```
  }

BaseGrid (<nb_entries>) {
        <e0> <e1> <e2>
  }

  CellTree (<nb_macro_elements>) {
   <index> <x-refinement> <y-refinement> <z-refinement> <nb_el> <sub_el0> <sub_el1> ...
   <sub_el_nb_el>
   ...
   ...
  }


Properties(<nb_macro_elements>) {
   <index> <nb_el> <prop_sub_el0> <prop_sub_el1> ... <prop_sub_el_nb_el>
   ...
   ...
  }

Region ("<region1>") {
   material = <material1>
        Edges    (<nb_edge>) { <edge0> <edge1> ... }
        Faces    (<nb_face>) { <face0> <face1> ... }
  }
Region ("<region2") {
   material = <material2>
        Vertices (<nb_vert>) { <vert0> <vert1> ... }
  }
Region ("<region3>") {
   material = <material3>
        Faces    (<nb_fac>)  { <fac0>  <fac1>  ... }
  }
Region ("<region4 >") {
   material = <material4>
        Elements (<nb_elts>) { <elt0>  <elt1>  ... }
  }

}
```

2.11 Boundary and grid

Boundary and grid files have a similar format. However, the interpretation of the contents is different. Each region must have one material name. The numbering of vertices, edges, and faces is given in Figure 6.6 on page 6.19 and Figure 6.7 on page 6.20. The second picture is only required when data must be stored on vertices or edges locally for each element.

An index to a face is signed. When negative, the real face index is computed by: $face\_index = -index-1$. The negative sign means that the edge order of the face as well as the edge orientation must be inverted. Face orientation is important to obtain a consistent orientation of the object (counterclockwise orientation of the edges when looking from outside of the solid including this face). An edge index is signed in the same way as a face index. A negative edge sign means that the first vertex and not the second is connected to the next edge of a loop.

NOTE The DF–ISE library provides a function to fix the orientation of the faces of a 3D boundary.

Elements have a shape code that defines the element geometry and the data following this code. Elements of different shape can be given in any order. The following elements are predefined:

Point	`0` *`vertex`*
Segment	`1` *`vertex0 vertex1`*
Triangle	`2` *`edge0 edge1 edge2`*
Rectangle	`3` *`edge0 edge1 edge2 edge3`*
Polygon	`4` *`nb_edges edge0 edge1...`*
Tetrahedron	`5` *`face0... face3`*
Pyramid	`6` *`face0... face4`*
Prism	`7` *`face0... face4`*
Brick	`8` *`face0... face5`*
Tetrabrick	`9` *`face0... face6`*
Polyhedron	`10` *`nb_faces face0 face1...`*

The edge and face lists of polygons and polyhedra do not have to be connected. They can be a set of loops and shells that define the outer boundary and internal holes of an element.

An edge orientation for each element is also predefined. The edge orientation is used when storing data on edges locally for each element. The vertex order for each edge of the elements are given below:

Segment:
- e0: v0 v1

Triangle:
- e0: v0 v1
- e1: v1 v2
- e2: v2 v0

Rectangle:
- e0: v0 v1
- e1: v1 v2
- e2: v2 v3
- e3: v3 v0

Polygon:
- e0: v0 v1
- e1: v1 v2
- …

Prism:
- e0: v0 v1
- e1: v1 v2
- e2: v2 v0
- e3: v0 v3
- e4: v1 v4
- e5: v2 v5
- e6: v3 v4
- e7: v4 v5
- e8: v5 v3

Tetrahedron:
- e0: v0 v1
- e1: v1 v2
- e2: v2 v0
- e3: v0 v3
- e4: v1 v3
- e5: v2 v3

Pyramid:
- e0: v0 v1
- e1: v1 v2
- e2: v2 v3
- e3: v3 v0
- e4: v0 v4
- e5: v1 v4
- e6: v2 v4
- e7: v3 v4

Brick:
- e0: v0 v1
- e1: v1 v2
- e2: v2 v3
- e3: v3 v0
- e4: v0 v4
- e5: v1 v5
- e6: v2 v6
- e7: v3 v7
- e8: v4 v5
- e9: v5 v6
- e10: v6 v7
- e11: v7 v4

