Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Triangulation of implicit surface with singularities

Master's thesis

2021
Bc. Kristína Korecová

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Triangulation of implicit surface with singularities

Master's thesis

Study Programme:	Computer Graphics and Geometry
Field of Study:	Mathematics
Department:	Department of Algebra and Geometry
Supervisor:	doc. RNDr. Pavel Chalmovianský, PhD.

Bratislava, 2021
Bc. Kristína Korecová

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Bc. Kristína Korecová
Study programme: Computer Graphics and Geometry (Single degree study, master II. deg., full time form)
Field of Study: Mathematics
Type of Thesis: Diploma Thesis
Language of Thesis: English
Secondary language: Slovak

Title: Triangulation of implicitely defined surfaces

Annotation: The student proposes an algorithm for triangulation of an implicitely defined surface. Given the singularities of the surface, one start to triangulate from the singular points. The regular parts should be triangulated adaptivly and uniformly. Final surface can be further optimized. The student should provide a way of numerical computation of singular points at least in case of algebraic surfaces of low degree.

Aim: Provide a triangulation of a surface given as a zero set of a function. Compare the results with known approaches in terms of quality and computational algorithm.

Literature: B. R. de Araújo, Daniel S. Lopes, Pauline Jepp, Joaquim A. Jorge, and Brian Wyvill. 2015. A Survey on Implicit Surface Polygonization. ACM Comput. Surv. 47, 4, Article 60 (July 2015), 39 pages. DOI:https://doi.org/10.1145/2732197

E. Hartmann: A marching method for the triangulation of surfaces, The Visual Computer (1998), 14, pp. 95–108

S. Akkouche & E Galin: Adaptive Implicit Surface Polygonization Using Marching Triangles, Computer Graphics Forum (2001), Vol. 20, pp. 67–80

Keywords: implicitly defined surface, triangulation, computational approach

Supervisor: doc. RNDr. Pavel Chalmovianský, PhD.
Department: FMFI.KAG - Department of Algebra and Geometry
Head of department: doc. RNDr. Pavel Chalmovianský, PhD.

Electronic version available:
prípustná pre vlastnú VŠ

Assigned: 19.10.2021

Approved: 20.10.2021 prof. RNDr. Július Korbaš, CSc.
Guarantor of Study Programme

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

.....
Student

.....
Supervisor

Acknowledgements: TODO Acknowledgements

Abstrakt

TODO Abstrakt po Slovensky

Kľúčové slová: TODO Kľúčové slová

Abstract

TODO Abstract in English

Keywords: TODO Keywords

Contents

List of Figures

List of Tables

Introduction

In mathematics, an implicit surface in $\mathbb{R}^3$ is a set of points where a given function $F : \mathbb{R}^3 \to \mathbb{R}$ is zero. Due to unification of constructive solid geometry modelling and implicit surfaces, we can now model and represent very complex surfaces using a single function.

In computer graphics, triangular mesh is the most common way of surface representation and visualization. Calculating the intersection of a ray and a triangular mesh is far easier task as calculating the intersection of a ray and an implicit surface.

Many surfaces contain singular points where all three partial derivatives vanish. These points may cause problems for meshing algorithms. Some meshing algorithms, such as Marching Cubes, simply ignore these points, which leads to insufficient surface representation.

In this thesis, we follow up on our effort in bachelor's thesis [13] to present an algorithm for creating the triangular mesh of regular implicit surfaces.

First, we reimplement the algorithm to be more effective, by using advanced data structures for mesh representation and 3D point search.

Next, we extend the algorithm to include the triangulation of certain types of isolated singularities – ADE singularities, and non-isolated singularities – curves on the intersection of two regular surfaces. We analyze the geometry of these singularities and propose an approach to create a mesh, which correctly captures the geometry of the singularities.

Lastly, we propose the adaptive technique, which changes the size of the triangles based on the local curvature of the surface.

Chapter 1

Theoretical background

1.1 Implicit surfaces

Implicit functions are a tool for surface representation and manipulation. In computer graphics, the implicit functions can be used to model complex surfaces using boolean operations, realistic animations, rendering and others.

Implicit functions do not define the surface explicitly. Instead, the surface is defined as a zero set of a function.

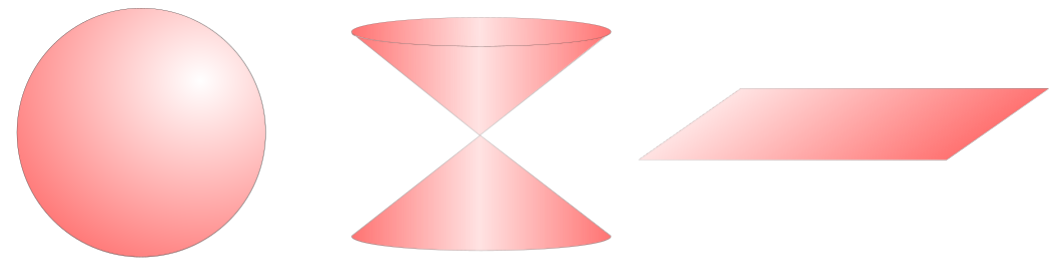
Definition 1 *Given a function* $F : \mathbb{R}^3 \to \mathbb{R}$*, one can define surface implicitly as a set of points* $(x, y, z) \in \mathbb{R}^3$ *that satisfy* $F(x, y, z) = 0$.

