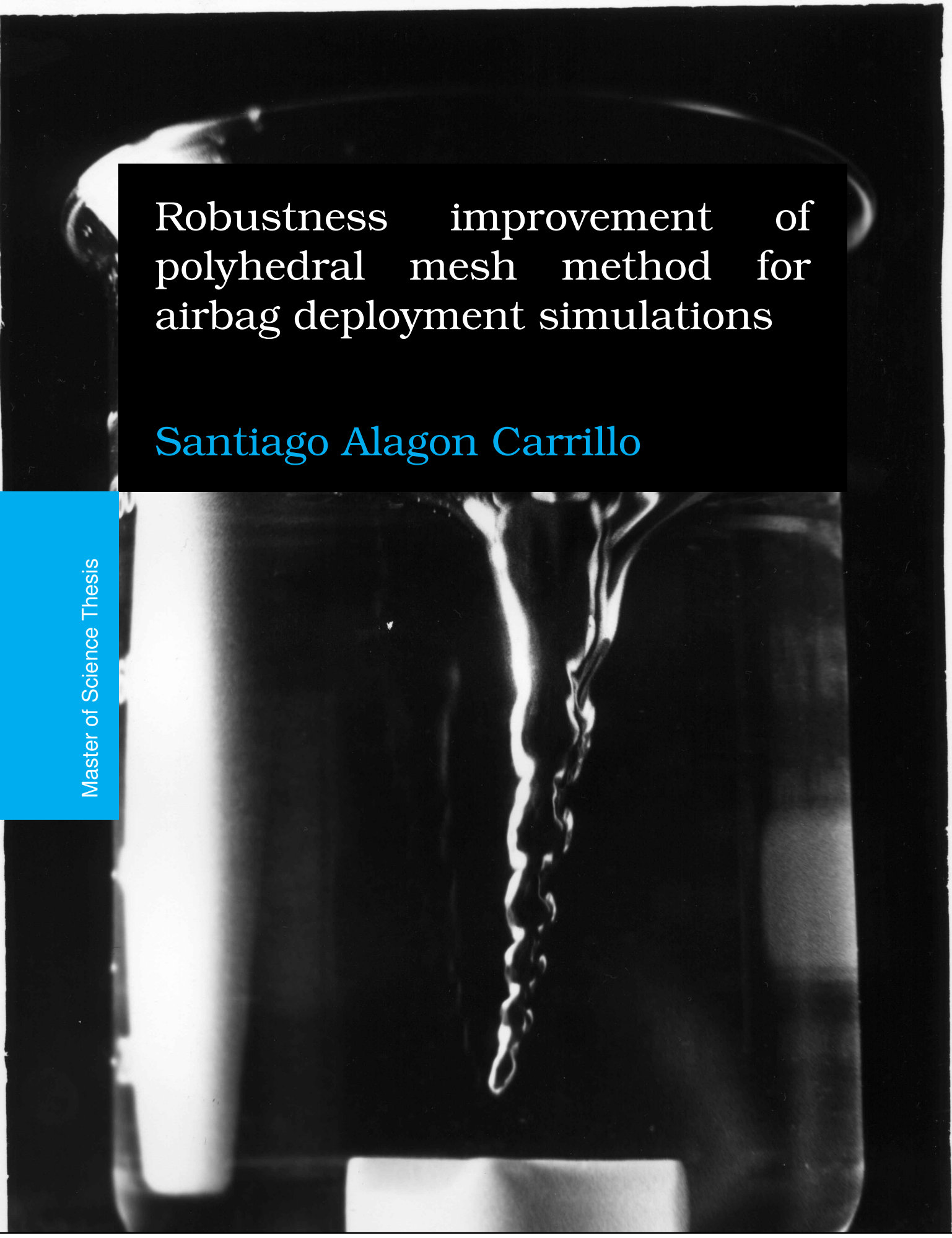
Robustness improvement of polyhedral mesh method for airbag deployment simulations

Santiago Alagon Carrillo

Master of Science Thesis



TU Delft Delft University of Technology

Numerical Analysis Department

Robustness improvement of polyhedral mesh method for airbag deployment simulations

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Applied Mathematics at Delft University of Technology

Santiago Alagon Carrillo

August 20, 2014

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) · Delft University of Technology

The work in this thesis was supported by TASS International . Their cooperation is hereby gratefully acknowledged.

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF
NUMERICAL ANALYSIS

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) for acceptance a thesis entitled

ROBUSTNESS IMPROVEMENT OF POLYHEDRAL MESH METHOD FOR AIRBAG DEPLOYMENT SIMULATIONS

by

SANTIAGO ALAGON CARRILLO

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE APPLIED MATHEMATICS

Dated: August 20, 2014

Supervisor(s):

Prof. dr. ir. C. Vuik

Dr. ir. M.R. Lewis

Reader(s):

Dr. ir. D.R. van der Heul

Dr. ir. W.T. van Horssen

Ir. E.H. Tazammourti

Abstract

In this project the problem of determining the inner and outer regions of the computational domain for an airbag deployment simulation is addressed. One approach to perform such simulation is via the structural equations of motion for the airbag fabric dynamics, Euler equations of fluid motion for the fluid inside the airbag and a coupling algorithm which defines the dependence between the two systems of equations.

The airbag fabric is discretized as triangular finite elements, and the Finite Volume mesh for the CFD solution inside the airbag is formed by two types of cells: structured cubic cells which have no interaction with the airbag fabric, and unstructured cells which are cubic cells that intersect with the airbag triangulation.

The unstructured cells that intersect the triangulation are called cut-cells and a proper description of their geometry is required to obtain an accurate solution via the finite volume solver. Two geometric properties of the cut-cells are particularly important: the exact geometry, i.e. the areas of the sections of the cell faces inside the flow, which is needed for proper calculations of the fluxes, and the characterization of the regions of the cut-cells as interior or exterior to the flux.

Developing a robust algorithm to determine the inside/outside regions of the cut-cells is the goal of this project.

Table of Contents

List of Figures

Chapter 1

Introduction

Nowadays, design phase in engineering relies strongly on computer simulation to decrease the cost of prototyping and testing, asses feasibility of a project, and to improve a developing product.

Vehicle industry is not the exception. Computer simulation is used to design and test performance and safety of structures and components inside and outside the vehicles. Examples of areas of application are: design of the tools to be used in manufacturing processes, stress analysis of the car-body or component structures, crashworthiness and occupant safety simulation, and fluid simulation of external aerodynamics.

Computational Fluid Dynamics has a wide area of applications in the automobile industry, some of them are: external aerodynamics, in-cylinder flow and combustion, coolant flow, engine compartment and passenger compartment analyses. [10]

A particular use of CFD is in the design process of airbags. This problem has an additional difficulty compared to all other CFD simulations used in vehicle design: airbags do not posses a fixed geometry. The shape of the airbag changes as the flow evolves inside of it and the flow reacts accordingly.

In this section the basic structure of a CFD solver for the coupled simulation of the flow and the airbag structure is presented. The emphasis of this project is on the treatment of the geometric computations at the moving boundaries of the airbag.

1-1 Literature Review. Euler Equations and Finite Volume Method.

The three basic ingredients of a scientific model are, [11]:

- Basic laws: express properties well established by measurements, theory, and observations, for a whole class of systems.
- Constitutive relations: express properties in the same way as basic laws but for a particular system.

- Conservation laws: express the fact that in many systems a particular measurable property of an isolated physical system does not change as the system evolves.

Euler's equations, Equation 1-1, follow from the conservation of mass, conservation of momentum and conservation of energy of inviscid fluids. [12].

$$\frac{d}{dt}\int_{\Omega}\mathbf{q}d\Omega+\int_{S}\Phi(\mathbf{q},\hat{n})dS=\mathbf{S}(\mathbf{q}) \tag{1-1}$$

for

$$\mathbf{q}=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho e\end{pmatrix}, \text{ and } \Phi=\begin{pmatrix}\rho v_n\\ \rho v_n u+p\cdot n_x\\ \rho v_n v+p\cdot n_y\\ \rho v_n w+p\cdot n_z\\ \rho v_n H\end{pmatrix} \tag{1-2}$$

where $\hat{n}=(n_x,n_y,n_z)$ is the outward normal of S, $\mathbf{v}=(u,v,w)$ is the velocity Cartesian vector and $\mathbf{S}(\mathbf{q})$ denotes a general source term.

One approach to simulate the deployment of an airbag is to approximate the solution of the flow inside the airbag using *Euler's equations*, approximate the solution of the movement of airbag fabric using the equations derived from elasticity, and a fluid-structure coupling condition.

The structure-fluid coupling for an inviscid fluid, with structure velocity $\mathbf{v}_s$ and flow velocity $\mathbf{v}_f$, results in

$$\mathbf{v}_f\cdot\mathbf{n}=\mathbf{v}_s\cdot\mathbf{n} \text{ and } p_f=p_s \tag{1-3}$$

A correction needs to be done at the interface with the airbag due to the movement of it produced by the flow. The correction results in the modified *Euler equation's* for the fluid domain given by

$$\frac{d}{dt}\int_{\Omega(t)}\mathbf{q}d\Omega+\int_{S(t)}\Phi(\mathbf{q},\mathbf{v}_s)\cdot\mathbf{n}dS=\mathbf{S}(\mathbf{q}) \tag{1-4}$$

where the flux function through the control volume boundary and the boundary state vectors are defined as

$$\mathbf{q_B}=\begin{pmatrix}\rho\\ \rho(\mathbf{v}_f-\mathbf{v}_s)\\ \rho e_B\\ \rho_1\\ \vdots\\ \rho_{N-1}\end{pmatrix}, \text{ and } \Phi(\mathbf{q},\mathbf{v}_s)=\begin{pmatrix}\rho(\mathbf{v}_f-\mathbf{v}_s)\\ \rho u(\mathbf{v}_f-\mathbf{v}_s)+p\mathbf{I}\\ \rho v(\mathbf{v}_f-\mathbf{v}_s)+p\mathbf{I}\\ \rho w(\mathbf{v}_f-\mathbf{v}_s)+p\mathbf{I}\\ \rho H(\mathbf{v}_f-\mathbf{v}_s)+p\mathbf{v}_s\end{pmatrix} \tag{1-5}$$

where $e_B=h-\frac{p}{\rho}+\frac{1}{2}|\mathbf{v}|^2$

This coupling approach in numerical methods is known as Arbitrary Lagrangian-Eulerian formulation.The discretization of the *Euler equations* is performed through the Method of Lines. This means that the spatial derivatives are replaced by algebraic approximations, using Finite Volumes approach (FVM), to obtain a system of Ordinary Differential Equations where only time remains as a variable. The next step is to apply an integration algorithm for initial value Ordinary Differential Equations to compute the approximate solution of the original Partial Differentia Equation. In other words, the Method of Lines discretizes the spatial and temporal derivatives separately.

The spatial discretization is performed by a Finite Volumes Method over hexahedral cells, the whole computational domain is divided into hexahedral cells which are classified into: active cells, if they belong to the flow region, inactive cells, if they belong outside of the flow region, and cut-cells, which are the cells intersected by the Finite Element triangulation (FE-triangulation) of the air bag. To obtain a second order space discretization, the state of the cell face is computed from the state at the cell center assuming a linear variation.

Boundary conditions for the cells are enforced through the specification of the fluxes at the boundary. For each cell face there are seven possible conditions, [12]

- wall
- subsonic inflow
- sonic inflow
- supersonic inflow
- subsonic outflow
- sonic outflow
- supersonic outflow

The discretization of elasticity equations for the airbag fabric is performed through a Finite Element Method (FEM) using triangular elements.

Conditions (1-3) are implemented through the definitions of the fluxes in Φ as defined in (1-5). The structural velocity at each cell center $\mathbf{x}_c$ is obtained through means of interpolation between the Finite Element nodes according to

$$\mathbf{v}_s(\mathbf{x}_c) = \sum_{i=1}^{3} N_i(\xi)\mathbf{v}_{\mathbf{N,i}}$$

where $\mathbf{v}_{N,i}$ is the nodal velocity and $N_i(\xi)$ is the local shape function evaluated in the FE-element transformed coordinate system.

The pressure in the cut-cells contributes to each of the nodal forces of the FE partition inside the cut-cell. The FE-elements inside each cut-cell are partitioned into N_{sp}-number of segment polygons, which each exert a force

$$\mathbf{f}_{p,j} = p_j(\mathbf{x}_c)\mathbf{n}_j S_j$$

on the FE-element. The total contribution to node i is computed according to

$$\mathbf{f}_{N,i} = \sum_{j=1}^{N_{sp}} N_i(\xi)\mathbf{f}_{p,j}.$$

The coupling is obtained by a standard *loose-coupling* procedure, *i.e.*, at each time step one evaluation of FE and flow solver are performed.

For known locations of the FE-elements, the fluid solution is advanced over one time step returning new contributions to the nodal forces $\mathbf{f}_n$ of the FE solver. The coupling procedure is represented in Figure1-1, [13]

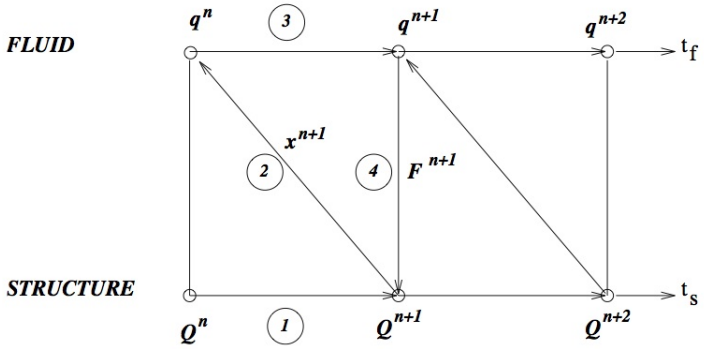


Figure 1-1: Schematic fluid-structure coupling algorithm.

1-2 Literature Review. Finite Volume Discretization

The space discretization for FVM can be performed in two types of meshes: structured and unstructured. In structured meshes, every cell can be referenced by a pair of indexes (i, j), or a triplet (i, j, k) in 3D, and the number of neighbors every cell possess is equal. In unstructured meshes every cell is referenced by a single index (i), [14]. Further more, structured meshes can be body-fitted rectangular or Cartesian immersed body, while unstructured meshes are always body-fitted, see Figure 1-2

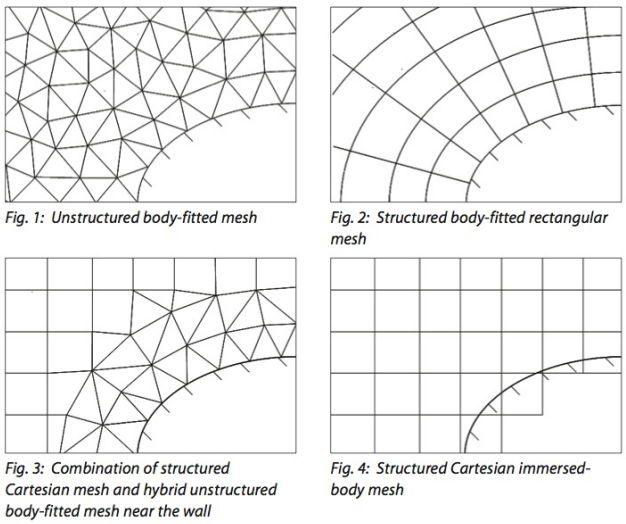


Figure 1-2: Mesh examples, reprint from [1]

Finite Volume Methods are suitable for equations derived from conservative principles because the Finite Volume Formulation is conservative by nature. In the FVM, the terms of the integral form of the equations are evaluated as fluxes at the surface of each grid cell or *finite volume*. The flux is defined in a way that the flux entering the volume is identical in growth or decrease to the flux leaving.

Thus the importance of the precise determination of the geometry of the control volumes into which the space is discretized, [15].

Nowadays the most used grids are *body fitted*, both structured and unstructured, but these techniques cannot be easily implemented in an automatic way and are cumbersome when dealing with complex geometries. Recently another grid generating technique has received more attention, the *Cartesian grids*. These are faster to generate, and have a straightforward implementation for moving boundaries and are suitable for automatic grid generation [16].

Cartesian grid methods differ from body-fitted methods in that they are non-body fitted. The whole domain is divided into a hexahedral grid system, a set of right parallelepipeds, extending through solid walls within the computational domain. The cells are then flagged as *inactive cells*, if they belong to the region where no flow computations will take place, *active cells*, if they belong to the region where the flow computations take place, and *cut-cells* for the cells intersected by the solid walls, see Figure 1-3.