Tetrabrick:

e0: v0 v1	e6: v2 v5
e1: v1 v2	e7: v2 v6
e2: v2 v3	e8: v3 v6
e3: v3 v0	e9: v4 v5
e4: v0 v4	e10: v5 v6
e5: v1 v5	e11: v6 v4

The order of appearance of `Region` blocks must correspond to the `regions` entry list of the `Info` block. Several regions with the same name are allowed.

```
DF-ISE text | binary

Info {
  version     = 1.0
  type        = boundary | grid
  dimension   = 1 | 2 | 3
  nb_vertices = <int>
  nb_edges    = <int>
  nb_faces    = <int>
  nb_elements = <int>
  nb_regions  = <int>
  regions     = [ "<region1>" "<region2>" ... ]
  materials   = [ <material1> <material2> ... ]
}
Data {

   CoordSystem {
       translate = [ <dx> <dy> <dz> ]
       transform = [ <xx> <xy> <xz> <yx> <yy> <yz> <zx> <zy> <zz> ]
  }
   for 1d, 2d, and 3d
   Vertices (<nb_vertices>) {
       <x0> <y0> <z0>
       <x1> <y1> <z1>
        ...
  }
   for 2d and 3d
   Edges (<nb_edges>) {
       <vertex0> <vertex1>
       <vertex0> <vertex1>
       ...
  }
   for 3d
   Faces (<nb_faces>) {
       <nb_edges> <edge0> <edge1> ... <edge_n>
       <nb_edges> <edge> <edge> ...
       ...
  }
   nb_vert. for 1d, nb_edges for 2d, nb_faces for 3d
   Locations (<nb_vertices> | <nb_edges> | <nb_faces>) {
       i (internal) | f (internal interface) |
       e (external interface) | u (undefined)
       ...
  }
Elements (<nb_elements>) {
        0 <vertex>
        1 <vertex0 <vertex1>
        2 <edge0 <edge1> <edge2>
        3 <edge0 <edge1> <edge2> <edge3>
        4 <nb_edges> <edge0> <edge1> ... <edge_n>
        5 <face0 ... <face3>
        6 <face0 ... <face4>
        7 <face0 ... <face4>
        8 <face0 ... <face5>
        9 <face0 ... <face6>
        10 <nb_faces> <face0> <face1> ... <face_n>
       ...
  }
Region ("<region1>") {
        material = <material1>
        Elements (<nb_elts>) { <elt0> <elt1> ... }
  }
Region ("<region2>") {
        material = <material2>
        Elements (<nb_elts>) { ... }
  }
}
```