Some examples of implicit surfaces and their equations can be seen on the Figure 1.1.

Unit normal vector of the implicit surface in point (x_0, y_0, z_0) is the normalized gradient of the implicit function required to be differentiable in that point provided it is non-zero.

Definition 2 *Gradient vector of a function* $F : \mathbb{R}^3 \to \mathbb{R}$ *is defined as*

$$\nabla F(x, y, z) = \left(\frac{\partial F(x, y, z)}{\partial x}, \frac{\partial F(x, y, z)}{\partial y}, \frac{\partial F(x, y, z)}{\partial z} \right).$$

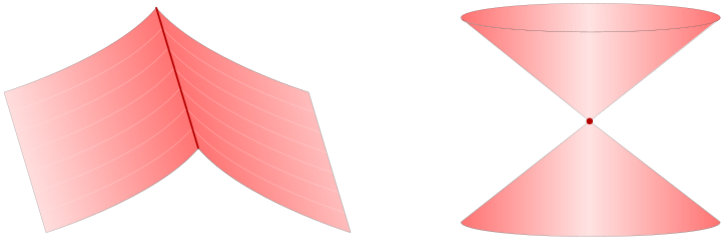


$F = x^2 + y^2 + z^2 - 1$ a) sphere

$F = x^2 + y^2 - z^2$ b) cone

$F = z$ c) plane

Figure 1.1: Implicit surfaces with corresponding equations.



a) non-isolated singularity b) isolated singularity

Figure 1.2: Isolated and non-isolated singularities of an implicitly defined surfaces.

If $\nabla F(x,y,z) \neq 0$*, we can define the unit normal vector of* F *as a normalized gradient vector*

$$N(F(x,y,z)) = \frac{\nabla F(x,y,z)}{\|\nabla F(x,y,z)\|}.$$

The oints lying on an implicit surface can be classified as regular or singular based on the value of the gradient vector at that point.

Definition 3 *The point* $P = (x,y,z)$ *lying on the implicit surface is considered regular if* $\nabla F(x,y,z) \neq \vec{0}$*. On the contrary, the point* P *is said to be singular if* $\nabla F(x,y,z) = \vec{0}$*.*

Singular points can be further classified as isolated or non-isolated.

Definition 4 *Singular point* P *of a surface* S *is said to be isolated, if there exists an open ball* $B_\varepsilon(P)$*, which does not contain any other singular point of* S*. Singular point* P *is said to be non-isolated if it is not isolated.*

In the Figure 1.2, one can see an example of isolated and non-isolated singularities.

1.1.1 Curvature of a surface

Curvature is a fundamental concept in differential geometry of curves and surfaces. In the case of curves, the curvature measures how much the curve differs from a straight line. It is defined as the inverse of the radius of the osculating circle, which is the best circle approximation of the curve.

For surfaces, the curvature is a measure of how much does the surface differ from a plane. The definition of the curvature of a surface is more complex than in the case of curves. The curvature of a curve on a surface depends on the choice of the tangent direction of a measured point.

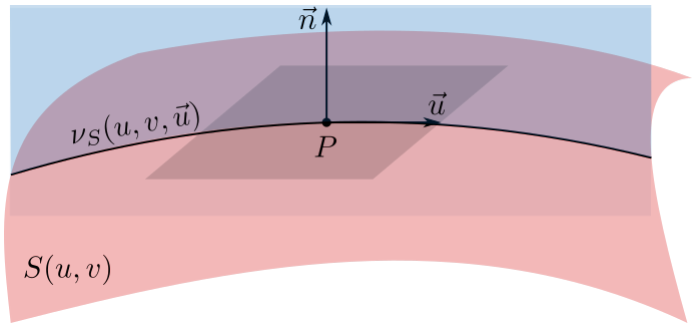


Figure 1.3: Normal cut of the parametric surface $S(u, v)$.

The idea of measuring the curvature of a surface has a long history in mathematics. One of the first contributors was the mathematician Carl Friedrich Gauss, who developed the idea of the Gaussian curvature of surfaces. In this subsection, we draw from the summary presented by Tiago Novello et al.[16].

Normal curvature of the surface

Let S be a parametric surface

$$S : D \subseteq \mathbb{R}^2 \to \mathbb{R}^3$$

$$S(u, v) = (S_x(u, v), S_y(u, v), S_z(u, v)), \quad (u, v) \in \mathbb{R}^2$$

Let the unit normal vector of the surface S at the point $S(u, v)$ be $\vec{n}(u, v)$.

We define the normal curvature of the surface as a function of the location of the point on the surface given by parameters u and v and the unit tangent vector in that point $\vec{t}$.

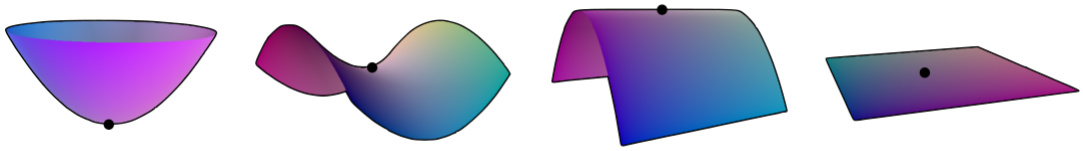
Definition 5 *The normal cut of a surface S at the regular point P in the direction of the unit tangent vector $\vec{t}$ is defined as an intersection of the surface S and a plane given by the vectors $\vec{t}$ and $\vec{n}(u, v)$ shifted to the point $S(u, v)$.*

The visualization of the normal cut is shown in the Figure 1.3. It is clear that the normal cut is a plane curve lying on the surface. We denote this normal cut as $\nu_S(u, v, \vec{t})$.

