SALOME Platform

General Fuse Algorithm,

Partition Algorithm,

Boolean Operations Algorithm

Backgrounds

SALOME Platform

TABLE OF CONTENTS

1. INTRODUCTION

This document describes the backgrounds of the following algorithms: General Fuse Algorithm, Partition Algorithm, Boolean Operation Algorithm (hereafter Algorithms).

2. OVERVIEW

For the moment there are two algorithms that solve similar tasks:

- Boolean Operations Algorithm in Open CASCADE 6x (BOA).

  BOA has been designed and developed in 2000. Since 2000 the algorithm is modified to fix the problems upon requests (in general the modifications were bug fixes).

- Partition Algorithm for SALOME platform (PA).

  PA has been designed and developed in 2005 and uses modern (at that time) algorithms of OCC as auxiliary tools, for e.g. unbalanced binary tree of overlapped bounding boxes.

  PA has the following features:

  - PA is based on General Fuse Algorithm (GF).
  - PA was developed taking into account the problems faced in BOA between 2000-2005 years.
  - The architecture of PA is history-based and has been designed to support modern trends in topology science.
  - The architecture of PA is expandable, that allows to create a lot of specific algorithms upon requests.

3. ANALYSIS OF OPERATORS

- The General Fuse operator can be applied to arbitrary number of arguments (in terms of TopoDS_Shape).

  The operator can be represented as the following:

$$R_{GF}=GF\ (S_1, S_2 \dots S_n) \qquad (3.1.1)$$

where

R_{GF} – result of the operation,

$S_1, S_2 \dots S_n$ - Arguments of the operation,

n - Number of arguments (n>1)

- The Partition operator can be applied to arbitrary number of arguments (in terms of TopoDS_Shape).

  The operator can be represented as the following:

$$R_{PA}=PA\ (S_1, S_2 \dots S_{nOB}, [T_1, T_2 \dots T_{nT}]) \qquad (3.1.2)$$

where

R_{PA} – result of the operation,

$S_1, S_2 \dots S_{nOB}$ - Arguments of the operation (Objects),

nOB - number of Objects (nOB>1)

$T_1, T_2 \dots T_{nT}$ - Arguments of the operation (Tools),

nT - number of Tools

It is evident that for **nT**=0

$$R_{PA}=PA\ (S_1, S_2 \dots S_{nOB})=R_{GF} \qquad (3.1.3)$$

Thus, PA is particular case of GF.

- BOA provides the operations (*Common, Fuse, Cut*) between two arguments (in terms of TopoDS_Shape).

  The operator can be represented as the following:

$$\mathbf{R_{BOA}=B_j\ (S_1,\ S_2)} \qquad (3.1.4)$$

  where

  $\mathbf{R_{BOA}}$ – result of the operation

  Bj – operation of type j (*Common, Fuse, Cut*),

  $\mathbf{S_1, S_2}$ - Arguments of the operation.

The result $\mathbf{R_{BOA}}$ (3.1.4) can be obtained from $\mathbf{R_{GF}}$ (3.1.1).

For e.g. for the two arguments $\mathbf{S_1, S_2}$ (see the Figure 1) the result $\mathbf{R_{GF}}$ will be

$$\mathbf{R_{GF}=GF\ (S_1,\ S_2) = S_{p1}+S_{p2}+S_{p12}} \qquad (3.1.5)$$

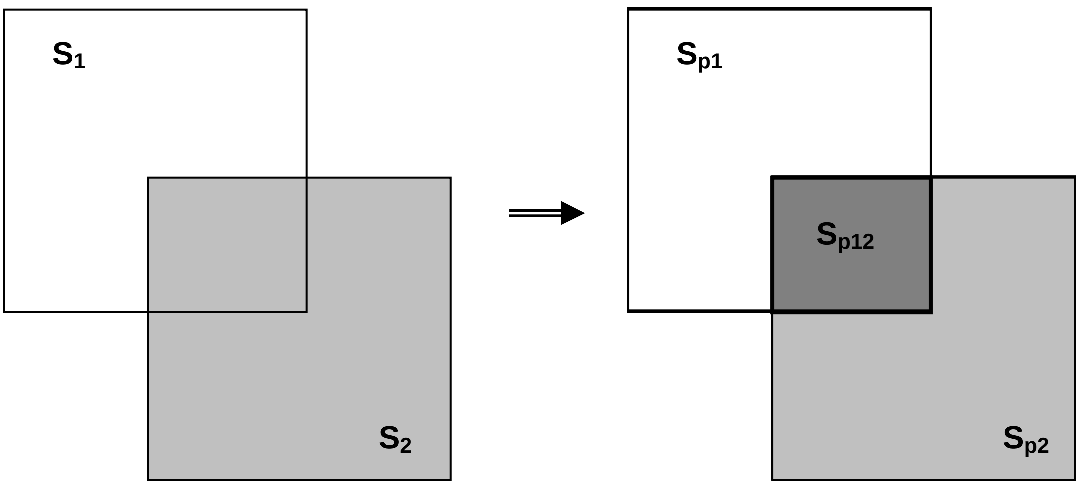


Figure 1

On the other hand

$$\mathbf{B_{common}\ (S_1,\ S_2) = S_{p12}}$$

$$\mathbf{B_{cut12}\ (S_1,\ S_2) = S_{p1}}$$

$$\mathbf{B_{cut21}\ (S_1,\ S_2) = S_{p2}} \qquad (3.1.6)$$

$$\mathbf{B_{fuse}\ (S_1,\ S_2) = S_{\ p1}+S_{\ p2}+S_{p12} = R_{GF}}$$

$$\mathbf{R_{GF}=GF\ (S_1,\ S_2) = B_{fuse} = B_{common}+ B_{cut12}+ B_{cut21}} \qquad (3.1.7)$$

The fact that the $\mathbf{R_{GF}}$ contains the components of $\mathbf{R_B}$ (3.1.7) allows to consider that GF is the general case of BOA. Thus it is possible to implement BOA, PA as subclasses of GF using C++ inheritance mechanism.

4. Terms and Definitions

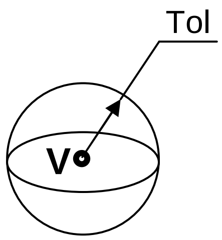
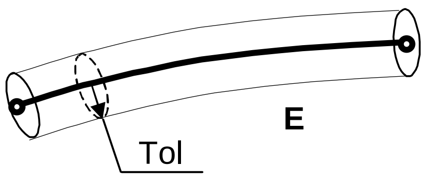
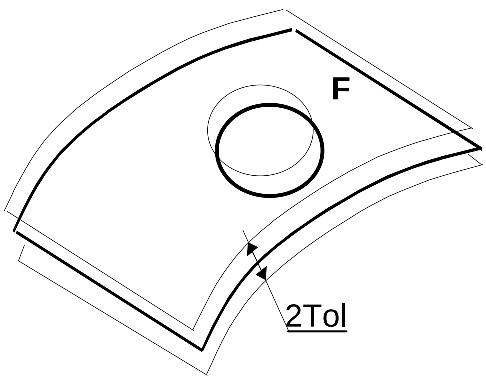
The chapter contains the background, terms, definitions that are necessary to understand how the algorithms work.

4.1. Tolerances

Each shape that has boundary representation in the Open CASCADE contains its own internal value of geometrical tolerance.

The meaning of the tolerance is different for different shapes (Table 1).

Table 1

No	Picture	Description
1		**Vertex** Value of the tolerance for a vertex V is the value of radii of a imaginary circumscribed sphere around the 3-D point of the vertex V
2		**Edge** Value of the tolerance for an edge E is the value of radii of a imaginary circumscribed pipe around the 3-D curve of the edge E
3		**Face** Value of the tolerance for a face F is the value of thickness of a imaginary offset surfaces to underlying surface of the face F

Tolerance is a way of stating how much precision is needed or conversely, how much error can be to accept the shape.

The tolerances in Open CASCADE are absolute values (in some units: mm, in, etc). Absolute tolerance defines the maximum permissible distance apart that two shapes (or sub-shapes) are permitted to be and still be considered to be *"close enough",* i.e. that these two shapes will be capable of being joined.

There are the following rules for the values of tolerances:

- For any edge **E** with vertices $\mathbf{V_i}$:

$$\mathbf{Tol\ (E) \le max\ (Tol\ (V_i)),\ i=1,\ 2\ldots Nb_V} \qquad (4.1.1)$$

where $\mathbf{Nb_V}$ – Number of vertices of the edge **E**.

- For any edge face **F** with edges $\mathbf{E_i}$:

$$\mathbf{Tol\ (F) \le max\ (Tol\ (E_i)),\ i=1,\ 2\ldots Nb_E} \qquad (4.1.2)$$

where $\mathbf{Nb_E}$ - Number of edges of the face **F**.

4.2. INTERFERENCES

The shapes interferes between each other in terms of theirs tolerances.

Shapes are interfered when there is a part of 3D space where the distance between the underlying geometry of shapes is less or equal to the sum of tolerances of the shapes.

For the three types of shapes (vertex, edge, face) there are six types of interferences.

4.2.1. Vertex/Vertex interference

A vertex $\mathbf{V_i}$ and a vertex $\mathbf{V_j}$ have Vertex/Vertex interference when $\mathbf{V_I}$ and $\mathbf{V_j}$ have the distance between corresponding 3D points that is less then sum of the tolerances $\mathbf{Tol(V_i)}$ and $\mathbf{Tol(V_j)}$ of the vertices (Figure 2).

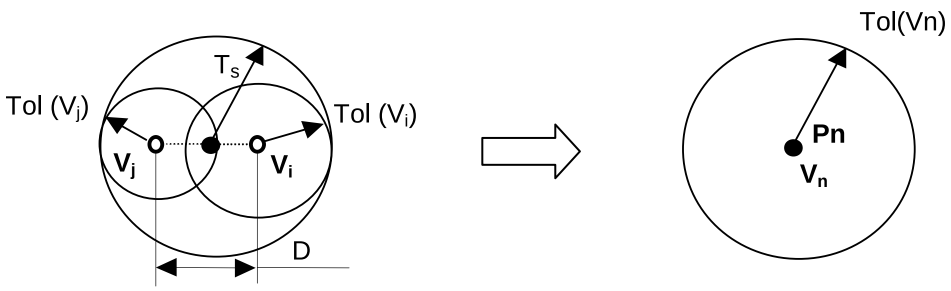


Figure 2

Result:

New vertex $\mathbf{V_n}$ with 3D point $\mathbf{P_n}$ and tolerance value $\mathbf{Tol(V_n)}$ that can be computed by the formulas:

$$\mathbf{P_{new} = 0.5 \bullet (P_i + P_j)}$$

$$\mathbf{Tol\ (V_n) = max\ (Tol(V_i),\ Tol(V_j)) + 0.5 \bullet D}$$

where

$$\mathbf{D = distance\ (P_i,\ P_j)}$$

SALOME Platform

4.2.2. Vertex/Edge interference

A vertex $\mathbf{V_i}$ and an edge $\mathbf{E_j}$ have Vertex/Edge interference when for the vertex $\mathbf{V_i}$ and the edge $\mathbf{E_j}$ the distance **D** between 3D point of the vertex and its projection on the 3D curve of the edge $\mathbf{E_j}$ is less or equal than sum of tolerances of the vertex $\mathbf{Tol(V_i)}$ and the edge $\mathbf{Tol(E_j)}$ (Figure 3).

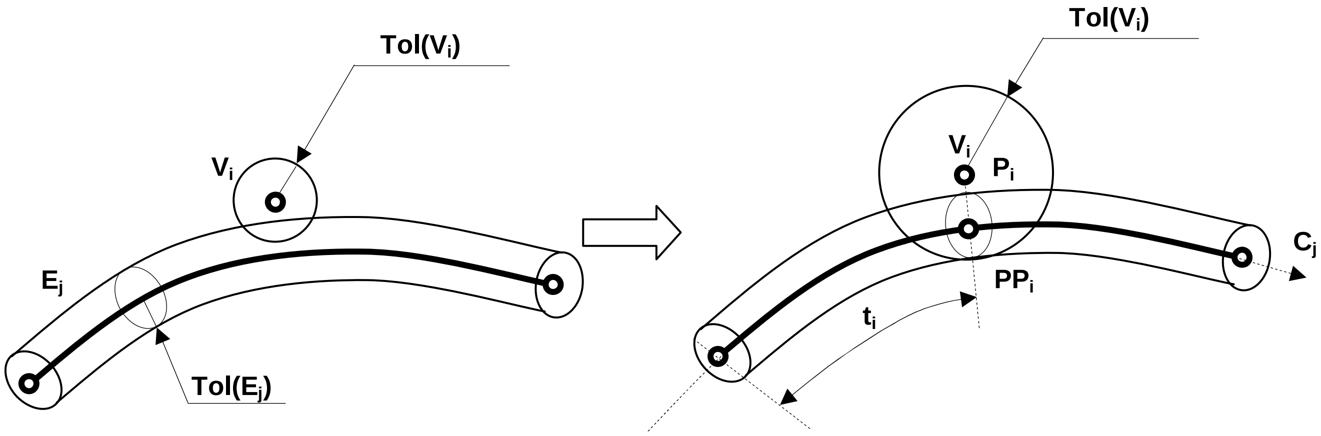


Figure 3

Result:

- Vertex $\mathbf{V_i}$ with the corresponding tolerance value.

$$\mathbf{Tol(V_i) = max\ (Tol(V_i),\ Tol(E_j)) + 0.5 \bullet D}$$

where

$$\mathbf{D = distance\ (P_i,\ PP_i)}$$

- Parameter $\boldsymbol{t_i}$ of the projected point $\mathbf{PP_i}$ on the 3D curve $\mathbf{C_j}$ of the edge $\mathbf{E_j}$.

4.2.3. Vertex/Face interference

A vertex $\mathbf{V_i}$ and a face $\mathbf{F_j}$ have Vertex/Face interference when for the vertex $\mathbf{V_i}$ and the face $\mathbf{F_j}$ the distance **D** between 3D point of the vertex and its projection on the surface of the face is less or equal than sum of tolerances of the vertex $\mathbf{Tol(V_i)}$ and the face $\mathbf{Tol(F_j)}$ (Figure 4).

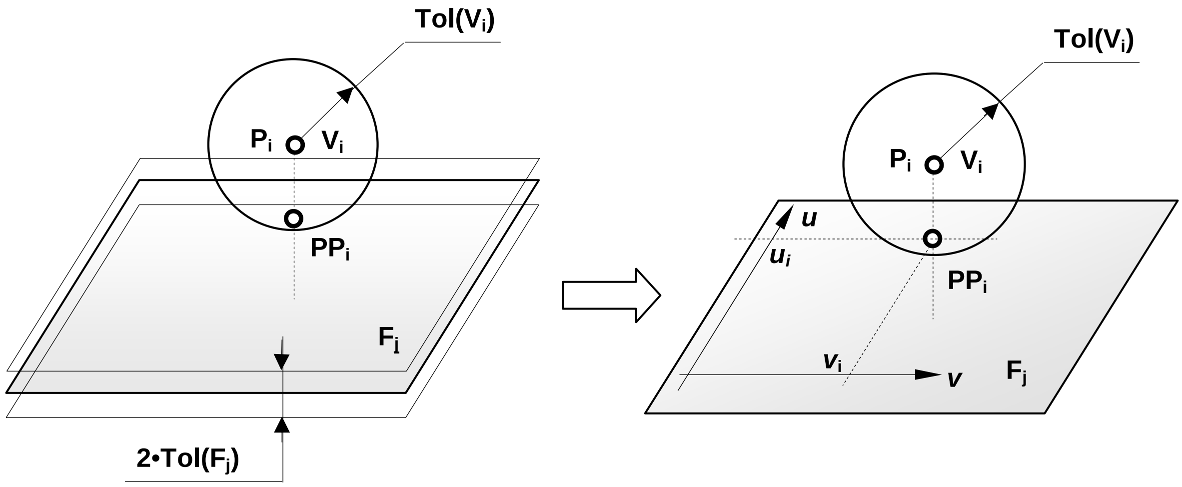


Figure 4

Result:

- Vertex $\mathbf{V_i}$ with the corresponding tolerance value.

$$\mathbf{Tol(V_i) = max\ (Tol(V_i),\ Tol(E_j)) + 0.5 \bullet D}$$

where

$$\mathbf{D = distance\ (P_i,\ PP_i)}$$

- Parameters $\boldsymbol{u_i}$**,** $\boldsymbol{v_i}$ of the projected point $\mathbf{PP_i}$ on the surface $\mathbf{S_j}$ of the face $\mathbf{F_j}$.