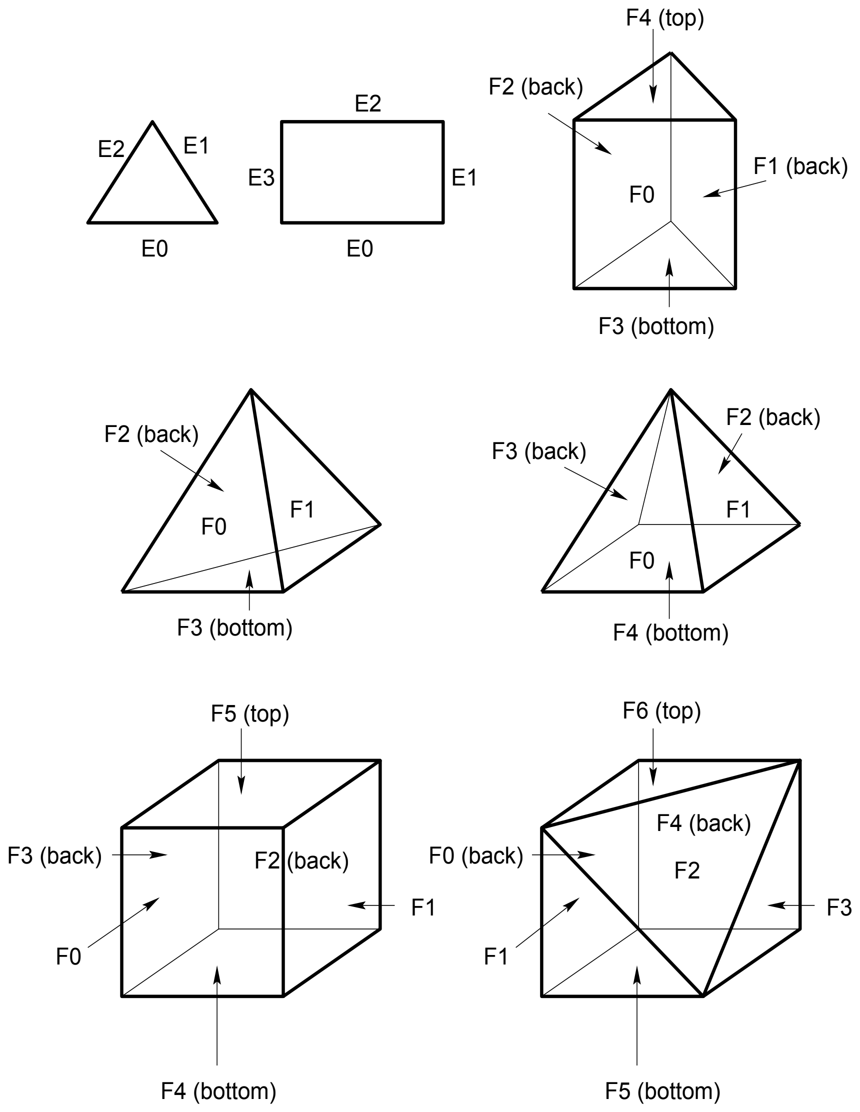

Figure 6.6 Edge and face numbering

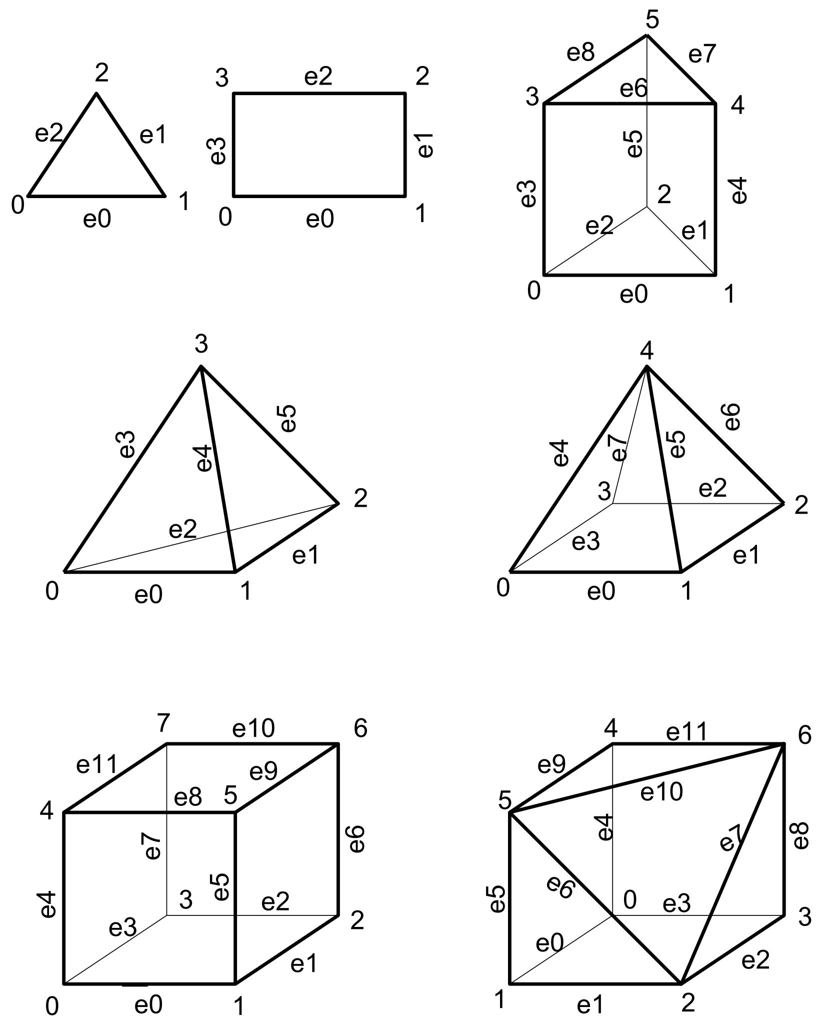

Figure 6.7 Edge and vertex numbering

2.12 Dataset

The `dataset` files add scalar or vector values to the nodes, edges, faces, elements, or regions of a `grid` file. The data *location* is currently limited to `vertex`, `edge`, `face`, `element`, or `region`. Two additional location codes are used to store Voronoï data: `vertex_element` and `edge_element`. These codes are only understood by DESSIS.

`dimension` gives the set number of data values per location (1 for scalars, > 1 for vectors). The `dimension` entry has a different meaning than the geometry dimension given in the `Info` block. The type of a dataset is currently either a `scalar` or `vector`. More types (`matrix`, `array`) will be added later.

`validity` gives the list of regions in which the dataset is defined. For consistency, data values should only be defined for the entries belonging to these validity regions.

NOTE The DF–ISE library provides some functions to access the values of a dataset for a given validity region. For the moment, this can only be used with datasets whose location is `vertex`.

Discontinuous datasets are a matter of interpretation of the data file format. Several datasets with the *same* quantity can be present in a single data file. Each instance of the dataset should be defined for a different set of regions (including the boundary points of a region). The entire dataset can, therefore, be discontinuous across boundaries. Because of the interpretational nature of this definition of discontinuous datasets, different tools may handle data files with such datasets differently. Refer to specific manuals for detailed descriptions of individual tools.

