Triangulation of 3D Domains

User Guide

March 26, 2012

Daniel Rypl

Czech Technical University in Prague
Faculty of Civil Engineering, Department of Mechanics

Thákurova 7, 160 00, Prague
Czech Republic

e-mail: drypl@fsv.cvut.cz
http://mech.fsv.cvut.cz/~dr/dr.html

Contents

1 Model Representation **3**

2 Mesh Size Specification **4**

3 Model Input Data Format **5**
3.1 Input Record of a Vertex 6
3.2 Input Record of a Curve 7
3.3 Input Record of a Surface 9
3.4 Input Record of a Patch 10
3.5 Input Record of a Shell 11
3.6 Input Record of a Region 12

4 Mesh Size Input Data Format **12**

5 Background Mesh Input Data Format **13**

6 Mesh Output Format **14**

7 Command Line Options **19**

8 Terminal Output **25**

9 Quality Evaluation **27**

10 Warnings and Error Messages **27**

11 Input File Example **28**

Appendix **31**

A Modelling of Conics **31**
A.1 Circular Arc 31
A.1.1 Circular Arc - Quadratic Curve 31
A.1.2 Circular Arc - Cubic Curve 31
A.2 Elliptic Arc 32
A.2.1 Elliptic Arc - Quadratic Curve 32
A.2.2 Elliptic Arc - Cubic Curve 33
A.3 Parabolic Arc 33
A.3.1 Parabolic Arc - Quadratic Curve 33
A.3.2 Parabolic Arc - Cubic Curve 34
A.4 Hyperbolic Arc 34
A.4.1 Hyperbolic Arc - Quadratic Curve 34
A.4.2 Hyperbolic Arc - Cubic Curve 35

B Run-Time Visualization **36**

1 Model Representation

The model is described by a boundary representation and consists of the following model entities: vertices, curves, surfaces, patches, shells, and regions. Topologically, each region is formed by a set of not self-intersecting boundary surfaces, patches, and shells, each of which is bounded by a set of curves. Each curve is given by two end vertices. Moreover, each boundary surface, patch, and shell points out to the regions on the side of its outer and inner normal. A curve keeps list of surfaces, patches, and shells sharing that curve. Similarly, a vertex stores the list of curves sharing that vertex. This basic topology is further restricted by the geometry of model entities. Both curves and surfaces are based on free-form representation in terms of tensor-product polynomial entities. This limits the number of curves bounding a surface to four. The number of curves bounding a patch or a shell is not limited (but must be at least two). While the patch is a planar model entity (trimmed plane), shell is constrained to a background surface (trimmed surface). To enhance the modelling capability an entity-to-entity fixation concept has been introduced. Generally, each model entity may be fixed to another model entity of the same or higher dimension. However, the fixed entity is not allowed to coincide with the boundary of the parent entity. Each model entity keeps the list of entities fixed to it. No further topological information is required for the description of a valid non-manifold domain of almost arbitrary complexity. Currently, rational Bezier entities are employed for free-form curves and surfaces representation. This allows to represent exactly conics and quadrics by entities starting with an order of three (quadratic curves and biquadratic surfaces).

The rational Bezier curve has the form

$$\boldsymbol{P}(t) = \frac{\sum_{i=0}^{n} \omega_i \boldsymbol{P}_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)}, \tag{1}$$

where $\boldsymbol{P}(t)$ is the point on the curve, $\boldsymbol{P}_i$ are Bezier control points, ω_i are weights of Bezier control points, $B_i^n(t)$ stand for Bernstein polynomials, t denotes an independent variable varying in range from 0 to 1, and n is the curve degree. The curve order is equal to $n+1$. $\boldsymbol{P}_0$ and $\boldsymbol{P}_n$ correspond to model vertices while the remaining points form the control polygon of the curve. They determine the bow of the curve and need not generally lie on the curve. The first and last segments of the control polygon coincide with the curve tangent in the starting and ending vertices respectively.

The rational Bezier surface can be written in a similar form

$$\boldsymbol{P}(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} \boldsymbol{P}_{ij} B_i^n(u) B_j^m(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} B_i^n(u) B_j^m(v)}, \tag{2}$$

where $\boldsymbol{P}(u,v)$ is the point on the surface, $\boldsymbol{P}_{ij}$ are Bezier control points, ω_{ij} are weights of Bezier control points, $B_i^n(u)$ and $B_j^m(v)$ stand for Bernstein polynomials, u and v denote independent parameters varying in range from 0 to 1, and n and m are surface degrees (orders are equal to $n+1$ and $m+1$) in u and v parametric directions, respectively. If the control points are arranged in a matrix $(n+1) \times (m+1)$ then the corner points correspond

to model vertices, the side points correspond to control polygons of model curves bounding the surface, and the remaining points form the control polygon of the surface and need not generally lie on the surface.
Bernstein polynomial can be expressed as

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{3}$$

or recursively as

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t), \tag{4}$$

where $B_0^0 = 1$.

The ordinary Bezier entities can be derived from rational Bezier entities when all weights are set to 1.

Two types of model entities are distinguished. The physical ones which are designated for the actual discretization and the virtual ones which serve as auxiliary for geometry description or mesh size specification. Note that there are some restrictions on the fixation between virtual and physical entities.

2 Mesh Size Specification

Three levels of mesh size specification are considered

- the global mesh size specification,
- the local mesh size specification, and
- the adaptive mesh size specification.

The global mesh size specification uses global weight functions to control the mesh size description over the domain. The local mesh size specification concept prescribes the desired mesh size variation on model entities. In the adaptive mesh size specification strategy, various mesh size sources, built according to the preceding problem analysis, are used.

Currently only the local and adaptive mesh size specifications are implemented. The local mesh size specification consists of two parts

- the required mesh size specification and
- the curvature-based mesh size control.

The former concept is used to explicitly prescribe the mesh size at individual model entities. Mesh size specification is stored at each vertex and at each control point of any curve or surface. These values are used to extract the mesh size specification on a curve or surface. Moreover, each model entity (except vertices) stores an upper bound limit on mesh size which is not allowed to be exceeded.

Similar expressions to the ones describing the geometry of rational Bezier curves and surfaces are used to interpolate the local mesh size specification at control points over the curve and surface. The mesh size extracted from a curve mesh size specification has the form

$$msz(t) = \frac{\sum_{i=0}^{n} \omega_i msz_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)}, \tag{5}$$

where $msz(t)$ is the required mesh size at point $\boldsymbol{P}(t)$ on the curve, msz_i are the mesh size specifications at Bezier control points, and the other symbols have the same meaning as in Eq. (1). A similar formula can be written for the extraction of the required mesh size on a surface

$$msz(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} msz_{ij} B_i^n(u) B_j^m(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} B_i^n(u) B_j^m(v)}, \tag{6}$$

where $msz(u,v)$ is the required mesh size at point $\boldsymbol{P}(u,v)$ on the surface, msz_{ij} are the mesh size specifications at Bezier control points. The remaining variables have the the same meaning as in Eq. (2).

The latter concept is employed to enable an accurate representation of a curve or surface by its discretization even if no particular mesh size is required. The criterion is based on the ratio (curvature rate) between the appropriate mesh size and the radius of curvature at a given location on the curve or surface. The default ratio is equal to 1, which corresponds to the discretization of a circle to 6 segments of the same length equal to the radius of the circle. The curvature-based mesh size control is performed implicitly. Its suppression may resolve in an unrecoverable error during mesh generation.

The adaptive mesh size specification is based on a background mesh with the mesh size specification at nodes and linear interpolation over the elements (edges, triangles, and tetrahedrons).

3 Model Input Data Format

The input data consist of a set of keywords and appropriate numeral or literal values associated with them. The first keyword on the line is specific and is obligatory. The order of following keywords is more or less compulsory but must respect some build-in logic. A unique positive identification number has to be assigned to each model entity. This number is then used to reference this model entity. Note that the numbering of model entities of different type is independent. The individual model entities must be ordered in the input file in such a way that any referenced model entity must already exist. Keywords may be typed in upper case, lower case, or mixed case. Everything on a line behind a # sign is treated as a comment. Any number of blank spaces may be used between the keywords and numbers. Empty lines are ignored. Too long lines may be split using the backslash. Note that each part of the splitted line must be marked by the # sign to comment it thoroughly. No further input data formating is required.

Note that some features are enabled or disabled if the source code is compiled with certain directives. Keyword coincide in vertex, curve, surface, patch, or shell specification is relevant only in the case that the code has been compiled with the *T3D_COINCIDE* directive otherwise it is ignored.