Definition 6 *The oriented normal curvature of the surface at the regular point P in the direction of the unit tangent vector $\vec{t}$ is defined as the oriented curvature of the normal cut $\nu_S(u, v, \vec{t})$. Non-oriented normal curvature is defined as an absolute value of the oriented normal curvature.*

Definition 7 *Minimal and maximal curvature at the regular point $P = S(u, v)$ are defined as*

$$\kappa_{min}(u, v) = \min_{\vec{t} \in T_P S} \nu_S(u, v, \vec{t}),$$



a) elliptic point b) hyperbolic point c) parabolic point d) planar point

Figure 1.4: Categorization of the surface points based on Gaussian curvature [15].

$$\kappa_{max}(u,v) = \max_{\vec{t} \in T_P S} \nu_S(u,v,\vec{t}),$$

where $T_P S$ is a tangent plane of the surface S in the point P.

Minimal and maximal curvatures are called principal.

Definition 8 *Gaussian curvature at a point $S(u,v)$ is defined as a product of principal curvatures at that point, i.e.*

$$\kappa_G(u,v) = \kappa_{min}(u,v)\kappa_{max}(u,v).$$

Gaussian curvature describes the shape of the surface in the local neighbourhood of the point. Points of the surface S, where the Gaussian curvature is positive, are called elliptic. Points of the surface S, where the Gaussian curvature is negative, are called hyperbolic. Points of the surface S, where only one of $\kappa_{min}, \kappa_{max}$ is zero, are called parabolic and points, where both κ_{min} and κ_{max} are zero, are called planar. The shape of the surface in the local neighbourhoods of the points, visualized on the Figure 1.4, is as follows:

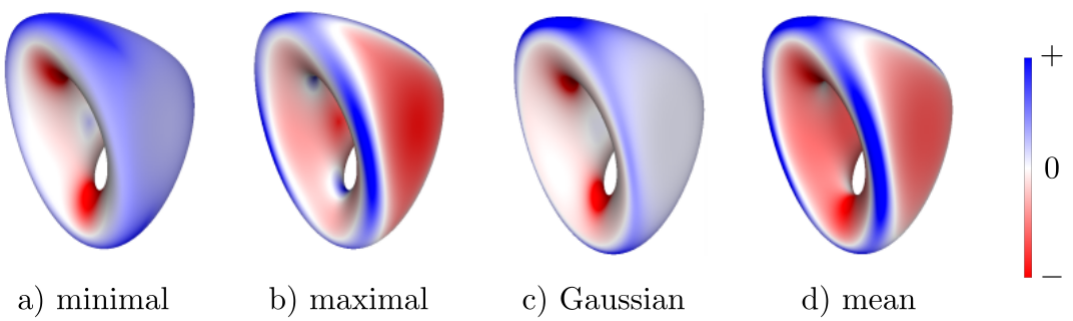
- elliptic points $\longrightarrow$ surface is locally curved like an ellipsoid,
- hyperbolic points $\longrightarrow$ surface is locally curved like a saddle,
- parabolic points $\longrightarrow$ surface is locally curved like a parabolic cylinder,
- planar points $\longrightarrow$ surface is locally flat like a plane.

Gaussian curvature is an intrinsic property, which means that it is also independent of the isometric image of the surface.

Definition 9 *Mean curvature of a surface S at a point $P = S(u,v)$ is defined as the arithmetic mean of corresponding principal curvatures*

$$\kappa_M(u,v) = \frac{\kappa_{min}(u,v) + \kappa_{max}(u,v)}{2}.$$

Minimal, maximal, Gaussian and mean curvature are visualized in the Figure 1.5, where the blue colour indicates a positive value of the corresponding visualized curvature, red colour indicates a negative value of the corresponding visualized curvature and the white colour indicates a zero value of the corresponding visualized curvature.



a) minimal b) maximal c) Gaussian d) mean

Figure 1.5: Visualization of minimal, maximal, Gaussian and mean curvatures of the double-torus [16]. Blue/white/red colour corresponds to the positive/zero/negative value of the corresponding curvature.

1.1.2 Curvature formulas for implicit surface

A version of curvature formulas for implicit surfaces appeared in [19] and were reformulated, summarized and proved by Ron Goldman in [8]. In this subsection, we point out these formulas.