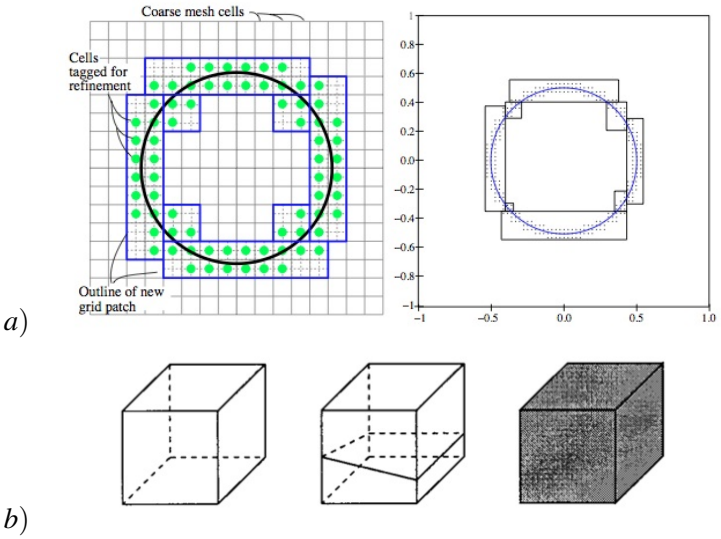


Figure 1-3: (a) Cell flagging; reprint from [2]; (b) Cell classification: flow cell, boundary cell, solid cell; reprint from [3]

This transforms the problem of conforming the mesh to the surface into the problem of characterizing and computing the intersection between the Cartesian grid and the surface geometry.

What distinguishes one type of Cartesian grid method from the other is the way boundary conditions are treated, [17] [5]. One approach is to choose a staircase description of the boundary, see Figure 1-4(c), and impose the boundary conditions using a forcing function and extrapolation of the variables. With this approach the solution on the boundary is smeared out to the width of the local cell and no sharp fluid-to-boundary interphase can be guaranteed. The second approach is defining the so called *cut-cells* by discarding the inactive region of each boundary cell, and using only the active cell region for computations, see Figure 1-4(b).

The remainder of this work is based on a cut-cell formulation of the boundary conditions for a Cartesian mesh where the surface of the immersed body, the airbag, consists of triangular elements obtained from a previous triangulation process.

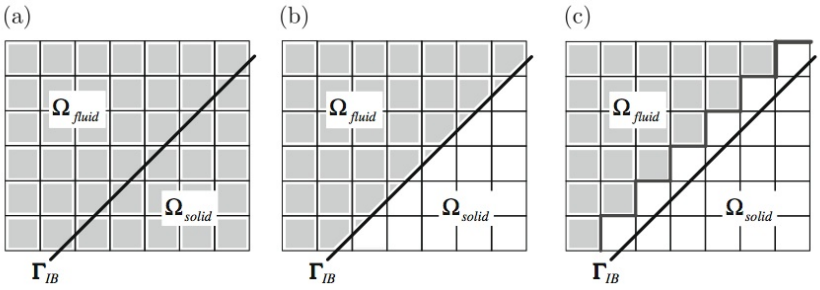


Figure 1-4: (a) Standard scheme, (b) Cut-cell scheme; (c) Staircase implementation; reprint from [4]

As it was already mentioned, the first step in generating a Cartesian grid is performing a cell division of the domain using a uniform hexahedral grid. Once the hexahedral cells have been defined, they are marked as *active cells*, *inactive cells* and *cut-cells*. Usually the inactive cells are discarded.

The regions of the cut-cells into which they are divided by the intersection with the immersed body, in this case the FE-triangulation, also have to be classified as active and inactive. Usually the inactive sections are discarded as well.

The geometry of the immersed boundary is analyzed for each cell and the Cartesian mesh can be locally refined to capture the geometry of the body in a more precise manner.

Cut-cells are intersected by the FE-triangulation in an arbitrary way which leads to complex intersections. Both Cartesian cells and triangles are convex so their intersections results in a convex polygon referred as *triangle-polygon*, *tp*. The edges of the *triangle-polygons* are obtained by clipping the edges of the triangles along the faces of the Cartesian cell resulting in the *face-segments*, *fs*. The division of the faces of the Cartesian cell along their intersection with the surface triangulation result in the *face-polygons*, *fp*. This can be seen for an arbitrary Cut-cell in Figure 1-19

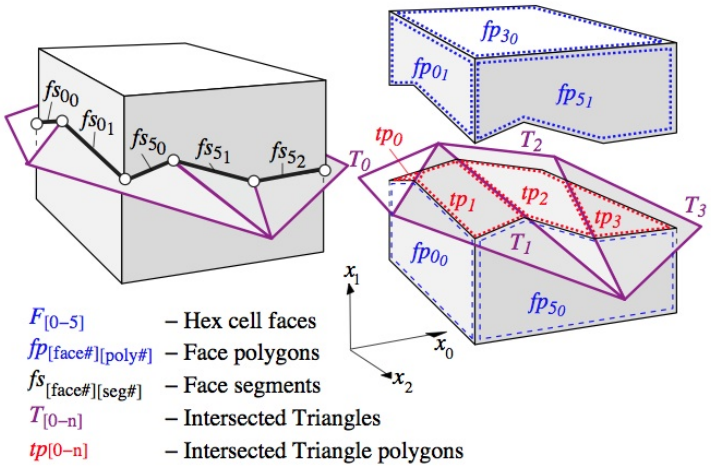


Figure 1-5: Anatomy of a cut cell; reprint from [2]

Because the surface can intersect the Cartesian mesh in an arbitrary way, the above mentioned process may produce small cut-cells which adversely affect the stability of the numerical method. To deal with this problems three procedures are found commonly in literature,*c.f.* [18], [5],[19]

1. *cell merging*, Figure 4-2
2. *cell linking*, Figure 1-7
3. *mixed approach*

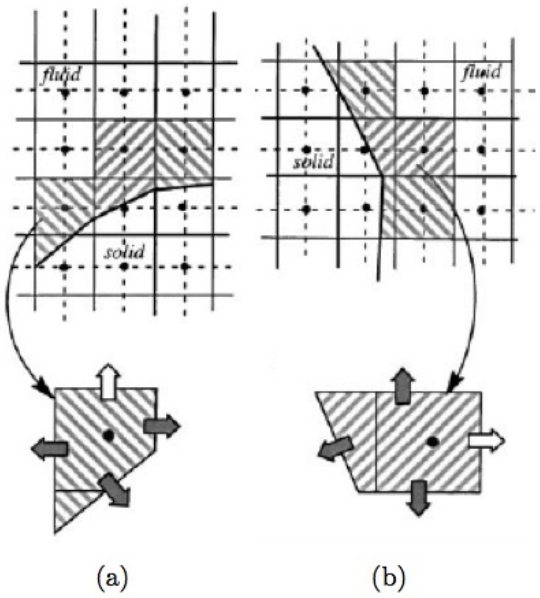


Figure 1-6: Cell merging for small cut-cells; reprint from [5]

Cartesian Meshes possess some disadvantages, *c.f.*[5],[2], [1]:

- Some geometrical features such as trailing edges and leading edges, require many levels of refinement
- Treatment of boundary conditions on irregular cut-cells results in complex coding.
- Cut-cells should not become too small in order to maintain good stability and convergence of the solution.
- Bodies should not be too close, a minimum of two cells apart.

But in general are well balanced by the advantages they possess, *c.f.*[5],[2], [1]:

- Easy to convert into an automatic procedure.

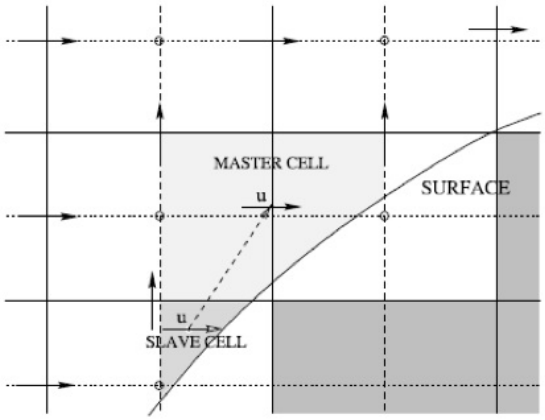


Figure 1-7: Cell linking for small cut-cells; reprint from [5]

- The meshing difficulties are restricted to lower order manifolds.
- It is possible to accurately impose the boundary conditions in the cut-cells.
- Cut-cell treatment of boundaries results in conservative schemes.
- Cut-cells are decoupled from the surface description.
- Easy implementation with adaptively refined grids.
- Away from the surfaces the good quality of the grid, uniform and orthogonal, imply better accuracy, lower discretization errors and better efficiency.
- The meshing process is not linked to a particular representation of the boundaries, (NURB, CAD, triangulation, etcetera). The meshing process may be done using “dirty geometries”, Figure 1-8
- Motion of the boundaries may be pre-programed and the implementation for moving geometries quite straight forward.
- Permit the use of high resolution methods.
- Good for iterative methods.
- Good for multiphase/multimaterial flows.

A Cartesian cut-cell method program is comprised of two mayor components, the flow solver and the geometry calculations. The focus of this work is on the geometry calculation, particularly on the determination of the active and inactive regions of the cut-cells. In computational geometry this problem is know as the *Point-in-Polyhedron* problem, Figure 1-9.

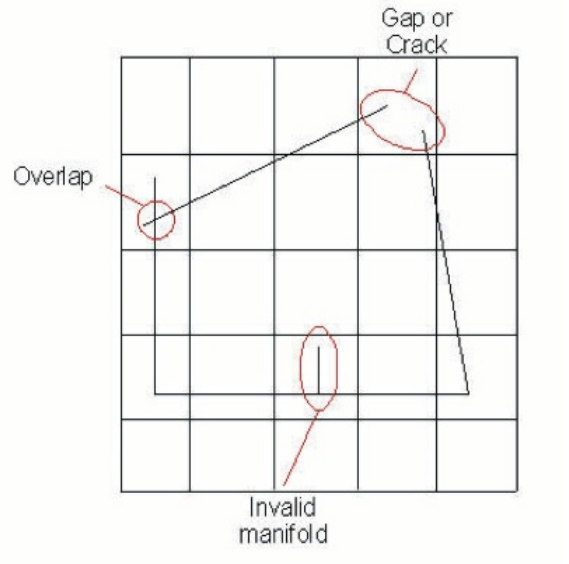


Figure 1-8: Schematics of dirty geometry; reprint from [1]

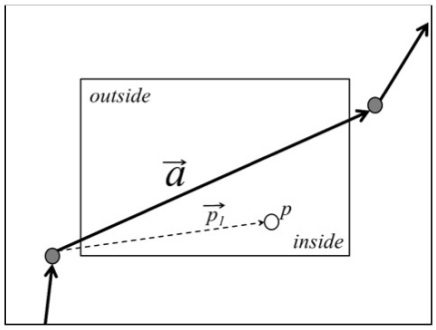


Figure 1-9: Point-in-Polyhedron Problem in meshing; reprint from [6]

1-3 Geometrical Calculations for a Cartesian Grid FVM

A precise determination of the exact geometry for cut-cells and a robust method to determine the active and inactive regions of each cut-cell are important for the accuracy of the Finite Volume solver. In this work the process to obtain the exact geometry of the cut-cells is called the *Search Algorithm*. The process which classifies the cells into, active cells, inactive cells and cut cells,is called the *Coloring Algorithm.*

To avoid a high computational cost, the whole domain is decomposed into *Blocks*, which are sets of neighboring cells, each containing the same number of cells.

Using the *Block* structure the Search Algorithm is performed in three stages.

- *Global search level*. This search determines a list of candidate triangular elements form the FE

triangulation, called the FE-elements, that may intersect each bock.

- *Euler search*. This search determines a set of FE-elements that may intersect every cell.
- *Exact Geometry*. This search determines all intersections between Euler cells and FE-elements using the intersection candidate list. Using this information, the polyhedrons into which a cut-cell is divided by the FE-triangulation are built.

Performing an exact geometry search for each cell would be too expensive computationally.

In order to describe the how the *Search Algorithm* determines the exact geometry, the following definitions based on [20] will be used:

A **point** is an entity that has a location in space but has no other properties: no volume, or dimension, Figure 1-10a. An **edge** is line segment joining two points, called **vertices**, if an edge has a direction associated to it, which means that the vertices have an order which defines them as the initial or final points then the edge is called **directed edge**, Figure 1-10b,c. A **polygonal chain** is a curve specified by a sequence of points, $\{e_1, e_2, ...e_n\}$, so that the curve consists of the edges between consecutive vertices, Figure 1-10a , a polygonal chain is said to be closed if the first and last point of the polygonal chain are linked by and edge. A **polygon**, denoted by P, is the region enclosed by a single closed polygonal chain with all its vertices on a single plane, that does not intersect itself, Figure 1-10d,e . A polygon divides the plane into a bounded region called the **interior** and an unbounded region. A **weakly simple polygon** is polygonal chain embedded in the plane divides it into two regions one of which is topologically equivalent to a disk, some sides can "touch" but not intersect, Figure 1-10e .

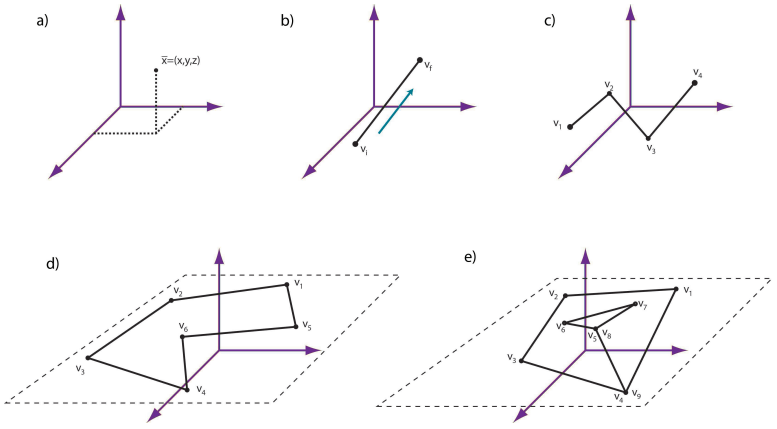


Figure 1-10: Polygon Classification

This work assumes that the polygons are defined in **counterclockwise** fashion **(CCW)**, that is, the **interior** of the polygon is on the left side of each directed edge.

A polygon P_2 is said to be **contained** within another P_1 if it lies in the interior region of P_1 and their boundaries do not touch, Figure1-11

A polygon P_{q_2} is said to be in the n-th level of containment respect to P_{q_1} if there exist a sequence of n polygons $P_{i_1} = P_{q_1}, P_{i_2}, P_{i_3}, \cdots, P_{i_n-1}, P_{i_n} = P_{q_2}$= where for each j, P_{i_j} is contained in $P_{i_{j-1}}$,Figure 1-11.

To be consistent with the counterclockwise description of each general polygon P, where the interior is always to the left of the boundary, in the case of contained polygons the outer contour is oriented counterclockwise fashion and the inner contours, in the first level of containment respect to P, are oriented in clockwise fashion, Figure 1-11

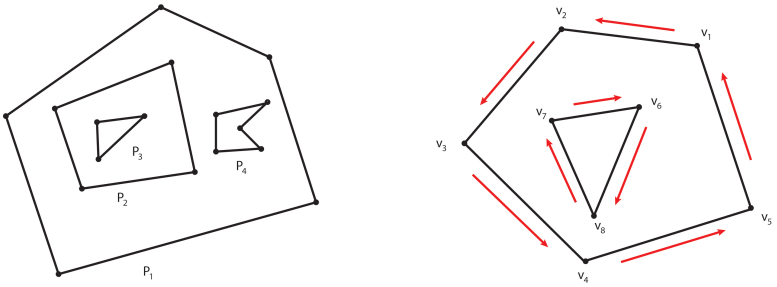


Figure 1-11: Contained polygons, P_3 is in the second level of containment respect to P_1

A **polyhedron**, denoted by $\mathcal{P}$, is a solid bounded by polygonal faces, see Figure 1-12. A polyhedron divides the space into two regions, a bounded region called the interior, and an unbounded region.

A point q is said to be inside $\mathcal{P}$ if q belongs to the interior of any polygon P obtained by the intersection of $\mathcal{P}$ and any plane containing q.

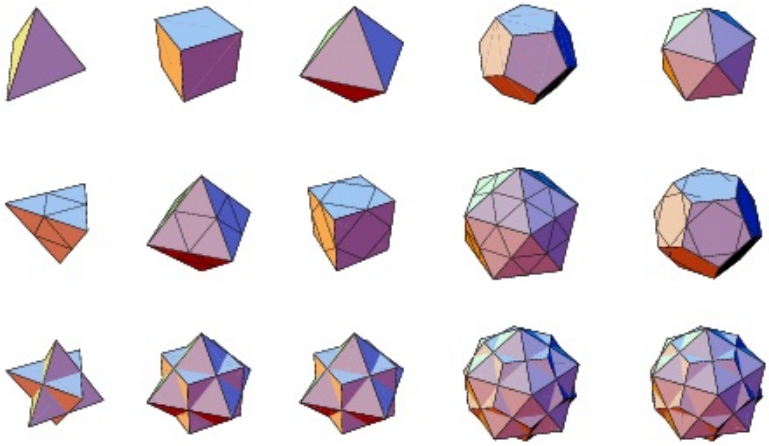


Figure 1-12: Examples of polyhedrons; reprint from [7]

To define a CCW description of a polyhedron we make use of what we already know for polygons, as it is described next.

A set of points in the two dimensional plane, in the case of this work a polygon, is said to be **convex** if for every two points, x_i and x_j, belonging to the set, every point in the straight line joining x_i and x_j belongs also to the set.