4.2.4. Edge/Edge interference

An edge $\mathbf{E_i}$ and an edge $\mathbf{E_j}$ have Edge/Edge interference when for two of edges $\mathbf{E_i}$ and $\mathbf{E_j}$ (with corresponding 3D curves $\mathbf{C_i}$, $\mathbf{C_j}$) there is (are) some place(s) where the distance between the curves is less than (or equals to) sum of tolerances of the edges.

Case 1: Two edges have common part(s) of 3D curves in terms of tolerance (Figure 5).

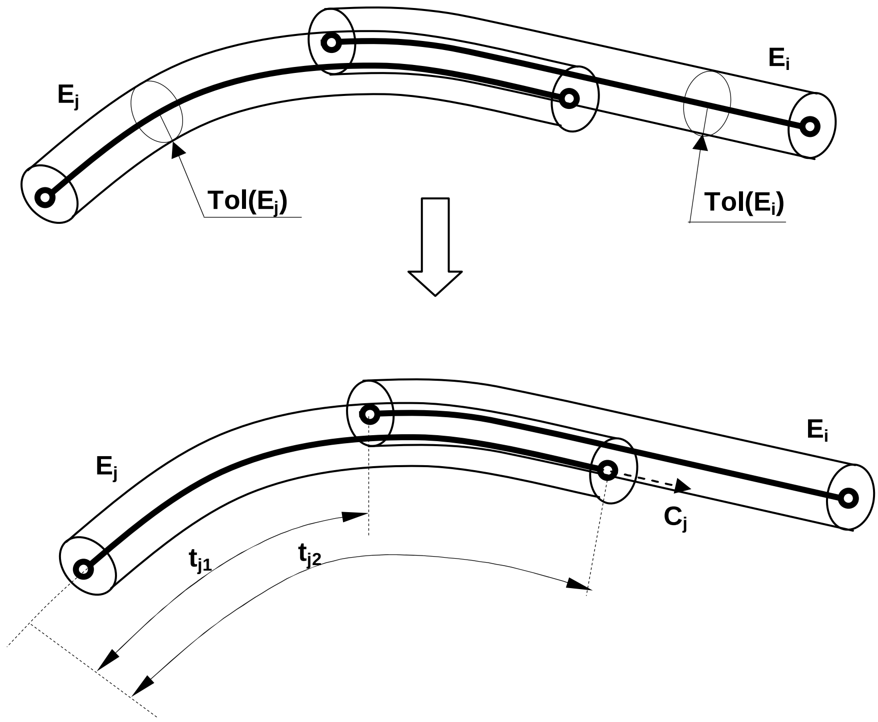


Figure 5

Result:

- Parametric range ***[t_{i1}, t_{i2}]*** for the 3D curve $\mathbf{C_i}$ of the edge $\mathbf{E_i}$.
- Parametric range ***[t_{j1}, t_{j2}]*** for the 3D curve $\mathbf{C_j}$ of the edge $\mathbf{E_j}$.

Case 2: Two edges have common point(s) in terms of tolerance (Figure 6).

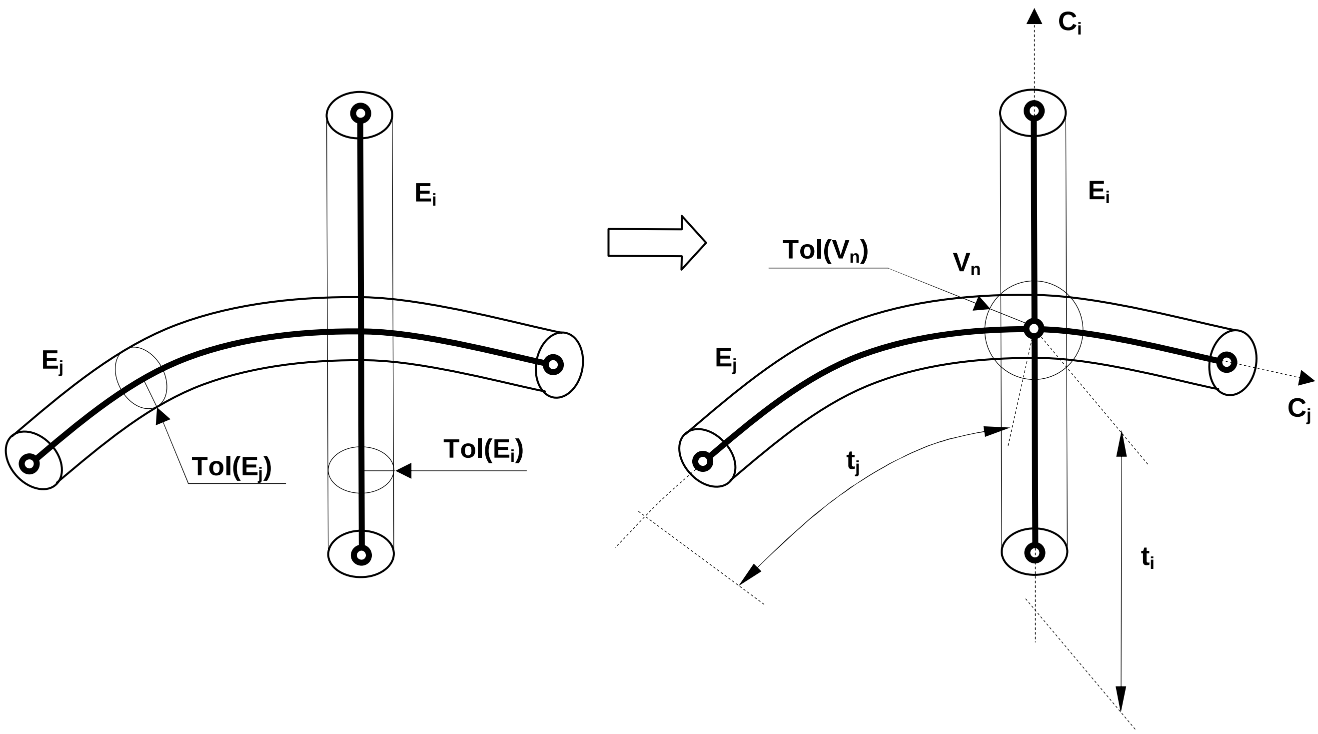


Figure 6

Result:

- New vertex (-ices) $\mathbf{V_n}$ with 3D point $\mathbf{P_n}$ and tolerance value $\mathbf{Tol(V_n)}$ that are calculated by the formulas:

$$P_n = 0.5\bullet(P_i + P_j)$$

$$Tol(V_n) = \max(Tol(E_i), Tol(E_j)) + 0.5\bullet D$$

where

$$D = distance\ (P_i, P_j)$$

$\mathbf{P_i, P_j}$ – the nearest 3D points for the curves $\mathbf{C_i, C_j}$

- Parametr t_i of $\mathbf{P_i}$ for the 3D curve $\mathbf{C_i}$.
- Parametr t_j of $\mathbf{P_j}$ for the 3D curve $\mathbf{C_j}$.

4.2.5. Edge/Face interference

An edge $\mathbf{E_i}$ and a face $\mathbf{F_j}$ have Edge/Face interference when for the edge $\mathbf{E_i}$ and the face $\mathbf{F_j}$ (with corresponding 3D curve $\mathbf{C_i}$**,** and surface $\mathbf{S_j}$) there is (are) some place(s) in 3D space where the distance between the $\mathbf{C_i}$ and the surface $\mathbf{S_j}$ is less than (or equal to) sum of tolerances of the edge $\mathbf{E_i}$ and the face $\mathbf{F_j}$**.**

Case 1: Edge $\mathbf{E_i}$ and Face $\mathbf{F_j}$ have common part(s) in terms of tolerance (Figure 7).

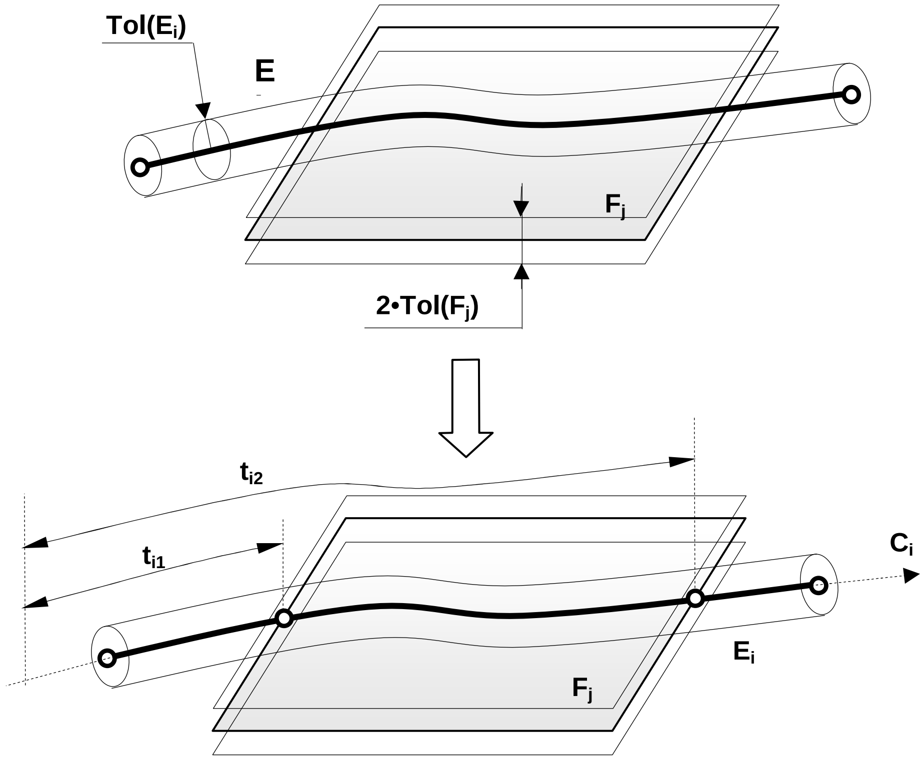


Figure 7

Result:

- Parametric range ***[t_{i1}, t_{i2}]*** for the 3D curve $\mathbf{C_i}$ of the edge $\mathbf{E_i}$.

Case 2: Edge $\mathbf{E_i}$ and Face $\mathbf{F_j}$ have common point(s) in terms of tolerance (Figure 8).

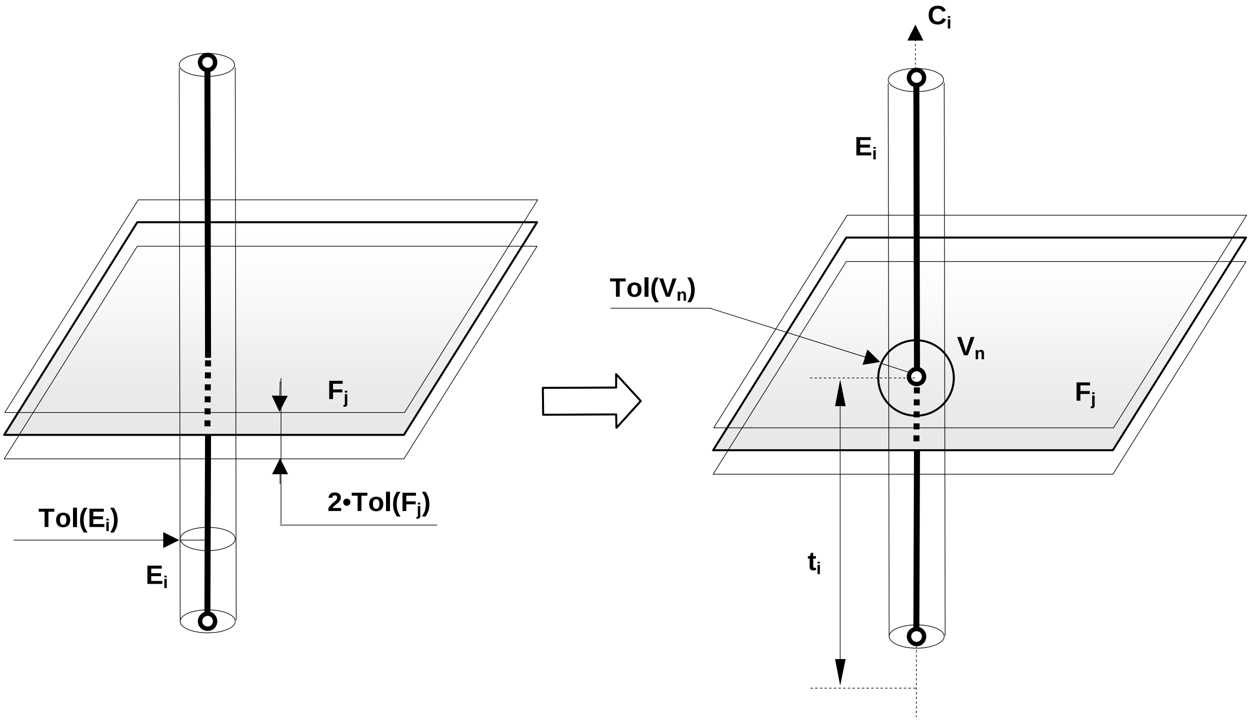


Figure 8

Result:

- New vertex (-ices) $\mathbf{V_n}$ with 3D point $\mathbf{P_n}$ and tolerance value $\mathbf{Tol(V_n)}$ that are calculated by the formulas:

$$\mathbf{P_n = 0.5 \bullet (P_i + P_j)}$$

$$\mathbf{Tol\ (V_n) = max\ (Tol\ (E_i),\ Tol\ (F_j)) + 0.5 \bullet D}$$

where

$$\mathbf{D = distance\ (P_i,\ P_j)}$$

$\mathbf{P_i, P_j}$ – the nearest 3D points for the curve $\mathbf{C_i}$ and surface $\mathbf{S_j}$

- Parameter $\boldsymbol{t_i}$ of $\mathbf{P_i}$ for the 3D curve $\mathbf{C_i}$.
- Parameters $\boldsymbol{u_i, v_i}$ of the projected point $\mathbf{PP_i}$ on the surface $\mathbf{S_j}$ of the face $\mathbf{F_j}$.

4.2.6. Face/Face Interference

A face F_i and a face F_j have Face/Face interference when for the face F_i and the face F_j (with corresponding surfaces S_i, S_j) there are some place(s) in 3D space where the distance between the surfaces is less than (or equal to) sum of tolerances of the faces.

Case 1: Face F_i and face F_j have common curve(s) C_{ijk} (k=0, 1, 2…k_n) in terms of tolerance (Figure 9).