Let $F : \mathbb{R}^3 \to R$ be an implicit function which defines surface by the equation $F(x, y, z) = 0$. Let us denote $F_t = \frac{\partial F}{\partial t}$ and $F_{ts} = \frac{\partial^2 F}{\partial t \partial s}$. Hessian matrix – the matrix of the second-order derivatives is defined as

$$H(F) = \begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{yx} & F_{yy} & F_{yz} \\ F_{zx} & F_{zy} & F_{zz} \end{pmatrix},$$

and the adjoint of the Hessian is defined as a transpose of its cofactor matrix.

$$H^*(F) = \begin{pmatrix} \begin{vmatrix} F_{yy} & F_{yz} \\ F_{zy} & F_{zz} \end{vmatrix} & -\begin{vmatrix} F_{xy} & F_{xz} \\ F_{zy} & F_{zz} \end{vmatrix} & \begin{vmatrix} F_{xy} & F_{xz} \\ F_{yy} & F_{yz} \end{vmatrix} \\ -\begin{vmatrix} F_{yx} & F_{yz} \\ F_{zx} & F_{zz} \end{vmatrix} & \begin{vmatrix} F_{xx} & F_{xz} \\ F_{zx} & F_{zz} \end{vmatrix} & -\begin{vmatrix} F_{xx} & F_{xz} \\ F_{yx} & F_{yz} \end{vmatrix} \\ \begin{vmatrix} F_{yx} & F_{yy} \\ F_{zx} & F_{zy} \end{vmatrix} & -\begin{vmatrix} F_{xx} & F_{xy} \\ F_{zx} & F_{zy} \end{vmatrix} & \begin{vmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{vmatrix} \end{pmatrix}.$$

Now, one can find the formulas of Gaussian, mean, minimal and maximal curvature for an implicit surface $F = 0$ at a regular point.

Gaussian curvature is given by

$$\kappa_G = \frac{\nabla F \cdot H^*(F) \cdot \nabla F^T}{|\nabla F|^4}.$$

The mean curvature of the implicit surface defined by function F is given by

$$\kappa_M = \frac{\nabla F \cdot H^*(F) \cdot \nabla F^T - |\nabla F|^2 Tr(H)}{2|\nabla F|^3}.$$

The principal curvatures κ_{min} and κ_{max} can be calculated from Gaussian curvature and mean curvature as

$$\kappa_{min}, \kappa_{max} = -\kappa_M \pm \sqrt{\kappa_M^2 - \kappa_G}.$$

1.2 ADE singularities

ADE singularities, also referred to as du Val singularities, are a specific class of simple, isolated surface singularities. They were classified by Arnold's [4] according to ADE classification [9] based on correspondence of these singularities to simply laced Dynkin diagrams [6]. We know infinitely many A singularities – $A_1, A_2, ...$, infinitely many D singularities – $D_4, D_5, ...$ and three E singularities – E_6, E_7 and E_8. ADE singularities are specified by their normal forms. When working in complex space, each singularity has a single normal form:

- $A_n \qquad F(x,y,z) = x^{n+1} + y^2 + z^2,$
- $D_n \qquad F(x,y,z) = yx^2 + y^{n-1} + z^2,$
- $E_6 \qquad F(x,y,z) = x^3 + y^4 + z^2,$
- $E_7 \qquad F(x,y,z) = x^3 + xy^3 + z^2,$
- $E_8 \qquad F(x,y,z) = x^3 + y^5 + z^2.$

Each ADE singularity on a surface can be locally expressed by their normal form.

In the real case, changing the signs in these equations produces different surfaces, and therefore, ADE singularities can be further classified by their signature.

Definition 10 *Let real surface singularities based on their signature be denoted as follows:*

- $A_{n\pm\pm} \qquad F(x,y,z) = x^{n+1} \pm y^2 \pm z^2,$
- $D_{n\pm\pm} \qquad F(x,y,z) = yx^2 \pm y^{n-1} \pm z^2,$
- $E_{6\pm\pm} \qquad F(x,y,z) = x^3 \pm y^4 \pm z^2,$
- $E_{7\pm\pm} \qquad F(x,y,z) = x^3 \pm xy^3 \pm z^2,$
- $E_{8\pm\pm} \qquad F(x,y,z) = x^3 \pm y^5 \pm z^2.$

An ordinary cone is the most common example of a surface with ADE singularity. Given as the zero set of the function $F(x,y,z) = x^2 - y^2 - z^2$, cone has a singular point $P = (0,0,0)$. This singular point is an example of A_{1--} singularity.

1.2.1 Correspondence between $SO(3,\mathbb{R})$ group and ADE singularities

$SO(3,\mathbb{R})$ is a special orthogonal group over the field of real numbers in three dimensions. It is also called a $3D$ rotation group, as it is a group of all rotations around axes passing through the origin in R^3.

Definition 11 *$SO(3,\mathbb{R})$ is a group of 3×3 orthogonal matrices of real numbers with determinant 1.*

$$SO(3,\mathbb{R}) = \left\{ A \in \mathbb{R}^{3\times 3} \,|AA^T = I,\ det(A) = 1 \right\}.$$

Simply laced Dynkin diagrams correspond to all finite subgroups of $SO(3,\mathbb{R})$. Finite subgroups of $SO(3,\mathbb{R})$ are the rotational symmetry groups of

- pyramid with n vertices (cyclic subgroup $\overline{C}_n$) – Figure 1.6 a),
- double pyramid with n vertices (dihedral subgroup $\overline{D}_n$) – Figure 1.6 b),
- platonic solids:
  - tetrahedron (tetrahedral subgroup $\overline{T}$) – Figure 1.6 c),
  - octahedron (octahedral subgroup $\overline{O}$) – Figure 1.6 d),
  - icosahedron (icosahedral subgroup $\overline{I}$) – Figure 1.6 e).

The correspondence is as follows:

- $A_n \iff \overline{C}_{n+1}$,
- $D_n \iff \overline{D}_{n+2}$,
- $E_6 \iff \overline{T}$,
- $E_7 \iff \overline{O}$,
- $E_8 \iff \overline{I}$.