The **convex hull** of a polygon P is the smallest convex polygon cP whose vertices are a subset of the vertices of P and that contains completely P, see Figure 1-13 .

Given a polygon P and its convex hull cP, the outward pointing normal to P is defined as the unitary vector

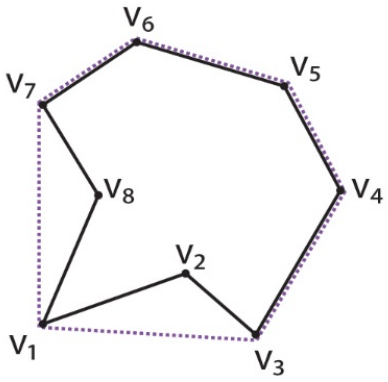


Figure 1-13: Polygon (continuous line) and its convex hull(dashed line).

.

$$\hat{n}_P = \frac{e_{i+1} \times e_i}{||e_{i+1} \times e_i||} \tag{1-6}$$

for any two non co-linear neighboring edges e_{i+1} and e_i of cP where the order of the indices is determined by the CCW description of cP. This vector is orthogonal to the plane where P lies.

A polyhedron $\mathcal{P}$ is said to be described in CCW fashion if for all its polygonal faces P_i there exists a point x in the interior of $\mathcal{P}$ arbitrarily close to P_i such that, from the point of view of x, P_i is described in CCW fashion

A polygon is topologically equivalent to a disk and a polyhedron is topologically equivalent to a sphere.

1-3-1 Exact Geometry

The algorithm that marks the active and inactive regions of each cut-cell is known as the *Coloring Algorithm*.To understand how certain implementations of the *Coloring Algorithm* can create robustness issues, in this section a review of the *Exact Geometry* search level of the *Search Algorithm* is presented.

As a result of the the *Global* search and *Euler* search, each cut-cell has a list containing a set of candidate FE-elements intersecting it. The purpose of the *Exact Geometry* search level is to detect if a certain FE-element of the list of candidates really intersect the cell and, if so, to calculate the intersection points to construct the geometry of the cut-cell.

To test wether a certain segment intersects a cell, it is straight forward to think of a verification process in terms of geometry. For example, to calculate the intersection point of a certain FE-element's edge with one of the cell's faces one can think of solving the system for the intersection of the plane containing the cell's face and the line containing the FE-elements's edge. This geometrical analysis also builds the intersection point in the process. The problem with geometrical methods is that they require the construction of "new geometry", called *constructors*, which have different precision than

the given data, are subject to round-off errors which at the end result in unknown accuracy for the intersection points, and are computationally expensive. [2].

Considering the problems that arise from solving the intersection problem in terms of geometrical descriptions, it is desired to first build topological tools to verify the actual occurrence of an intersection and if such an intersection really happens then to build the intersection point using geometrical tools. This way the use of intersection constructors is kept to a minimum and also their effect on accuracy and on the use of computational resources. After all, the intersection problem is a question of topology, not of geometry.

To achieve this, two topological tests are used

- *Edge test*. For each FE edge test wether it intersected by a certain cell's face, see Figure 1-14b.
- *Slice test*. For each FE segment test wether it is intersected by a certain cell's edge, see Figure 1-14c.

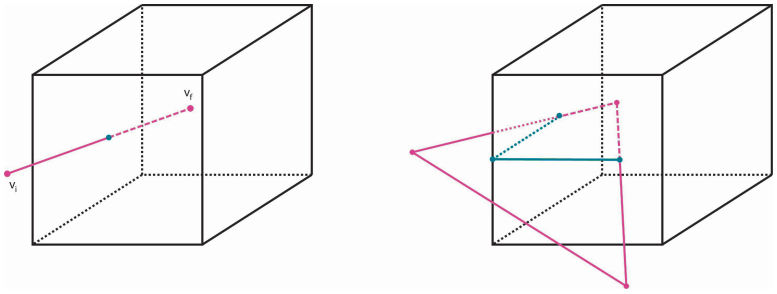


Figure 1-14: Exact Tests

In order to understand how these tests work, the concept of the *Signed Volume* for a tetrahedron, presented in the next paragraph, needs to be known.

Signed Volume of a Simplex

A simplex is the generalization of a triangle to n-dimensions. The simplex in 3 dimensions is a tetrahedron.

A known property for simplices is the computation of the volume in determinant form. This property states that the volume $V(T)$ of the simplex T with vertices $(v_0, v_1, \dots v_d)$ in d-dimensions is:

$$V(T_{v_0 v_1 \dots v_d}) = \frac{1}{d!} \begin{pmatrix} v_{0_0} & v_{0_1} & \dots & v_{0_{d-1}} & 1 \\ \dots & \dots & \dots & \dots & \dots \\ v_{d_0} & v_{d_1} & \dots & v_{d_{d-1}} & 1 \end{pmatrix} \tag{1-7}$$

which, for a tetrahedron T_{012a} reads:

$$V(T_{012a}) = \frac{1}{3!} \begin{pmatrix} v_{0_0} & v_{0_1} & v_{0_2} & 1 \\ v_{1_0} & v_{1_1} & v_{1_2} & 1 \\ v_{2_0} & v_{2_1} & v_{2_2} & 1 \\ v_{a_0} & v_{a_1} & v_{a_2} & 1 \end{pmatrix} \tag{1-8}$$

This volume is positive if the triangle $\Delta_{0,1,2}$ forms a counterclockwise circuit when viewed form a point located on the side of the plane defined by $\Delta_{0,1,2}$ which is opposite from d, *c.f* [2], see Figure 1-15

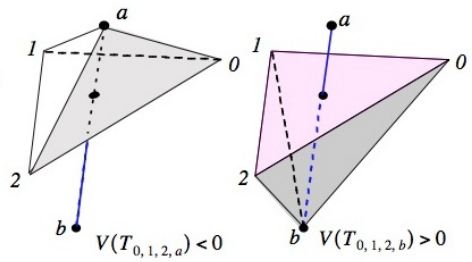


Figure 1-15: Signed Volume Property; reprint form [2]

The half space where the point a is with respect to the plane containing $\Delta_{0,1,2}$ can be determined by the sign of the volume obtained via the determinant. If this volume is zero the point a is coplanar with $0, 1, 2$.

Using the signed volume property, two tests are defined, the *Pierce test* and the *Inside test*

Pierce test

This test determines weather a line segment between two points a and b pierces trough the plane defined by a triangle $\Delta_{0,1,2}$.

Applying the signed volume test to the triangle $\Delta_{0,1,2}$ and the segment $\overline{ab}$, see Figure 1-16, $\overline{ab}$ crosses the pane if and only if the signed volumes $V(T_{012a})$ and $V(T_{012b})$ have opposite signs

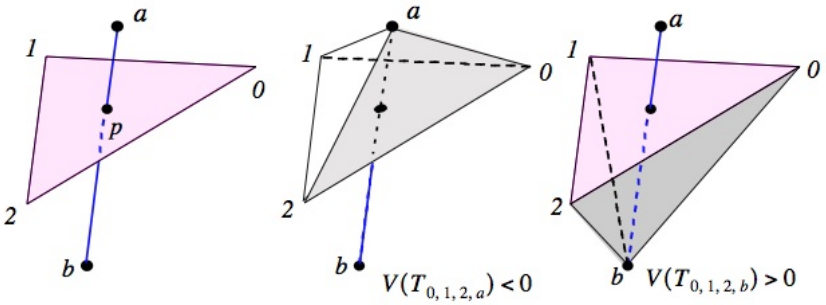


Figure 1-16: Pierce test; reprint form [2]

Inside test

Once it has been verified that the segment $\overline{ab}$ indeed pierces through the plane where the triangle $\Delta_{0,1,2}$ resides, what need to be determined is if the segment pierces inside the triangle.

According to the signed volume property, piercing occurs inside the triangle if the the volume of the tetrahedrons connecting the end points of segment $\overline{ab}$ with the end points of the edges of the triangle $\Delta_{0,1,2}$ have all the same sign, that is if:

$$[V(T_{a,1,2,b}) < 0 \wedge V(T_{a,0,1,b}) < 0 \wedge V(T_{a,2,0,b}) < 0]$$

or

$$[V(T_{a,1,2,b}) > 0 \wedge V(T_{a,0,1,b}) > 0 \wedge V(T_{a,2,0,b}) > 0]$$

see Figure 1-17 for a graphic representation of this test.

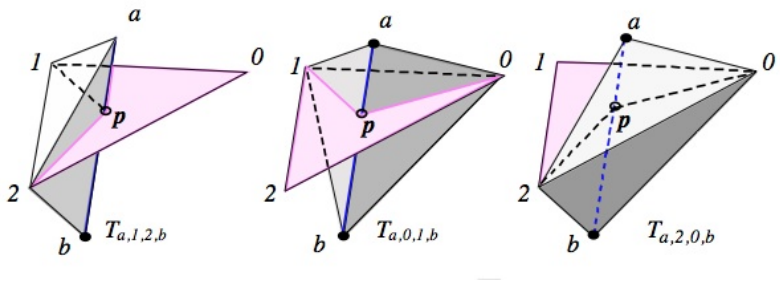


Figure 1-17: Inside test, reprint form [2]

Topological Characteristics

To obtain the topological characteristics of a cut-cell the first step is to apply the *Edge test* to determine which cell faces are pierced by which edges of the FE-elements from the candidate list.

For each of the FE-edges and for each of the cell's faces, the *Edge Test* is performed in two stages. First the *Pierce Test* is used to determine which of the cell face's planes the edge pierces. Second, the *Inside Test* is performed to determine if the segment pierces the plane inside the cell's face.

Then it needs to be determined is which of the triangular FE-elements of the FE-mesh cut each of the Euler cell's edges. This is done using the *Slice Test*.

The *Slice Test* is performed in two stages. First the *Pierce Test* is performed to determine if a certain Euler cell's edge pierces trough any of the FE-element's planes. Second, the *Inside Test* is performed to determine if the cell's edge pierces inside the FE-element.

Degenerate cases There exist six cases where the *Pierce Test* or *Slice Test* may fail:

1- FE-node coincides with a face, see Figure 1-18a.

2- FE-node coincides with a cell edge, see Figure 1-18b.

3- FE-node coincides with cell vertex, see Figure 1-18c.

4- FE-edge intersects with a cell edge, see Figure 1-18d.

5- FE-edge intersects with a cell vertex, see Figure 1-18e.

6- FE-element intersects with a cell vertex, see Figure 1-18f.

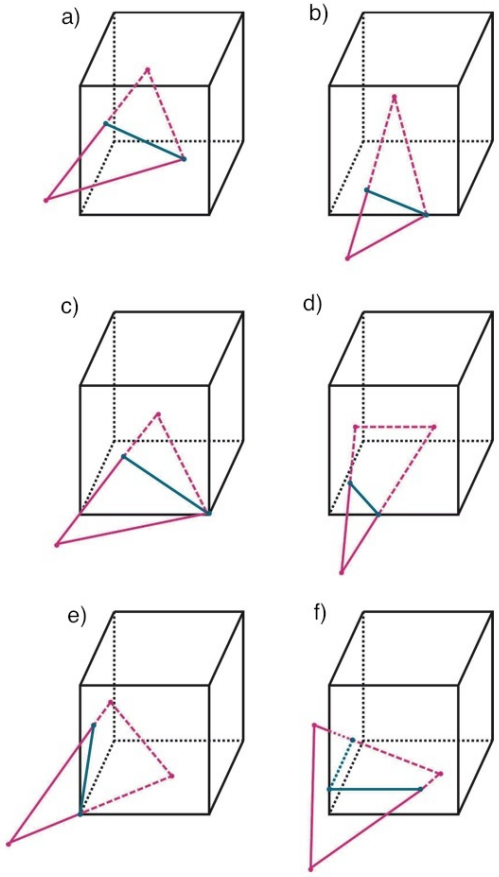


Figure 1-18: Degenerate cases

When cases 1, 2 or 3 occur the *Edge test* concludes no piercing because one of the signed volumes is zero. When cases 2 and 3 are found also the *Slice test* fails because one of its components gives a zero volume.

When cases 4 and 5 occur the *Edge test* fails because one of the components of the *Inside test* has a zero volume.

When case 6 occurs the *Edge test* fails because one of the components of the *Pierce test* has zero volume.

In this cases the computation of the polypoint is performed knowing that the location of the polypoint will be coplanar with the cell face.

Geometrical Characteristics

After applying the above mentioned tests, the FE-edges intersecting each cell face it are known, as well as the FE-elements cutting each of the cells' edges. The points where an FE-edge pierces a cell's face and the points where a cell's edge intersects a FE-element are known as **polypoints**, denoted by pp_i. To build the exact geometry of the cut-cells the polypoints have to be calculated.

Geometrical constructors were originally rejected because the problem of wether or not an edge pierces a plane inside a certain region is a problem of topology and not of geometry, but once the existence of polypoints has been established geometrical constructors should be used to obtain the actual location of the polypoints, [2], and the polyhedrons into which the cut-cells are divided by the airbag.

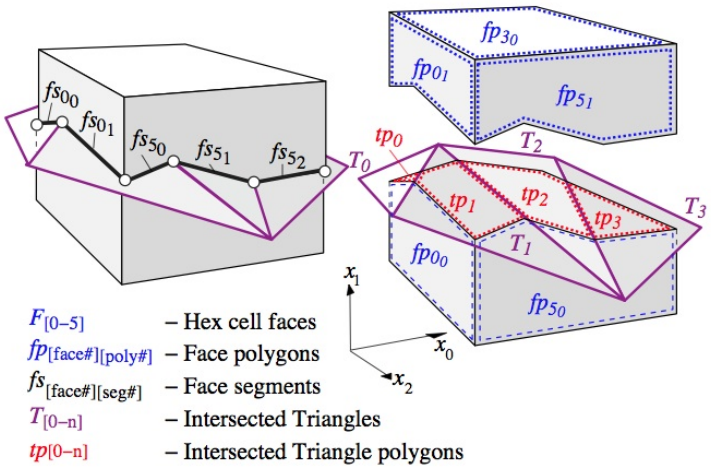


Figure 1-19: Anatomy of a cut cell; reprint from [2]

To describe the geometry of the cut-cells the following definitions will be employed.

A FE-element is said to *slice a cell face* if at least one vertex v_i of the FE-element is in one half-space into which the whole space is divided by the cell face's plane, a second vertex v_j of the same FE-element is on the other half-space, and the intersection between the FE-element and the cell face, result in a section of straight line called **face segment**, fs, see Figure 1-20a. To distinguish this type of face segments from other types which will be mentioned, the prefix s is used, **s-face segment**, sfs.

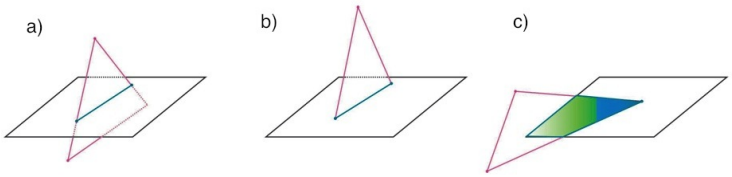


Figure 1-20: Classification of face segments and polypoints

A FE-element is said to *slice with an edge a cell face* if exactly two of its vertices v_i and v_j are coplanar

with the cell-face and the intersection of the edge joining v_i and v_j with the cell face results in a line segment, which is also a type of face segment. For this type of face segments the prefix t will be used, **t-face segment**, tfs, see Figure 1-20b.

A FE-element is *coplanar with a cell-face* if its three vertices are co-planar with the cell face's plane. For a given cell face, P_{cf}, the intersection of a coplanar FE-element edges with P_{cf} results in a third type of face segments for which the prefix p will be employed, **p-face segment**, pfs, see Figure 1-20c.

The three types of face segments can coincide exactly with a section of a cell face edge.

Inside/Outside Determination

Once the polyhedrons constituting each cut-cell are constructed by the *Search Algorithm*, it is necessary to determine if they are active or inactive, as well as their volume, and area of the polygonal faces to make the flux calculations for the Finite Volume Solver.

One way to determine the active polyhedrons in each cut-cell is through the determination of the active face polygons. This transforms the three dimensional problem of the activity determination for polyhedrons into a two dimensional problem of the activity determination for their polygonal faces.

The remainder of this work assumes that for each cut-cell's face **all observations are performed from a point of view of an observer inside the cut-cell**, which means that, when speaking of a cell face and its face polygons, it is assumed that the outward pointing normal of that face, with respect to the cell polyhedron, points away from the view point. It is assumed also that an order for the vertices of the cell face is determined, *i.e.* there is one vertex known as the first vertex, see Figure1-21

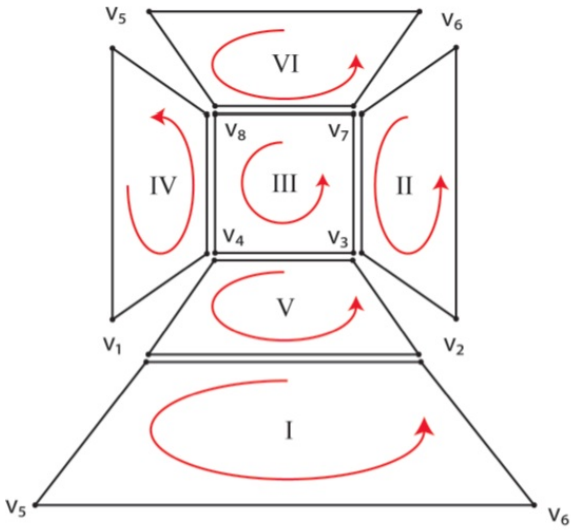


Figure 1-21: View point for cut cell

The airbag is a polyhedron $\mathcal{P}_a$. When $\mathcal{P}_a$ is sliced by a plane l, the plane slices the polyhedron into a set of polygons $\{P_a\}_i$, Figure 1-22.