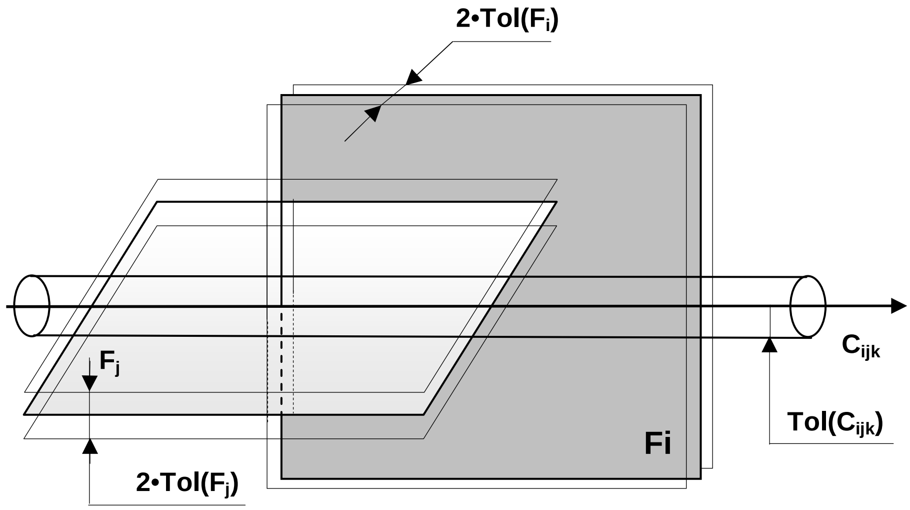


Figure 9

Result:

- Intersection curves C_{ijk} (k=0, 1, 2…k_n, k_n - number of intersection curves) with correspondent values of tolerances **Tol(C_{ijk})**.

Case 2: Face $\mathbf{F_i}$ and face $\mathbf{F_j}$ have common point(s) in terms of tolerance (Figure 10).

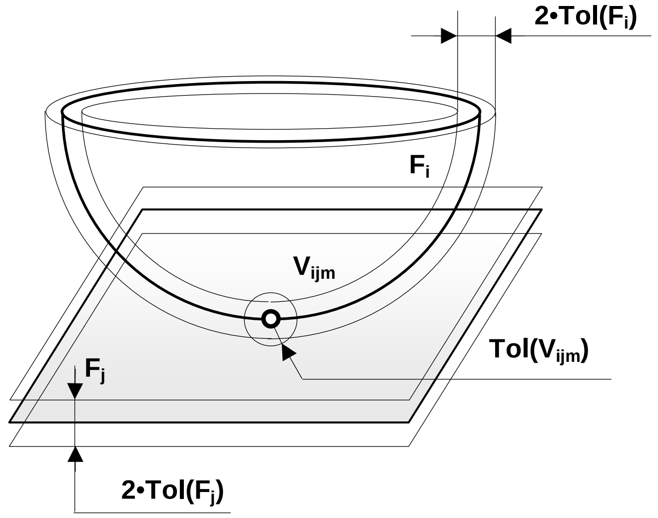


Figure 10

Result:

- New vertices $\mathbf{V_{ijm}}$ (m=0, 1, 2, m_n, - number of intersection points).
- New vertex (-ices) $\mathbf{V_{ijm}}$ with 3D point $\mathbf{P_n}$ and tolerance value $\mathbf{Tol(V_n)}$ that are calculated by the formulas:

$$\mathbf{P_n = 0.5 \bullet (P_i + P_j)}$$

$$\mathbf{Tol(V_{ijm}) = max\ (Tol(F_i), Tol(F_j)) + 0.5 \bullet D}$$

where

$$\mathbf{D = distance\ (P_i, P_j)}$$

$\mathbf{P_i, P_j}$ – the nearest 3D points for the surface $\mathbf{S_i}$ and surface $\mathbf{S_j}$.

- Parameters $\boldsymbol{u_i}$ **,** $\boldsymbol{v_i}$ of the projected point $\mathbf{PP_j}$ on the surface $\mathbf{S_j}$ of the face $\mathbf{F_j}$
- Parameters $\boldsymbol{u_i}$ **,** $\boldsymbol{v_i}$ of the projected point $\mathbf{PP_i}$ on the surface $\mathbf{S_i}$ of the face $\mathbf{F_i}$

4.2.7. Computation Order

The interferences between shapes are computed on the basis of increasing of the dimension value of the shape in the following order: ***Vertex/Vertex***, **Vertex/Edge, Edge/Edge, Vertex/Face, Edge/Face, Face/Face**. The reason is to avoid the computation of redundant interferences between upper shapes $\mathbf{S_i}$, $\mathbf{S_j}$ when there are interferences between lower sub-shapes $\mathbf{S_{ik}}$, $\mathbf{S_{jm}}$.

4.2.8. Results

- The result of the interference is a shape that can be either interfered shape itself (or its part) or a new shape.
- The result of the interference is a shape with the dimension value that is less or equal to the minimal dimension value of interfered shapes. For e.g. the result of Vertex/Edge interference is a vertex, but not an edge.
- The result of the interference splits source shapes on parts each time as it can do that.

4.3. PAVES

The result of interferences of the type Vertex/Edge, Edge/Edge, Edge/Face in a number of cases is a vertex (new or old) lying on the edge.

The result of interferences of the type Face/Face in a number of cases is intersection curves. These curves go through the vertices lying on the faces.

The position of the vertex $\mathbf{V_i}$ on the curve **C** is determined by a value of a parameter $\mathbf{t_i}$ of the 3D point of the vertex on the curve.

A Pave $\mathbf{PV_i}$ on a curve **C** is a structure containing the vertex $\mathbf{V_i}$ and correspondent value of the parameter $\mathbf{t_i}$ of the of the 3D point of the vertex on the curve. The curve **C** can be 3D or 2D curve. (Figure 11)

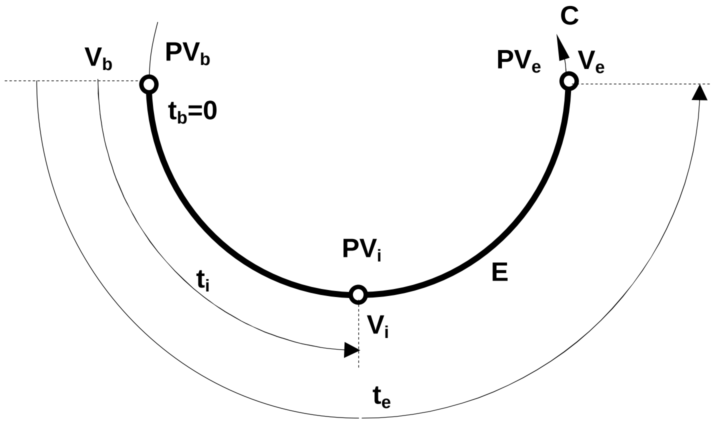


Figure 11

Two paves $\mathbf{PV_1, PV_2}$ on the one curve **C** can be compared using the parameter value. $\mathbf{PV_1 > PV_2}$ if $\mathbf{t_1 > t_2}$.

The usage of paves allows making the difference between the one vertex **V** shared between arbitrary number of edges $\mathbf{E_1, E_2, \dots E_i}$

The usage of paves allows to bind the vertex to the curve (or any structure that contains a curve: edge, intersection curve).

4.4. PAVE BLOCKS

A set of paves **PV$_i$** (i=1, 2...nPV, nPV- number of paves) of the curve **C** can be sorted in increasing order using the value of parameter **t** on the curve **C**.

A pave block **PBi** is a part of the object (edge, intersection curve) between neighboring paves. (Figure 12)

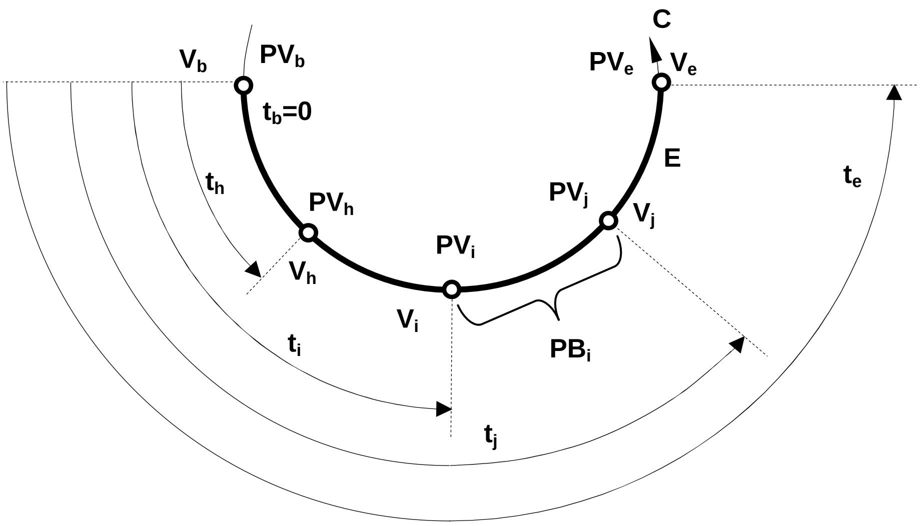


Figure 12

Any finite source edge **E** has the one bounding pave block that contains two paves **PV$_b$**, **PV$_e$**. The pave **PV$_b$** corresponds to the vertex **V$_b$** with minimal parameter **t$_b$** on the curve of the edge. The pave **PV$_e$** corresponds to the vertex **V$_e$** with maximal parameter **t$_e$** on the curve of the edge.

4.5. SHRUNK RANGE

A pave block **PV** (Figure 13) of a curve **C** is bounded by vertices **V_1, V_2** with the values of tolerances **Tol(V_1), Tol(V_2).**

The curve **C** has a value of tolerance **Tol(C)**. In case of edge the value of tolerance is tolerance of the edge. In case of intersection curve the value of tolerance is obtained from an intersection algorithm.

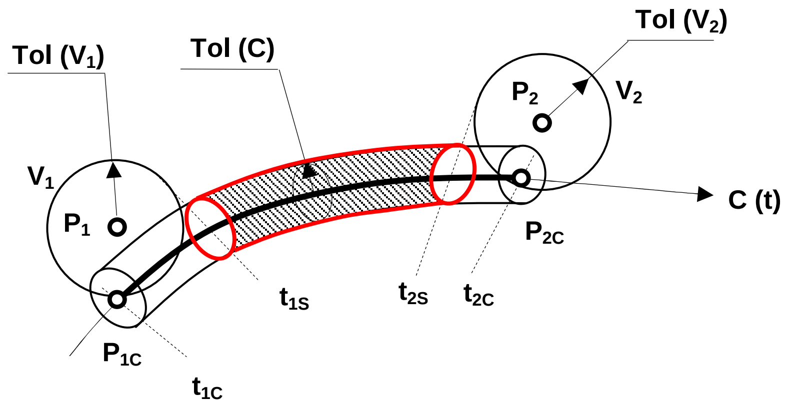


Figure 13

The theoretical parametric range of the pave block is **[t_{1C}, t_{2C}].**

The positions of the vertices **V_1, V_2** of the pave block can be different and are determined by the following conditions:

$$\textbf{Distance } (P_1, P_{1c}) \leq Tol(V_1)+Tol(C)$$

$$\textbf{Distance } (P_2, P_{2c}) \leq Tol(V_2)+Tol(C)$$

The Figure 13 shows that each tolerance sphere of a vertex can reduce the parametric range of the pave block to **[t_{1S}, t_{2S}].** The range **[t_{1S}, t_{2S}]** is the *shrunk range of the pave block*. At limiting state **t_{1S}=t_{1C}, t_{2S}=t_{2C}**.

The shrunk range of the pave block is the part of 3D curve that can interfere with other shapes.

4.6. COMMON BLOCKS

The interferences of the type **Edge/Edge, Edge/Face** can produce results as common parts. In case of **Edge/Edge** interference the common parts are pave blocks having different edges as a base. (Figure 14)

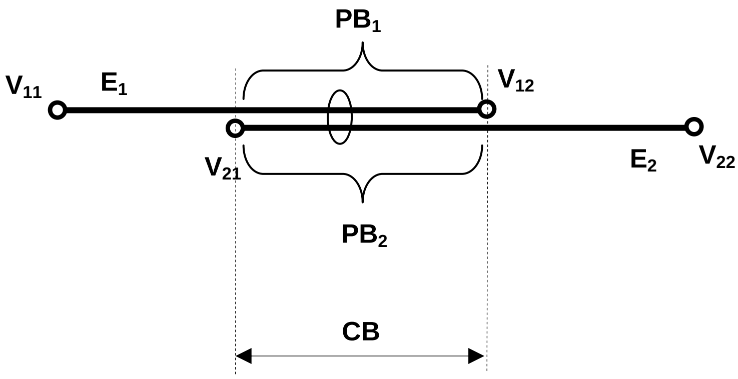


Figure 14

If the pave blocks **PV_1, PV_2…PV_{NbPV}** (**NbPV** – number of the pave blocks) have the same bounding vertices and geometrically coincide, the pave blocks form common block **CB**.

In case of **Edge/Face** interference the common parts are pave block lying on a face(s) (Figure 15).

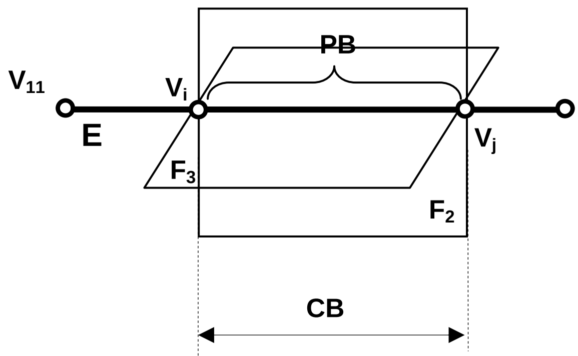


Figure 15

If the pave blocks **PV_i** geometrically coincide with a face **F_j**, the pave block form common block **CB**.

In general case a common block **CB** contains:

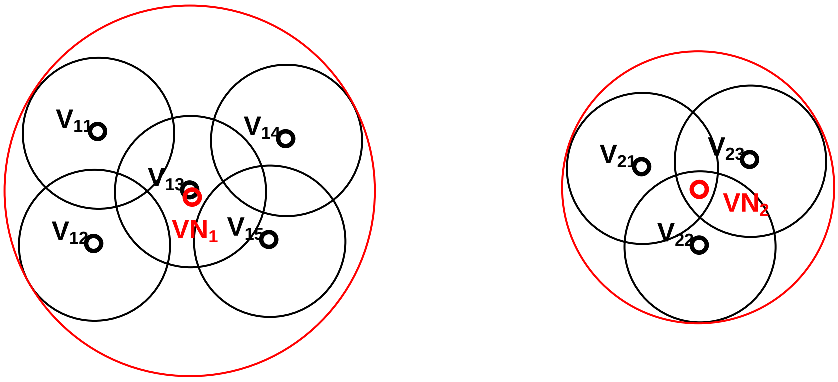
- Pave blocks **PV_i** (i=2, 3… NbPV)
- A set of faces **F_j** (j=0,1..NbF, NbF – number of faces).

4.7. Connexity Chains

The interfering shapes can produce connexity chains.

The term connexity chains can be explained by the following examples

- *Interfering vertices*. (Figure 16)



Chain 1 Chain 2

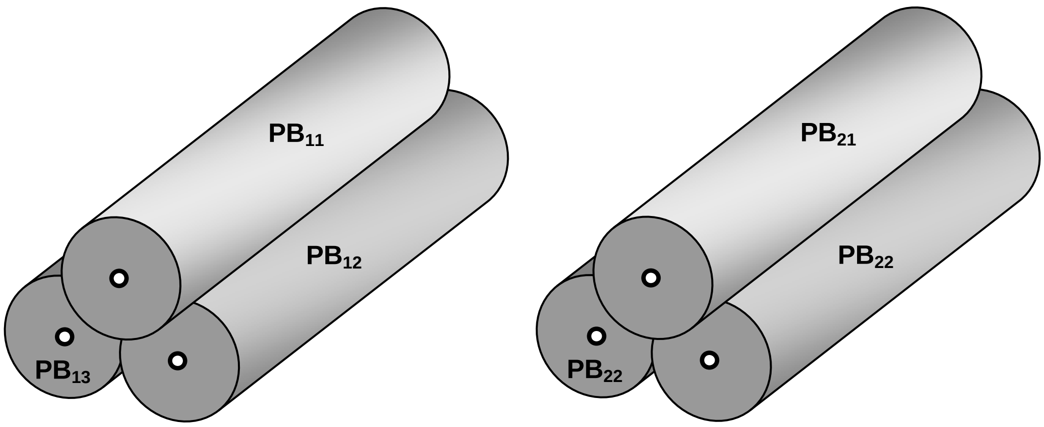
Figure 16

The pairs of interfered vertices are: (nV_{11}, nV_{12}), (nV_{11}, nV_{13}), (nV_{12}, nV_{13}), (nV_{13}, nV_{15}), (nV_{13}, nV_{14}), (nV_{14}, nV_{15}), (nV_{21}, nV_{22}), (nV_{21}, nV_{23}), (nV_{22}, nV_{23})

The pairs produce the two connexity chains: (nV_{11}, nV_{12}, nV_{13}, nV_{14}, nV_{15}), (nV_{21}, nV_{22}, nV_{23}).