The conclusion is that ADE singularities correspond to finite subgroups of $SO(3,\mathbb{R})$, which represent these types of symmetries in $\mathbb{R}^3$.

1.3 CSG modelling for implicit surfaces

This section will discuss how constructive solid geometry can be used for modelling complex implicit surfaces. The first major publication regarding constructive solid geometry was published in 1977 by Requicha and Voelcker [18]. More detailed mathematical foundations were published a year later, in 1978, by Requicha and Tilove [17].

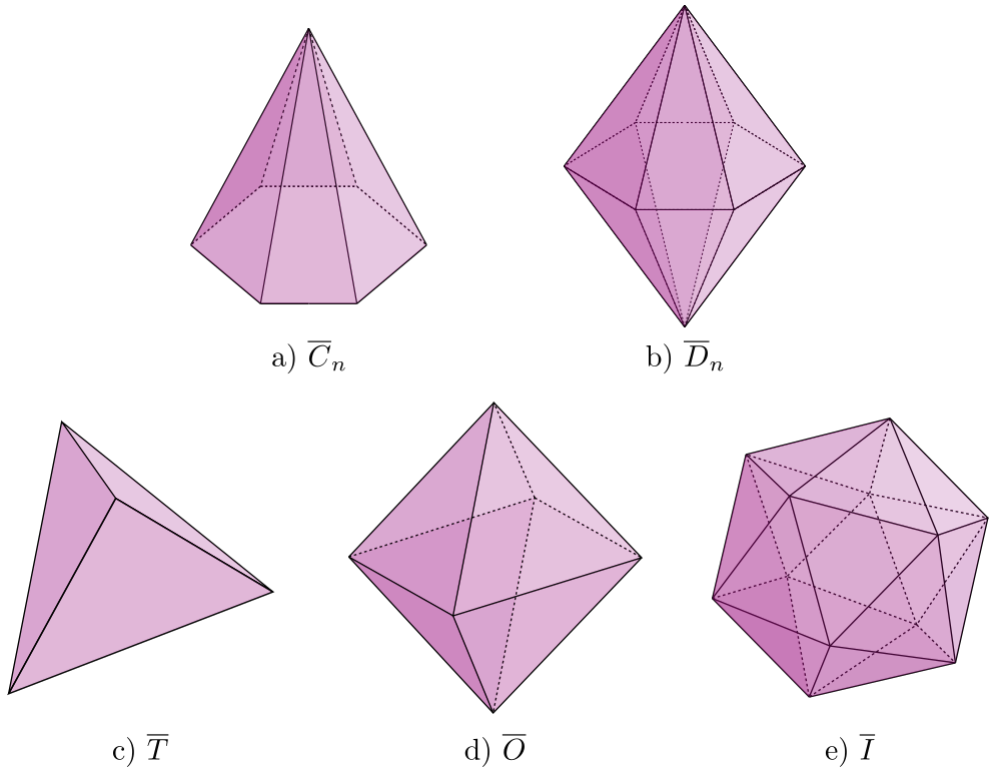


Figure 1.6: Shapes representing the finite subgroups of $SO(3, \mathbb{R})$ group.

1.3.1 Constructive solid geometry (CSG)

Constructive solid geometry is a technique used for modelling complex geometric objects using boolean operations on sets of points – union, intersection and difference. It gained popularity in the 1980s as a powerful tool to create complex shapes from sets of geometrical primitives, such as cylinders, spheres and cones. The resulting model can be represented as a CSG tree containing geometrical primitives in its leaves and boolean operations in its internal nodes [7]. An example of such a CSG tree can be seen in the Figure 1.7.

CSG for implicit surfaces

As the implicit surface divides space into inside ($F(x, y, z) < 0$) and outside ($F(x, y, z) > 0$), one can easily combine the idea of implicit surfaces and CSG modelling. Given two implicit surfaces represented by implicit functions F and G, the surface of the intersection of the interiors can be defined by the implicit function $H = min(F, G)$. Similarly, the surface of the union of the interiors can be defined by the implicit function $H = max(F, G)$ and lastly, the surface of the difference of the interiors can be defined by the implicit function $H = max(F, -G)$. Two-dimensional example of this idea can be seen in the Figure 1.8.

Theorem 1 *The minimum function* $min(F, G)$ *of two functions* $F : \mathbb{R}^3 \to \mathbb{R}$ *and*

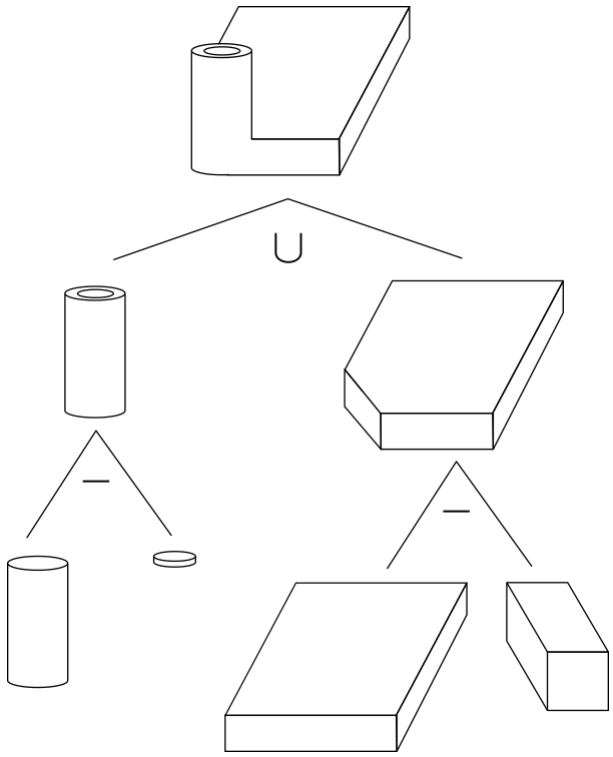


Figure 1.7: Example of CSG tree [7].