The following notation will be used in the description of input records for individual model entities:

[ ] - optional parameter
{ } - obligatory parameter
( ) - repeated parameter
| - logical XOR
|| - logical OR
- number

3.1 Input Record of a Vertex

Vertex # {*xyz* # # # || (*fixed* {*vertex* # | *curve* # [*t* #] | *surface* # [*uv* # #] }) } \
[*size* {# | *def* | *uni* | *ver* | *cur* | *sur*} [* #]] [*weight* #] [*factor* #] \
[*property* #] [*virtual*] [*hidden*] [*output* {*yes* | *no*}] [*coincide vertex* (#)]

Vertex, as the compulsory keyword, is followed by its identification number. A vertex may be determined either by three coordinates preceded by keyword *xyz* or by fixation to either of vertex, curve, or surface. This is expressed by keyword *fixed* followed by the appropriate entity keyword (*vertex*, *curve*, or *surface*) and its identification number. When the fixation to curve or surface is used parametric coordinates preceded by keyword *t* in the case of curve or *uv* in the case of surface must be provided otherwise the parametric position of the vertex on the curve or surface is calculated using its coordinates specified after keyword *xyz* or inherited from the parent vertex. If the position of the vertex on a parent entity is overdetermined the redundant information is used to avoid ambiguity of the vertex position. Note that the original position (given by real coordinates) of the vertex and its position on the parent entity (given by parametric coordinates) are not allow to differ by more than user defined epsilon specified on the command line. A vertex may be fixed simultaneously to a vertex and curve or to a vertex and surface. In this case, repeated use of keyword *fixed* is allowed. A size specification may be assigned to a vertex. This can be done using keyword *size* followed by a concrete number or a special keyword, optionally followed by a multiplication factor preceded by an asterisk. Special keyword *def* stands for a default mesh size specified on the command line when running the program, *uni* denotes the uniform mesh size specified on the command line, *ver*, *cur*, and *sur* is used in the case of fixation when the mesh size is to be the same as on the appropriate parent model entity (vertex, curve, or surface) at vertex location. By default, a vertex fixed to a parent entity inherits mesh size from that entity. In the case of a multiple fixation to vertex and curve or surface the mesh size of parent curve or surface is used. If no size is specified and the vertex is not fixed to any entity the default size from command line is used. Only positive mesh size is allowed at vertices. Note that

the size assigned to a vertex is used for interpolation of mesh size over curves and surfaces sharing that vertex. The unit weight is assumed for all non-fixed vertices unless a weight is specified after keyword *weight*. Only positive weight specification is accepted. A vertex fixed to a parent entity inherits weight from that entity. Note that this inherited weight can be overridden by new weight specification (keyword *weight*) only in the case of vertex to vertex fixation. Keyword *factor* is used to specify the vertex mesh size multiplication factor. Note that this factor is not used when interpolating mesh size specification over curves and surfaces sharing the vertex. Setting it to zero will cause excluding this vertex from local mesh size control. Negative mesh size factors are not accepted. Default value of mesh size factor is equal to 1. An integer property number may be assigned to a vertex after keyword *property*. Keyword *virtual* marks a vertex as virtual. Keyword *hidden* marks it as a hidden one, which means that the location of the mesh node associated with this vertex is not fixed and may change during the smoothing process. The output of node generated at vertex position may be enforced or suppressed by setting *yes* or *no* after *output* keyword. By default, each physical non-hidden vertex is designated for output, except those fixed vertices with no physical ancestor designated for output. Suppressed output for a vertex bounding a physical curve or fixed to a physical curve designated for output is ignored. If generation of elements on a model entity between the vertex being just defined and other close vertices (already defined) bounding that entity is to be prevented, the other vertices should be specified after keywords *coincide vertex*. Neither from the coinciding vertices is allowed to be fixed to physical vertex or vertex with physical ancestor.

In the current implementation, the keyword *hidden* is ignored.

3.2 Input Record of a Curve

Curve # {*vertex* # #} [*order* # || *fixed* {*curve* # | *surface* #}] \
[*size* {# | *def* | *uni* | *cur* | *sur*} [* #]] [*rate* {# | *def*} [* #]] [*factor* #] \
[*property* #] [*virtual*] [*hidden*] [*count* #] [*output* {*yes* | *no*}] \
[*polysize* ([* #] {# | *def* | *uni*})] [*coincide curve* (#)] \
[*density* {# | *def*} [* #]] [*bassoc* {*yes* | *no*}]

Polygon # {*xyz* # # # | *poly* #} [*size* {# | *def* | *uni* | *cur*} [* #]] [*weight* #]

Curve, as the compulsory keyword, is followed by its identification number. If the curve order is not provided and the curve is not fixed to any entity the default value 2 is used as the order and no consequent polygon records are expected. When the curve is fixed to an entity, the curve order may but needs not be specified. In the former case, curve order must be greater or equal to the order of the appropriate parent entity or set to zero (the order of the parent entity is used in that case). Identification numbers of the starting and ending vertices are specified after keyword *vertex* and determine the curve orientation. When the curve is fixed to a surface its end vertices must lie on the same parametric curve of the surface (the user supplied epsilon is used as a tolerance) and must be fixed to surface itself or to a curve top parent of which is fixed to the surface. A physical curve fixed to a physical solid surface is not allowed to be intersected by another physical curve fixed to the same

surface. The orientation of the fixed curve is independent of the orientation of the parent model entity. The meaning of keywords *property* and *virtual* is the same as explained in Section 3.1. The size specification preceded by keyword *size* is also similar to the one in Section 3.1 but two differences should be mentioned. Firstly, no default value is used when the size specification is missing and secondly the size specification (if provided) is treated as an upper bound of the required size extracted from the mesh size specification of curve control points. For all model entities the (nonzero) size is recursively propagated to boundary entities of the next lower dimension untill a smaller (nonzero) size is encountered. Keyword *density* determines the upper bound of the mesh size as the ratio between the length of the curve and the density which is specified as a concrete number or as default value (controlled by the command line option) using a special keyword *def*, optionally followed by a multiplication factor preceded by an asterisk. The value of the density has no upper bound limit and must not be negative. If the mesh size is specified in multiple ways, the smaller nonzero value is considered. Keyword *factor* specifies the curve mesh size multiplication factor which is applied to both curve mesh size specification and mesh size extracted from curve control points as well. Keyword *hidden* marks curve as a hidden one which allows the repositioning of mesh elements and nodes of this curve during the smoothing process without respecting geometrical restriction of their fixation to this curve. Keyword *rate* is used to define the accuracy of the geometrical representation of the curve by its discretization in terms of a ratio between the appropriate mesh size and radius of curvature at any location on the curve. The rate is specified as a concrete number or as default value (controlled by the command line option) using a special keyword *def*, optionally followed by a multiplication factor preceded by an asterisk. The value of the rate has no upper bound limit and must not be negative. The *polysize* keyword is used for fixed curves only if the inheritance of mesh size specification from the parent model entity is not desirable. In that case, the size specification must be provided for each internal control point in subsequent order. The repetition number preceded by an asterisk may be used if the specification is the same for several subsequent control points. The meaning of keywords *def* and *uni* is the same as described in Section 3.1. Keyword *count* can be used to enforce a regular division of the curve to a specified number of segments. Setting this number to zero disables regular division. The number of segments may be also controlled by size specification (acting as an upper bound limit on segment size). In this case, count or size specification that yields a larger number of segments is used. Note that on curves of order larger than 2 the regularity of the division can be disrupted due to the eventual curvature-enforced refinement possibly also leading to larger number of segments. The count specification is reflected only by physical curves not bounding any physical solid surface, patch, or shell. The output of segments on the curve may be enforced or suppressed by setting *yes* or *no* after *output* keyword. By default, curves bounding a physical solid surface, patch, or shell are not designated for output. A curve may be degenerated into a single point (collapsed curve) if it is bounded by a single vertex and all its control points are merged at this vertex. Collapsed curve is not allowed to be fixed to a curve or surface. Similarly as in Section 3.1, if generation of elements on a model entity between the curve just being defined and other close curves (already defined) bounding that entity is to be prevented, the other curves should be specified after keywords *coincide curve*. Neither from the coinciding curves is allowed to be fixed to physical curve or curve with physical ancestor. Keyword *bassoc* followed by specification *yes* or *no* controls separate output of triangles

(ids) classified on surfaces, patches, and shells bounded by the curve. Note that a specific command line option controls whether all or none of the physical curves will be by default designated for output of associated triangles.

A set of internal control points must be provided for each non-fixed curve of degree $n > 1$. It means that input record of such a curve must be followed by $n - 1$ records for each internal control point of the curve. The number of the control point in the range from 1 to $n - 1$ specified after keyword *Polygon* is used as the control point sequence number with respect to the curve orientation and can be referenced only in the context of the current curve. The control point may be specified by three coordinates preceded by keyword *xyz* or may be associated with any already specified polygon control point using keyword *poly* followed by the sequence number of that control point. Keywords *size* and *weight* have a similar meaning to the one explained in Section 3.1 but even negative values of size and weight specification can be used. Note however that the denominator in Eqs (1), (2), (5), and (6) must not be negative or zero.

In the current implementation, keyword *hidden* is ignored, unless only two planar model entities (physical, solid, non-hidden surface, patches, and shells) with the same property and with collinear normals of the same orientation are sharing the given curve with no physical parent or child. Note that using hidden curves has an impact on the performance because the surfaces, patches, and shells sharing those curves are subjected to an additional smoothing.

3.3 Input Record of a Surface

Surface # {*curve* # # # #} [*order* # # || *fixed surface* #] \
[*size* {# | *def* | *uni* | *sur*} [* #]] [*rate* {# | *def*} [* #]] [*factor* #] \
[*property* #] [*virtual*] [*hidden*] [*hole*] [*output* {*yes* | *no*}] \
[*polysize* ([* #] {# | *def* | *uni*})] [*bassoc* {*yes* | *no*}] \
[*coincide* {*surface* (#) | *patch* (#) | *shell* (#)}]

Polygon # # {*xyz* # # # | *poly* # #} [*size* {# | *def* | *uni* | *sur*} [* #]] [*weight* #]

