10} (behind E_9), E_{12} (behind E_{11}), E_{14} (behind E_{13}), E_{20} (behind E_{19}).

The vertices and quadrilateral elements of surface mesh file `meshsurf/isonurb.mhs` and 3-D curve/surface file `region3d/isonurb.cs3` are illustrated in Figure B.5.

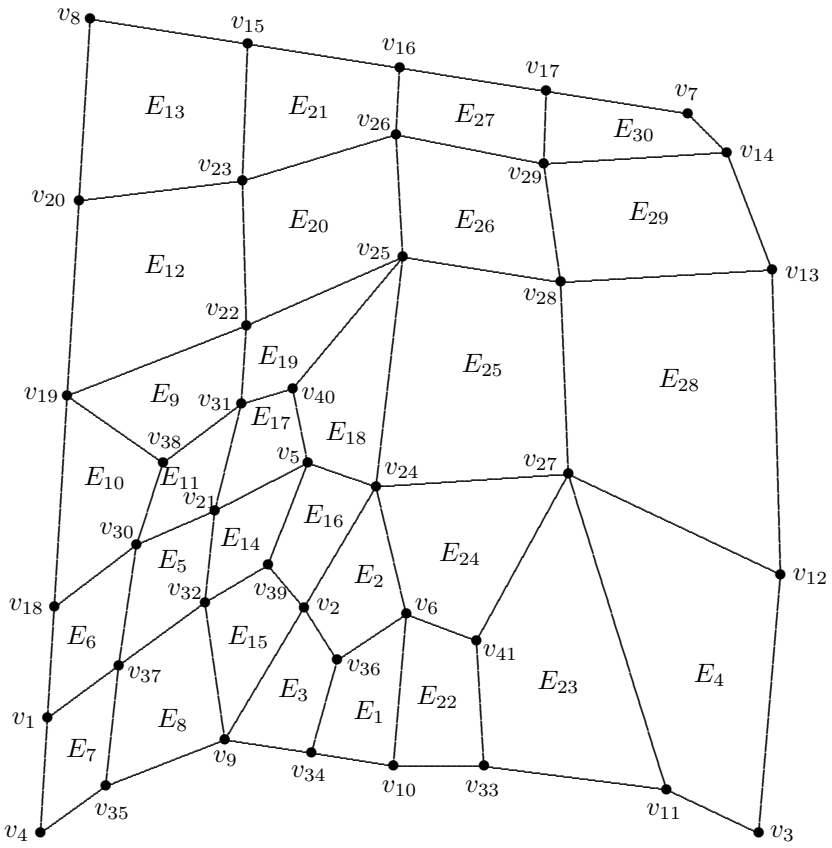

Figure B.5: Illustration of surface mesh file. Vertices are v_1 to v_{41}. Quadrilateral elements are E_1 to E_{30}.