Each connexity chain is used to create new vertex: VN_1, VN_2.

- *Interfering edges* (Figure 17)



Chain 1 Chain 2

Figure 17

The pairs of coincided pave blocks are: (PB_{11}, PB_{12}), (PB_{11}, PB_{13}), (PB_{12}, PB_{13}), (PB_{21}, PB_{22}), (PB_{21}, PB_{23}), (PB_{22}, PB_{23})

The pairs produce the two connexity chains: (PB_{11}, PB_{12}, PB_{13}), (PB_{21}, PB_{22}, PB_{23}).

- *Interfering faces* (Figure 18)

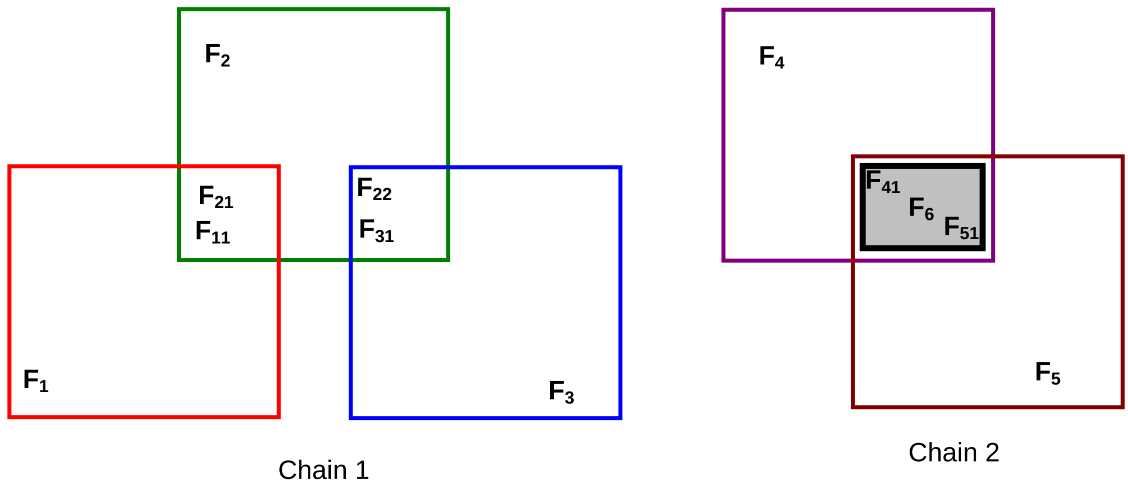


Figure 18

The pairs of same domain faces are: (F_{11}, F_{21}), (F_{22}, F_{31}), (F_{41}, F_{51}) , (F_{41}, F_6) , (F_{51}, F_6)

The pairs produce the three connexity chains: (F_{11}, F_{21}), (F_{22}, F_{31}), (F_{41}, F_{51}, F_6)

A connexity chain contains shapes that are *Same Domain* Shapes (same domain vertices, same domain edges, same domain faces).

4.8. Split Edges

The set of pave blocks PB_1, PB_2… PB_{nPB} (nPB – number of pave blocks) for the edge E is used to build split edges S_{pi} (i=1, 2…nPB).

4.9. Section Edges

The set of pave blocks PB_1, PB_2… PB_{nPB} (nPB – number of pave blocks) for the intercestion curve C is used to build section edges S_{ci} (i=1, 2…nPB).

5. Main Parts of Algorithms

The Algorithms (GF, PA, BOA) consist of three parts:

5.1. Data Structure

Data Structure (DS) is used tob

- Store information for data and intermediate results
- Provide the access to the information
- Provide the links between chunks of information

The contents of the information are:

- Arguments
- New shapes
- Interferences
- Pave Blocks
- Common Blocks

5.2. Intersection Part

Intersection Part (IP) is used to

- Initialize DS
- Compute interferences between the arguments (and their sub-shapes)
- Compute same domain vertices, edges
- Build split edges
- Build section edges
- Build p-Curves
- Store all obtained information in DS

IP uses DS as input data.

5.3. Building Part

Building Part (BP) is used to

- Build sub-shapes of the result of the operation
- Build the result of the operation

- Provide history information (in terms of *BRepBuilderAPI_MakeShape::Generated(), ::Modified(), :IsDeleted()*)

BP uses DS as input data.

6. General Fuse Algorithm

6.1. Arguments

- The arguments are shapes (in terms of *TopoDS_Shape*).
- Number of arguments is non-limited
- Each argument is valid shape (in terms of *BRepCheck_Analyzer*);
- Each argument can be one of the following types (see theTable 2)

Table 2

No	Type	Index of Type
1	COMPOUND	0
2	COMPSOLID	1
3	SOLID	2
4	SHELL	3
5	FACE	4
6	WIRE	5
7	EDGE	6
8	VERTEX	7

- The argument of type 0 (COMPOUND) can include any number of shapes of any type (0, 1…7).
- The argument should not be self-interfered, i.e. all sub-shapes of the argument that have geometrical coincidence through any topological entities (vertices, edges, faces) must share these entities.

6.2. RESULTS

- During the operation the argument Si can be splitted on parts (Si_1, Si_2… Si_{1NbSp}, NbSp - number of the parts). The set (Si_1, Si_2… Si_{1NbSp}) is *image* of the argument Si.
- The result of GF operation is a compound. Each sub-shape of the compound (resulting shape) corresponds to the concrete argument shape S_1, S_2…S_n and has shared sub-shapes in accordance with interferences between the argument and the others.
- For the arguments of the type EDGE, FACE, SOLID the result contains split parts of the argument
- For the arguments of the type WIRE, SHELL, COMPSOLID, COMPOUND the result contains the image of the shape with the corresponding type (i.e. WIRE, SHELL, COMPSOLID, and COMPOUND.

The types of resulting shapes depend on the type of the corresponding argument (Table 3).

Table 3

No	Type of argument	Type of resulting shape	Comments
1	COMPOUND	COMPOUND	The resulting COMPOUND is built from • Images of sub-shapes of type COMPOUND COMPSOLID, SHELL, WIRE, VERTEX • Sets of splitted sub-shapes of type SOLID, FACE, EDGE
2	COMPSOLID	COMPSOLID	The resulting COMPSOLID is built from split SOLID-s
3	SOLID	Set of split SOLID-s	
4	SHELL	SHELL	The resulting SHELL is built from split FACE-s
5	FACE	Set of split FACE-s	
6	WIRE	WIRE	The resulting WIRE is built from split EDGE-s
7	EDGE	Set of split EDGE-s	
8	VERTEX	VERTEX	

The following set of examples is to clarify the definition above.

6.2.1. Example 1

The arguments are three edges E_1, E_2, E_3 (Figure 19), that intersect in one 3D point.

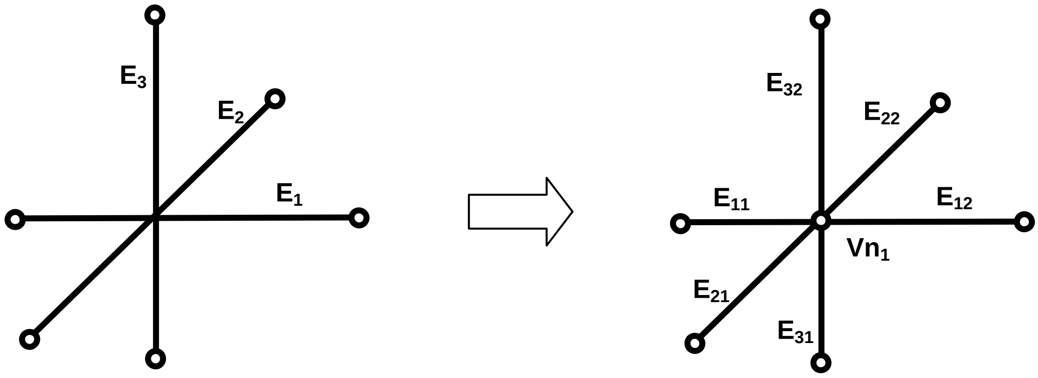


Figure 19

The result of GF operation is compound.

The compound contains 6 new edges E_{11}, E_{12}, E_{21}, E_{22}, E_{31}, and E_{32}. These edges have one shared vertex Vn_1.

In this case:

- the argument edge E_1 has resulting split edges E_{11}, E_{12} (image of E_1),
- the argument edge E_2 has resulting split edges E_{21}, E_{22} (image of E_2),
- the argument edge E_3 has resulting split edges E_{31}, E_{32}. (image of E_3).

6.2.2. Example 2

The arguments are two wires W_1 (Ew_{11}, Ew_{12}, Ew_{13}), W_2 (Ew_{21}, Ew_{22}, Ew_{23}) and the edge E_1 (Figure 20).

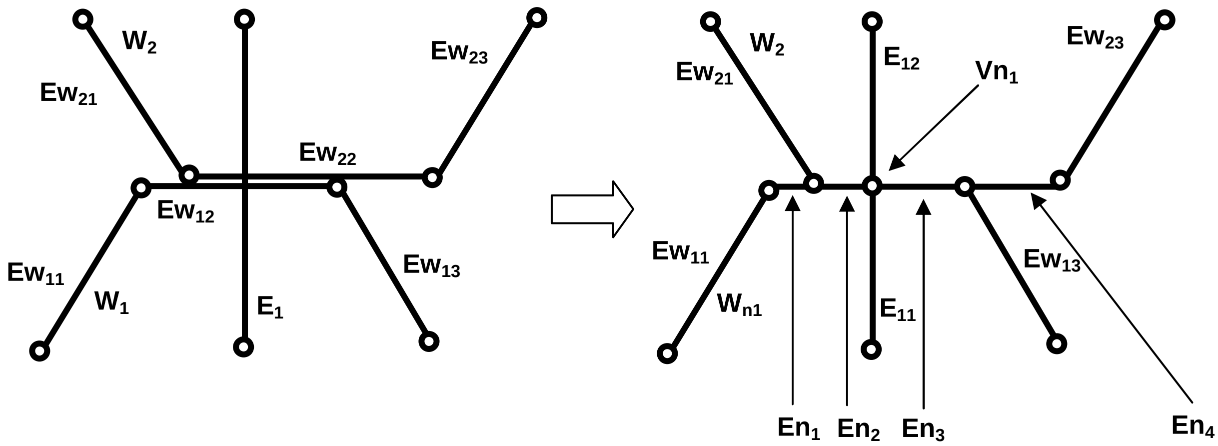


Figure 20

The result of GF operation is compound.

The compound consist of 2 wires Wn_1 (Ew_{11}, En_1, En_2, En_3, Ew_{13}), Wn_2 (Ew_{21}, En_2, En_3, En_4, Ew_{23}) and edges E_{11}, E_{12}.

In this case

- the argument W_1 has image Wn_1,
- the argument W_2 has image Wn_2,
- the argument edge E_1 has split edges E_{11}, E_{22}. (image of E_1)

The edges En_1, En_2, En_3, En_4 and the vertex Vn_1 are new shapes created during the operation so as the edge Ew_{12} has split edges En_1, En_2, En_3, the edge Ew_{22} has split edges En_2, En_3, En_4.

6.2.3. Example 3

The arguments are the edge E_1 and the face F_2 (Figure 21).

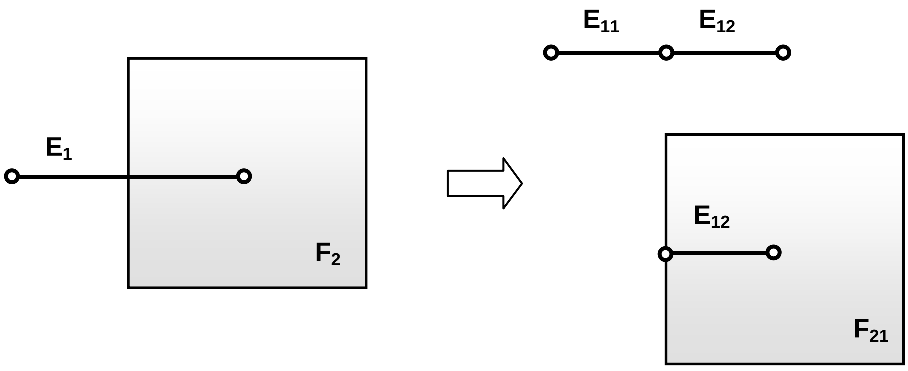


Figure 21

The result of GF operation is compound.

The compound consists of 3 shapes:

- Split parts of the edge E_{11}, E_{12} (image of E_1)
- New face F_{21} with internal edge E_{12}. (image of F_2)

6.2.4. Example 4

The arguments are the edge E_1 and the face F_2 (Figure 22).

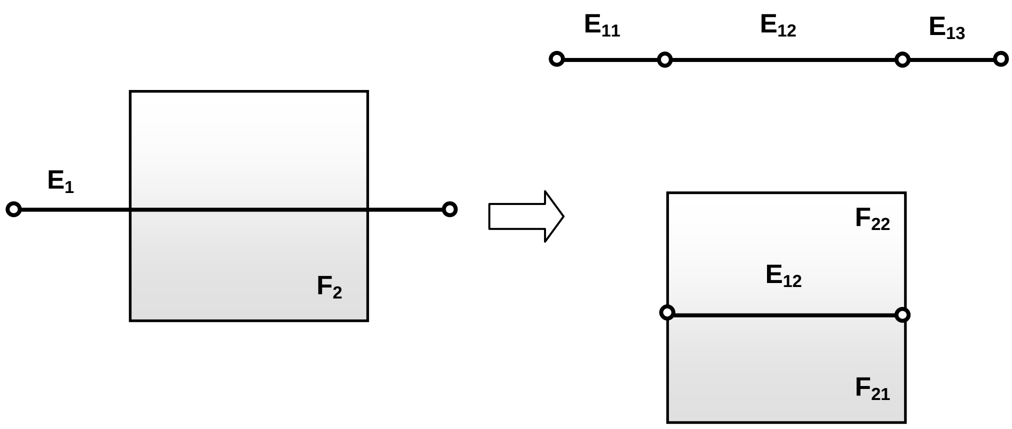


Figure 22

The result of GF operation is compound. The compound consists of 5 shapes:

- Split parts of the edge E_{11}, E_{12} , E_{13} (image of E_1),
- Split parts of the face F_{21}, F_{22} (image of F_2)

6.2.5. Example 5

The arguments are the edge E_1 and the shell Sh_2 that consists of 2 faces F_{21}, F_{22} (Figure 23).