Surface, as the compulsory keyword, is followed by its identification number. If surface orders are not provided and the surface is not fixed to any surface the orders are extracted from the curves bounding the surface. When the surface is fixed to a surface, the orders may but need not be specified. In the former case, order in any direction must be greater or equal to the order of the parent surface or set to zero (the order of the parent surface is used in that case). Identification numbers of the four bounding curves in clockwise order with respect to the surface outer side (normal) are specified after keyword *curve*. The first and second curves determine the u and v directions, respectively, on the surface with the origin at their common vertex. The surface outer normal is then given by the vector product of vectors tangent to the parametric curves in u and v directions and oriented in the positive u and v directions, respectively. The orientation of bounding curves is not relevant. No two adjacent curves are allowed to be degenerated into a single point and no two opposite or adjacent curves are allowed to be the same or coinciding. This disables creation of surfaces

degenerated into a curve or point. When the surface is fixed to a surface its bounding curves must be fixed to the parent surface itself. The ordering of bounding curves of the fixed surface must be in agreement with bounding curves ordering of the parent surface. Since physical curves fixed to a physical solid surface are not allowed to intersect each other, the physical surfaces fixed to the same physical solid surface cannot overlap. The meaning of keywords *size*, *factor*, *rate*, *property*, *virtual*, and *hidden* is the same as explained in Section 3.2. It should be mentioned however that the smallest of both principal radii of curvature on the surface is considered when the rate is specified. Only surface shared by two solid physical regions of the same property may be hidden. Also note that for surfaces and curves with nonzero size the (nonzero) value of factor is recursively propagated to the boundary entities of the next lower dimension untill a smaller (nonzero) factor is encountered. Keyword *hole* is used to specify that the currently defined surface forms a hole. The output of triangles on a surface may be enforced or suppressed by setting *yes* or *no* after *output* keyword. By default, surfaces bounding a physical solid region are not designated for output. Also the meaning of keyword *polysize* is the same as explained in Section 3.2. Note that the control points on a surface are ordered in such a way that the first index (corresponding to u direction) is running faster. Similarly as in Section 3.1, if generation of elements on a model entity between the surface just being defined and other close surfaces, patches, or shells (already defined) bounding that entity is to be prevented, the other surfaces, patches, or shells should be specified after appropriate model entity keyword (*surface*, *patch*, or *shell*) preceded by keyword *coincide*. Keyword *bassoc* followed by setting *yes* or *no* controls separate output of tetrahedrons (ids) classified to region bounded by the surface. Note that a specific command line option controls whether all or none of the physical surfaces will be by default designated for output of associated tetrahedrons.

A set of internal control points must be provided for each non-fixed surface of degrees greater than 1 in both directions. It means that input record of such a surface of degree $n \times m$ must be followed by $(n-1)(m-1)$ records for each internal control point of the surface. The numbers of the control point (in u direction ranging from 1 to $n-1$ and in v direction ranging from 1 to $m-1$) specified after keyword *Polygon* are used as the control point sequence numbers and can be referenced only in the context of the current surface. The control point may be specified by three coordinates preceded by keyword *xyz* or may be associated with any already specified polygon control point using keyword *poly* followed by the sequence numbers of that control point. Keywords *size* and *weight* have the same meaning as explained in Section 3.2.

Planar surface must be convex. Note that using hidden surfaces has an impact on the performance because the regions sharing those surfaces are subjected to an additional smoothing.

3.4 Input Record of a Patch

Patch # {*normal* # # # (*boundary curve* (#)) } [*origin* # # #] [(*subpatch* (#))] \
[(*fixed* {*vertex* (#) | *curve* (#) })] [*size* {# | *def* | *uni*} [* #]] \
[*factor* #] [*property* #] [*hidden*] [*hole*] [*output* {*yes* | *no*}] [*epsilon* #] \
[*coincide* {*surface* (#) | *patch* (#) | *shell* (#) }] [*bassoc* {*yes* | *no*}]

Patch, as the compulsory keyword, is followed by its identification number. The outer normal vector of the patch plane is specified after keyword *normal*. The position of the plane may be specified after keyword *origin*. Otherwise the plane is located in such a way that the plane is exactly in the middle of the extent in the normal direction given by all boundary and fixed vertices of the patch. The planarity of the patch is checked for all boundary and fixed vertices and curves of the patch as the distance from the patch plane. Should this distance exceed a user given tolerance the execution is premeturaly terminated. The tolerance may be specified after keyword *epsilon*, otherwise the global user defined epsilon given by a command line option (see *–e* option in Section 7) or its default value is used. The list of signed identification numbers of curves bounding the patch (from outside or inside) is preceded by keywords *boundary curve*. The positive number indicates the anticlockwise orientation of the curve (given by its starting and ending vertex) with respect to the patch when viewing it against the outer normal. The negative number indicates the clockwise orientation of the curve. The sign of identification number of collapsed curve forming the boundary of the patch is irrelevant (as the curve itself). The list of identification numbers of patches surrounded by the currently defined patch may be specified after keyword *subpatch*. Each subpatch must be coplanar with the patch, however its normal orientation is arbitrary. If a subpatch is touching the patch along its boundary curve, this curve must be specified also in the list of curves bounding the patch (including proper orientation). Furthermore, identification numbers of vertices and curves inside the patch can be enumerated after appropriate model entity keyword (*vertex* or *curve*) preceded by keyword *fixed*. The meaning of keywords *size*, *property*, and *hidden* is the same as explained in Section 3.2. Only patch shared by two solid physical regions of the same property is allowed to be hidden. The factor specification preceded by keyword *factor* is effective only if size specification is provided. The meaning of keywords *hole*, *output*, *coincide*, and *bassoc* is the same as described in Section 3.3.

Note that using hidden patches has an impact on the performance because the regions sharing those patches are subjected to an additional smoothing. Note also, that all the generated nodes classified to the patch are exactly in the patch plane, independently of the position of boundary nodes deviating eventually from the plane (by not more than the user prescribed tolerance).

3.5 Input Record of a Shell

Shell # {*bgsurface* # (*boundary curve* (#))} [(*subshell* (#))] \
[(*fixed* {*vertex* (#) | *curve* (#) })] [*size* {# | *def* | *uni*} [* #]] [*factor* #] \
[*rate* {# | *def*} [* #]] [*property* #] [*hidden*] [*hole*] [*output* {*yes* | *no*}] \
[*coincide* {*surface* (#) | *patch* (#) | *shell* (#) }] [*bassoc* {*yes* | *no*}]

Shell, as the compulsory keyword, is followed by its identification number. The background surface of the shell is specified after keyword *bgsurface*. Note that the background surface is also part of the model and will be discretized unless marked as hole or virtual. The list of signed identification numbers of curves bounding the shell (from outside or inside) is preceded by keywords *boundary curve*. The positive number indicates the anticlockwise orientation of the curve (given by its starting and ending vertex) with respect to the shell when viewing

it against the outer normal. The negative number indicates the clockwise orientation of the curve. The sign of identification number of collapsed curve forming the boundary of the shell is irrelevant (as the curve itself). Orientation of the outer normal vector of the shell is given by the background surface. The list of identification numbers of shells surrounded by the currently defined shell may be specified after keyword *subshell*. Furthermore, identification numbers of vertices and curves inside the shell can be enumerated after appropriate model entity keyword (*vertex* or *curve*) preceded by keyword *fixed*. The meaning of keywords *size*, *rate*, *property*, and *hidden* is the same as explained in Section 3.3. Only shell shared by two solid physical regions of the same property is allowed to be hidden. The factor specification preceded by keyword *factor* is effective only if size specification is provided. The meaning of keywords *hole*, *output*, *coincide*, and *bassoc* is the same as described in Section 3.3.

Note that using hidden shells has an impact on the performance because the regions sharing those shells are subjected to an additional smoothing.

3.6 Input Record of a Region

Region # {(*boundary* {*surface* (#) | *patch* (#) | *shell* (#)})} [(*subregion* (#))] \
[(*fixed* {*vertex* (#) | *curve* (#) | *surface* (#) | *patch* (#) | *shell* (#)})] \
[*size* {# | *def* | *uni*} [* #]] [*factor* #] [*property* #] [*hole*] [*output* {*yes* | *no*}]

Region, as the compulsory keyword, is followed by its identification number. The list of signed identification numbers of surfaces, patches, and shells bounding the region is specified after appropriate model entity keyword (*surface*, *patch*, or *shell*) preceded by keyword *boundary*. The positive number means that the outer normal of the surface, patch, or shell points out of the region, the negative number indicates that the outer normal points inside the region. The list of identification numbers of regions surrounded by the region may be specified after keyword *subregion*. If a subregion is touching the region along its boundary surface, patch, or shell, this boundary entity must be specified also in the appropriate list of entities bounding the region (including proper orientation). Furthermore, the identification numbers of vertices, curves, surfaces, patches, and shells inside the region can be enumerated after appropriate model entity keyword (*vertex*, *curve*, *surface*, *patch*, or *shell*) preceded by keyword *fixed*. The meaning of keywords *size*, *factor*, and *property* is the same as in Section 3.4. Keyword *hole* indicates that currently defined region is a hole. The output of tetrahedrons in region may be enforced or suppressed by setting *yes* or *no* after *output* keyword. By default, each physical solid region is designated for output.

4 Mesh Size Input Data Format

The format of the local mesh size control which is part of model description has been explained in Section 3.

The adaptive mesh size control is based on a background mesh. The required mesh size is specified at nodes of this mesh and is interpolated over the individual elements of the

background mesh. The following elements can form the background mesh

- linear edge (2 nodes),
- linear triangle (3 nodes) and
- linear tetrahedron (4 nodes).

If quadratic elements are used, the midside nodes are ignored. The background mesh is generally independent of the model being discretized but it can be (more likely) tightly connected to it. It is usually identical with the mesh generated during the previous step of an adaptive analysis. The adaptive mesh size input data file is provided by the application analysis code but its manual creation (as an startup mesh size control) is also possible. The description of the background mesh input data format is provided in the following section.

5 Background Mesh Input Data Format

There are no keywords in the background mesh input data file (the keywords used in following subsection are just for better understanding). Everything on a line behind a # sign is treated as a comment. Any number of blank spaces may be used between individual numbers. Empty lines are ignored.