For a polygon P_{a^*} for which each of its edges is the result of a FE-element slicing the plane l, the outward normal, $\hat{n}$, of each triangular FE-element is known, so an outward pointing normal to each

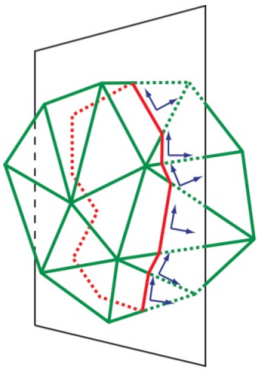


Figure 1-22: Equivalence to two-dimensional problem

face segment, with respect to the airbag, can be determined by projecting $\hat{n}$ over the plane l. We identify this projection by

$$\hat{pn} \tag{1-9}$$

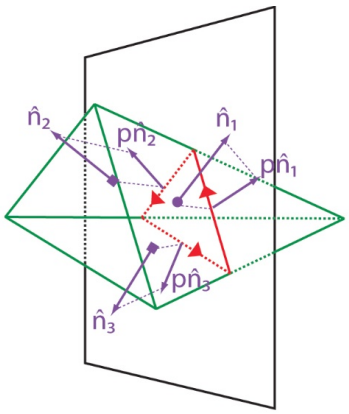


Figure 1-23: Projected and rotated normal

Rotating $\hat{pn}$ by $+90^\circ$ over the plane l, with respect of the *interior observer point of view*, a direction is obtained for each *s-face segment* on the boundary of P_{a^*}, this direction is called the normal induced CCW (**niCCW**) direction. See Figure 1-23.

The interior points of P_{a^*} when its boundary is traversed in the **niCCW** direction of its face segments are also iterior points of the airbag polyhedron $\mathcal{P}_a$. In this sense the interior points of the polygon P_{a^*} according to the **niCCW** direction of its face segments are consistent with the interior points of the polyhedron $\mathcal{P}_a$.

The same occurs for intersections of the FE-triangulation, $\mathcal{P}_a$, and a cell face's plane l_{cf}. The plane l_{cf} is divided into four types of regions: interior points to the intersection of the set $\{P_a\}_i$ with the cell face polygon P_{cf}; interior points only to the set $\{P_a\}_i$; interior point only to the cell face polygon P_{cf},

and point which do not belong to either $\{P_a\}_i$ or P_{cf}. To determine the active face polygons, what needs to be determined is the points that belong to both the interior of P_{cf} and $\{P_a\}_i$, see Figure 1-24.

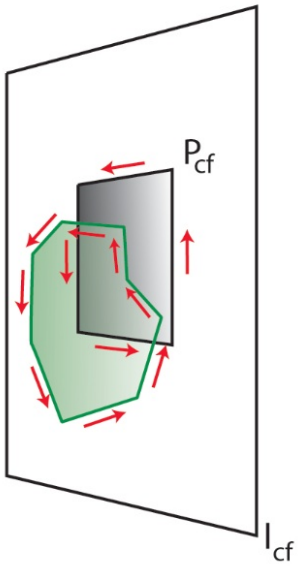


Figure 1-24: Cell face plane regions

If all the edges of the polygons $\{P_a\}_i$ are s-face segments, then each of them has an niCCW direction, and the same for the edges of the cell face polygon P_{cf}. In this case, active face polygons will be the points for which the CCW direction of their boundary is consistent with the niCCW direction of their edges, see Figure 1-24

This requires the description of all face polygons in CCW fashion for comparison with the niCCW direction of the edges. This is straigth forward for the *Simple polygons*, but because the airbag polyhedron is in general non-convex, a plane might slice the polyhedron in a set of simple polygons that can be contained one into the other, see Figure1-25.

For the contained polygons, these are transformed into weakly simple polygons by joining the first vertex of an internal contour with the first vertex of the polygon containing it using a dummy edge. The dummy edge is traversed in both directions and each of the simple polygons is described on counterclockwise fashion. See Figure 1-26.

An extreme geometrical case, known as collapsing FE-segments, occurs when the airbag folds causing two neighboring triangular FE-segments to become almost co-planar. In this project the problem of collapsing FE-segments will be ignored, see Figure 1-27.

Unfortunately the inside/outside determination method described above assumes that all face segments are s-face segments, so a niCCW direction can be assigned to all of them, but this is not always the case, as it happens when the face polygons contain t-face segments or p-face segments, see Figure1-28a. This method is also uncapable of determining the activity status of faces flagged as cut faces but which are cut at a single polypoint, Figure1-28b

The goal of this work is to develop a robust method to determine the inside/outside polyhedrons of each cut-cell in order to solve the deficiencies of the *Signed Volume Test*.

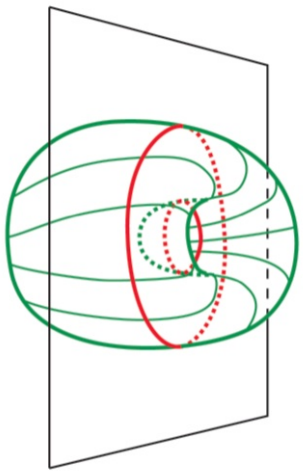


Figure 1-25: Plane slicing a non-convex polyhedron

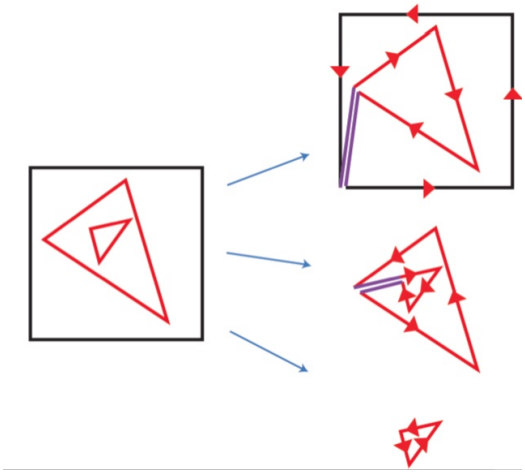


Figure 1-26: Polygonal decomposition

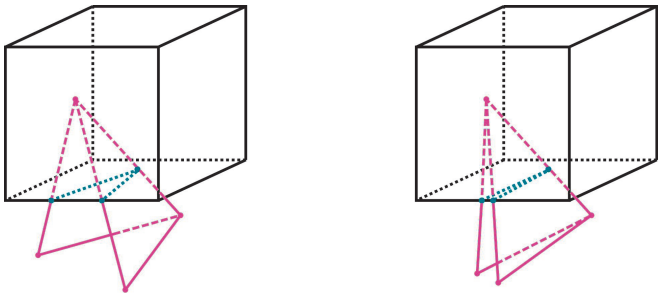


Figure 1-27: Collapsing triangles

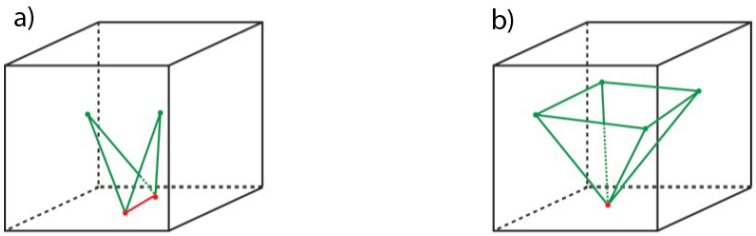


Figure 1-28: Cases where no niCCW can be determined.

1-4 Research Question

In the previous section an overview of the *Search Algorithm* was presented. Once the exact geometry is known, the goal of the *Coloring Algorithm* is to mark the active and inactive cells as well as the active and inactive regions of the cut-cells. The goal of this work is to answer the following question:

How to determine which region in a cut-cell belongs to the flow and which out-side in a consistent way and for cases where no niCCW direction can be chosen for the face segments or where a cut cell is cut at a single polypoint?

The only known fact about the cut-cells, in general, is their classification as such, but not which regions in them belong to the flow and which outside.

Thinking of a test to define the inner and outer regions of a cut-cell seems straight forward in terms of any of the standard tests for the *Point in Polyhedron Problem* used un computational geometry.

For example the *Line Crossing Test*, Figure 1-29, which determines if a point is inside a polygon by counting the number of times a ray emanating from the point being tested to the "infinity" crosses the boundary of the polygon, could be used taking any test point in any of the two regions of a cut-cell and the centroid of any of the active cells, tracing a line segment between those two points and counting the number of intersections would reveal the answer. But soon problems arise, for example:

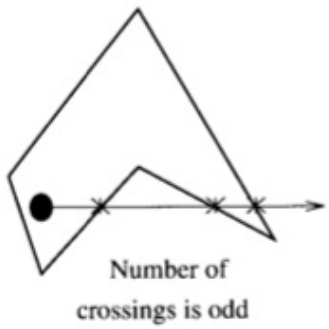


Figure 1-29: Line crossing principle; reprint from [8]

- How to choose the test point in the cut-cell?

- How does the choice of the active cell affects the accuracy and what role does connectivity plays?
- What happens if the test point in the cut-cell is too close to the airbag's boundary?

This problems become more complicated taking into account the fact that the boundary in the problem at hand is a moving, flexible boundary.

The above mentioned research question is the research topic that this work will address.

Chapter 2

Geometrical Modeling and Test problems

2-1 Abstract Geometrical Modeling

A method to determine the active and inactive face polygons of the cut-cells, which we will call the *Consistency Method*, was suggested in the previous chapter. This method works by verifying the consistency between the CCW description of the face polygons and the niCCW traversing direction of their edges, but it is not applicable in a number of cases. Nonetheless the underlying ideas of that method can be used to design other strategies.

In what follows of this section, a mathematical formulation for geometric modeling, based on [9], is used to support and to give a formal proof of the ideas behind the *Consistency Method*, then, a classification for the cut cells will be presented, from which the test cases are defined.

For both the airbag polyhedron $\mathcal{P}_a$ and each cell polyhedron $\mathcal{P}_c$ there is a counter-clock wise description and the outward pointing normals of their faces are known.

As it is depicted in Figures 1-22, Figure 1-25, and Figure 2-1 when one of this polyhedrons is sliced transversally by an infinite plane, the intersection of the polyhedron with the plane results in a polygon for which a normal induced counter-clock wise (niCCW) description can be obtained.

In the case of a cut-cell and its cut faces, considering the plane to which one of its faces P_{cf} belongs, we know that both the cell face polygon, P_{cf}, and the polygon P_a resulting from the intersection of the airbag and the plane, can be described in CCW fashion such that the interior points to these polygons according to this description are consistent with the interior points of the polyhedrons, $\mathcal{P}_a$ and $\mathcal{P}_c$, see Figure 2-1

In [9] an algebra of simplices is used to represent any solid in n-dimensions in a unique manner. With this algebra, operations such as intersection and union between solids can be performed obtaining unique representations for the result.

The set of objects for which this algebra is defined are the simplices. A simplex in n-dimensions is a generalization of a triangle to the n-dimensional space, in the plane a simplex is a triangle and in space a simples is a tetrahedron.

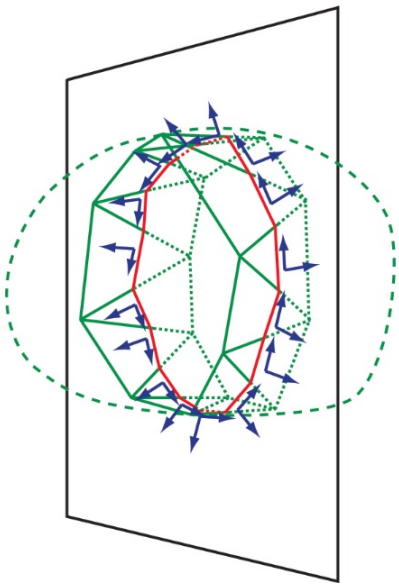


Figure 2-1: Interior points of $\mathcal{P}_a$, $\mathcal{P}_c$, P_{cf} and P_a

.

On the plane with origin O, for a polygon P on it, we define: the simplices S_i obtained for each edge $\overline{v_i v_j}$ of P as the region enclosed by the vertices O, v_i and v_j; the triangles T_i as the oriented triangle (O, v_i, v_j) according to the order of the vertices v_i, v_j in P; and the integer coefficients α_i as $\alpha_i = sign(Area\_sign(T_i))$ where $sign$ is the sign function and $Area\_sign$ is the signed area of T_i. See Figure2-2

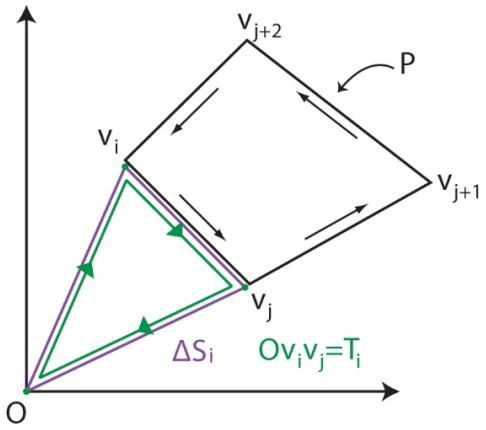


Figure 2-2: Objects of the algebra of simplices.

.

In [9] it is shown that any polygon P can be associated to a simplicial chain of the form

$$\chi = \sum_{i=1}^{m} \alpha_i S_i \tag{2-1}$$

and, for a point $x \in \mathbb{R}^d$, to the characteristic function

$$f_\chi : \mathbb{R}^2 \to \mathbb{Z},\ f_\chi(x) = \sum_{i=1}^{m} \alpha_i \tag{2-2}$$

That is, in each point in $\mathbb{R}^2$ the characteristic function takes the values of the sum of the coefficients of the simplices containing x.

The association between a given polygon and a polygonal chain is done through the characteristic function f_χ, any point x in the plane is determined to be in P if $f_\chi \neq 0$, that is, if the sum of the coefficients of the simplices that contain the point is different from zero. In [9] it is shown that this association is unique. Figure 2-3 presents an example of the association of a polygon with its simplicial chain.

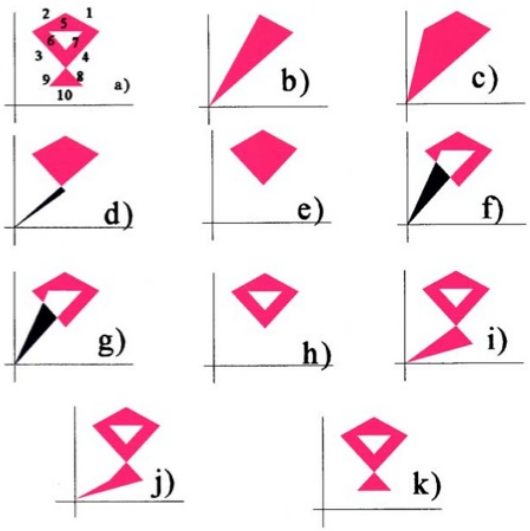


Figure 2-3: Example of simplicial chain association, reprint from [9]

.

In [9] the following theorem is proven

Theorem 1. *Let P_1 and P_2 be polyhedral solids in $\mathbb{R}^d$, and χ_1 and χ_2 their associated chains respectively*

$$\chi_1 = \sum_{i=1}^{n} \alpha_i S_i \, , \, \chi_2 = \sum_{j=1}^{m} \beta_i T_i$$

Then the associated normal chain to the intersection solid $P_\chi = P_1 \cap P_2$ is

$$\chi = \sum_{i=1}^{n} \sum_{j=1}^{m} (\alpha_i \cdot \beta_j) \cdot Sim(S_i \cap T_j)$$

where $Sim(S_i \cap T_j)$ is the simplex obtained by as the intersection of the simplices S_i and T_j.

As it was noted in Section 1-3-1, by slicing the airbag polyhedron with the infinite plane l_{cf} of a cell face polygon P_{cf}, the intersection results in a polygon P_A, know in counterclockwise order. The cell-face P_{cf} is also a polygon, contained in l_{cf}, known in counterclockwise order, so, applying **Theorem 1**, the intersection of the polygons P_A and P_{cf}, is representable by a normal simplicail chain

$$\chi_{AC} = P_A \cap P_C = \sum_{i=1}^{n} \sum_{j=1}^{m} (\alpha_i \cdot \beta_j) \cdot Sim(S_i \cap T_j)$$

where the coefficient $c_{ij} = (\alpha_i \cdot \beta_j)$ is given by

$$c_{ij} = \begin{cases} 1 \text{ where the intersection is non-empty} \\ 0 \text{ where the intersection is empty} \end{cases}$$

the intersection polygon χ_{AC} is known in counterclockwise order.