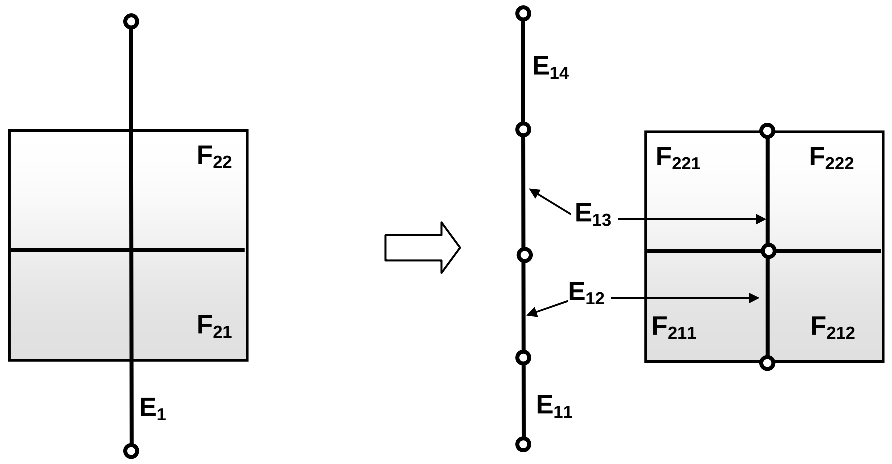


Figure 23

The result of GF operation is compound. The compound consists of 5 shapes:

- Split parts of the edge E_{11}, E_{12} , E_{13}, E_{14} (image of E_1)
- Image shell Sh_{21} (that contains split parts of the faces F_{211}, F_{212}, F_{221}, F_{222})

6.2.6. Example 6

The arguments are the wire W_1 (E_1, E_2, E_3, E_4) and the shell Sh_2 (F_{21}, F_{22}) (Figure 24).

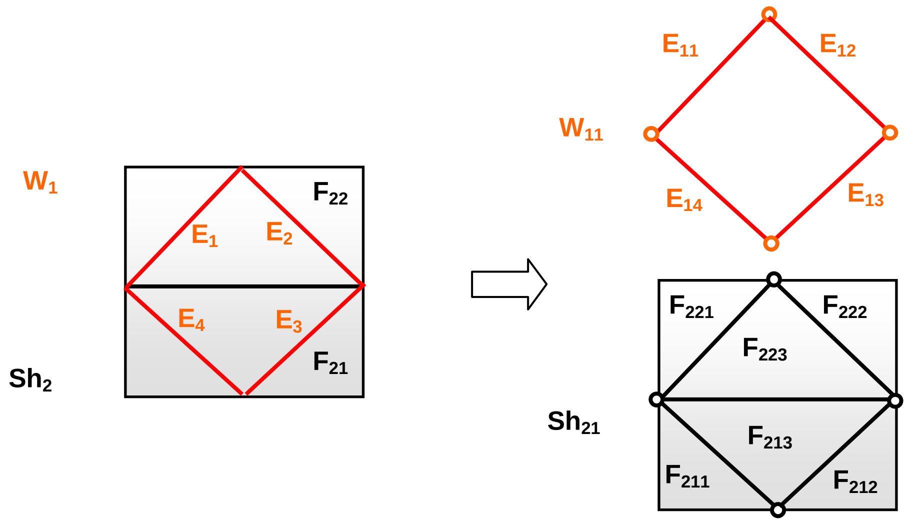


Figure 24

The result of GF operation is compound. The compound consists of 2 shapes:

- Image wire W_{11} that consist of split parts of the edges of W_1: E_{11}, E_{12} , E_{13}, E_{14}
- Image shell Sh_{21} that contains split parts of the faces F_{211}, F_{212}, , F_{213}, F_{221}, F_{222}, F_{223}

6.2.7. Example 7

The arguments are 3 faces F_1, F_2, F_3 (Figure 25).

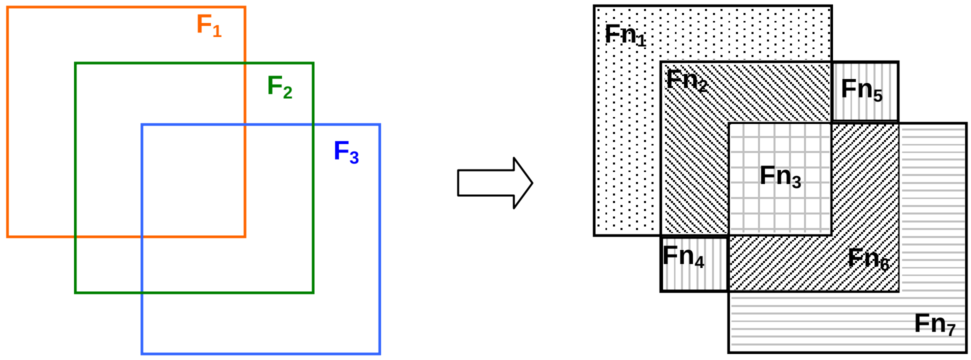


Figure 25

The result of GF operation is compound. The compound consists of 7 shapes:

- Split parts of the faces Fn_1, Fn_2, Fn_3, Fn_4, Fn_5, Fn_6, Fn_7

6.2.8. Example 8

The arguments are the shell Sh_1 (F_{11}, F_{12}, F_{13}) and the face F_2 (Figure 26).

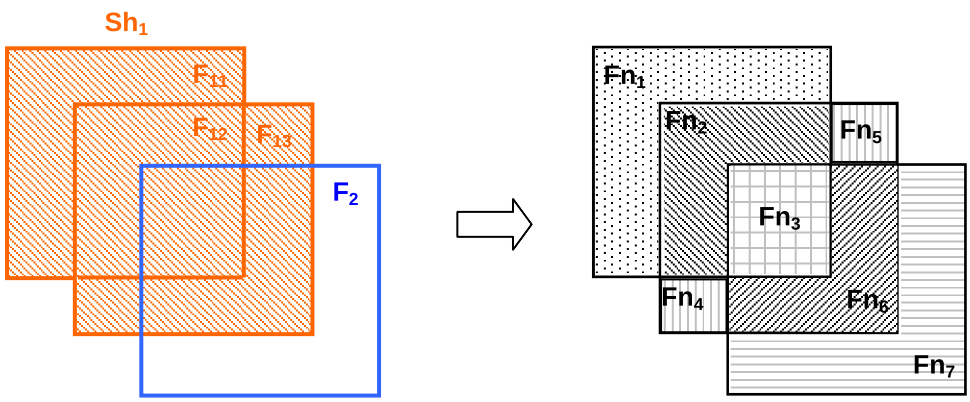


Figure 26

The result of GFoperation is compound. The compound consists of 4 shapes:

- Image shell Sh_{11} that consist of split parts of the faces of Sh_1: Fn_1, Fn_2, Fn_3, Fn_4, Fn_5, Fn_6
- Split parts of the face F_2: Fn_3, Fn_6, Fn_7

6.2.9. Example 9

The arguments are the shell Sh_1 (F_{11}, F_{12}…F_{16}) and the solid So_2 (Figure 27).

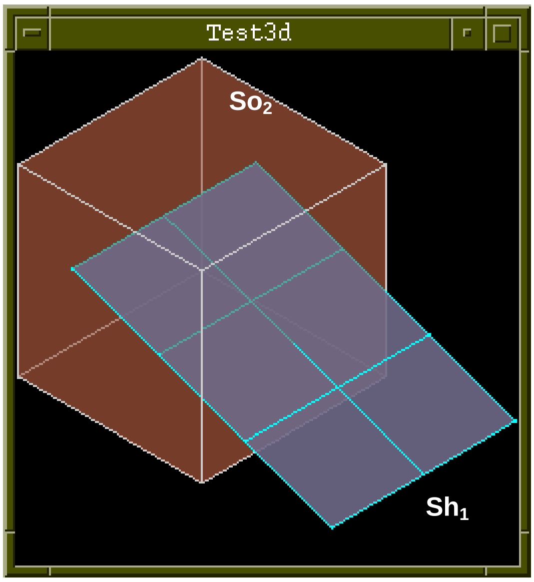


Figure 27

The result of GF operation is compound.

The compound consists of 2 shapes (Figure 28)

- Image shell Sh_{11} that consist of split parts of the faces of Sh_1: Fn_1, Fn_2….Fn_8.
- Solid So_{21} with internal shell. (image of So_2)

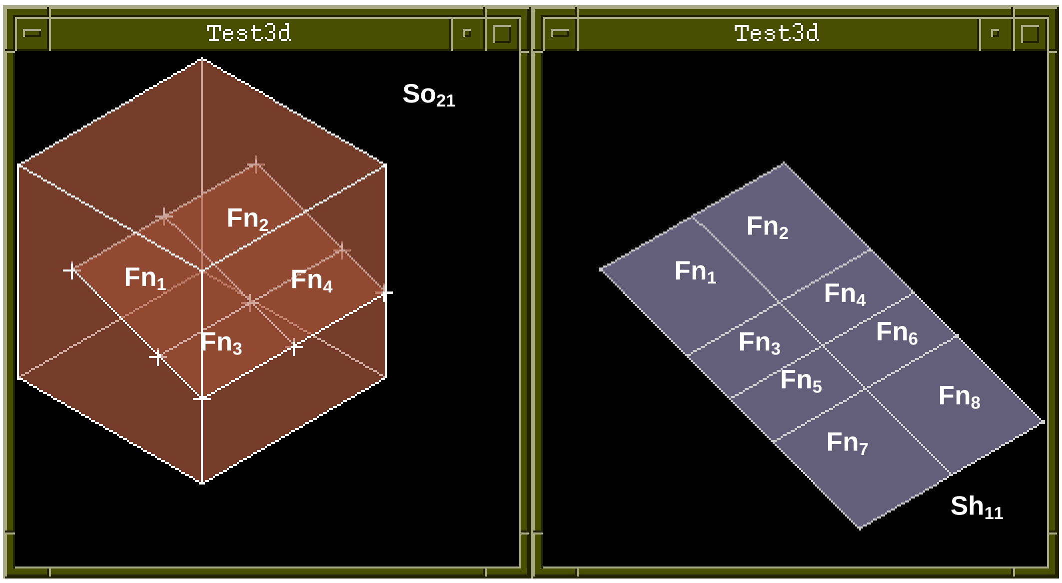


Figure 28

6.2.10. Example 10

The arguments are the compound Cm_1 (of 2 solids So_{11}, So_{12}) and the solid So_2 (Figure 29).

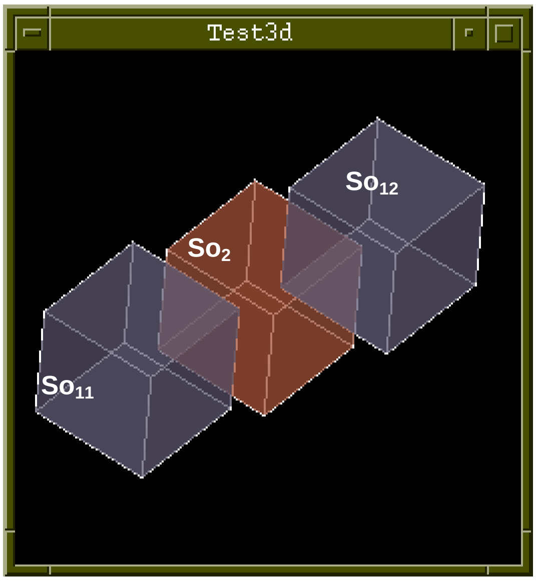


Figure 29

The result of GF operation is compound. The compound consists of 4 shapes (Figure 30)

- Image compound Cm_{11} that consists of split parts of the solids of So_{11}, So_{12} (Sn_1, Sn_2, Sn_3, Sn_4).
- Split parts of the solid So_2 (Sn_2, Sn_3, Sn_5).

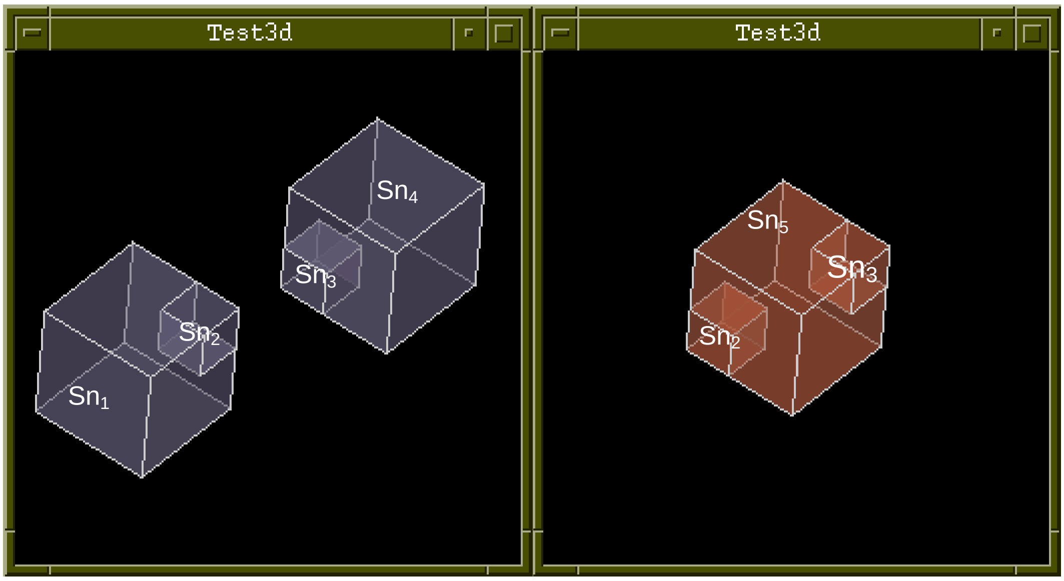


Figure 30

7. PARTITION ALGORITHM

7.1. ARGUMENTS

The arguments of the algorithm are shapes (in terms of *TopoDS_Shape*). The main requirements for the arguments are described in 6.1

The arguments in PA are separated in two groups:

- Shapes
- Tools

7.2. RESULTS

The main difference between GF and PA algorithms is in the rules of building the result.

- For arguments of **Shapes** group the rules are the same as it has been described in 6.2.
- For the arguments of **Tools** group there is the following rule: The result should not contain any shape **St**$_i$ from **Tools** group or image of the shape **St**$_i$ (**St**$_{i1}$**, St**$_{i2}$**...Sti**$_{Nbt}$ **Nbt** - number of Tools) from **Tools** group if these shapes are not included in images of **Shapes** group.
- The result will contain shapes that have complexity (in terms of TopAbs_ShapeEnum) that is less or equal to given **Limit.** The values of Limit can have any value. In case when **Limit** =TopAbs_SHAPE the result will contain shapes with complexity that corresponds to the complexity of the arguments.

7.2.1. Example 1

The arguments are

- Shapes: edges E1, E2
- Tools: edge E3 (Figure 31)
- Limit : TopAbs_SHAPE

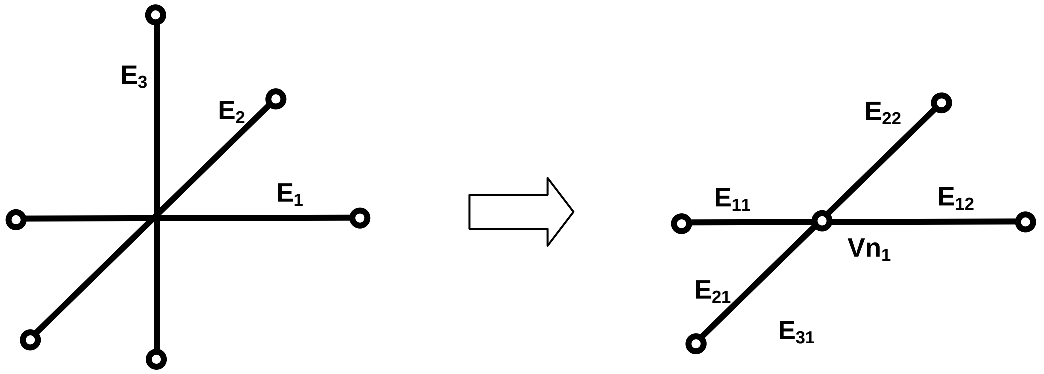


Figure 31

The result of PA operation is compound.

The compound contains 4 new edges E_{11}, E_{12}, E_{21}, E_{22}. These edges have one shared vertex Vn_1.