The first record contains the mesh type (only simplicial meshes are supported), element degree (only linear elements are considered, midside nodes of quadratic elements are ignored), and the overall numbers of background nodes, edges, triangles, and tetrahedrons.

mesh_type elem_degree
nodes edges trias tetras

For each background node, the following record describing the position of the node and associated non-negative mesh size is expected. Eventual zero mesh size specification is treated as size corresponding to the model extent.

node_id coord_x coord_y coord_z mesh_size

Background edges, triangles, and tetrahedrons are described by the following records

edge_id nd1_id nd2_id [mid_nds_ids]
tria_id nd1_id nd2_id nd3_id [mid_nds_ids]
tetra_id nd1_id nd2_id nd3_id nd4_id [mid_nds_ids]

which are repeated for each background edge, triangle, and tetrahedron. Note that the numberings of background nodes, edges, triangles, and tetrahedrons are independent always starting from one. The nodes, edges, triangles, and tetrahedrons do not have to be ordered according to their identification number, however, the ordering nodes – edges – trias – tetras is obligatory.

6 Mesh Output Format

The output of the mesh data is formatted in such a way that it can be read by another application either in a fixed or free format. There are no keywords in the mesh output format (the keywords used in the following subsection are just for better understanding). The output is organized in blocks separated by a free line (if there is no output in a particular block the corresponding free line is omitted as well). The corresponding fixed format in Fortran style is enclosed in parenthesis at the end of the same line. If the description of a particular record of output format is too long it is split to several lines. In that case the corresponding format in Fortran style is split equivalently. The actual number of items on output is provided in brackets if necessary. Since the actual output is controlled by a command line option (see *–p* option in Section 7), items of the output format which are dependent on that option are enclosed in brackets. If some of them are not requested for the output the corresponding part of the format should be skipped. Note that some features of the output are enabled or disabled if the source code is compiled with certain directives. Output of neighbouring elements is supported only if the code has been compiled with the *T3D_OUTPUT_NEIGHBOUR* directive.

Block No 1 contains mesh type (3 - for the current implementation), element degree (1 - linear, 2 - quadratic), applied renumbering type (0 - none renumbering, 1 - node renumbering, 2 - element renumbering), and output type (see *–p* option in Section 7) followed by the number of nodes, edges, triangles, and tetrahedrons.

elem_type elem_degree renum_type output_type	(4I10)
nodes edges trias tetras	(4I10)

Block No 2 consists of records for each node in one of the following formats depending on the node classification to the parent model entity (note that mesh entities are always classified to the non-hidden model entity of the lowest possible dimension).

- vertex (entity type = 1)

node_id coords[3] ent_type ent_id ent_prop	(I10, 3F15.6, I5, I10, I10)

- curve (entity type = 2)

node_id coords[3] ent_type ent_id ent_prop	(I10, 3F15.6, I5, I10, I10,
[tangent[3]] [par_coords[1]]	[5X, 3F10.6], [5X, F10.6])

- surface (entity type = 3)

node_id coords[3] ent_type ent_id ent_prop	(I10, 3F15.6, I5, I10, I10,
[normal[3]] [par_coords[2]]	[5X, 3F10.6], [5X, 2F10.6])

- region (entity type = 4)

node_id coords[3] ent_type ent_id ent_prop	(I10, 3F15.6, I5, I10, I10)

• patch (entity type = 5)

node_id coords[3] ent_type ent_id ent_prop *[normal[3]] [par_coords[2]]*	(I10, 3F15.6, I5, I10, I10, [5X, 3F10.6], [5X, 2F10.6])

• shell (entity type = 6)

node_id coords[3] ent_type ent_id ent_prop *[normal[3]] [par_coords[2]]*	(I10, 3F15.6, I5, I10, I10, [5X, 3F10.6], [5X, 2F10.6])

Note that parametric coordinates are not available for midside nodes of quadratic elements classified to non-linear curves or non-planar surfaces and shells if boundary non-conforming elements are generated, and therefore zeros are used in that case. The tangent and normal are provided as if boundary conforming elements would have been generated. Also note that nodes classified to a vertex or curve are always classified to the most top physical parent vertex or curve, respectively. The only exception are the nodes on hidden curves, surfaces, patches, and shells when generating linear elements, in which case the nodes are classified to the appropriate from the two model entities of the next higher dimension between which the original model entity is hidden. Note that this dynamic reclassification is not performed when quadratic elements are required. The parametric coordinates of a node on a patch refers to a fictitious rectangular bilinear planar surface not defined in the input model (and are provided just for completeness). The tangent, normal, and parametric coordinates in block No 2 are provided only if appropriate output specification (*–p* option) has been used.

Important note: In the case of quadratic elements (see option *–k*), output of tangent, normal, and parametric coordinates is suppressed for the whole mesh if a hidden entity on input was detected (regardless whether the hiding request was later ignored for some reason).

Block No 3 consists of records for each edge in one of the following formats depending on element type (linear or quadratic). Entity type is equal to 2 (curve).

• linear edge

edge_id node_id[2] ent_type ent_id ent_prop	(I10, 2I10, I5, I10, I10)

• quadratic edge

edge_id node_id[3] ent_type ent_id ent_prop	(I10, 3I10, I5, I10, I10)

Note that edges classified to a curve are always classified to the most top physical parent curve. The node numbering of an edge element is provided in Figure 1.

Figure 1: Node numbering of an edge element.

Block No 4 consists of records for each triangle in one of the following formats depending on element type (linear or quadratic). the entity type is equal to 3, 5, or 6 according to the classification of the triangle to a surface, patch, or shell model entity, respectively.

• linear triangle

tria_id node_id[3] ent_type ent_id ent_prop	(I10, 3I10, I5, I10, I10,
[ngb_tria_id[3]] [bnd_curve_id[3] bnd_curve_prop[3]]	[3I10], [6I10])

• quadratic triangle

tria_id node_id[6] ent_type ent_id ent_prop	(I10, 6I10, I5, I10, I10,
[ngb_tria_id[3]] [bnd_curve_id[3] bnd_curve_prop[3]]	[3I10], [6I10])

The bounding curve ids (zero means no boundary curve) are provided to simplify the boundary condition specification. The node and edge numbering of a triangular element is depicted in Figure 2. Note that the numbering of neighbouring triangles is in agreement with the edge numbering.

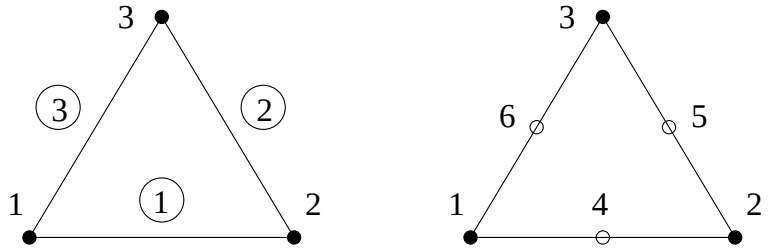

Figure 2: Node and edge numbering of a triangular element.

Block No 5 consists of records for each tetrahedron in one of the following formats depending on element type (linear or quadratic). Entity type is equal to 4 (region).

• linear tetrahedron

tetra_id node_id[4] ent_type ent_id ent_prop	(I10, 4I10, I5, I10, I10,
[ngb_tetra_id[4]]	[4I10]
[bnd_ent_id[4] bnd_ent_type[4] bnd_ent_prop[4]]	[4I10, 4I5, 4I10])
[bnd_curve_id(6) bnd_curve_prop(6)]	[6I10, 6I10])

• quadratic tetrahedron

tetra_id node_id[10] ent_type ent_id ent_prop	(I10, 10I10, I5, I10, I10,
[ngb_tetra_id[4]]	[4I10]
[bnd_ent_id[4] bnd_ent_type[4] bnd_ent_prop[4]]	[4I10, 4I5, 4I10])
[bnd_curve_id(6) bnd_curve_prop(6)]	[6I10, 6I10])

The bounding model entity ids are provided to simplify the boundary condition specification. If there is no boundary model entity/curve on a particular face of a tetrahedron then corresponding boundary entity/curve id and boundary entity type are equal to zero. The node and face numbering of a tetrahedral element is displayed in Figure 3. Note that the numbering of neighbouring tetrahedrons is done with respect to the face numbering.

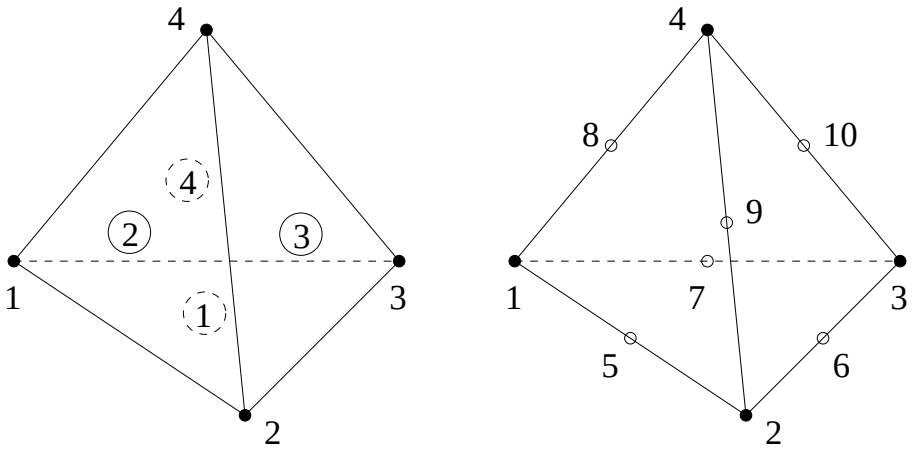

Figure 3: Node and face numbering of a tetrahedral element.

Note that there is only one global element numbering (it means that different types of elements are not numbered independently). The output of neighbouring elements and boundary entities in blocks No 4 and 5 are provided only if appropriate output specification (*–p* option) has been used. Also note that the output of the following blocks (blocks No. 6, 7, 8, 9, and 10) is dependent on the *–p* option in the similar way as was the case for the previous blocks. The actual form of the output may be checked by executing the program with *–F* option and appropriate *–p* option.

Block No 6 consists of the number of curves, surfaces, patches, and shells for which the list of associated elements is provided.