The plane l_{cf} where the cell face P_{cf} lies, is divided into four types of regions: interior points to the intersection of the set $\{P_a\}_i$ with the cell face polygon P_{cf}, interior points only to P_A, interior points only to the cell face polygon P_{cf}; and points which do not belong to either P_A or P_{cf}. To determine the active face polygons, what needs to be determined is the points that belong to both the interior of P_{cf} and P_A, see Figure 2-4.

Theorem 1 proves that χ_{AC}, that is, the active face polygon, is the one whose edges, when traversed in CCW direction are traversed in the same direction as the niCCW direction.The other three types of regions can be characterized by how the boundaries of P_A and P_{cf} are traversed relative to arbitrary points in each region. Analyzing an arbitrary point p totally contained in each region, keeping in mind the $\chi\chi_{AC}$ representation of $P_A \cap P_{cf}$, and given the characterization of the interior points to a polygon as the points always to the left of the boundary when traversed in counterclockwise direction, the four regions are characterized as follows, see Figure 2-5.

1- If $p \in P_A \cap P_{cf}$, the description of both P_A and P_{cf}, when described in counterclockwise order with respect to the point of view of q, is consistent with their general counterclockwise description in the plane.

2- If $p \notin P_A$ and $p \notin P_{cf}$, the description of both P_A and P_{cf}, when described in counterclockwise order with respect to the point of view of q, is not consistent with their general counterclockwise description in the plane.

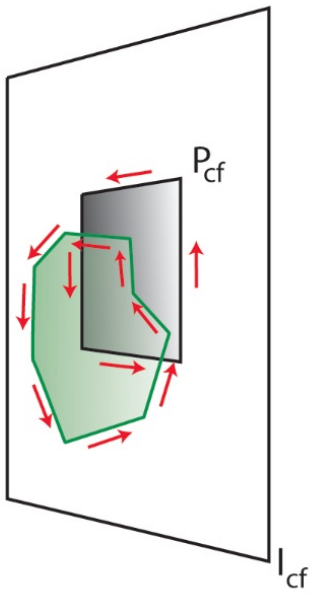


Figure 2-4: Cell face plane regions

3- If $p \in P_A$ and $p \notin P_{cf}$, the description of P_A, when described in counterclockwise order with respect to the point of view of q, is consistent with the general counterclockwise description in the plane, but the description of P_{cf}, when described in counterclockwise order with respect to the point of view of q, is not consistent with the general counterclockwise description in the plane.

4- If $p \notin P_A$ and $p \in P_{cf}$, the description of P_{cf}, when described in counterclockwise order with respect to the point of view of q, is not consistent with the general counterclockwise description in the plane, but the description of P_A, when described in counterclockwise order with respect to the point of view of q, is consistent with the general counterclockwise description in the plane.

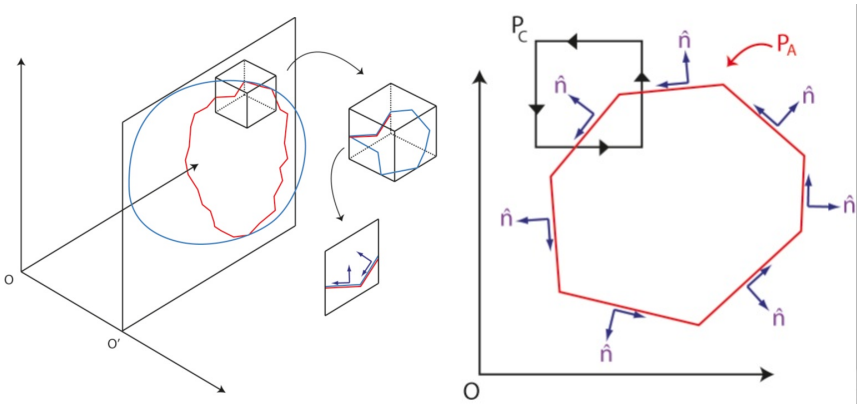


Figure 2-5: Regions for the airbag polygon and cell faces

.

The active face polygons of each cut-cell face can be obtained by computing the simplicial chains of P_A and P_{cf} and then computing the intersection. The disadvantage of this approach is that it requires the complete simplicial characterization of P_A for every cut-cell face plane. Given the number of cells and the anisotropic nature of the grid, this is not an efficient approach.

For the interior of the cut-cell's face polygons P_{cf}, the former approach can be modified to obtain a local method. The polygonal regions inside P_{cf} satisfy one of the two characterizations **1** or **4** mentioned above. From the description of the face polygons in CCW fashion, it follows that the interior regions to the flow are the ones for which, for an arbitrary point p inside such polygonal region, the CCW description of the boundary from the point of view of p is consistent with the niCCW description of P_A and P_{cf}. In this way the complete niCCW description of P_A is not required, which is the drawback of a non-local method, because the outward normal to each FE- segment is known, by projecting the normal into the cell face P_{cf} and rotating 90° the niCCW direction of the segments of P_A inside P_{cf} is known.

Result 1. The polygonal regions inside each cut cell face P_a belonging to the flow regions are those for which the CCW description of their boundaries, is completely consistent with the niCCW face segments directions of the polygons P_a and P_{cf}.

2-2 Test Problems

To verify that the solution methods work, a set of test problems must be constructed. These test cases must be representative of all the possible cut-cell cases that may appear in the meshing process.

In order to construct the test cases in a general way in a manner which allows to cover all possibilities that can occur, the test cases are classified according to the number of zeros in the Signed Volume test that each FE-element can have for its vertices with respect to a single cell face plane.

Given a cell face polygon P_{cf} in the plane l_{cf}, and a set of FE-elements, P_{e_i}, sharing a vertex v^* the following can occur.

- At most one zero and only s-face segments, (**Test case 1**): All FE-elements P_{e_i} slice P_{cf} into a s-face segments, which means that at most one zero occurs in the signed volume test between any of the vertices of a FE-element and any three vertices of P_{cf}.

  This case results into face polygons with edges for which a niCCW can be directly obtained form the FE-elements outward normal. Contained polygons can be the result of this type of intersections. See Figure 2-6

- Two zeros and one t-face segment for each FE-element: there exist two vertices v_i and v_j of an FE-element with zero Signed Volume with respect to the vertices of P_{cf}. These two vertices, v_i and v_j, are linked by a t-face segment for which a niCCW direction cannot be chosen in a straight forward manner because it is an edge shared by two FE-elements.

  This case must be further divided into cases where: the FE-elements sharing the t-face segment are on the same side of l_{cf} (**Test case 2**), see Figure 2-7a; and cases where the FE-elements sharing the t-face segment are on opposite sides of l_{cf} (**Test case 3**), see Figure 2-7b.

- Three zeros and p-face segments, (**Test case 4**): at least one of the FE-elements P_{e_i} is coplanar with the cell face P_{cf}, with non empty intersection, this results into p-face segments for which a straight forward choice of niCCW direction does not exists, see Figure 2-8

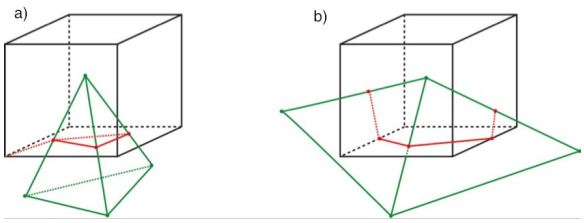


Figure 2-6: Test case 1.

.

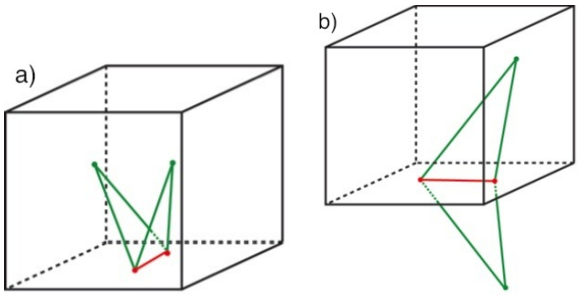


Figure 2-7: a) Test case 2, b) Test case 3

.

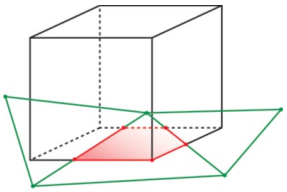


Figure 2-8: Test case 4

.

- Only one zero at v^* and no FE-element's edges piercing l_{cf}, (**Test case 5**): for this case no face segments exist, the cut face contains one single polypoint. There are no edges to attempt to define a niCCW direction and apply the *Consistency Method*, Figure 2-9

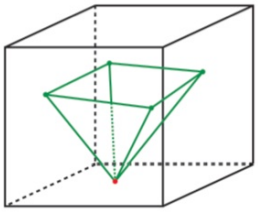


Figure 2-9: Test case 5

.

- Only one polypoint at v^* no face segments inside P_{cf} with FE vertices on both sides of the plane l_{cf}: for this case no face segments exist, the cut face contains one single polypoint. There are no edges to attempt to define a niCCW direction and apply the *Consistency Method* and FE-elements are on both sides of l_{cf}, the polypoint can occur either at a vertex of P_{cf}, (**Test case 6**), or at an edge of P_cf, (**Test case 7**), Figure 2-10

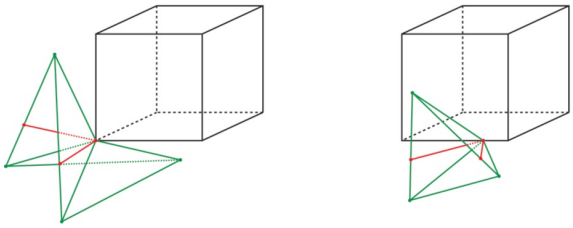


Figure 2-10: Test case 6 and Test case 7

.

- Only one polypoint at an edge of an FE-element or at the interior of an FE-element: for this cases no face segment exist, the cut face contains one single polypoint. There are no edges to attempt to define a niCCW direction and apply the *Consistency Method* and FE-elements are on both sides of l_{cf}. This cases will be shown to be treatable as Test cases 6 and 7. Examples of these cases are shown in Figure 2-11

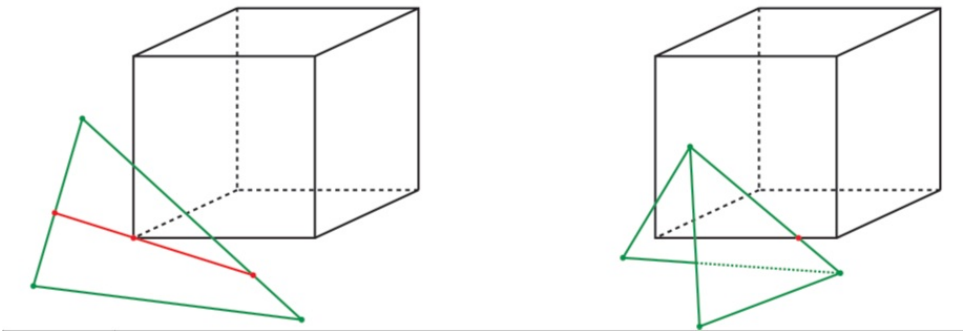


Figure 2-11: Test case 8 and Test case 9

.

Chapter 3

Coloring Algorithm, Design

This chapter presents the *Coloring Algorithm*. Different methods are used depending on the way the FE triangulation intersects every cut-cell to determine the active and inactive face polygons. The airbag polyhedron is referred as $\mathcal{P}_a$, and all methods are explained for a particular cut cell $\mathcal{P}_c$. When no ambiguity exists about the cell face being treated, the cell face polygon is simply referred as P_{cf} and its face polygons as P_i, all of which belong to the cell face plane l_{cf}.

3-1 Extended Consistency Method, treatment of Test Cases 1, 2 and 4

According to the *Consistency method*, the status of a *face polygon* can be determined by comparing its CCW description with the niCCW direction of the face segments.

For each cut-cell face, each of its *face polygons* is traversed in counter-clock wise direction and if the direction of the face segments on its boundary is contrary to the niCCW direction of the *face segments*, then the polygon is determined to belong outside the flow region. Figure3-1 presents an example.

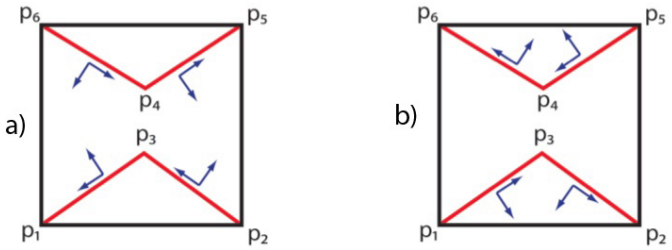


Figure 3-1: Traversing direction consistency.

.

In Figure3-1 a), the cell face is cut into the polygons $P_1 = p_1p_2p_3$, $P_2 = p_1p_3p_2p_5p_4$ and $P_3 = p_6p_4p_5$, for this case P_1 and P_3 are consistent with the directions at which the *face segments* are traversed, while

P_2 is not. It is concluded that the interior of P_1 and P_3 belong to the flow region while the interior of P_2 belongs outside.

In Figure 3-1 b), the cell face is cut into the polygons $P_1 = p_1p_2p_3$, $P_2 = p_1p_3p_2p_5p_4$ and $P_3 = p_6p_4p_5$, for this case P_2 is consistent with the directions at which the *face segments* are traversed, while P_2 and P_3 are not. It is concluded that the interior of P_2 belongs to the flow region while the interior of P_1 and P_3 belong outside.

This method works as well for cell faces with contained face polygons. The determination is done using the description of the contained *face polygons* as a weakly simple polygons using a dummy edge which is traversed in both directions in the CCW description. See for example Figure 3-2

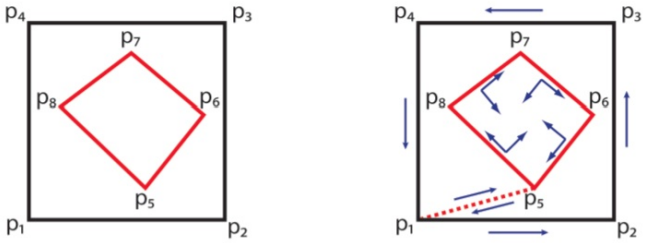


Figure 3-2: Non-simple polygon degeneracy.

.

The *Consistency Method* works for cell faces where all the face segments are *s-face segments*, such as in Test case 1, because for this type of face segments a niCCW direction is defined in terms of the projection of the FE-element's normal onto the cell face.

When a face polygon has *p-face segments* or *t-face segments* on its boundaries, this method fails because there is no straight forward way to determine a niCCW direction for these face segments, such as Test cases 2, 3 and 4. This method also fails for cut cell faces marked as cut faces where the intersection occurs at a single polypoint, in these cases no face segments exist.

A way to extend the *Consistency Method* so that it works for the problematic test cases 2 and 4, is to define in a systematic way an outward pointing normal for the FE-element's edges, and so, a niCCW direction for the face segments.

Given the airbag polyhedron $\mathcal{P}_a$ and an edge e^* joining the vertices v_i and v_j, shared by two of its triangular faces FE_k and FE_l with outward pointing normals $\hat{n}_k$, $\hat{n}_l$ respectively, the **edge-outward pointing normal** $\hat{n}^*$ of e^* is given by

$$\hat{n}^* = \frac{\hat{n}_k + \hat{n}_l}{||\hat{n}_k + \hat{n}_l||} \tag{3-1}$$

Figure 3-3 represents the edge-outward normal.

When a face polygon contains on its boundary an edge which happens to be a *p-face segment* of *t-face segment*, its edge-outward pointing normal is determined using 3-1. The edge-outward pointing normal is then projected onto the cell face and rotated 90° to obtain the niCCW direction of the face polygon's edge, and the *Consistency method* can be applied to determine the activity, Figure 3-4

The extension of the *Consistency Method* using the edge-outward pointing normal is not applicable for cases where an edge e^* of the face polygon is a *t-face segment* for which the FE-elements FE_k and

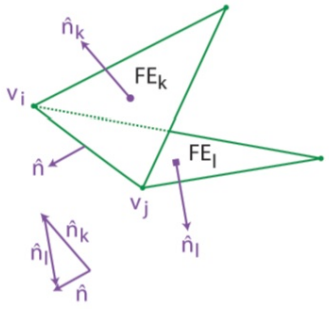


Figure 3-3: Edge-outward normal.

.

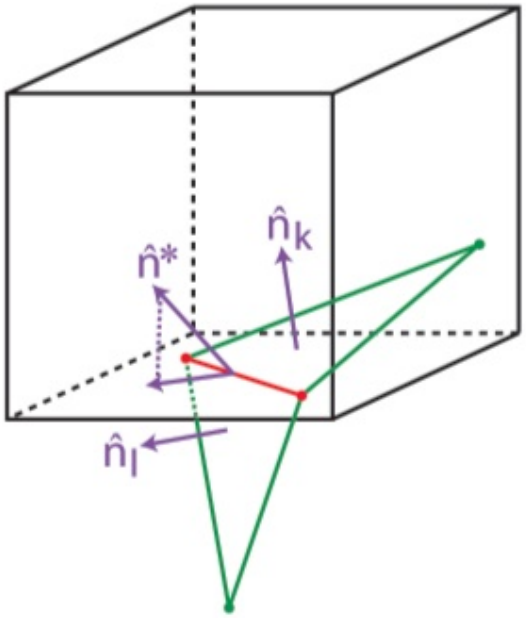


Figure 3-4: Modification for the Consitency method.