In this case:

- the argument edge E_1 has resulting split edges E_{11}, E_{12} (image of E_1),
- the argument edge E_2 has resulting split edges E_{21}, E_{22} (image of E_2)

7.2.2. Example 2

The arguments are

- Shapes: wire W_1 (Ew_{11}, Ew_{12}, Ew_{13}) and the edge E_1
- Tools: wire W_2 (Ew_{21}, Ew_{22}, Ew_{23}) (Figure 32).
- Limit : TopAbs_SHAPE

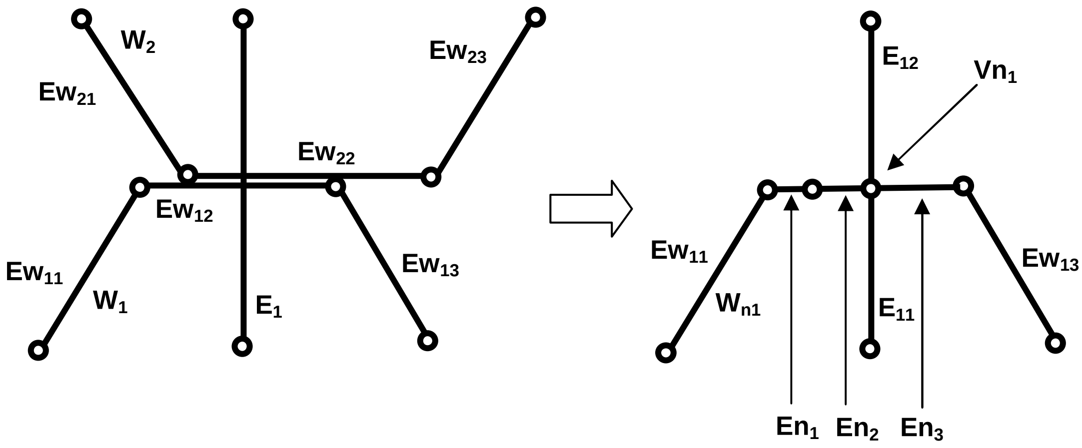


Figure 32

The result of PA operation is compound.

The compound consist of wire Wn_1 (Ew_{11}, En_1, En_2, En_3, Ew_{13}), and edges E_{11}, E_{12}.

In this case

- the argument W_1 has image Wn_1
- the argument edge E_1 has split edges E_{11}, E_{22}. (image of E_1)

The edges En_1, En_2, En_3, En_4 and the vertex Vn_1 are new shapes created during the operation so as the edge Ew_{12} has split edges En_1, En_2, En_3, the edge Ew_{22} has split edges En_2, En_3, En_4.

7.2.3. Example 3

The arguments are

- Shapes: the face F2
- Tools: the edge E1 (Figure 33).
- Limit : TopAbs_FACE

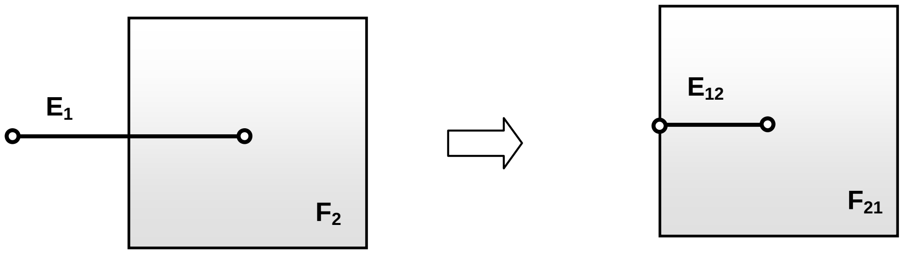


Figure 33

The result of PA operation is compound.

The compound consists of 3 shapes:

- New face F_{21} with internal edge $E_{12.}$ (image of F_2)

8. BOOLEAN OPERATIONS ALGORITHM

8.1. ARGUMENTS

- The arguments are shapes (in terms of *TopoDS_Shape*).
- Number of arguments: 2 (**Object, Tool**).
- Each argument is valid shape (in terms of *BRepCheck_Analyzer*).
- Each argument can be one of the following types (Table 4).

Table 4

No	Type	Index of Type	Dimension
1	COMPOUND	0	One of number 0, 1, 2, 3
2	COMPSOLID	1	3
3	SOLID	2	3
4	SHELL	3	2
5	FACE	4	2
6	WIRE	5	1
7	EDGE	6	1
8	VERTEX	7	0

- The argument of type 0 (COMPOUND) can include any number of shapes of any type (0, 1…7).
- Each argument should have constant value of the dimension to (Table 4)
- The argument should not be self-interfered, i.e. all sub-shapes of the argument that have geometrical coincidence through any topological entities (vertices, edges, faces) must share these entities.
- For Boolean operation *Fuse* the arguments should have equal dimensions.
- For Boolean operation *Cut* the dimension of **Object** should be not less then the dimension of **Tool**.

8.2. RESULTS

- The results of BOA[1] is defined by the formulas:

$$\mathbf{B_{common}=Object \cap Tool} \tag{8.2.1}$$

$$\mathbf{B_{fuse}=Object + Tool} \tag{8.2.2}$$

$$\mathbf{B_{cut12}=Object - Tool} \tag{8.2.3}$$

$$\mathbf{B_{cut21}= Tool - Object} \tag{8.2.4}$$

- The result of BOA operation is a compound **Bj**. Each component of the compound has shared sub-shapes in accordance with interferences between the **Object** and **Tool**.
- Bj contains components obtained by the formulas (8.2.1 - 8.2.4). The components are containers of sub-shapes having same dimensions in accordance with Table 4 (in terms of connexity for cases of edges, faces).
- The result $\mathbf{B_{common}}$ should have the dimension that is equal to the lower dimension of the arguments.

[1] The rules are valid for Open CASCADE 5.0 and higher

8.2.1. Example 1

The arguments are (Figure 34)

- Object: edge E_1
- Tool: edge E_2

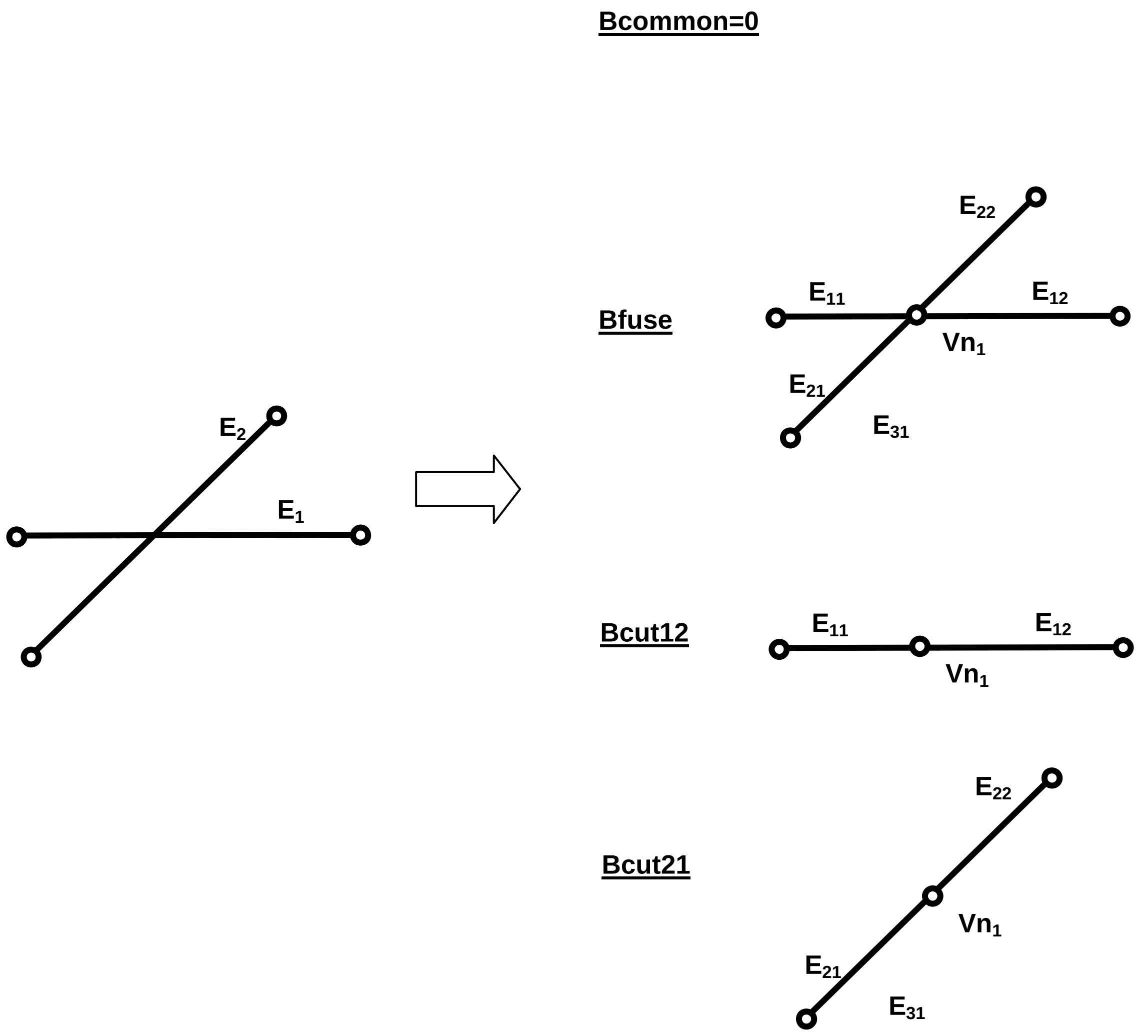


Figure 34

The result of BOA operation is compound.

B_{common}	The compound contains noting because the dimension of intersection (vertex) is less than minimal dimension of arguments.
B_{fuse}	The compound contains a wire of 4 new edges E_{11}, E_{12}, E_{21}, E_{22}. The edges have one shared vertex Vn_1.
B_{cut12}	The compound contains a wire of 2 new edges E_{11}, E_{12}. The edges have one shared vertex Vn_1.
B_{cut21}	The compound contains a wire of 2 new edges E_{21}, E_{22}. The edges have one shared vertex Vn_1.

8.2.2. Example 2

The arguments are (Figure 35):

- Object: edge E_1
- Tool: edge E_2

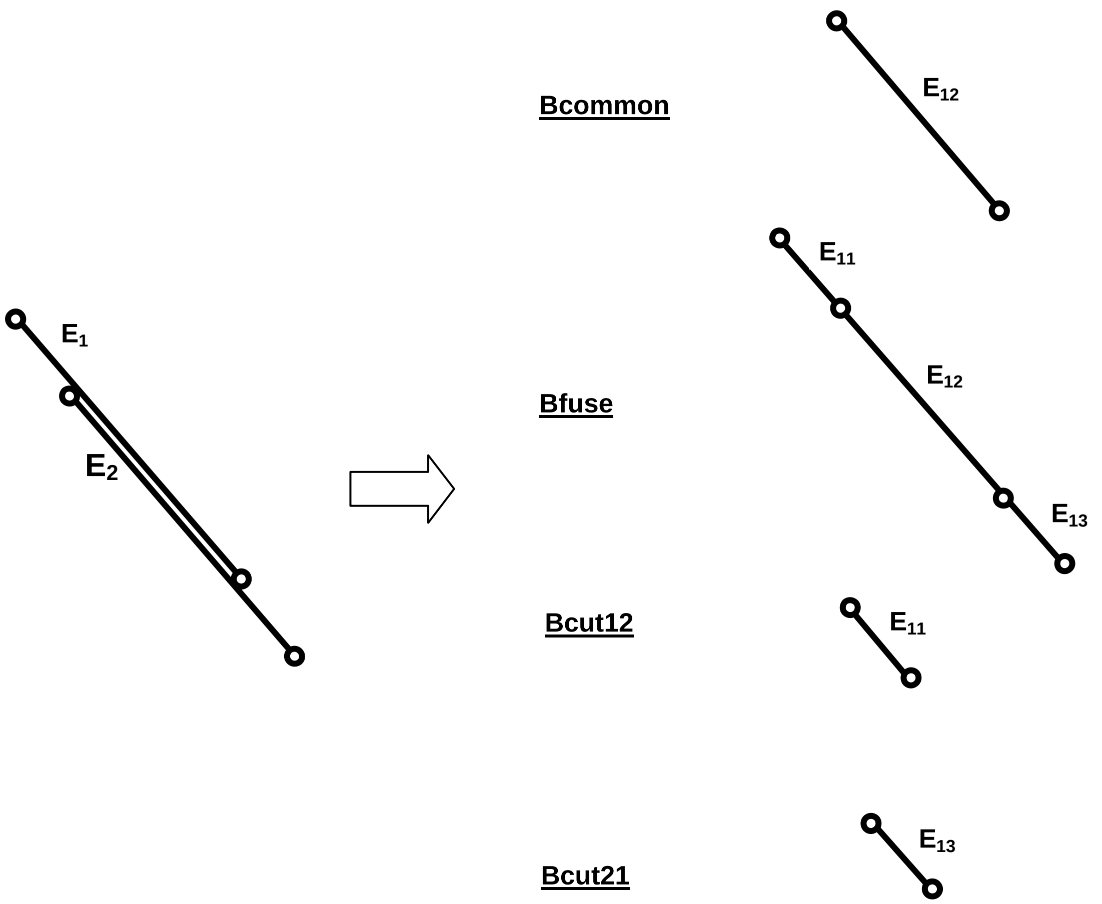


Figure 35

The result of BOA operation is compound.

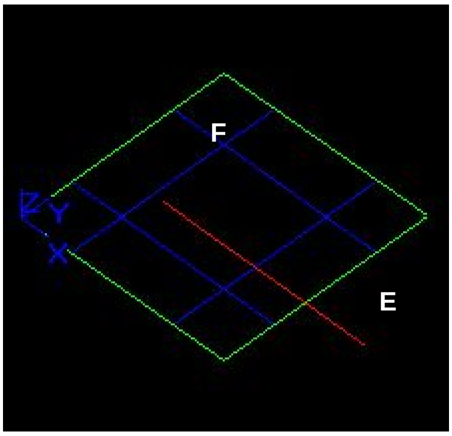
B$_{common}$	The compound contains a wire of 1 new edge E_{12}.
B$_{fuse}$	The compound contains a wire of 3 new edges E_{11}, E_{12}, E_{13}. The edges have shared vertices.
B$_{cut12}$	The compound contains a wire of 1 new edge E_{11}.
B$_{cut21}$	The compound contains a wire of 1 new edge E_{13}.

8.2.3. *Example 3*

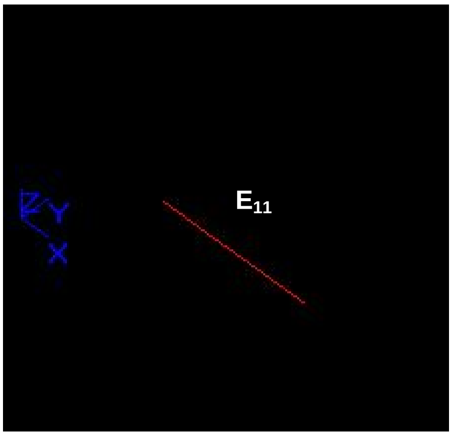
The arguments are (Figure 36)

- Object: face F
- Tool: edge E

Arguments



B$_{common}$



B$_{fuse}$=0

B$_{cut12}$=0

B$_{cut21}$

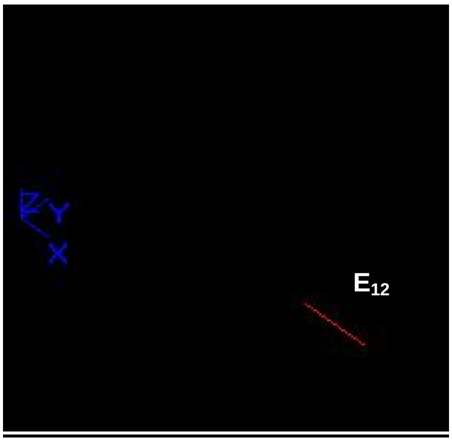


Figure 36

The result of BOA operation is compound.