Data values are given in the order predefined by the `location` entry. When vectors are stored, all components are given per item (`vertex`, `edge`, ...) in the `Values` block.

The order of appearance of `Dataset` blocks must correspond to the `datasets` entry list of the `Info` block. Functions are references to the `function` blocks stored in the property file. Several datasets with the same name are allowed (with different validity regions, this may be used to define data discontinuities).

```
DF-ISE text | binary

Info {
  version     = 1.0
  type        = dataset
  dimension   = 1 | 2 | 3
  nb_vertices = <int>
  nb_edges    = <int>
  nb_faces    = <int>
  nb_elements = <int>
  nb_regions  = <int>
  datasets    = [ "<dataset1>" "<dataset2>" ... ]
  functions   = [ <function1> <function2> ... ]
}

Data {

  Dataset ("<dataset1>") {
        function  = <function1>
        type      = scalar | vector
        dimension = 1 (1 for scalar, > 1 for vectors or arrays)
        location  = vertex | edge | face | element | region
        validity  = [ "<region0>" "<region1>" ... ]
        Values (<nb_values>) { <value0> <value1> ... }
  }

  Dataset ("<dataset2>") {
        function  = <function2>
  ...
}
```

2.13 XYPlot

The `xyplot` files are used to store curves. The property list gives additional information for each data column of the file. The x-axis is not always in the first column, since the user can freely select the dataset mapped on the x-axis. For undefined values (discontinuities in the curve), the character ‘u’ is used as a placeholder:

```
DF-ISE text | binary

Info {
  version    = 1.0
  type       = xyplot
  datasets   = [ "<dataset1>" "<dataset2>" ... ]
  functions  = [ <function1> <function2> ... ]
}

Data {
  <v0_d1> <v0_d2> ... (one value for each data set)
  <v1_d1> <v1_d2> ... (one value for each data set)
  ...
}
```

2.14 Examples

This section provides examples of the DF–ISE file format.