.

FE_l sharing it belong to the same half-space with respect to the face polygon's plane, as in Test case 3 . In this case it can occur that the edge-outward pointing normal $\hat{n}^*$ has the same direction as the outward pointing normal of the cell face where the face polygon lies, and the projection of $\hat{n}^*$ onto the cell face's plane results in a vector of magnitude zero, see Figure 3-5

3-2 Local half-space determination method 1, treatment of Test Case 3

In general, when a face polygon, P_i, has an edge e^* which is a problematic *t-face segments*, *i.e*, where the FE-elements sharing e^* are on the same half-space defined by l_{cf}, e^* can be treated as a dummy

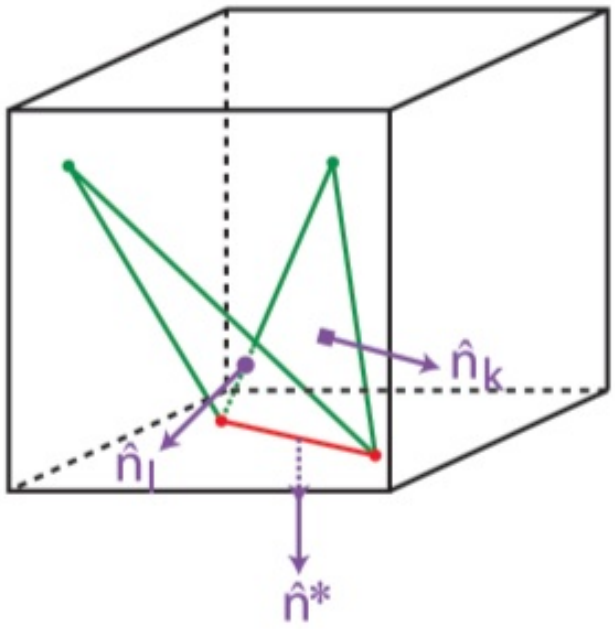


Figure 3-5: projection with magnitude zero.

.

face segment, and the activity of such face polygon is obtained via the *Extended Consistency Method* as long as P_i has on its boundary more face segments for which a niCCW direction can be determined, see Figure 3-6

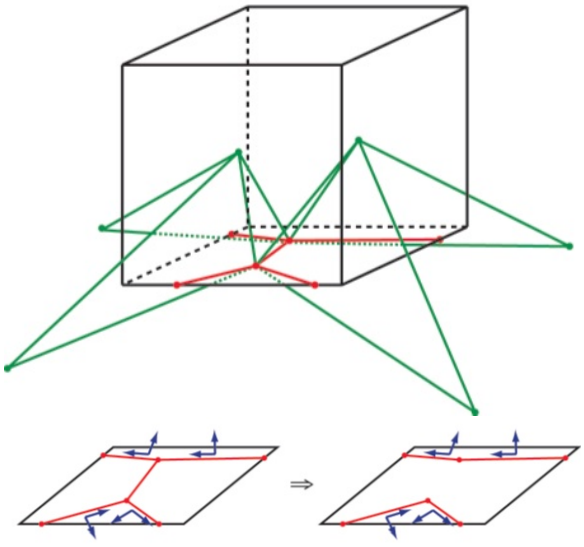


Figure 3-6: Test Case 2, non-isolated t-face segment.

.

For a cell face P_{cf} where a problematic *t-face segment* e^* is the only information available, and the FE elements FE_k and FE_l that share e^*, and the set FE elements $\{FE\}_i$ which share edges with FE_k and FE_l, all belong to one half-space into which the whole space is divided by l_{cf}, then the cell face P_{cf}, whose outward pointing normal is $\hat{n}_{cf}$, belongs completely inside $\mathcal{P}_a$ or completely outside, see Figure 3-7.

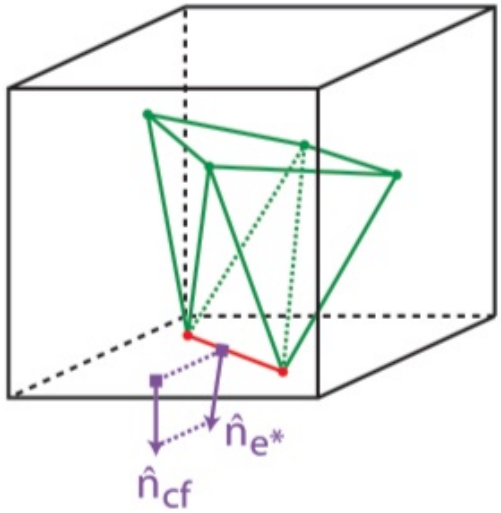


Figure 3-7: Test Case 2.

.

To determine P_{cf} belongs to the flow region, the edge-outward normal $\hat{n}_{e^*}$ of e^* is projected onto $\hat{n}_{cf}$, depending on the sign of the magnitude of the projection and the half-space to which FE_k, FE_l and $\{FE\}_i$ belong, the activity of P_{cf} can be determined.

If FE_k, FE_l and $\{FE\}_i$ are on the same half-space as the cell polyhedron $\mathcal{P}_c$, with respect to the plane l_{cf}, and the projection of $\hat{n}_{e^*}$ onto $\hat{n}_{cf}$ has positive magnitude, then P_{cf} is inactive, if the magnitude of the projection is negative, then P_{cf} is active. On the other hand, if FE_k, FE_l and $\{FE\}_i$ are on the complementary half-space to that of the cell polyhedron $\mathcal{P}_c$, with respect to the plane l_{cf}, and the projection of $\hat{n}_{e^*}$ onto $\hat{n}_{cf}$ has positive magnitude, then P_{cf} is active, if the magnitude of the projection is negative, then P_{cf} is inactive. See Figure 3-7

3-3 Local half-space determination method 2, treatment of Test Case 5

The way this case it treated follows the same ideas as the *Local half-space determination method* for Test case 3. If the cell face P_{cf} is cut at a single polypoint, p^*, which is also a vertex v^* of $\mathcal{P}_a$, and no other information is available for P_{cf}, then the activity determination of P_{cf} is based on the magnitude of the projection of a well chosen outward normal of $\mathcal{P}_a$ onto $\hat{n}_{cf}$.

The number of FE elements that share v^* is arbitrary, so defining an "averaged" outward normal for all of them not only is not straight forward but also it can also result on computational robustness issues.

To solve this problem, the information of the normal of a single FE element is used. Based on the angle α_i of the edges emanating from v^* with respect to the plane l_{cf}, the FE element FE^* closest

to P_{cf} is chosen. The outward pointing normal $\hat{n}_{FE^*}$ of FE^* is projected onto $\hat{n}_{cf}$, and the activity determination of P_{cf} follows the same rules as in the previous method for Test case 3, see Figure 3-8.

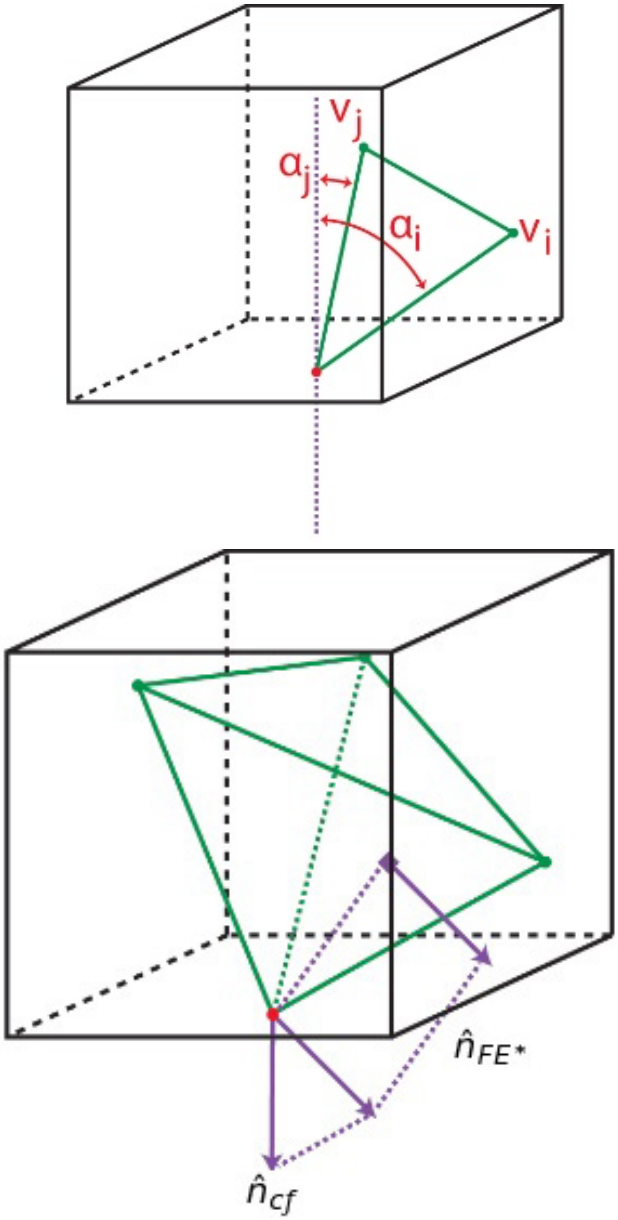


Figure 3-8: Test Case 5, projected normal.

.

3-4 Local half-space determination method 3, treatment of Test Cases 6, 7, 8 and 9

Suppose that for a given cell face P_{cf} the cut point occurs at a single polypoint pp^* which is also a vertex of the cell face and a vertex of $\mathcal{P}_a$, see Figure 3-9

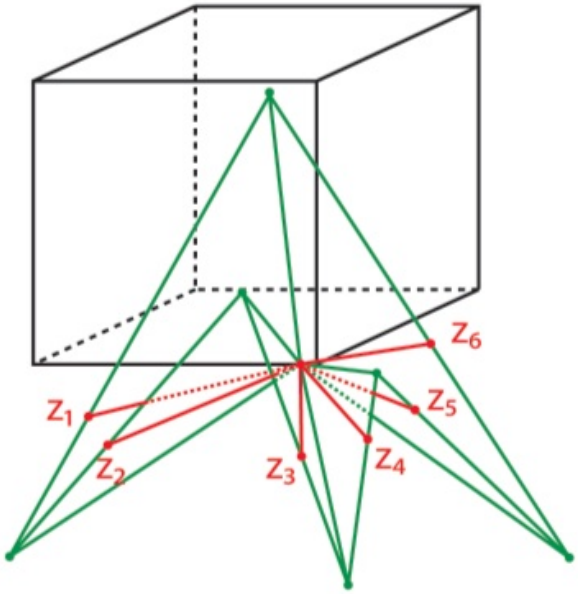


Figure 3-9: Test Case 6.

.

The cell face P_{cf} has a CCW order and thus a first and second edge ordering, ce_i and ce_{i+1}, can be assigned to its edges sharing pp^*. These edges have outward normals $\hat{n}_i$ and $\hat{n}_{i+1}$ with respect to P_{cf} respectively.

Some of the FE elements sharing pp^* cut through the plane l_{cf}, outside the polygon P_{cf}, and the edges of those FE elements intersect l_{cf} at the polypoints pp_1, pp_2, ... pp_n, from which the cut segments $cs_1 = \overline{(pp^*, pp_1)}$, $cs_2 = \overline{(pp^*, pp_2)}$, ... $cs_n = \overline{(pp^*, pp_n)}$ are defined, see Figure 3-10

An ordering of the cut segments cs_i is defined based on the angle α between each cs_i and the cell face edge ce_i, and thus a first cut segment cs^* can be chosen. For the cut segment cs^* an edge-outward pointing normal, $\hat{n}^*$, is obtained by projecting the normal of the FE element to which cs^* belongs onto the plane l_{cf}, see Figure 3-10

The activity of P_{cf} is determined by analyzing the sign of the magnitude of the projections $n_i^* \hat{n}_i$ and $n_{i+1}^* \hat{n}_{i+1}$ of $\hat{n}^*$ onto $\hat{n}_i$ and $\hat{n}_{i+1}$.

The activity of the cell face is determined as follows:

For $0 < \alpha < 90^\circ$

- if $n_i^* < 0$ the cell face is inactive.

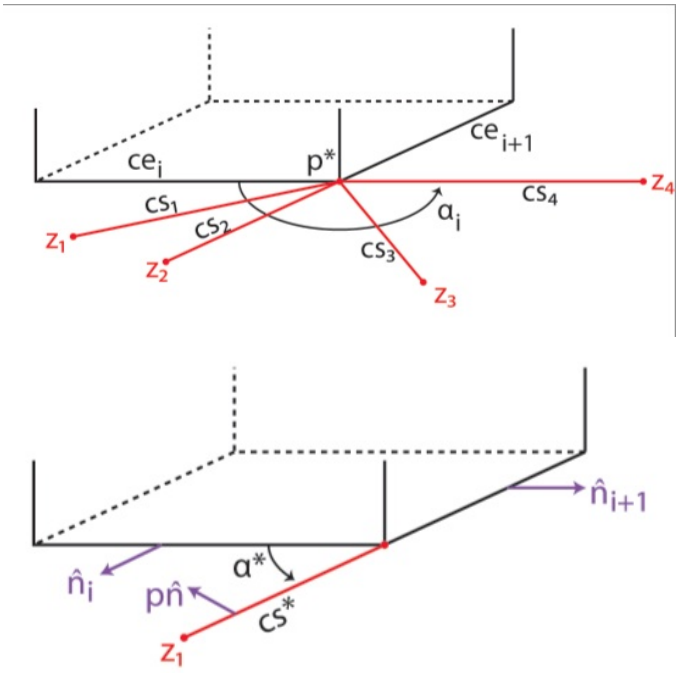


Figure 3-10: Test Case 6, first cut segment and outward normal.

.

- if $n_i^* > 0$ the cell face is active.

For $90° < \alpha < 270°$

- if $n_i^* < 0$ the cell face is active.
- if $n_i^* > 0$ the cell face is inactive.

For $\alpha = 90°$

- if $n_{i+1}^* < 0$ the cell face is active.
- if $n_{i+1}^* > 0$ the cell face is inactive.

The same idea is applicable for Test cases 7 and 8, where the application is seen directly by dividing the FE elements further by dummy edges connecting the polypoint pp^* with the vertices of the FE elements, this transforms these cases into Test case 6. In practice it is not necessary to define the dummy edges, but it helps to visualize the extension of the method.

The method is simplified for Test case 7 where the outward normal information of only one edge of P_{cf} is used, see Figure 3-11

The activity of the cell face is determined as follows:

For $0 < \alpha < 90°$

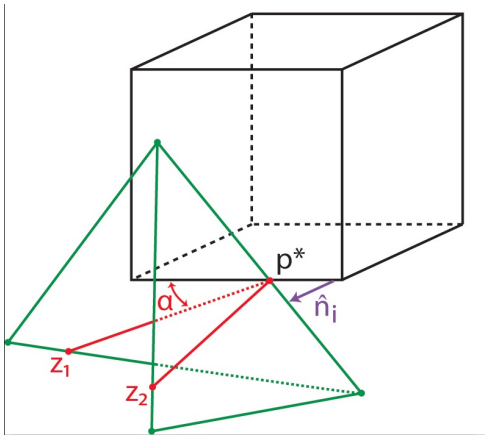


Figure 3-11: Simplification for Test Case 7.

.

- if $n_i^* < 0$ the cell face is inactive.
- if $n_i^* > 0$ the cell face is active.

For $90° < \alpha < 180°$

- if $n_i^* < 0$ the cell face is active.
- if $n_i^* > 0$ the cell face is inactive.

Rotating $\hat{n}_i$ CCW 90° to obtain $\hat{n}_i^{\perp}$ and projecting $\hat{n}^*$ onto $\hat{n}_i^{\perp}$ to obtain $n^{*\perp}\hat{n}_i^{\perp}$ then

For $\alpha = 90°$

- if $n^{*\perp} < 0$ the cell face is inactive.
- if $n^{*\perp} > 0$ the cell face is active.

3-5 Cell faces with no poly points

The methods developed in the previous sections threat the cell faces which are flagged as cut faces, that is, faces which contain at least one poly point. To determine the activity status of the faces which are not cut faces using only local information, it is first noted that these faces' polygons are formed completely by cell edges, the determination of the activity of these faces is through the use of an activity marker for the cell edges as it is explained in the next paragraphs.

During the construction of the cut cells the cell edges must be flagged as either neutral, if a cell edge contains a polypoint or section of face segment, or inactive, if the cell edge does not intersect the FE triangulation in anyway. For a cut cell, the *Coloring Algorithm* first evaluates the activity of the cell faces flagged as cut. Whenever a cut face has a face polygon with a complete cell edge, ie, an uncut

cell edge, on its boundary, the cell edge activity flag is left as inactive if the face polygon is determined to be inactive, or it is switched to active if the face polygon is determined to be active.