[curves surfaces patches shells] ([4I10])

Block No 7 contains curve entity type (equal to 2), curve id, and number of edges on the curve. This first record is followed by records (for each of the curve edges) containing number of triangles connected to the edge and list of ids of these connected triangles. The order in which these records appear on the output is identical with the edge ordering if the curve would have been designated for output. The order of connected triangles is not defined. If the parent entity of the connected triangle has not been designated for the output the triangle does not appear in the list of connected triangles. If neither from the 2D model entities connected to the curve is designated for the output, the curve does not appear in this block.

[curve_ent_type curve_id curve_prop edges] ([4I10])

[trias trias_ids[trias]] ([I10,[trias]I10])

Block No 7 is repeated for all physical curves for which the output of associated triangles has not been disabled on the input (see keyword *bassoc* in Section 3.2) or during run-time and the total number of which is given in block No 6.

Block No 8 contains surface entity type (equal to 3), surface id, and number of triangles on the surface. This first record is followed by records (for each of the surface triangles) containing number of tetrahedrons connected to the triangle and list of ids of these connected tetrahedrons. The order in which these records appear on the output is identical with the triangle ordering if the surface would have been designated for output. The order of connected tetrahedrons is not defined. If the region of the connected tetrahedron has not been designated for the output the tetrahedron does not appear in the list of connected tetrahedrons. If neither from the regions bounded by the surface is designated for the output, the surface does not appear in this block.

[surface_ent_type surface_id surface_prop trias] ([4I10])
[tetras tetras_ids[tetras]] ([I10,[tetras]I10])

Block No 8 is repeated for all physical solid surfaces for which the output of associated tetrahedrons has not been disabled on the input (see keyword *bassoc* in Section 3.3) or during run-time and the total number of which is given in block No 6.

Block No 9 contains patch entity type (equal to 5), patch id, and number of triangles on the patch. This first record is followed by records (for each of the patch triangles) containing number of tetrahedrons connected to the triangle and list of ids of these connected tetrahedrons. The order in which these records appear on the output is identical with the triangle ordering if the patch would have been designated for output. The order of connected tetrahedrons is not defined. If the region of the connected tetrahedron has not been designated for the output the tetrahedron does not appear in the list of connected tetrahedrons. If neither from the regions bounded by the patch is designated for the output, the patch does not appear in this block.

[patch_ent_type patch_id patch_prop trias] ([4I10])
[tetras tetras_ids[tetras]] ([I10,[tetras]I10])

Block No 9 is repeated for all physical solid patches for which the output of associated tetrahedrons has not been disabled on the input (see keyword *bassoc* in Section 3.4) or during run-time and the total number of which is given in block No 6.

Block No 10 contains shell entity type (equal to 6), shell id, and number of triangles on the shell. This first record is followed by records (for each of the shell triangles) containing number of tetrahedrons connected to the triangle and list of ids of these connected tetrahedrons. The order in which these records appear on the output is identical with the triangle ordering if the shell would have been designated for output. The order of connected tetrahedrons

is not defined. If the region of the connected tetrahedron has not been designated for the output the tetrahedron does not appear in the list of connected tetrahedrons. If neither from the regions bounded by the shell is designated for the output, the shell does not appear in this block.

[shell_ent_type shell_id shell_prop trias] ([4I10])
[tetras tetras_ids[tetras]] ([I10,[tetras]I10])

Block No 10 is repeated for all physical solid shells for which the output of associated tetrahedrons has not been disabled on the input (see keyword *bassoc* in Section 3.5) or during run-time and the total number of which is given in block No 6.

7 Command Line Options

The program may be executed by typing

t3d [option [parameter]] ...

on the command line prompt with the following command line options:

- *–h* - displays program usage description
- *–P* - displays program command line parameters description
- *–F* - displays format of input file, output file, and background mesh file;
  note that the description of output file format takes into account the actually applied *–p* and *–k* options; the other options are ignored
- *–Q* - silent mode (all messages to standard error channel are discarded)
- *–W* - suppresses warning messages
- *–V* - includes virtual entities into the domain extent
- *–C* - enables curvature violation on surface and shell; (triangles with node normals differing by more than 120° are allowed)
- *–B* - generates boundary conforming elements;
  this option is relevant only for quadratic elements (see option *–k*)
- *–A* - disables default designation of boundary model entities (curves, surfaces, patches, and shells) for output of boundary associated elements (triangles and tetrahedrons);
  this option is relevant only if output of boundary associated elements is required (see option *–p*)
- *–S* - disables nodal smoothing for surface, patch, shell, and region triangulation;
  note that this option prevents removing slivers from region triangulation
- *–i* - (string) input model file name;
  if this option is not present an error occurs

–o - (string) output mesh file name;
the output is realized only if this option is used

–g - (string) output log file name;
standard error channel is used as default

–m - (string) input background mesh file name

–x - (string) output graphics (Elixir) file name

–n - (6x int #) number of model entities;
this option specifies the estimate on number of individual entities (including virtual ones) in model representation; the entities are ordered: vertices, curves, surfaces, patches, shell, and regions; must not be negative; by default zero number of model entities of each type is assumed; if applied no model entity of corresponding type is allowed to have larger id than the specified value; need not be applied to all types of model entities

–f - (2x int #) numbering shift;
this option specifies the shift for node and element numbering; must not be negative; no numbering shift is applied by default

–L - (int #) default listing size;
this option defines the length of listing which is allocated as a contiguous block of memory; ranges from 10 to 10000; default value is 1000

–T - (int #) hash table size;
this option defines the size of hash table used to access model entities; the larger value the faster access; must be positive; default value is 100

–r - (int #) renumbering type;
some cumulative combinations of the following specifications may be used

0 - no renumbering (default)
1 - renumbering of nodes
2 - renumbering of elements (disabled)
4 - fast renumbering

specification 4 is relevant only if used cumulatively with specification 1 or 2; it attempts to shorten the renumbering time by skipping passes the are not likely to produce the best renumbering

–s - (int #) type of weighting applied during the Laplacian smoothing;
this option is relevant only if option *–S* is not used; any cumulative combination of the following types may be used

0 - no weighting (default)
1 - mesh size weighting
2 - connectivity weighting

–k - (int #) element degree;
one of the following types must be used

1 - linear (default)
2 - quadratic

note that midside nodes on quadratic elements do not follow possible curvature of model boundary unless the *–B* option is specified

–p - (int #) additional output specification;
any cumulative combination of the following specifications may be used

- 0 - no additional output (default)
- 1 - output of parametric coordinate(s)
  (only nodes classified to a curve, surface, patch, or shell)
- 2 - output of node tangent
  (only nodes classified to a curve)
- 4 - output of node normal
  (only nodes classified to a surface, patch, or shell)
- 8 - output of boundary model entities
  (boundary curve for adjacent triangles and boundary
  surface, patch, or shell for adjacent tetrahedrons)
- 16 - output of elements associated with boundary model entities
  (triangles associated with a curve and tetrahedrons associated with
  a surface, patch, or shell)
- 32 - output of neighbouring elements
  this specification is enabled only if the code has been compiled with
  the *T3D_OUTPUT_NEIGHBOUR* directive
- 128 - output of nodes in T3d native order
  native ordering follows the mesh generation order: vertex, curve, surface, patch, shell, region linear nodes, curve, surface, patch, shell, region quadratic nodes;
  this specification is relevant only if used together
  with node renumbering request (see *–r*)
- 512 - complete output of boundary model entities
  (boundary curve for adjacent triangles and tetrahedrons and boundary
  surface, patch, or shell for adjacent tetrahedrons)
  note that this specification automatically activates
  specification 8
- 1024 - mesh output even if zero element quality detected
- 4096 - simple output limited to output of blocks 1 – 5;
  output of nodes is performed without classification to model entities and
  without any additional data irrespectably to remaining output specification
- 8192 - output is split into separate files according to individual output blocks
  files are distinguished according to suffix added to supplied output file name:
  - block No 1 - suffix **.numbers**
  - block No 2 - suffix **.nodes**
  - block No 3 - suffix **.edges**
  - block No 4 - suffix **.faces**
  - block No 5 - suffix **.tetras**
  - blocks No 6 – 10 - suffix **.bassoc**

–q - (int #) type of mesh quality report;
any cumulative combination of the following types may be used

0 - no report
1 - element quality report (default)
2 - dihedral angle report
4 - nodal connectivity report
8 - report distribution of selected quantities
16 - disable triangle quality report
32 - disable tetrahedron quality report
64 - report all model entities
128 - per model entity report
256 - color triangle element quality
512 - color tetra element quality

note that if no from element quality, dihedral angle, or nodal connectivity is included in the mesh quality report type then no report is accomplished; implicitly only model entities designated for output are subjected to the quality report; midnodes are ignored for quality evaluation; color quality types work only together with option *–X*

–y - (int #) type of graphics output file;
one of the following types must be used

0 - no specific graphics output (default)
1 - Elixir graphics output
2 - VRML 2.0 graphics output
3 - VTK graphics output

note that this option is relevant only if used together with option *–x* and if the code has been compiled with the *T3D_GRAPHICS_SUPPORT* directive

–z - (int #) graphics output specification;
some cumulative combinations of the following specifications may be used

0 - no graphics output specification(default)
1 - wireframe graphics output
2 - mesh size contours output
4 - reverted mesh size contours output
8 - black & white mesh size contours output

specification 1 is applied only if VRML output is required; specifications 2, 4, and 8 are irrelevant if used together with specification 1; specifications 4 and 8 are relevant only if used cumulatively with specification 2; note that this option is effective only if used together with nonzero option *–y*; this option is independent of options *–M*; note that contour visualization might not be supported by all VRML viewers; also note that some VRML viewers do not support wireframe view unless explicitly specified; this option is relevant only if the code has been compiled with the *T3D_GRAPHICS_SUPPORT* directive; note that midnodes in case of quadratic elements are displayed as if the *–B* option had been used