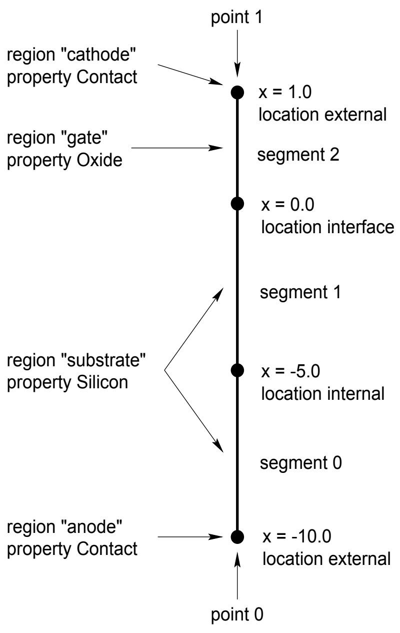

Figure 6.8 One-dimensional grid

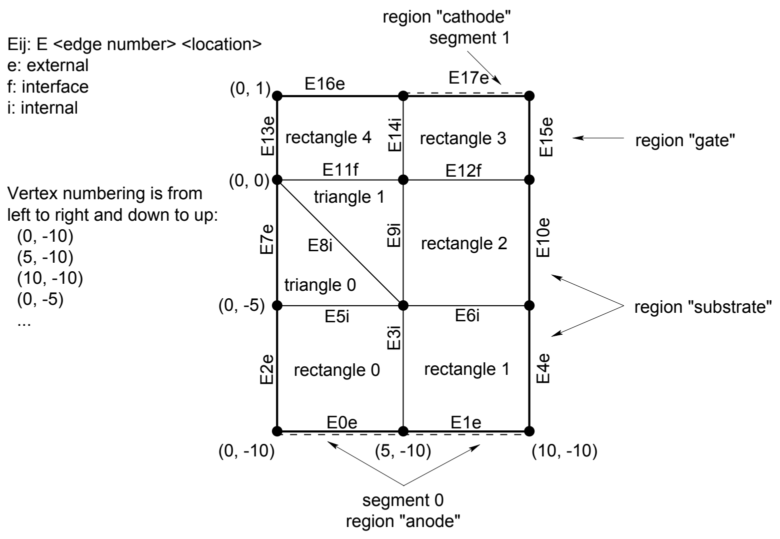

Figure 6.9 Two-dimensional grid

.

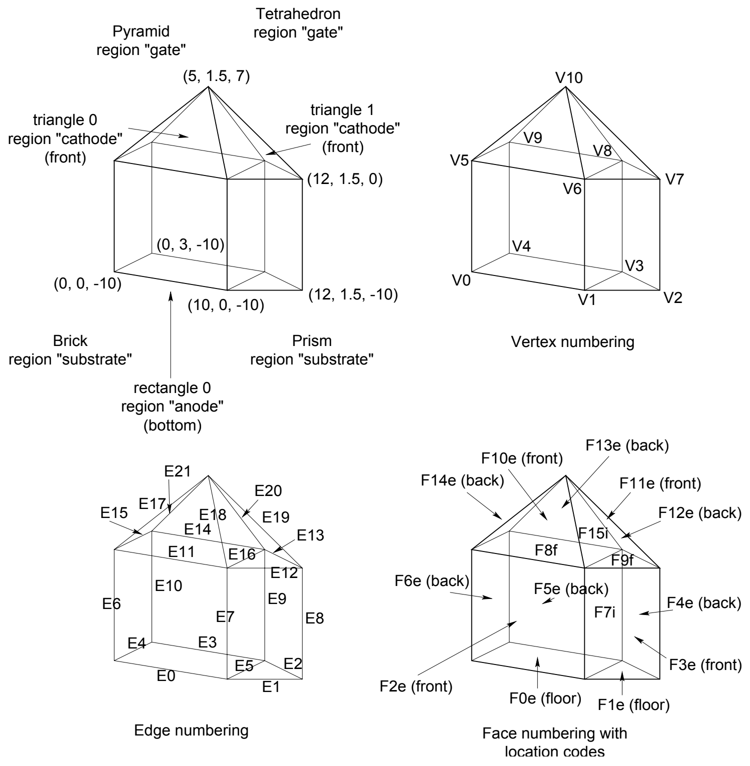

Figure 6.10 Three-dimensional grid