After determining the face polygon's activity for all cut faces, all the uncut cell edges of these cell faces are also flagged as either active or inactive, based on this information the activity status for the neighboring uncut cell faces is determined. If an uncut cell face contains at least one cell edge flagged as active then this cell face polygon is flagged as active.

Chapter 4

Coloring Algorithm Implementation

4-1 Exact Geometry Requirements

In order to present the implementation of the methods of the previous chapter, first we have to go back one step and analyze what these methods require that the Exact Geometry Algorithm provide.

In Chapter 1 we described how the Exact Geometry algorithm works based on topological tests and geometrical constructors in order to calculate the polypoints and to build the face polygons.

Here we briefly describe the requirements on information that the Coloring Algorithm requires from the Exact Geometry Algorithm.

For the active and inactive cells, the exact geometry information consists of two of their vertices, the first and the last one, from which the faces can be obtained.

For the cut cells the following information is required:

- two diagonally opposing vertices, v_1 and v_7 as in Figure 1-21
- a list of FE elements which intersect the faces or are inside the cut cell.
- a list of (six) cell faces, each containing its own geometrical information

The cell faces are divided into types according to the method of the Coloring Algorithm that needs to be used to analyze the activity of its face polygons. Faces as in Test cases 1,2 and 4 are the first type, faces as in Test Case 3 are the second type, faces as in Test Case 5 are the third type, and faces as in Test Cases 6,7,8 and 9 are the fourth type. As it was described in Chapter 2, the classification of the Test Cases depends on the number of zeros of the Signed Volume, which is information that can be obtained from the Exact Geometry Algorithm.

For the cut cell faces the following information is required:

- a marker to know if it is a cut face or not.

- a list of cell edges each with an activity flag which can be either active, inactive or indeterminate.
- Two diagonally opposing vertices, which could be the first and third vertex of each cell according to the cell face CCW description starting at the vertex with the lowest global index of a particular face, as it is shown in Figure 1-21.
- A list of FE-elements which intersect the face.
- A list of polypoints.
- A list of the face segments obtained from the intersection with the FE-triangulation.
- A list of the face edges with a marker to flag them as cut edge, active or inactive.
- A list of the face polygons in CCW fashion
- The type of cut cell face.

For the face segments the following information is required:

- The type of face segment, according to the classification given in Chapter 1.
- If the face segment is a s-face segment, then also its niCCW orientation.
- The FE-elements to which they belong.

For the polypoints the following information is required:

- The FE-Elements to which it belongs. It can be one FE-element if it is the result of a cell edge piercing a FE-element, two if it is the result of a FE-edge piercing a cell face, or three or more if it coincides with a FE vertex.
- The number of the edges of the FE elements to which it belongs.

For the face polygons the following information is required:

- An indicator of its activity.
- The vertices.
- The edges in CCW fashion.

4-2 Implementation

In this section the algorithms for each method described in Chapter 3 are presented. The algorithms are described separately although the implementation of the Coloring Algorithm would contain the four methods together and the application of each would follow from the type of face being treated.

To show the application of each algorithm a single cut cell is used. In each case the cut cell has a geometry representative of the test case for which the algorithm is designed.

Prior to the application of the Coloring Algorithm, the faces of the cut cells are colored in blue, and the FE triangulation is represented in green. After the application of the Coloring Algorithm, the face polygons interior to the airbag are colored in blue and the face polygons exterior are colored in red.

4-2-1 Implementation: Extended Consistency Method.

In this subsection the application of the Extended Consistency Method is shown. For this purpose a test cell and a set of elements of the FE triangulation are chosen such that one of the FE-elements is coplanar with one of the cell faces, see Figure 4-1.

The pseudocode of the coloring routine is in Algorithm 1.

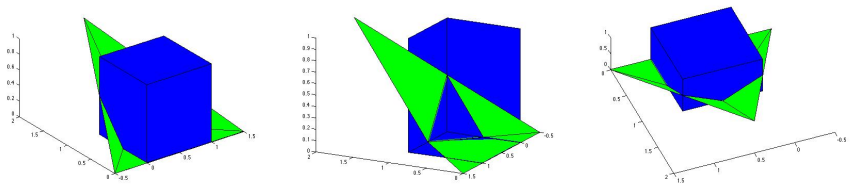


Figure 4-1: Test case for Extended Consistency Method.

Algorithm 1: Extended Consistency Method.

Input: CellFace(k) $\rightarrow \mathbf{P}_{cf} = (vc_1, vc_2, vc_3, vc_4)$
Output: Status of face polygons

read: face segments.
read: face polygons.
n =number of face polygons
for *i:* 1 ***to*** n **do**
  read: polygon(i)
  set polygon(i) activity = active
  m =number of edges polygon(i)
  for *j:* 1 ***to*** m **do**
    if *edge(j) is face segment* **then**
      read: type of face segment.
      if *edge(j) is s-face segment* **then**
        if *edge(j) traversing direction* $\neq$ *niCCW* **then**
          set polygon(i) activity = inactive
      if *(edge(j) is t-face segment)* $\bigvee$ *(edge(j) is p-face segment)* **then**
        read: FE-elements of edge(j) $\rightarrow \{FE_1, FE_2\}$
        read: normals of $\{FE_1, FE_2\} \rightarrow \{\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2\}$
        compute:edge-outward normal $\hat{\mathbf{n}}^*$
        project: $\hat{\mathbf{n}}^*$ onto CellFace(k) $\rightarrow \mathbf{n}_k^*$
        normalize: $\mathbf{n}_k^* \rightarrow \hat{\mathbf{n}}_k^*$
        rotate: $\hat{\mathbf{n}}_k^* \rightarrow \hat{\mathbf{n}}_k^{*\perp_{ccw}}$
        set: edge(j) niCCW direction $\rightarrow \hat{\mathbf{n}}_k^{*\perp_{ccw}}$
        if *edge(j) traversing direction* $\neq$ *niCCW* **then**
          set polygon(i) activity = inactive

The result is can be seen in Figure 4-2, which shows the FE-elements and their outward pointing normals and rotated images of the resulting cut-cell.

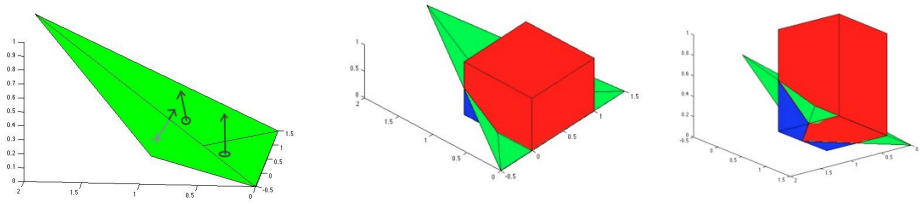


Figure 4-2: Results for Extended Consistency Method

4-2-2 Implementation: Half-space Determination Method 1.

This subsection presents the pseudocode of the Half-space Determination Method 1, see Algorithm 2. The notation signVol(a,b,c,d) is used for signed volume of a tetrahedron with triangular base (a,b,c) and fourth vertex d, where (a,b,c) is in CCW order with respect to d.

Algorithm 2: Half-space Determination Method 1.

Input: CellFace(k) $\rightarrow \mathbf{P}_{cf} = (vc_1, vc_2, vc_3, vc_4)$
Output: Status of face polygons

read: face segments.
read: face polygons.
n =number of face polygons // n=1
for *i:* 1 ***to*** n **do**
    read: polygon(i)
    set polygon(i) activity = 1
    get polygon(i) face segment $\rightarrow e^* = (v_{e^*1}, v_{e^*2})$
    read: FE-elements of $e^* \rightarrow \{FE_l = (v_{e^*1}, v_{e^*2}, v_3), FE_k(v_{e^*1}, v_{e^*2}, v_4)\ \}$
    read: normals of $\{FE_l, FE_k\ \} \rightarrow \{\hat{\mathbf{n}}_l, \hat{\mathbf{n}}_k\ \}$
    compute:edge-outward normal $\hat{\mathbf{n}}^*$
    project: $\hat{\mathbf{n}}^*$ onto CellFace(k) normal $\hat{\mathbf{n}}_{cf} \rightarrow \mathbf{n}_k^*$
    normalize: $\mathbf{n}_k^* \rightarrow \hat{\mathbf{n}}_k^*$
    if *signVol*$(vc_1, vc_2, vc_3, v_3) > 0$ **then**
        if $||\hat{\mathbf{n}}_k^* + \hat{\mathbf{n}}_{cf}||$=2 **then**
            set polygon(i) activity = 0
    if *signVol*$(vc_1, vc_2, vc_3, v_3) < 0$ **then**
        if $||\hat{\mathbf{n}}_k^* + \hat{\mathbf{n}}_{cf}||$=0 **then**
            set polygon(i) activity = 0

4-2-3 Implementation: Half-space Determination Method 2.

In this subsection the application of the Half-space Determination Method 2 is shown. For this purpose a test cell and a set of elements of the FE triangulation are chosen such that the vertex shared by all the FE-elements is the only polypoint at one of the cell faces, see Figure 4-3

The pseudocode of the coloring routine is in Algorithm 3.

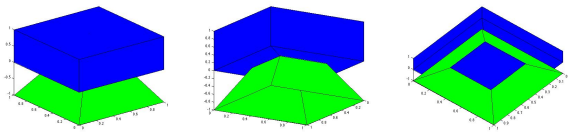


Figure 4-3: Test case for Half-space Determination Method 2.

The result is can be seen in Figure 4-4, which shows the FE-elements and their outward pointing normals and rotated images of the resulting cut cell.

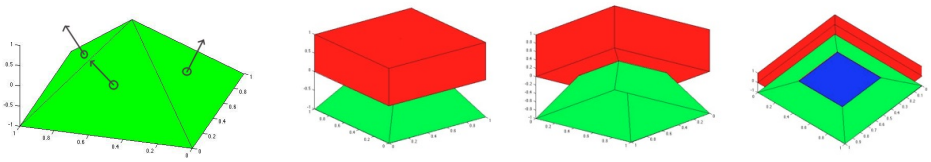


Figure 4-4: Results for Half-space Determination Method 2.

Algorithm 3: Half-space Determination Method 2.

Input: CellFace(k) $\rightarrow \mathbf{P}_{cf} = (vc_1, vc_2, vc_3, vc_4)$
Output: Status of face polygons

read: face polypoint $\rightarrow p^*$
read: FE edges at $p^* \rightarrow (e_1 = (p^*, v_1), e_2 = (p^*, v_2), \cdots, e_j = (p^*, v_j))$
read: cell face normal $\rightarrow \hat{\mathbf{n}}_{cf}$
set face polygon activity = 1
$\alpha \leftarrow 1$
$FE_1^* \leftarrow$null
$FE_2^* \leftarrow$null
$FE^* \leftarrow$null
$e^* \leftarrow$null
$\beta_{\min} \leftarrow 1$
//Loop to obtain the FE-element closes to the the cell face
for *i:* 1 ***to*** *j* **do**
    $\beta = \hat{\mathbf{n}}_{cf} \cdot \frac{e_i}{||e_i||}$
    if $\beta < \alpha$ **then**
        $\alpha = \beta$
        $e^* \rightarrow e_i = (p^*, v_i)$
        read: FE-elements sharing $e_i \rightarrow \{FE_l, FE_k\}$
get: eges e_{FE_l}, e_{FE_k} of (FE_l, FE_k) at p^* different from e^*
compute: angle between e_{FE_l} and $\hat{\mathbf{n}}_{cf} \rightarrow \beta_l$
compute: angle between e_{FE_k} and $\hat{\mathbf{n}}_{cf} \rightarrow \beta_k$
if $\beta_k < \beta_l$ **then**
    set: $FE^* \leftarrow FE_k$
if $\beta_l < \beta_k$ **then**
    set: $FE^* \leftarrow FE_l$
read: FE^* normal $\rightarrow \hat{\mathbf{n}}_{FE^*}$
$\hat{\mathbf{n}}_{test} = \hat{\mathbf{n}}^* + vc_1$
if *signVol*$(vc_1, vc_2, vc_3, v_i) > 0$ **then**
    if *signVol*$(vc_1, vc_2, vc_3, \hat{\mathbf{n}}_{test}) < 0$ **then**
        set face polygon activity = 0
if *signVol*$(vc_1, vc_2, vc_3, v_i) < 0$ **then**
    if *signVol*$(vc_1, vc_2, vc_3, \hat{\mathbf{n}}_{test}) > 0$ **then**
        set face polygon activity = 0

4-2-4 Implementation: Half-space Determination Method 3.

In this subsection the application of the Half-space Determination Method 3 is shown. For this purpose a test cell and a set of elements of the FE triangulation are chosen such that the vertex at which the FE-elements join is the only polypoint at one of the cell faces and the FE elements do not slice the cell face but some slice through the cell face plane, see Figure 4-5

The pseudocode of the coloring routine is in Algorithm 4.

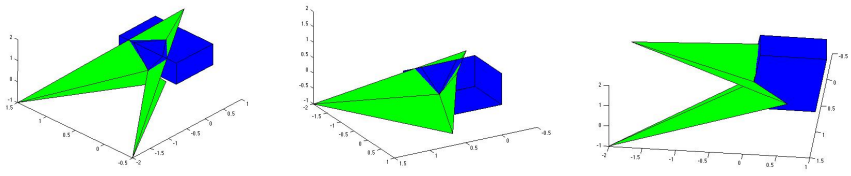


Figure 4-5: Test case for Half-space Determination Method 3.

Algorithm 4: Half-space Determination Method 3.

Input: CellFace(k) $\rightarrow \mathbf{P}_{cf} = (vc_1, vc_2, vc_3, vc_4)$
Output: Status of face polygons

set face polygon activity = 1
read: face polypoint $\rightarrow p^*$
read: cell face edges $ce_i = (vc_i, vc_{i+1})$, $ce_{i+1} = (vc_{i+1}, vc_{i+2})$
read: cell face edges normals $\hat{\mathbf{n}}_i$, $\hat{\mathbf{n}}_{i+1}$
read: FE-elements at $p^* \rightarrow (FE_1, FE_2, \cdots, FE_j)$
read: polypoints of FE-elements on l_{cf} plane $\rightarrow \{z_1, z_2, \cdots z_k\}$
create: face segments $cs_i = (p^*, z_i)_{i\in(1,2,\cdots k)}$
$\beta \leftarrow \frac{3\pi}{2}$
$FE^* \leftarrow$null
//Loop to obtain the FE-element closes to the the cell face
for *i:* 1 ***to*** *k* **do**
    compute: angle between cs_i and $\hat{\mathbf{n}}_i \rightarrow \beta_i$
    if $\beta_i < \beta$ **then**
        $\beta = \beta_i$
        $FE^* \leftarrow FE_i$
read: FE^* normal $\rightarrow \hat{\mathbf{n}}_{FE^*}$
project: $\hat{\mathbf{n}}_{FE^*}$ onto $\hat{\mathbf{n}}_i$ and normalize $\rightarrow \hat{\mathbf{n}}^i_{FE^*}$
project: $\hat{\mathbf{n}}_{FE^*}$ onto $\hat{\mathbf{n}}_{i+1}$ and normalize $\rightarrow \hat{\mathbf{n}}^{i+1}_{FE^*}$
if $0 < \beta_i < \frac{\pi}{4}$ **then**
    if $||\hat{\mathbf{n}}_i + \hat{\mathbf{n}}^i_{FE^*}||$=0 **then**
        set polygon(i) activity = 0
if $\frac{\pi}{4} < \beta_i < \frac{3\pi}{4}$ **then**
    if $||\hat{\mathbf{n}}_{i+1} + \hat{\mathbf{n}}^{i+1}_{FE^*}||$=0 **then**
        set polygon(i) activity = 0
if $\frac{3\pi}{4} < \beta_i < \frac{3\pi}{2}$ **then**
    if $||\hat{\mathbf{n}}_i + \hat{\mathbf{n}}^i_{FE^*}||$=2 **then**
        set polygon(i) activity = 0

The result is can be seen in Figure 4-6, which shows the FE-elements and their outward pointing

normals and rotated images of the resulting cut cell.

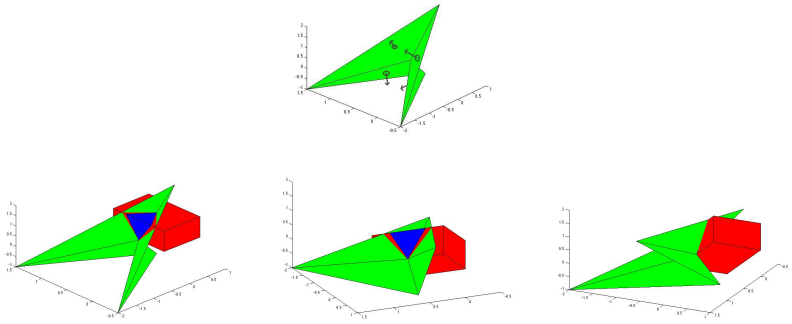


Figure 4-6: Results for Half-space Determination Method 2.