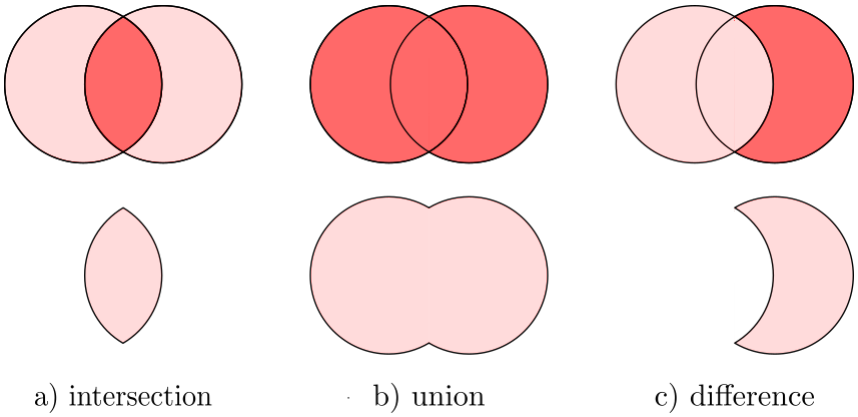


Figure 1.8: Boolean operations on implicit curves.

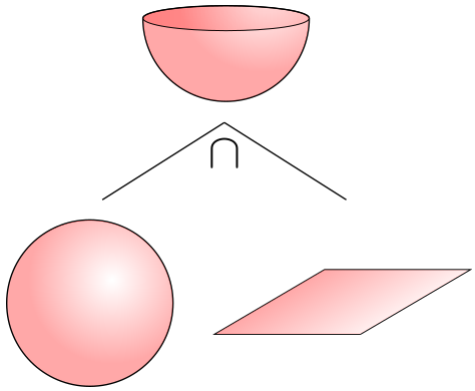


Figure 1.9: intersection of a sphere and a plane.

$G : \mathbb{R}^3 \to \mathbb{R}$ *is defined as*

$$min(F, G) = F + G - \sqrt{F^2 + G^2}$$

. The maximum function $max(F, G)$ *of two functions* $F : \mathbb{R}^3 \to \mathbb{R}$ *and* $G : \mathbb{R}^3 \to \mathbb{R}$ *is defined as*

$$max(F, G) = F + G + \sqrt{F^2 + G^2}$$

.

These formulas allow us to model implicit surfaces, which are the result of performing a finite number of operations union, intersection and difference on arbitrary implicit functions.

An easy example is creating an implicit equation representing half of a sphere. Let us have an implicit function $F(x, y, z) = x^2 + y^2 + z^2 - 1$, which represents a unit sphere and another implicit function $G(x, y, z) = z$ which represents the xy plane. The minimum function of these two functions

$$min(F, G) = x^2 + y^2 + y^2 - 1 + z - \sqrt{(x^2 + y^2 + y^2 - 1)^2 + z^2}$$

represents the surface of a half sphere. The visualization of these surfaces can be seen in the Figure 1.9.

1.4 Non-isolated surface singularities

As we already mentioned, the non-isolated singular point of the surface S is the singular point, for which every open ball $B_\varepsilon(P)$ contains other singular points of the surface S. Non-isolated singular points are also called singular curves.

In the CSG modelling, singular curves most commonly arise as a result of intersection, union and difference operations on the interiors of these surfaces. When performing the intersection, union or difference operation on the interiors of two regular

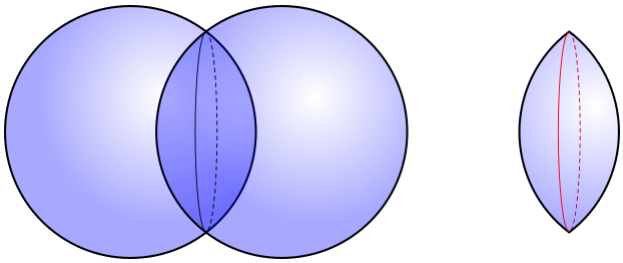


Figure 1.10: intersection of two spheres.