–d - (fpn #) default mesh size;
default mesh size is assigned to vertices and control points with zero mesh size specification if local mesh size control is applied (see *–v* option); is used in input file; must be positive

–u - (fpn #) uniform mesh size;
uniform mesh size is propagated over all model entities; is used in input file; must be positive

–v - (fpn #) mesh size multiplication factor;
each mesh size specification except the background mesh size specification is multiplied by this factor; must not be negative; default value is 1; zero value disables local mesh size control

–c - (fpn #) default curvature rate;
it is assigned to model entities without curvature rate specification; controls the ratio between the appropriate mesh size and radius of curvature; default value is 1; if set to 0 curvature based mesh size control is disabled for entities that do not specify any curvature rate; is used in input file; must not be negative

–w - (fpn #) curvature rate multiplication factor;
curvature rate specified for each curve and surface is multiplied by this factor; must not be negative; default value is 1; zero value disables the curvature based mesh size control

–t - (fpn #) default curve density;
it is assigned to model curves without density specification; dtermines the mesh size as the ratio between the length of the curve and the density; default value is 1; if set to 0 density based mesh size control is disabled for entities that do not specify any density; is used in input file; must not be negative

–a - (fpn #) background mesh size multiplication factor;
background mesh size specification is multiplied by this factor; must not be negative; zero value disables background mesh size control; default value is 1

–e - (fpn #) user defined epsilon;
the value of epsilon is used to resolve the geometrical ambiguity; the default value is 1.0e-5

–R - (fpn #) size of the root octant;
this option is ignored if the specified value is smaller then the largest dimension of the bounding box of the domain; must be positive; note that the value may be increased when *–U* option is specified

–U - (3x fpn #) origin of the root octant;
note that this option may cause increase of root octant size specified with *–R* option; by default root octant origin is placed in such a way that the root octant center coincides with the center of the domain bounding box

–E - (fpn #) octant size excess;
octant size excess controls the maximum ratio between the terminal octant size and the actual required element size; ranges from 1 to 1.5; default value is 1

–X - enables interactive X-window interface based on Elixir graphic library;
this option is available only when source code was compiled with the directive *T3D_ELIXIR* and all prerequisite libraries (Ckit, Elixir, X Window) and header files were available

–D - sets demonstration mode for run without user intervention;
this option works only together with option *–X*

–M - enables drawing of mesh size contours;
this option works only together with option *–X*

–N - enables drawing of background mesh;
this option works only together with option *–X*; when used together with option *–M* mesh size contours will be applied to background mesh only

–Y - (3x int #) sets plotting specifications for model, mesh, and octree (in this order);
the purpose of this option is to save memory when using graphic interface;
this option works only together with option *–X*
any cumulative combination of the following types may be used for the model drawing specification

0 - do not draw model
1 - draw model vertices (default)
2 - draw model curves (default)
4 - draw model surface, patches, and shells (default)
8 - draw model regions (default)
16 - draw model vertex numbers (default)
32 - draw model curve numbers (default)
64 - draw model surface, patch, and shell numbers (default)
128 - draw model region numbers (default)

any cumulative combination of the following types may be used for the mesh drawing specification

0 - do not draw mesh
1 - draw mesh nodes (default)
2 - draw mesh edges (default)
4 - draw mesh triangles (default)
8 - draw mesh tetrahedrons (default)
16 - draw mesh node numbers (default)
32 - draw mesh edge numbers (default)
64 - draw mesh triangle numbers (default)
128 - draw mesh tetrahedron numbers (default)

one of the following types may be used for the octree drawing specification

0 - do not draw octree
1 - draw octree (default)

note that drawing of the whole model, mesh, and octree is performed by default, which corresponds to *–Y 255 255 1* plot specification

–@ - enables phase by phase run;
this option works only if *T3D_PC_VERSION* directive is not defined

–$ - enables step by step run;
this option works only if *T3D_PC_VERSION* directive is not defined

–# - specifies element removal when discretizing regions;
this option works only together with options *–X* and *–$*

When a t3d parametric option is specified more than once only the last value is accepted. This is not true for file name specification which is supposed to be unique.

The following command line options are not processed by *t3d* and are passed to the Elixir graphic interface (if available and required)

–display (string) - redirects the output

–geometry (string) - specifies the geometry

–fn (string) - specifies the font

–font (string) - specifies the font

–defcmap - sets the default color map

–defvisual - sets the default visual

–bestvisual - sets the best visual

8 Terminal Output

During the runtime, the user is notified about completion of individual phases of the mesh generation, about the number of generated mesh entities, about the nominal profile before and after renumbering (if nodal renumbering was required), about the mesh quality (if quality report was required), and about the time (if running without graphic interface) and memory consumed by the mesh generation (only mesh entities are considered).

The number of mesh entities is provided separately for the total number of mesh entities in the mesh (total) and for the number of the mesh entities in the output of finite element mesh (FE).
The nominal profile is defined as number of entries in the upper triangular part of the connectivity matrix (Laplacian matrix of the mesh graph), which have at least one nonzero entry with the same column index and smaller row index, increased by number of nodes. In the case of quadratic elements, the full connectivity is considered. The nominal profile multiplied by square of degrees of freedom per node can be used to estimate memory requirements for characteristic matrix of the discretized governing differential equation in the finite element analysis.
The quality report depends on the actually used *–q* option. By default, it provides the arithmetic and harmonic means of selected quality quantity, the worst quality (including

the number of the worst element (in parenthesis), if its quality falls into the worst quality interval), and the distribution (the first three lines of the quality report in terminal output bellow) of the quality into three quality intervals expressed as the number of elements falling into each interval (interval bounds are in parenthesis) and corresponding percentage of the total number of elements. If required, a more detailed quality distribution is generated.

An example of terminal output follows:

```
T3D - Triangulation of 3D Domains
Copyright: Daniel Rypl, 1995-2008
=================================

t3d -i wheel.in -o wheel.out -d 10 -r 1

Program started          21:23:25
Options analyzed         21:23:25
Input data analyzed      21:23:25
Octree built             21:23:25
Vertices discretized     21:23:25
Curves discretized       21:23:25
Surfaces discretized     21:23:26
Regions discretized      21:23:36
Renumbering completed    21:23:40
Discretization done      21:23:40
Output data printed      21:23:41
Program finished         21:23:41

Number of nodes:       23266 (FE)     23266 (total)
Number of edges:           0 (FE)    141288 (total)
Number of trias:           0 (FE)    227297 (total)
Number of tetras:     109280 (FE)    109280 (total)

Nominal profile:   167098536 (old)    5960348 (new)

Tetras quality:    94125 : (1.000 - 0.750)  86.13 %
                   15012 : (0.750 - 0.500)  13.74 %
                     143 : (0.500 - 0.000)   0.13 %
                 0.85555 - arithmetic mean quality
                 0.84286 - harmonic mean quality
                 0.21759 - worst quality (9554)

Real time consumed          15.40 sec
Memory consumed             35779 kB
```

9 Quality Evaluation

The quality is evaluated separately for triangular and tetrahedral elements. In the current implementation, the midnodes of quadratic elements are not considered for the quality evaluation. By default only elements on the output are subjected to the quality evaluation. There are recognized three distinct quality quantities

- element shape based quality,
- dihedral angle based quality, and
- connectivity based quality.

The elements shape based quality is evaluated according to the following formulas (quality is normalized $q \in \langle 0, 1 \rangle$)

- triangular element

  $$q = 4\sqrt{3}\ \frac{A}{a^2 + b^2 + c^2}$$

  where A is the area of the triangle and a, b, and c are the lengths of its sides,

- tetrahedral element

  $$q = 216\sqrt{3}\ \frac{V^2}{(A + B + C + D)^3}$$

  where V denotes the volume of the tetrahedron and A, B, C, and D are the areas of its faces.

The dihedral angle based quality is evaluated according to the following formula

$$q = \frac{\delta_{min}}{\delta_{max}},$$

where δ_{min} is the minimal and δ_{max} the maximal dihedral angle in a particular element.
The connectivity based quality compares the valence of individual nodes with the optimal valence which is exactly 6 for triangular mesh and approximately 12 for tetrahedral mesh.

10 Warnings and Error Messages

By default, warnings and error messages are printed on terminal output. They may be suppressed using *–Q* option or redirected to a file (together with standard terminal output) using *–g* option. There are several types of error messages each one with different exit code

- option error message (exit code 5),
- manipulation error message (exit code 10),
- memory allocation error message (exit code 15),
- file manipulation error message (exit code 20),

- input data error message (exit code 25), and
- output data error message (exit code 30).

Each error contains except its type also more or less specific explanation of the error. The place where the error was generated can be identified by the line number, function name, and module name. In the case of input/output data error, the line number and the input/output file name is also provided.

11 Input File Example

The format of input file is demonstrated on a simple example, a bus shelter, which is depicted in Figure 4c). The vertex and curve numbering is shown in Figures 4a) and 4b). The final mesh obtained via executing the command line specified at the top of the input file is displayed in Figure 4d).