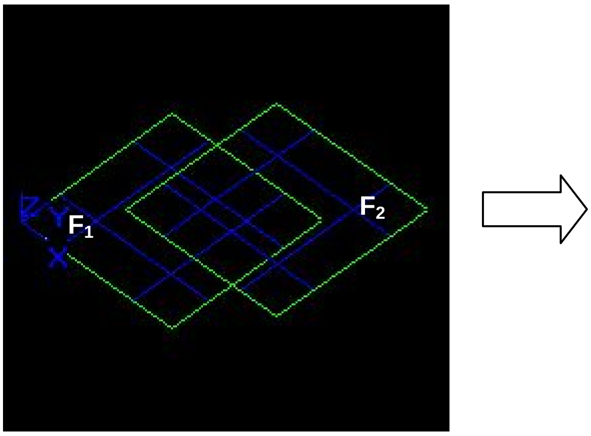
B$_{common}$	The compound contains new edge E_{11}.
B$_{fuse}$	The compound contains nothing (different dimensions of the arguments).
B$_{cut12}$	The compound contains nothing.
B$_{cut21}$	The compound contains new edge E_{12}.

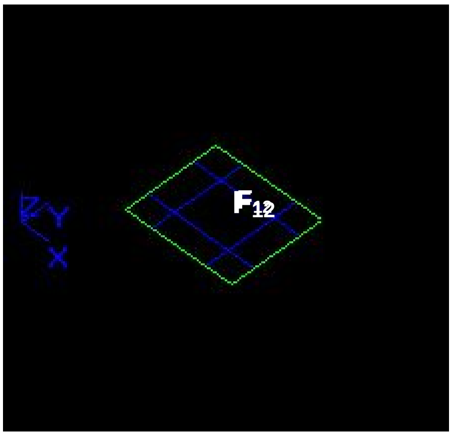
8.2.4. Example 4

The arguments are (Figure 37)

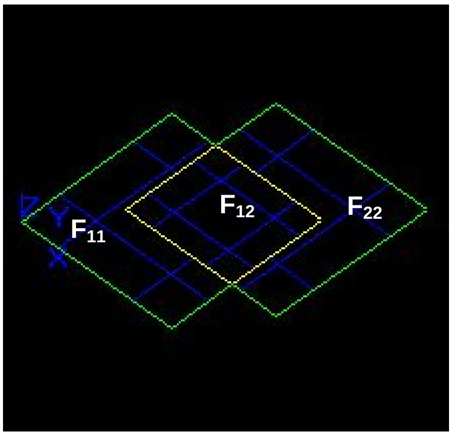
- Object: face F_1
- Tool: face F_2

Arguments **B_{common}**

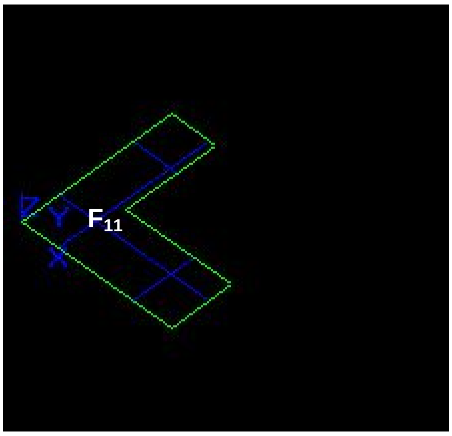




B_{fuse}



B_{cut12}



B_{cut21}

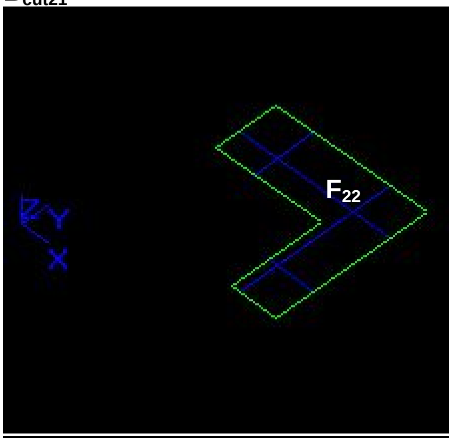


Figure 37

The result of BOA operation is compound.

B_{common}	The compound contains new face F_{12}.
B_{fuse}	The compound contains a shell of 3 new faces F_{11}, F_{12}, F_{22}.
B_{cut12}	The compound contains new face F_{11}.
B_{cut21}	The compound contains new face $F_{22.}$

9. LIMITATIONS OF ALGORITHMS

The chapter is devoted to the problems that should be considered as limitations of Algorithms. In most cases the reason of failure is a superposition of different factors (self-interfered arguments, inappropriate, ungrounded values of the tolerances of the arguments, adverse mutual position of the arguments, tangency, etc).

The description below is just to illustrate the limitations and do not exhaust all problems that can be faced in practice.

Each case of failure of Algorithms is the matter of the maintenance service.

9.1. ARGUMENTS

9.1.1. Common requirements

Each argument should be valid (in terms of *BRepCheck_Analyzer*), or conversely, if the argument is considered as not-valid (in terms of *BRepCheck_Analyzer*), it can not be used as an argument of the algorithm.

The class *BRepCheck_Analyzer* is to check the overall validity of a shape. For a shape (or its sub-shapes) to be valid in Open CASCADE, it must respect certain criteria. If the shape is determined as not valid, the problems can be fixed by tools from *ShapeAnalysis, ShapeUpgrade, ShapeFix* packages.

The class *BRepCheck_Analyzer* is just hand-made tool that has its own problems. Examples of such problems:

- The *Analyzer* checks the distances between the two 3D-points P_i, PS_i of an edge on a face F. The point P_i is obtained from 3D-curve (at the parameter t_i) of the edge. PS_i is obtained from 2D-curve (at the parameter t_i) of the edge on surface S of the face F. To be valid the distance should be less then Tol(E). The number of these check-points is constant value (say 23). It is supposed that edge E is valid (in terms of *BRepCheck_Analyzer*).
- Split the edge E onto two split edges E_1, E_2. Each split edge has 3D-curve and 2D-curve as the edge E has. Let's check E_1 (or E_2). The *Analyzer* again checks the distances between the two 3D-points P_i, PS_i. The number of these check-points is again constant value (23). But there are not guarantee that the distance should be less then Tol(E), because the points for E_1 are not the same as for E.
- So, in case when E_1 is not valid, the edge E also should not be valid. But E is supposed as valid. Thus the Analyzer is wrong for E.

The fact that the argument of Algorithm is valid shape (in terms of *BRepCheck_Analyzer*) is necessary requirement but not sufficient to produce valid result of the Algorithms.

SALOME Platform

9.1.2. Pure self-interference

The argument should not be self-interfered, i.e. all sub-shapes of the argument that have geometrical coincidence through any topological entities (vertices, edges, faces) should share these entities.

9.1.2.1. Example 1

The compound of two edges E_1, E_2 (Figure 38) is self-interfered shape and can not be used as the argument of the Algorithms.

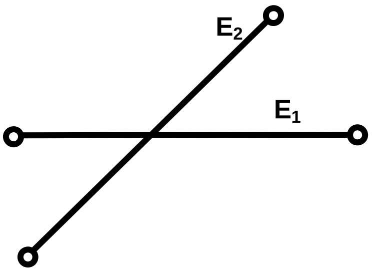


Figure 38

9.1.2.2. Example 2

The edge E (Figure 39) is self-interfered shape and can not be used as the argument of the Algorithms.

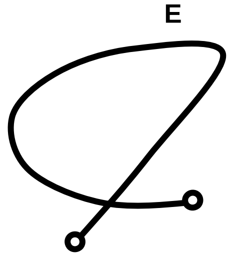


Figure 39

9.1.2.3. Example 3

The face F (Figure 40) is self-interfered shape and can not be used as the argument of the Algorithms.

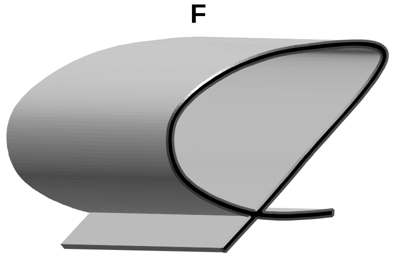


Figure 40

9.1.2.4. Example 4

The face F (Figure 42) has been obtained by revolution of the edge E around the line L (Figure 41)

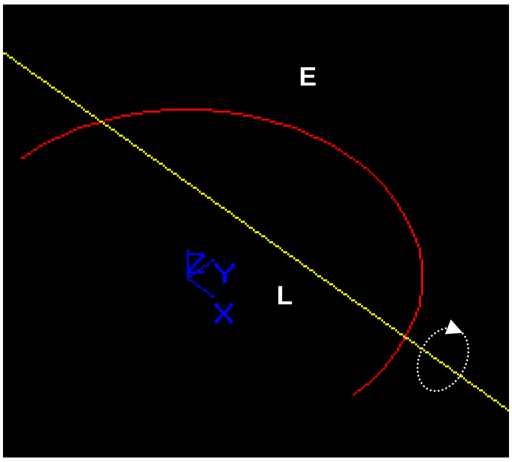


Figure 41

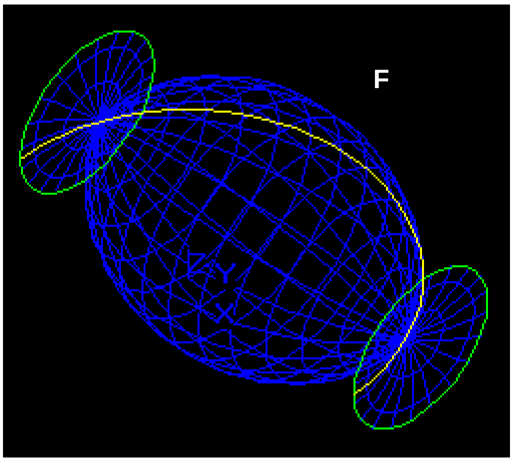


Figure 42

In spite of the fact that the face F is valid (in terms of *BRepCheck_Analyzer*) the face F is self-interfered shape and can not be used as the argument of the Algorithms.

9.1.3. Self-interferences due to tolerances

9.1.3.1. Example 1

The edge E (Figure 43) is based on non-closed circle.

The distance between the vertices of E is D=0.69799.

The values of the tolerances Tol(V_1)=Tol(V_2)=0.5 (Figure 44).

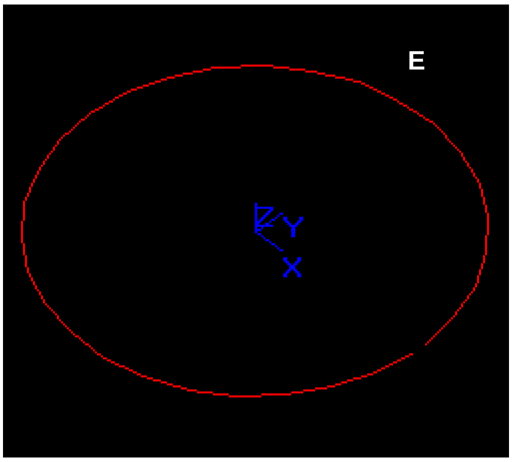


Figure 43

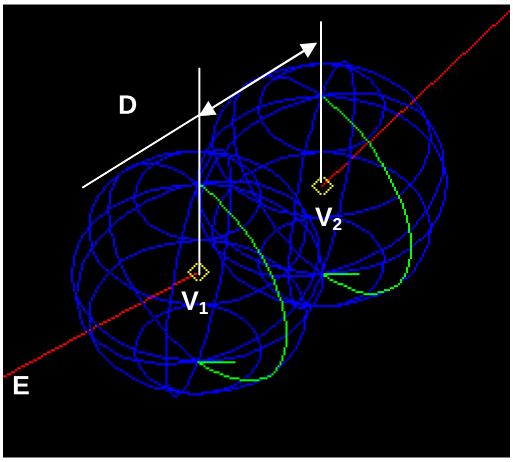


Figure 44

In spite of the fact that the edge E is valid (in terms of *BRepCheck_Analyzer*) the edge E is self-interfered shape because its vertices are interfered. Thus the edge E can not be used as the argument of the Algorithms.

9.1.3.2. Example 2

The solid S (Figure 45) contains the vertex V. The value of the tolerance Tol(V)= 50.000075982061 (Figure 46). [pkv/901/A9]

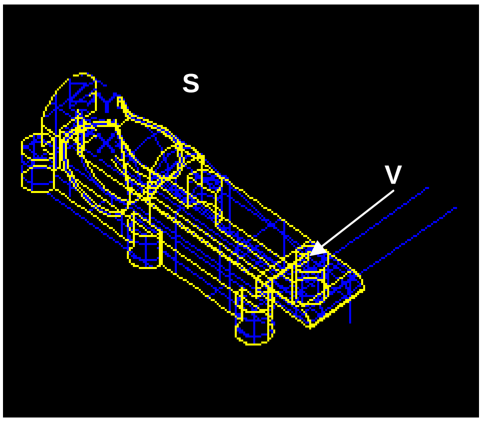


Figure 45

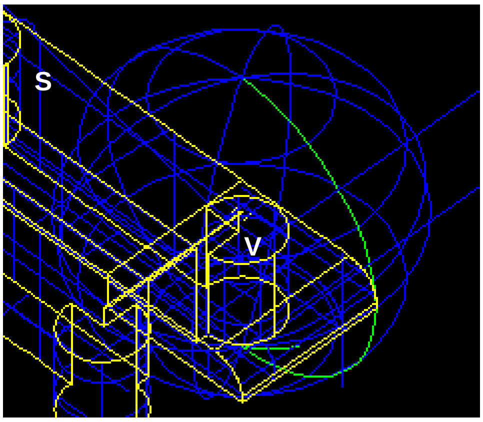


Figure 46

In spite of the fact that the solid S is valid (in terms of *BRepCheck_Analyzer*) the solid S is self-interfered shape because the vertex V is interfered with a lot of sub-shapes of S without any topological connection with them. Thus the edge E can not be used as the argument of the Algorithms.

9.1.4. Parametric representation

The parametrization of some surfaces (cylinder, cone, surface of revolution) can be the cause of limitations.

9.1.4.1. Example 1

The parameterization range for cylindrical surface is:

$$\mathbf{U: [0, 2*\pi], V: }]-\infty, +\infty[\qquad (9.1.4)$$

The range on U coordinate is always restricted; meanwhile the range on V coordinate is non- restricted.

For clearness

- The face (cylinder-based, radii R=3, H=6) (Figure 47). The Figure 48 shows p-Curves for the cylinder.

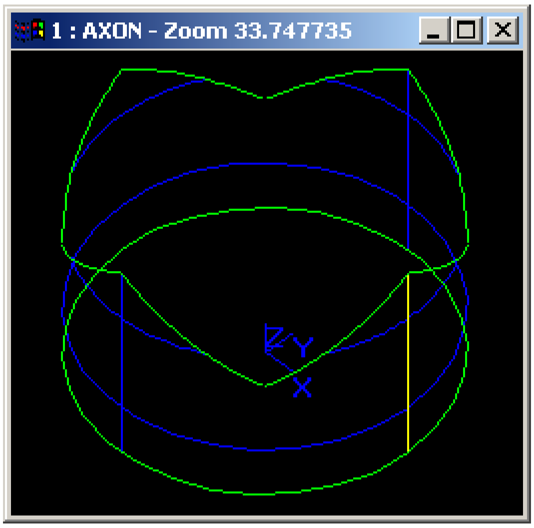


Figure 47

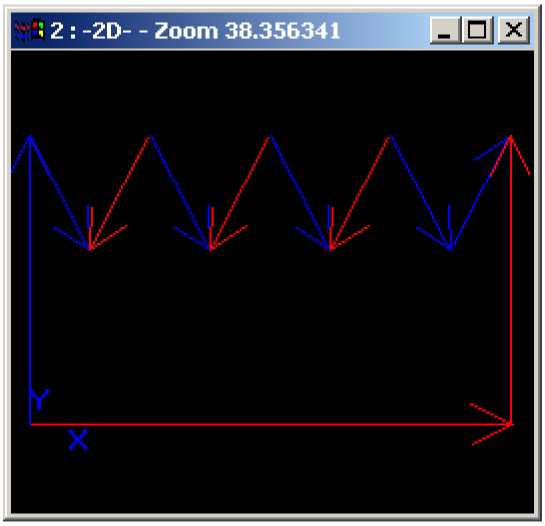


Figure 48

- The face (cylinder-based, radii R=3000, H=6000, scale factor **ScF=1000**) (Figure 49). The Figure 50 shows p-Curves for the cylinder. Please, draw attention on Zoom value on the Figures.