4-3 Implementation Results

The application of the different coloring routines that comprise the Coloring Algorithm successfully determines the active and inactive polygonal faces of the test cut cells.

The test cut cells chosen are representative of the different types of cut cells that can occur according to the classification presented in Chapter 2.

The Coloring Algorithm presented is capable of correctly determining the active and inactive regions of the different types of cut cells that can occur.

Chapter 5

Conclusions and Future Work

5-1 Contributions

In this project an algorithm, known as the *Coloring Algorithm*, was developed to determine the active and inactive polygonal faces of the cut cell for a Cartesian cut-cell method.

The complete *Coloring Algorithm* comprises of four routines that determine the activity of the face polygons for different types of cell faces. The classification of the cell faces depends on topological characteristics which are all verified via the *Signed Volume Test*.

The proposed *Coloring Algorithm* presents three qualities which makes it adequate for implementation for a *Cartesian cut cell* method:

- A classification of the cut cells was presented based on topological characteristics. Based on this classification the *Coloring Algorithm* was designed to be able to treat all the types of cell faces.
- The determination of the activity of the face polygons depends only on local information available for each cell. This makes it suitable also for parallel implementations.
- The determination of the activity of the polygonal faces utilizes only topological tests. This implementation avoids problems derived from machine precision and truncation errors.

The proposed *Coloring Algorithm* was implemented in **Matlab** and it was shown to be able to handle the test problems correctly.

5-2 Conclusions

Although many methods to determine the inside and outside regions of polygons and polyhedra can be found in the literature, these are in general not suitable for the problem treated in this work. All these

methods correspond to the solution of the so-called *Point in Polyhedron problem*, which generally deals with single objects in space, and most of them rely on a proper selection of a test point and the use of geometrical constructors.

Due to the large number of cells of a Cartesian mesh and the need to maintain computational efficiency and accuracy, the usual approaches for the solution for the *Point in Polyhedron problem* is not suitable.

The *Coloring Algorithm* designed in this work addresses and solves the deficiencies of the usual solutions to the *Point in Polyhedron problem*. Based on the classification for the cut-cells presented in this work and the choice of the test problems, it is concluded that the *Coloring Algorithm* designed, presumably determines correctly the active and inactive regions for all possible geometries of cut-cells.

Appendix A

Matlab code

This appendix contains the Matlab implementation of the Coloring Algorithm comprised of the different routines developed in this work. Topological tests, on the FE-elements and cell faces, are performed during the execution of the Coloring Algorithm in order to define the type of cell faces and the coloring routines that must be used.

As an input this algorithm receives a cut cell and all it geometrical information.

A-1 Coloring Algorithm

```
1  %This scrit contians the methods: Extended concistency method, Local half
2  %space determination method 2 and Local Half space determination method 3
3  %presented the thesis
4
5  %it receives an object of the class myCell, which contains al the
       geometric
6  %information of the cut cell.
7
8  %there are three types of cell faces: face type 1 are faces treated with
9  %the Extended concistency method, this faces are examples of test cases
       1,2
10 %and 4; face type 3 are faces treated with the Local half space
11 %determination method 2, these faces are examples of test cases 5; face
12 %type 3 are faces treated with the Local half space determination method
       2,
13 %these faces are examples of test cases 6,7, 8 and 9;
14
15 function colorCell(myCell)
16
17
18 v1=[0,0,0];
19 v2=[1,0,0];
```

```
20 v3=[1,1,0];
21 v4=[0,1,0];
22 v5=[0,0,1];
23 v6=[1,0,1];
24 v7=[1,1,1];
25 v8=[0,1,1];
26
27 act = myCell.Activity;
28
29 %if the cell is active
30 if (act == 1)
31     %loop over faces of the cell to color each polygon
32     for f=1:6;
33
34
35         %read the type of cell face
36         faType = myCell.Faces{f}.Type;
37
38         switch faType
39
40
41
42
43             case 1 %test cases 1,2,4
44                 segments=myCell.Faces{f}.Segments;
45
46                 numPol=myCell.Faces{f}.NumPol;
47
48                 numSeg=myCell.Faces{f}.NumSeg;
49
50                 %loop over the oplygons to analize thir activity
51                 if (numSeg>0)
52                     for p=1:numPol
53
54                         %get the polygon p of the list of polygons
55                         polygon=myCell.Faces{f}.Polygons{p};
56
57                         edges=polygon.Edges;
58                         numEdges=myCell.Faces{f}.Polygons{p}.NumEdge;
59
60                         %loop to traverse the polygon
61                         for e=1:numEdges
62
63                             %loop to verify each edge e against the
                                segment s
64                             for s=1:numSeg
65                                 %veriication only to look for incorrect
66                                 %directions, by deault polygons are
                                    active
67
68                                 %this if verifies if the first point of
                                    the
```

```
69                                  %edge e is the same as the second point
                                        of the
70                                  %segment s.
71                                  if (edges{e}(1)==segments{s}(4))&&(edges{
                                        e}(2)...
72                                          ==segments{s}(5))&&(edges{e}(3)
                                                ==...
73                                          segments{s}(6))
74
75                                      %if the above condition was satiffied
                                            , then
76                                      %this verifies if the second point
                                            the edge
77                                      %e is the same as the first point of
                                            the
78                                      %segment s
79                                      if (edges{e}(4)==segments{s}(1))&&...
80                                              (edges{e}(5)==segments{s}(2))
                                                    &&...
81                                              (edges{e}(6)==segments{s}(3))
82
83                                          %this changes the activity of
                                                polygon p
84                                          %to inactive==0.
85                                          myCell.Faces{f}.Polygons{p}.
                                                Activity=0;
86                                      end
87                                  end
88
89                              end
90                          end
91                          myCell.Faces{f}.Polygons{p}.Activity;
92                      end
93                  end
94
95
96
97
98              case {3,4} %test cases 5,6,7,8 and 9
99                  Ppoint=myCell.Faces{f}.Polygons{2}.Polygon{:}
100                 innerVertex=myCell.Faces{f}.InnerVertex
101                 outVer=myCell.Faces{f}.OuterVertex
102                 %innerVertex.PpCoordinate
103
104                 sigZero=0;
105                 sigPos=0;
106                 sigNeg=0;
107                 ang=1;
108                 edge=0;
109
110                 %extract the vertices of the cell face
111                 cellvert=myCell.Cellvertices;%matrix 8X3 of vertices
                        whole cell
```

```
        cellface=myCell.Cellfaces(f,:); %vertices of cell f

        %get the first three vertices (doesnt matter which)
        vert1=cellvert(cellface(1),:);
        vert2=cellvert(cellface(2),:);
        vert3=cellvert(cellface(3),:);

        %extract FE elements to analize side of plane
        fevert=myCell.FEVert;
        feelems=myCell.FEElems;

        [numVert,dummy1]=size(fevert);

        %loop over vertices to obtain signed volume of the vertices
        %of the FE elements w.r.t. te cell face, this is used to
        %determine if the polypoint corresponds to a problem of the
        %type test case 5 or test cases 6,7,8,or 9
        for fev=1:numVert
            %fev
            testVertex=fevert(fev,:);

            e1=vert1-testVertex;
            e2=vert2-testVertex;
            e3=vert3-testVertex;

            vol=sign(1/6 *det([e1;e2;e3]));

            %next two if's count the possitive and negative
            if vol==1
                sigPos=sigPos+1;
            end

            if vol==-1
                sigNeg=sigNeg+1;
            end

        end

        sigPos
        sigNeg

        %-%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %condition for test case 5
        if (sigPos==0 || sigNeg==0)
            sigZero=56;
            normal=myCell.Cellnormals(f,:)
            incidentEdges=innerVertex.EdgesPp;

            [numEdges,n]=size(incidentEdges);
```

```
161
162                 %compute the angle between each edge and the fixed
163                 %normal fixEdge
164                 for inED=1:numEdges
165                     insED=incidentEdges(inED,4:6)-Ppoint;
166                     insED=insED/norm(insED);
167                     angt=dot(normal,insED);
168
169                     %loop to save the closest edge to cell face and
                            to
170                     %get the FE element to which it belongs
171                     if (abs(angt) <= ang)
172                         ang=abs(angt);
173                         edge=inED;
174                         FE1ind=incidentEdges(inED,7);
175                         FE2ind=incidentEdges(inED,8);
176                     end
177
178                     %get the FE element to which it belongs
179                     FE1=innerVertex.FEelems(FE1ind);
180                     FE2=innerVertex.FEelems(FE2ind);
181
182
183                 end
184
185                 %get edges of elements sharing edge closes to cell
                        face
186                 edgesFE1=FE1.Edges
187                 edgesFE2=FE2.Edges
188
189                 %initialize variable for sum of angles
190                 angFE1=0;
191                 angFE2=0;
192                 %initialze working FE element
193                 FEdef=0;
194
195                 %loop to verify angle of all edges of FE1 and FE2
                        that
196                 %have Pppoint and obtain the closest
197                 for i=1:3
198                     angFE1=0;
199                     angFE2=0;
200                     if (isequal(edgesFE1(i,1:3),Ppoint))
201                         FE1E1=edgesFE1(i,4:6)-Ppoint;
202                         FE1E1=FE1E1/norm(FE1E1);
203                         angFE1=dot(normal,FE1E1);
204                     end
205
206                     if (isequal(edgesFE2(i,1:3),Ppoint))
207                         FE2E1=edgesFE2(i,4:6)-Ppoint;
208                         FE2E1=FE2E1/norm(FE2E1);
209                         angFE2=dot(normal,FE2E1);
210                     end
```

```
                        if (isequal(edgesFE1(i,4:6),Ppoint))
                            FE1E2=edgesFE1(i,1:3)-Ppoint;
                            FE1E2=FE1E2/norm(FE1E2);
                            angFE1=angFE1+dot(normal,FE1E2);
                        end

                        if (isequal(edgesFE2(i,4:6),Ppoint))
                            FE2E2=edgesFE2(i,1:3)-Ppoint;
                            FE2E2=FE2E2/norm(FE2E2);
                            angFE2=angFE2+dot(normal,FE2E2);
                        end

                        if (abs(angFE1)<= abs(angFE2) && abs(angFE1)>0)
                            FEdef=FE1;
                        end

                        if (abs(angFE2)<= abs(angFE1) && abs(angFE2)>0)
                            FEdef=FE2;
                        end

                    end

                    %compute angle between face normal and FE normal
                    FENor=FEdef.OutNormal;
                    angNorFaNorFE=dot(normal,FENor);


                    %consitions to determine the activity of the cell
                        face
                    if (angNorFaNorFE > 0)
                        if(sigNeg >0)
                            myCell.Faces{f}.Polygons{1}.Activity=0;
                        end
                        if(sigPos >0)
                            myCell.Faces{f}.Polygons{1}.Activity=1;
                        end
                     end

                    if (angNorFaNorFE < 0)
                        if(sigNeg >0)
                            myCell.Faces{f}.Polygons{1}.Activity=1;
                        end
                        if(sigPos >0)
                            myCell.Faces{f}.Polygons{1}.Activity=0;
                        end
                    end
                end


                %
                    -%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
261                 %condition for test case 6,7,8 or 9
262                 if (sigPos>0 && sigNeg>0)
263                     fT=faType;
264                     FEelements=outVer.FEelems;
265                     numElemMin=1;
266                     normal=outVer.FaceEdgeNormal
267                     csList=outVer.CSedges
268
269                     [m,n]=size(csList);
270
271                     tePpoint=[Ppoint,Ppoint,0];
272
273                     %loop over cs segments to deternmine the closest to
                            the
274                     %edge
275                     for ics=1:m
276                         csList(ics,:)=csList(ics,:)-tePpoint;
277
278                         tcs=csList(ics,4:6)/norm(csList(ics,4:6));
279
280                         angTemp=dot(normal,([-1,.1,0]/norm([-1,.1,0])))
281
282                         %keeps the number of FE element closest to cell
283                         %face
284                         if (angTemp <= numElemMin)
285                             numElemMin=csList(ics,7)
286                         end
287                     end
288
289                     elementNormal=FEelements(3).OutNormal
290
291                     %projection of the outward normal of the FE-element
292                     %onto the cell edge normal
293                     signOfProjection=sign(dot(normal,elementNormal))
294
295                     if (signOfProjection==-1)
296                         myCell.Faces{f}.Polygons{1}.Activity=0;
297
298                     end
299
300                     if (signOfProjection==1)
301                         myCell.Faces{f}.Polygons{1}.Activity=0;
302                     end
303
304
305                 end
306
307
308
309         end
310
311
312
```

```
313
314         %section which colors the polygons and returns a graphic output
                of
315         %the cut cell
316         numPol=myCell.Faces{f}.NumPol;
317         for p=1:numPol
318             pol=[];
319             temPol=myCell.Faces{f}.Polygons{p};
320
321             numVert=temPol.NumVert;
322             %this builds a matrix pol with the vertices in column form
                    for
323             %the polygon p on face f
324             for v=1:numVert
325                 pol=[pol;temPol.Polygon{v}];
326             end
327
328             p1=pol(:,1);
329             p2=pol(:,2);
330             p3=pol(:,3);
331
332             if (temPol.Activity==1)
333                 fill3(p1,p2,p3,'b');
334                 hold on,
335             end
336
337             if (temPol.Activity==0)
338                 fill3(p1,p2,p3,'r');
339                 hold on,
340             end
341
342         end
343
344
345         %
346         %
347         %here ends the color section
348
349     end
350
351     %section to graph the FE elements
352     FEvert=myCell.FEVert;
353     FEelems=myCell.FEElems;
354
355     [m,n]=size(FEelems);
356     %
357     p1=[];
358     p2=[];
359     p3=[];
360
361     elem=[];
362
363     for k=1:m
```

```
364             FEvert(FEelems(k,1),:);
365             FEvert(FEelems(k,2),:);
366             FEvert(FEelems(k,3),:);
367             elem=[FEvert(FEelems(k,1),:);FEvert(FEelems(k,2),:);...
368                 FEvert(FEelems(k,3),:)];
369
370             p1=elem(:,1);
371             p2=elem(:,2);
372             p3=elem(:,3);
373
374             fill3(p1,p2,p3,'g');
375             hold on,
376         end
377         %end graphing FE elems
378
379
380 end
```

Bibliography

[1] M. G. Whitepaper, "Advanced immersed boundary cartesian meshing technology in floefd," 2011.

[2] M. Aftosmis, *Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries*. Von Karman Institute for Fluid Dynamics Lecture Series, March 1997.

[3] G. Yang, D. M. Causon, and D. M. Ingram, "Calculation of compressible flows about complex moving geometries using a three-dimensional cartesian cut cell method," *International Journal of Numerical Methods in Fluids*, 2008.

[4] S. Kang, G. Iaccarino, F. Ham, and P. Moin, "Prediction of wall-pressure fluctuation in turbulent flows with an immersed boundary method," *Journal of Computational Physics*, 2009.

[5] H. Bandringa, *Immersed boundary method*. PhD thesis, University of Groningen, 2010.

[6] J. Peteker, *Point in Polygon Detection*. PhD thesis, University of California, 2010.

[7] H. Cundy and Rollett, *Mathematical Models*. Oxford: Tarquin Publications, 1989.

[8] E. Haines, "Poin in olygon strategies," in *Graphics Gems IV*, Academic Press, 1994.

[9] F. Feito and M. Rivero, "Geometric modeling based on simplicial chains," *Computer and Graphics*, no. 5, pp. 611–619, 1998.

[10] G. Lonsdale, *Numerical simulation in automotive design.* C and C Research laboratories, NEC Europe Ltd., St. Augustin, Germany.

[11] E. van Groesen and J. Molenaar, *Continuum modeling in the physical sciences*. Philadelphia: Society of Industrial and Applied Mathematics, 2007.

[12] C. Hirsch, *Numerical Computations of Internal and External Flows.* Salisbury: Elesevier, 2007.

[13] F. Cirak and R. Radovitzky, "A lagrangian-eulerian shell-fluid coupling algorithm based on level set methods," *Computers and structures*, 2005.

[14] D. Causon, C. Mingham, and L. Qian, "Inrtroductory finite volume methods for pdes," Manchester Metropolitan University, 2011.

[15] R. J. Leveque, *Finite Volume Methods for Hyperbolic Problems*. Cambridge: Cambridge text in Applied Mathematics, 2004.

[16] P. Tucker and Z. Pan, "A cartesian cut cell method for incompressible viscous flow," *Applied Mathematics Modeling*, 2000.

[17] R. Mittal and G. Iccarino, "Immersed boundary method," *Annuel Review of Fluid Mechanics*, 2005.

[18] M. Meyer, A. Devesa, S. Hickel, X. Hu, and N. Adams, "A conservative immersed interface method for large-eddy simulation of incompressible flows," *Journal of Computational Physics*, 2010.

[19] D. Hartmann, M. Meinke, and W. Schröder, "A strictly conservative cartesian cut-cell method for compressible viscous flows on adaptive grids," *Comput. Methods Appl. Mech. Engrg.*, 2011.

[20] Y. Peng, J. Yong, W. Dong, H. Zhang, and J. sun, "A new algorithm for boolean operations on general polygons.," *Computers and Graphics 29*, 2005.