The actual content of the input file follows:

```
###############
# Bus shelter #
###############

## command line
## t3d -i shelter.in -o shelter.out -d 0.5 -X -$

# base vertices
vertex 1 xyz 0 0 0
vertex 2 xyz 5 0 0
vertex 3 xyz 0 2 0
vertex 4 xyz 5 2 0

# roof vertices
vertex 11 xyz 0 0 2
vertex 12 xyz 5 0 2
vertex 13 xyz 0 2 2
vertex 14 xyz 5 2 2

# columns (z-direction)
curve 1 vertex 1 11
curve 2 vertex 2 12
curve 3 vertex 3 13
curve 4 vertex 4 14 count 2

# sill and beams (x-direction)
curve 11 vertex 1 2
curve 12 vertex 11 12
curve 13 vertex 13 14


# sill, beams and arches (y-direction)
curve 21 vertex 1 3
```

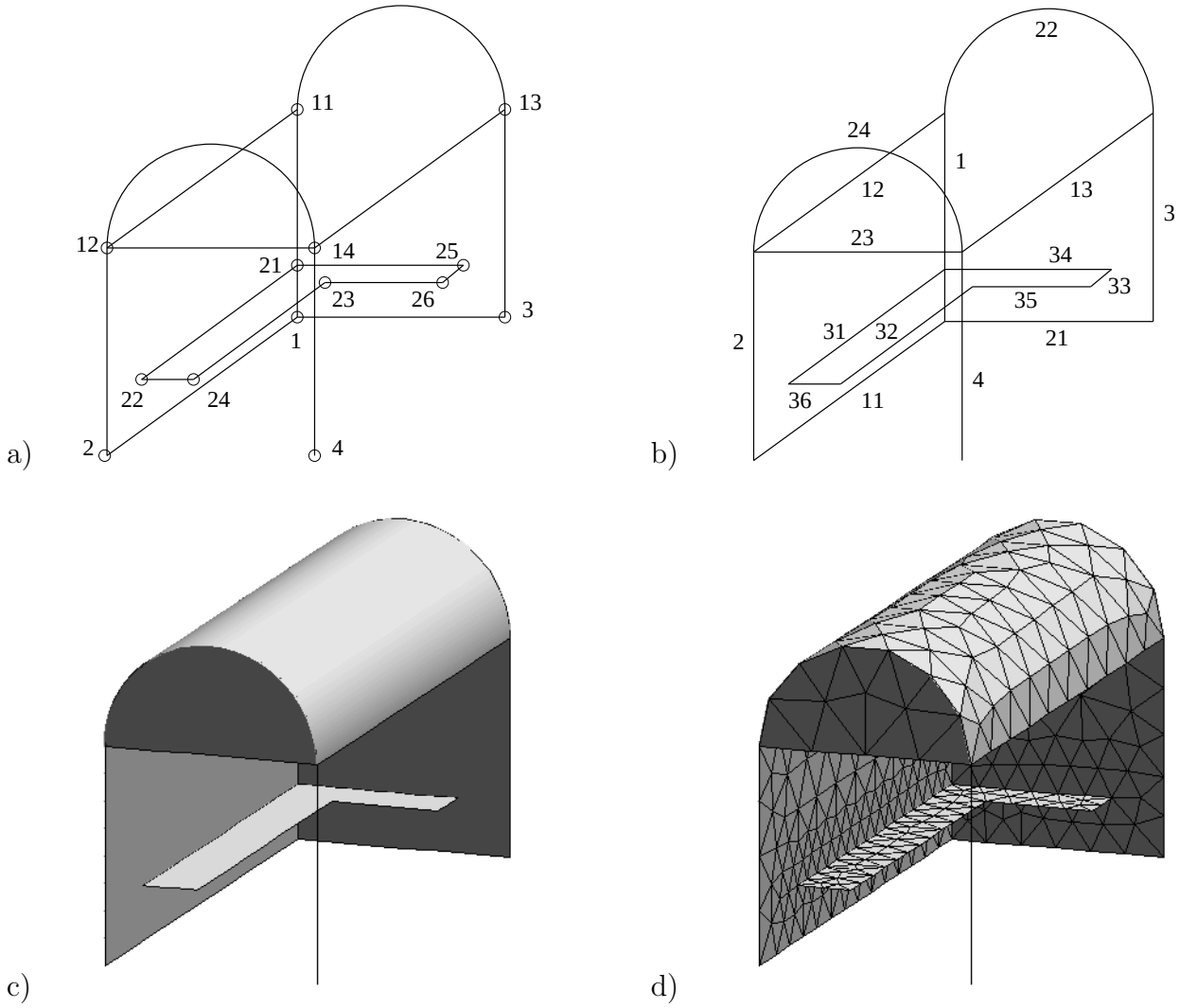

Figure 4: Bus shelter: (a) vertex numbers, (b) curve numbers, (c) model, (d) mesh.

```
curve 22 order 4 vertex 11 13
polygon 1 xyz 0 0 4 weight 0.3333333333
polygon 2 xyz 0 2 4 weight 0.3333333333
curve 23 vertex 12 14
curve 24 order 4 vertex 12 14
polygon 1 xyz 5 0 4 weight 0.3333333333
polygon 2 xyz 5 2 4 weight 0.3333333333

# bench vertices
vertex 21 xyz 0 0 0.5 fixed curve 1
vertex 22 xyz 4 0 0.5
vertex 23 xyz 0.5 0.5 0.5
vertex 24 xyz 4 0.5 0.5
vertex 25 xyz 0 1.5 0.5
vertex 26 xyz 0.5 1.5 0.5

# bench curves
```

```
curve 31 vertex 21 22 factor 0.5
curve 32 vertex 23 24
curve 33 vertex 25 26
curve 34 vertex 21 25 factor 0.5
curve 35 vertex 23 26
curve 36 vertex 22 24

# walls
patch 1 normal 0 1 0 boundary curve -11 1 12 -2 fixed curve 31 size def
patch 2 normal 1 0 0 boundary curve 21 3 -22 -1 fixed curve 34 size def
patch 3 normal 1 0 0 boundary curve 23 -24 size def

# roof
surface 1 curve 12 22 13 24

# bench
patch 11 normal 0 0 1 boundary curve 31 -32 -33 -34 35 36 size def
```

A Modelling of Conics

In this appendix, the exact representation of conical arcs by rational Bezier curves of quadratic and cubic degree is described in terms of the geometry of the control polygon and weights of its vertices. Note that the formulas for cubic curves have been derived by expansion of a quadratic curve and that negative values of weights have been considered.

A.1 Circular Arc

A.1.1 Circular Arc - Quadratic Curve

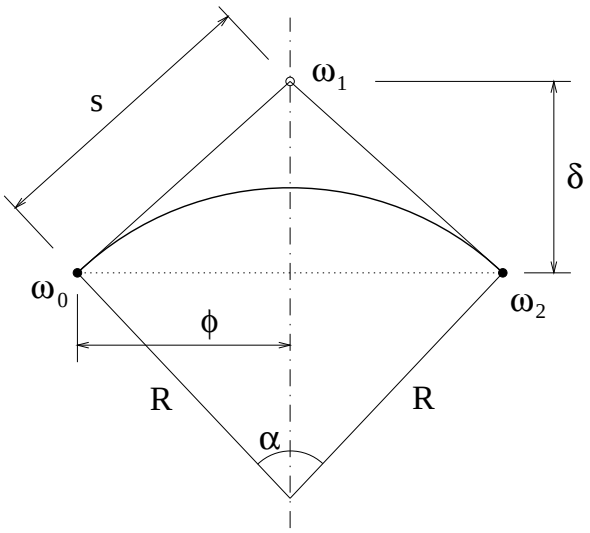

Figure 5: Circular arc - quadratic curve.

$$\delta = R\frac{\sin^2\frac{\alpha}{2}}{\cos\frac{\alpha}{2}} \qquad \phi = R\sin\frac{\alpha}{2} \qquad s = R\frac{\sin\frac{\alpha}{2}}{\cos\frac{\alpha}{2}}$$

$$\omega_0 = \omega_2 = 1 \qquad \omega_1 = \cos\frac{\alpha}{2}$$

$$\alpha \in (0; 2\pi) \setminus \{\pi\}$$

A.1.2 Circular Arc - Cubic Curve

$$\delta = 2\,R\frac{\sin^2\frac{\alpha}{2}}{1+2\cos\frac{\alpha}{2}} \qquad \phi = R\frac{\sin\alpha}{1+2\cos\frac{\alpha}{2}} \qquad s = 2\,R\frac{\sin\frac{\alpha}{2}}{1+2\cos\frac{\alpha}{2}}$$

$$\omega_0 = \omega_3 = 1 \qquad \omega_1 = \omega_2 = \frac{1+2\cos\frac{\alpha}{2}}{3}$$

$$\alpha \in (0; 2\pi) \setminus \{\frac{4}{3}\pi\}$$

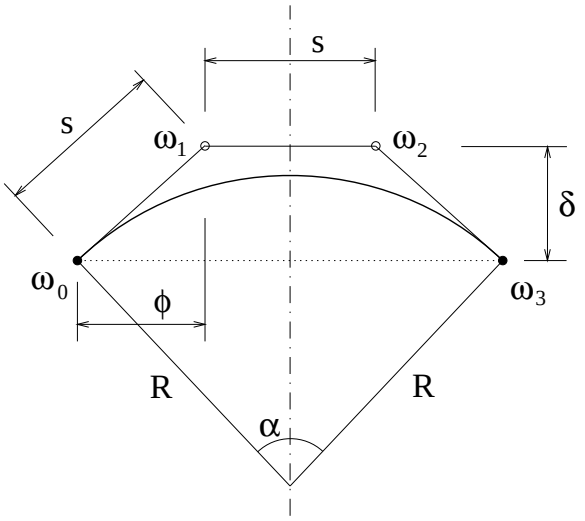

Figure 6: Circular arc - cubic curve.