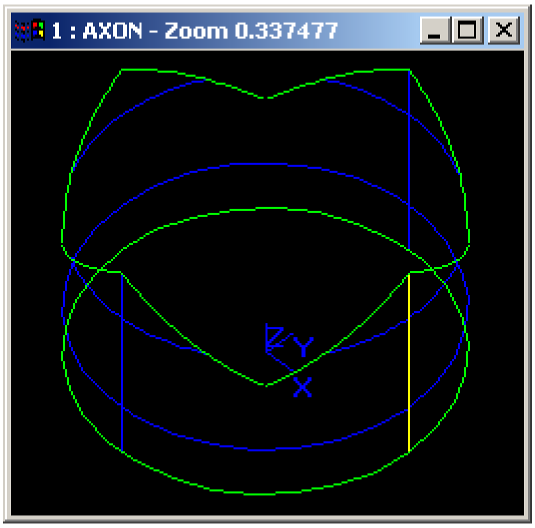


Figure 49

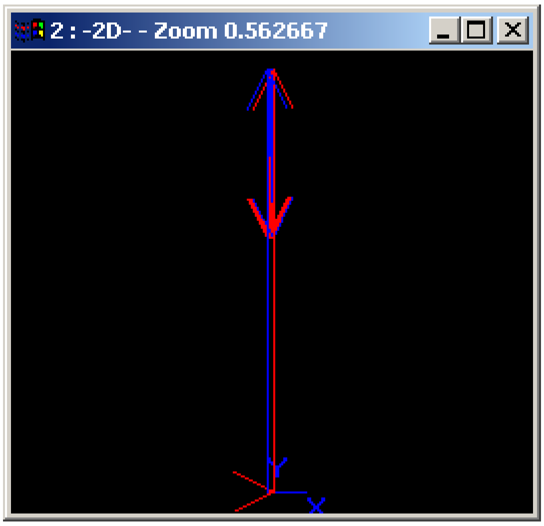


Figure 50

It is evident that starting with some value of **ScF** (say **ScF>**1000000.) all sloped p-Curves on the Figure 48 will be almost vertical (like on the Figure 50). At least there will be not difference between the values of its angles computed by standard C Run-Time Library functions like *double acos(double x)*. The loss of accuracy in computation of angles can be the cause of failure of some sub-algorithms of BP (building faces from set of edges, building solids from set of faces)

9.1.5. Stop-gaps

Due to tolerance model it is possible to create shapes that use sub-shapes of lower order to stop gups.

9.1.5.1. Example 1

The face F has two edges E_1, E_2 and two vertices, plane {0,0,0, 0,0,1} (Figure 51).

- The edge E_1 is based on line {0,0,0, 1,0,0} Tol(E_1)=1.e-7.
- The edge E_2 is based on line {0,1,0, 1,0,0} Tol(E_2)=1.e-7.
- The vertex V_1, point {0,0.5,0} Tol(V_1)=1.
- The vertex V_2, point {10,0.5,0} Tol(V_2)=1.

The face F is valid (in terms of *BRepCheck_Analyzer*).

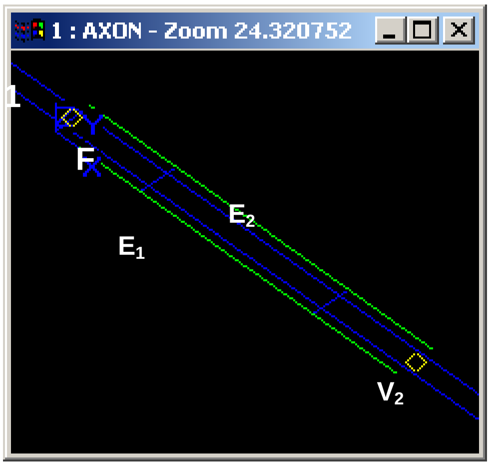


Figure 51

The values of tolerances Tol(V_1), Tol(V_2) are big enough to fix the gups between the ends of the edges. But the vertices V_1, V_2 does not contain an information about trajectories of connection between corresponding ends of the edges. The trajectories are undefined. The fact will be the cause of failure of some sub-algorithms of BP. The sub-algorithms for building faces from set of edges for e.g. use information about all edges connected in a vertex. The situation when one vertex will have several pairs of edges of kind above will not be solved in right way.

9.2. INTERSECTION PROBLEMS

9.2.1. *Pure intersections and common zones*

9.2.1.1. Example 1

Intersection between two edges (Figure 52):

- E_1 – based on line: {0,-10,0, 1,0,0} , Tol(E_1)=2.
- E_2 – based on circle: {0,0,0, 0,0,1}, R=10, Tol(E_3)=2.

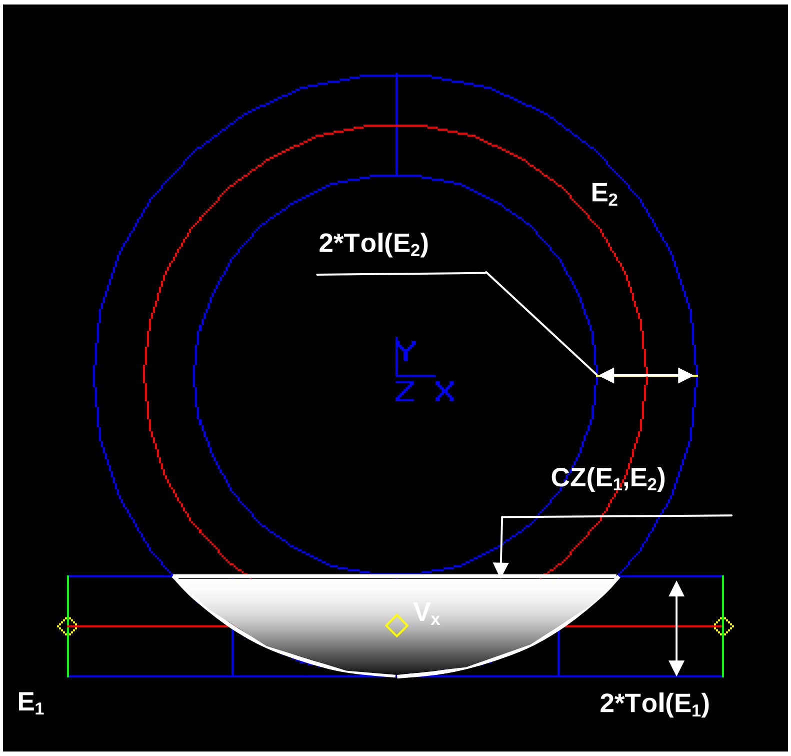


Figure 52

The result of pure intersection between E_1 and E_2 is vertex Vx {0,-10,0}.

The result of intersection taking into account tolerances is the common zone CZ (part of 3D-space where the distance between the curves is less than (or equals to) sum of tolerances of the edges.

As for IP of Algorithms it uses result of pure intersection Vx instead of CZ, because of the reasons:

- The Algorithms does not produce Common Blocks between edges based on underlying curves of *explicitly different type* (*Line / Circle* for this case). The rule of thumb is the following: the different type of curves is special sign to produce the result of type "vertex". The rule does not work for non-analytic curves (*Bezier, B-Spline*) and theirs combinations with analytic curves.
- The algorithm of intersection between two surfaces (*IntPatch_Intersection*) does not compute CZ of intersection curve (-s), point (-s). So even if CZ was computed by Edge/Edge intersection algorithm, its result could not be treated by Face/Face intersection algorithm.

SALOME Platform

9.2.2. Tolerances

The limitations are due to modeling errors or inaccuracies.

9.2.2.1. Example 1

The arguments are vertex V_1 and edge E_2 (Figure 53).

The vertex V_1 interferes with the vertex V_{12}. The vertex V_1 interferes with the vertex V_{22}.

So the vertex V_{21} should interfere with the vertex V_{22}.

But it is impossible because the vertices V_{21}, V_{22} are the vertices of the edge E_2, thus $V_{21} \neq V_{22}$.

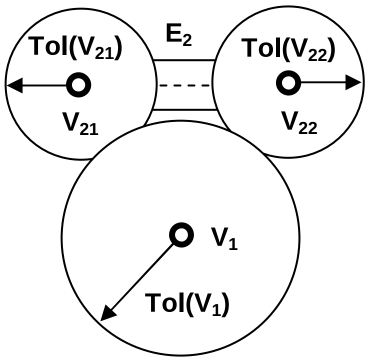


Figure 53

The problem can not be solved in general, because the length can be as small as possible to provide validity of E_2 (in extreme case: Length $(E_2)=Tol(V_{21}) + Tol(V_{22}) + \varepsilon$, $\varepsilon \to 0$).

In particular case the problem can be solved by decreasing the values $Tol(V_{21})$, $Tol(V_{22})$ $Tol(V_1)$ (refinement of arguments).

It is easy to see that if the E_2 were slightly above than tolerance sphere of V_1 the problem wouldn't appear at all.

9.2.2.2. Example 2

The arguments are two planar rectangular faces F_1, F_2 (Figure 54).

Intersection curve between the planes is the curve C_{12}. The curve produces new intersection edge EC_{12}. The edge goes through the vertices V_1, V_2 thanks to big values of the tolerances of the vertices $Tol(V_1)$, $Tol(V_2)$. The situation is: two straight edges (E_{12}, EC_{12}) go through the two vertices. It is impossible for this case.

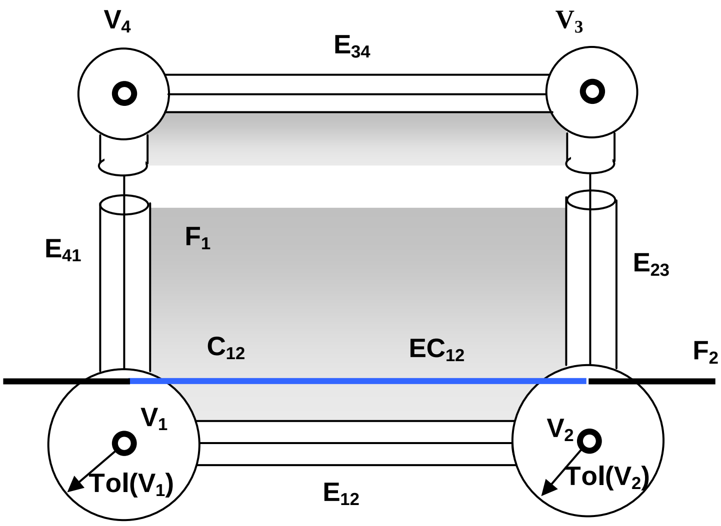


Figure 54

The problem can't be solved in general, because the length of E_{12} can be infinite and the values of $Tol(V_1)$ and $Tol(V_2)$ theoretically can be infinite too.

In particular case the problem can be solved by several ways:

- Decrease if possible the values $Tol(V_1)$, $Tol(V_2)$ (refinement of F_1).
- Analysis of the value of $Tol(EC_{12})$. Increase $Tol(EC_{12})$ in order to get common part between the edges EC_{12}, E_{12}. The common part then will be rejected as already existing edge E_{12} for the face F_1.

It is easy to see that if the C_{12} were slightly above than tolerance spheres of V_1, V_2 the problem wouldn't appear at all.

9.2.2.3. Example 3

The arguments are two edges E1, E2 (Figure 55):

- The edges E_1, E_2 have common vertices V_1, V_2.
- The edges E_1, E_2 have 3D-curves C_1, C_2.
- $Tol(E_1)$=1.e-7, $Tol(E_2)$=1.e-7

C_1 practically coincides in 3D with C_2. The value of deflection is Dmax (say Dmax=1.e-6)

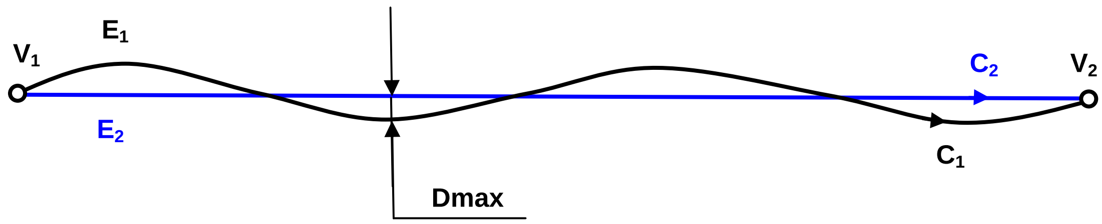


Figure 55

The evident and prospective result should be Common Block between E_1, E_2. But the result of intersection is on the Figure 56

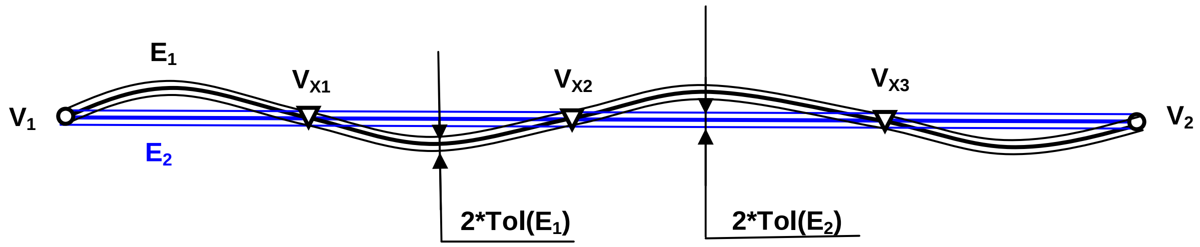


Figure 56

The result contains three new vertices Vx_1, VX_2, Vx_3 and 8 new edges (between whiles V_1, Vx_1, VX_2, Vx_3, V_2) and no Common Blocks. The result (Figure 56) is quite correct due to source data: $Tol(E_1)$=1.e-7, $Tol(E_2)$=1.e-7, Dmax=1.e-6.

In particular case the problem can be solved by several ways:

- Increase if possible the values $Tol(E_1)$, $Tol(E_2)$ to get coincidence in 3D between E_1, E_2 in terms of tolerance.
- Replace E_1 by more by the more accurate model.

The example can be extended from 1D (edges) to 2D (faces) (Figure 57).

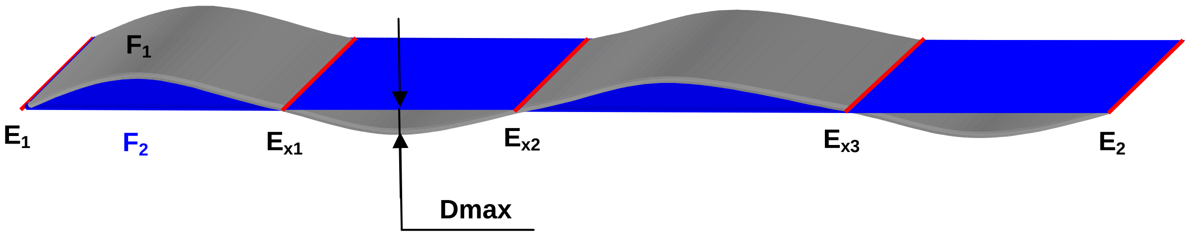


Figure 57

The comments and recommendations are the same as for 1D case.