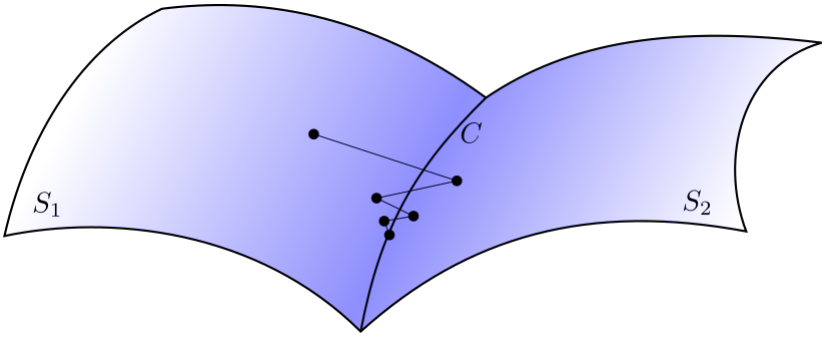


Figure 1.11: Projecting a point to the implicit curve.

surfaces, the singular curve is the curve given by the intersection of these surfaces. An example of an intersection of two spherical interiors is shown in the Figure 1.10. The red colour displays the resulting singular curve given by an intersection of the two spheres.

Projecting point to the implicit curve

Implicit curve C is given by two implicit equations - the equations of two surfaces which intersect in the curve C. To project a point close to the surface to the given surface, we use the method where we project the point in the direction of the normal vector using the iterative Newton-Rhapson method. An extension of this method can be used to project a point to the implicit curve by alternately projecting to the two surfaces until the point lies on both with the required precision. A visualization of this approach can be seen in the Figure 1.11. We denote the projection function $proj_C$.

Differential geometry of implicit curves

An implicit curve C is given by two implicit equations $F_1, F_2 : \mathbb{R}^3 \to \mathbb{R}$ representing two implicit surfaces S_1 and S_2, respectively, which intersect in the implicit curve C. The unit tangent vector $\vec{t}_C$ of the curve C in the point $P \in C$ is given by the cross product of unit normal vectors of the surfaces S_1 and S_2

$$\vec{t}_C(P) = \vec{n}_{S_1}(P) \times \vec{n}_{S_1}(P).$$

Chapter 2

Mathematical model of the proposed triangulation

2.1 Triangulation adaptive to the local curvature

As we explained at the beginning of chapter 1.1, the curvature of a surface measures how much the surface bends.

The triangulation of the surface should be accurate enough but also memory efficient. It can be achieved by creating a triangulation which is locally adaptive to the curvature of the surface. Therefore, having smaller triangles where the surface is curved and bigger triangles where the surface is flatter.

In this section, we present our implementation of the triangulation adaptive to the local curvature.

In the original algorithm, the height of the triangle which is projected to the surface is set to the constant value $\frac{\sqrt{3}}{2}e$, where e is the required length of the side of the triangle. To achieve the adaptivity of the size of the triangles, we set the height of the triangle to depend on the curvature in the given point, as shown in the Figure 2.1.

To identify the curved areas, we decided to use the curvedness defined by Koendering and Doorn in 1992 [12]. As the authors explained, the Gaussian curvature and the mean curvature are not very descriptive of the local shape of the surface. Principal curvatures taken as a pair are more informative. The curvedness was introduced to measure the curvature in the given point by a single number instead of a pair of

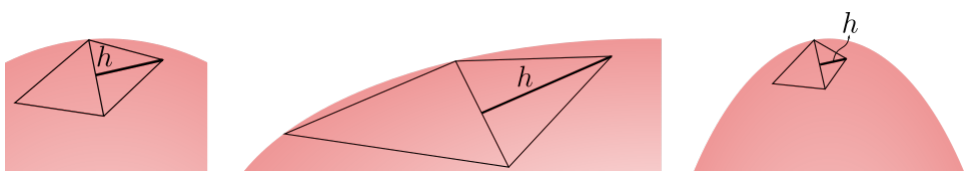


Figure 2.1: Adaptive height of the new triangle.

numbers. The authors defined curvedness as

$$c = \sqrt{\frac{\kappa_1^2 + \kappa_2^2}{2}},$$

where κ_1 and κ_2 are the principal curvatures in the given point.

Curvedness has some obvious properties:

- if $\kappa_1 == \pm\kappa_2$, then $c = |\kappa_1| = |\kappa_2|$,
- if $|\kappa_1| < |\kappa_2|$, then $|\kappa_1| < c < |\kappa_2|$,
- $c = 0$ only in planar points,
- $c \geq 0$.

One could say that curvedness says only about the amount of the curvature in the given point, not about the way the surface is curved at that point.

As an example, we describe the curvedness of the sphere and cylinder:

- **Sphere:** As the principal curvatures of a sphere are both equal to the inverse of the radius of the sphere, curvedness is also equal to the inverse of the radius of the sphere.
- **Cylinder:** One of the principal curvatures of a cylinder is equal to zero, and the other is equal to the inverse of the radius of the cylinder, curvedness is therefore equal to half of the inverse of the radius of the cylinder.

Let us define the curvedness radius as $r_c = \frac{1}{c}$. It can be perceived as the radius of a sphere, with curvedness c.

For the triangulation adaptive to the local curvature of the surface, an input size of the edge e is not perceived as the approximate size of the triangulation triangle. Instead, it is perceived as a measure of the level of detail. For the given edge size e, the level of detail for the triangulated surface is the same as if we wanted to triangulate the unit sphere uniformly using the triangles with the edge size e.

Let us define the constant of detail as $k_d = \frac{1}{e}$. It can be perceived as the number of times the edge size e would fit into the radius of the unit sphere.

Then, for given point P, lying on the surface, the size of the edge, which should be used around that point to achieve the desired detail, is calculated as the size of the edge which would fit k_d–times int r_c. A maximum edge size is set to avoid the edge case in the planar point.