A.2 Elliptic Arc

A.2.1 Elliptic Arc - Quadratic Curve

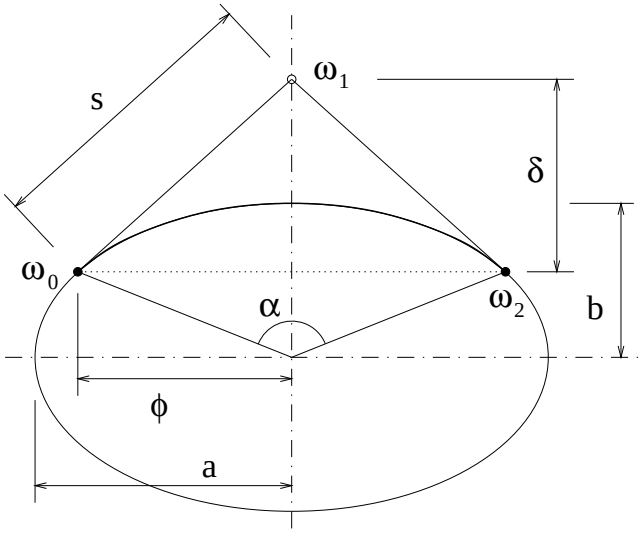

Figure 7: Elliptic arc - quadratic curve.

$$\delta = b\frac{\sin^2\frac{\alpha}{2}}{\cos\frac{\alpha}{2}} \qquad \phi = a\sin\frac{\alpha}{2} \qquad s = \frac{\sin\frac{\alpha}{2}}{\cos\frac{\alpha}{2}}\sqrt{a^2\cos^2\frac{\alpha}{2} + b^2\sin^2\frac{\alpha}{2}}$$

$$\omega_0 = \omega_2 = 1 \qquad \omega_1 = \cos\frac{\alpha}{2}$$

$$\alpha \in (0; 2\pi) \setminus \{\pi\}$$

A.2.2 Elliptic Arc - Cubic Curve

$$\delta = 2\,b\frac{\sin^2\frac{\alpha}{2}}{1+2\cos\frac{\alpha}{2}} \qquad \phi = a\frac{\sin\alpha}{1+2\cos\frac{\alpha}{2}} \qquad s = \frac{2\sin\frac{\alpha}{2}}{1+2\cos\frac{\alpha}{2}}\sqrt{a^2\cos^2\frac{\alpha}{2}+b^2\sin^2\frac{\alpha}{2}}$$

$$\omega_0 = \omega_3 = 1 \qquad \omega_1 = \omega_2 = \frac{1+2\cos\frac{\alpha}{2}}{3}$$

$$\alpha \in (0;2\pi) \setminus \{\frac{4}{3}\pi\}$$

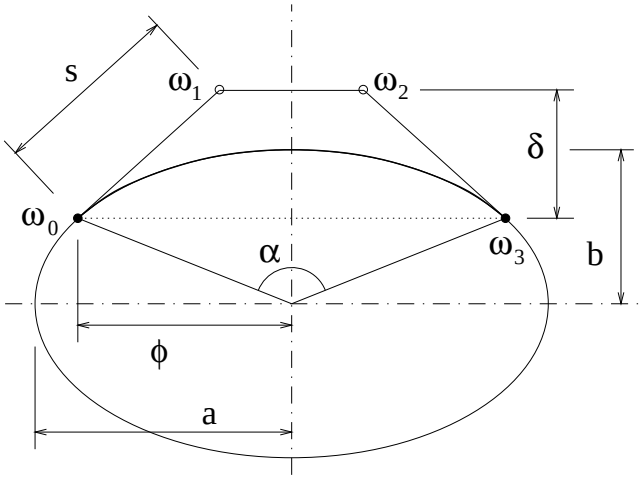

Figure 8: Elliptic arc - cubic curve.

A.3 Parabolic Arc

A.3.1 Parabolic Arc - Quadratic Curve

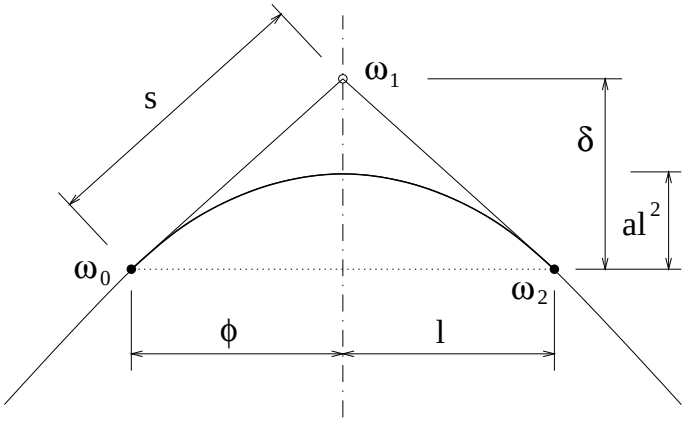

Figure 9: Parabolic arc - quadratic curve.

$$\delta = 2al^2 \qquad \phi = l \qquad s = l\sqrt{4a^2l^2+1}$$

$$\omega_0 = \omega_1 = \omega_2 = 1$$

A.3.2 Parabolic Arc - Cubic Curve

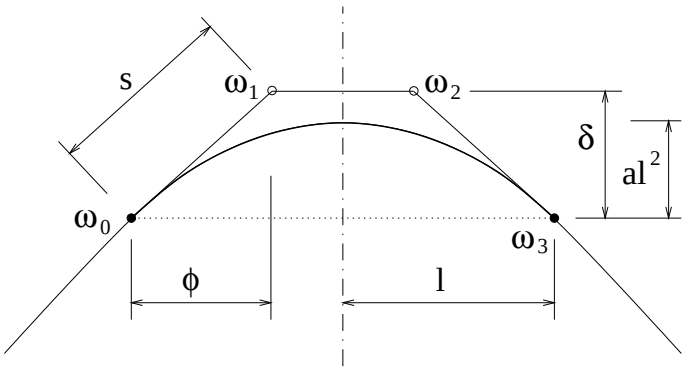

Figure 10: Parabolic arc - cubic curve.

$$\delta = \frac{4}{3}al^2 \qquad \phi = \frac{2}{3}l \qquad s = \frac{2}{3}l\sqrt{4a^2l^2+1}$$

$$\omega_0 = \omega_1 = \omega_2 = \omega_3 = 1$$

A.4 Hyperbolic Arc

A.4.1 Hyperbolic Arc - Quadratic Curve

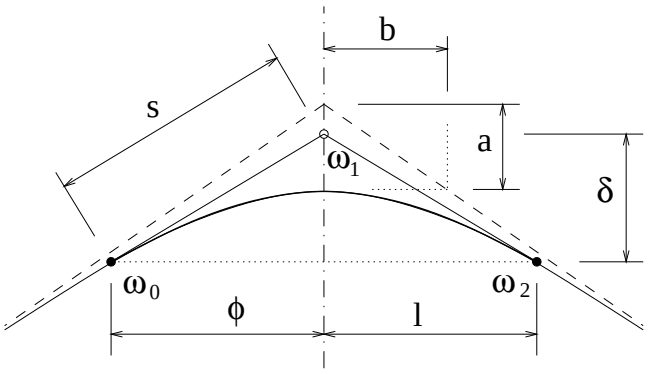

Figure 11: Hyperbolic arc - quadratic curve.

$$\delta = \frac{al^2}{b\sqrt{b^2+l^2}} \qquad \phi = l \qquad s = \frac{l\sqrt{l^2(a^2+b^2)+b^4}}{b\sqrt{b^2+l^2}}$$

$$\omega_0 = \omega_2 = 1 \qquad \omega_1 = \frac{\sqrt{b^2+l^2}}{b}$$

A.4.2 Hyperbolic Arc - Cubic Curve

$$\delta = \frac{2al^2}{b(b+2\sqrt{b^2+l^2})} \qquad \phi = \frac{2l\sqrt{b^2+l^2}}{b+2\sqrt{b^2+l^2}} \qquad s = \frac{2l\sqrt{l^2(a^2+b^2)+b^4}}{b(b+2\sqrt{b^2+l^2})}$$

$$\omega_0 = \omega_3 = 1 \qquad \omega_1 = \omega_2 = \frac{b+2\sqrt{b^2+l^2}}{3b}$$

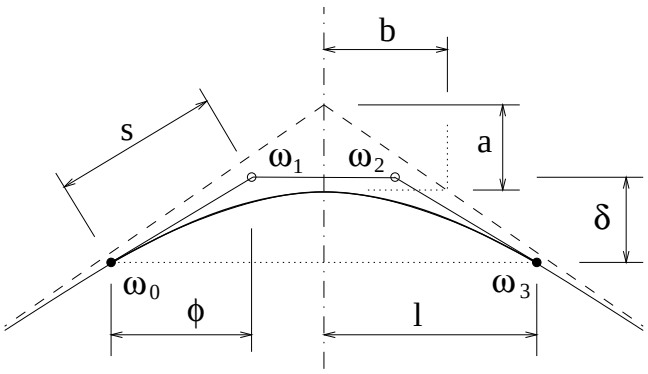

Figure 12: Hyperbolic arc - cubic curve.

B Run-Time Visualization

When running T3d with Elixir graphic interface, the bottom menu palette provides the user with all the facilities to control the run of T3d and to visualize the results. There are few useful key bindings

Accelerators

Ctrl a	fit all (all drawing windows are affected)
Ctrl p	proceed run
Ctrl s	stop run
Ctrl x	exit (no results will be printed)

Fast viewing

B1	view window (choose two opposite corners)
Ctrl B1	pan view
Ctrl B2	zoom view
Shift B2	fit all (only active drawing window is affected)
Ctrl Shift B1	rotate view
B3	done

Graphics selection

B1	select
Ctrl B1	select by window (choose two opposite corners)
Shift B1	select nearest point (confirm by B1 or select next one by Shift B1)
B2	accept selection
B3	reject selection

Handler control

Ctrl B3	suspend handler
B3	resume handler

B1, B2, and B3 stand for left, middle, and right mouse button.

In the case, an error in input data has been detected, the user is interactively asked for starting a graphic interface (to view at least the part of input data which has been successfully parsed). A time out is used to prevent blocking. Note that this is an optional feature and might not be available.