During the algorithm, for a given edge E with the length l_E, the point C is created as displayed on the Figure 2.2. The point C lies in the plane of the incident triangle of the edge E on the line perpendicular to E and passing through M_E–midpoint of the

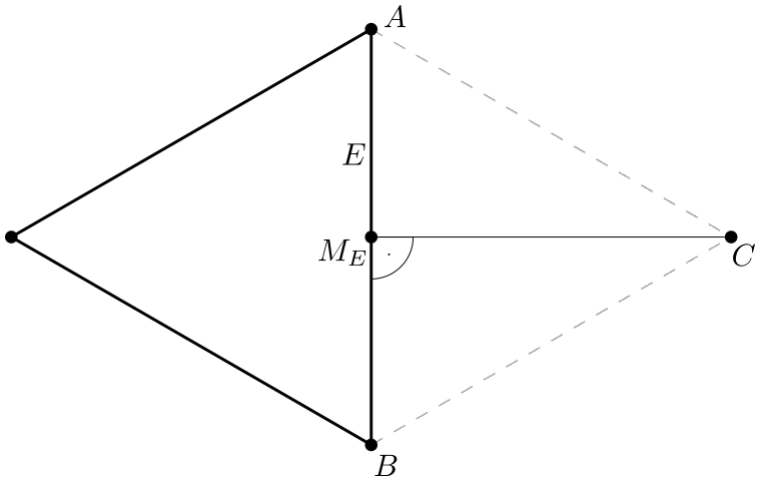


Figure 2.2: Creating the point C for the given edge E.

edge E. The distance between M_E and C is the height of the equilateral triangle with the edge E.

The curvedness is computed in each of the points A, B and C. The maximal value of the three computed values is used. To avoid fast changes of the triangle size, some constraints are introduced.

Inspired by the article by Akkouche and Galin [3], let us define l_{avg} as the average length of the set of edges consisting of the edges incident with the points A and B on the boundary of the mesh and the edge E. The rate at which the triangle height can scale compared to the height of the equilateral triangle with the edge length l_{avg} is set to 30%.

We calculate the edge length for the given constant of detail as $\frac{r_c}{k_d}$ which is equal to $\frac{e}{c}$, where e is the size of the edge on the input and c is curvedness in the point P_E. The new point is then created on the same line perpendicular to the edge E in the distance $\frac{\sqrt{3}e}{2c}$ from the point M_E.

The result of the adaptive triangulation is best visible on a torus, as the curvedness of the torus is maximal in the inner part of the torus and minimal in the outer part of the torus. The resulting mesh for four different constants of detail for a torus is displayed on the Figure 2.3 in the top row. One may note the difference in the sizes of the triangles in the inner part of the torus and the outer part of the torus.

In contrast, the uniform triangulation of the same torus for four different edge lengths is displayed in the bottom row in the Figure 2.3. One may note that the triangles in the inner part of the torus are approximately the same size as those in the outer part.

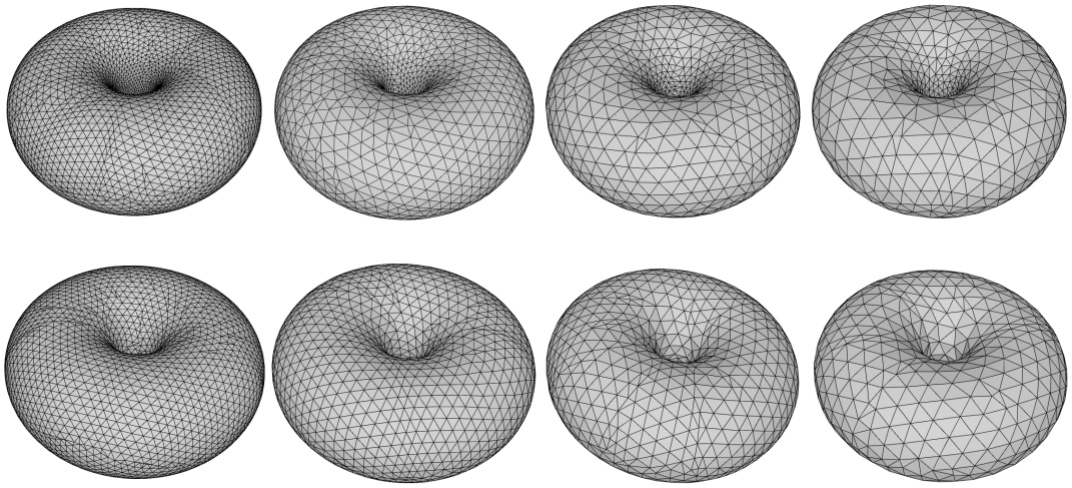


Figure 2.3: Adaptive triangulation of a torus with four different constants of detail (top row). Uniform triangulation of a torus with four different constants of detail (bottom row).

2.2 Triangulation of ADE singularities

2.2.1 Analysis of the geometry of ADE singularities

ADE singularities are simple, isolated surface singularities which can be expressed by corresponding implicit equations.

We already know that A_{1--} singularity is locally represented as a cone. This section discusses the geometric structure of other ADE surface singularities.

Definition 12 *Let us define the branch of ADE singularity as the part of the implicit surface connected to the rest of the implicit surface only by the singular point.*

For our needs, we pick one triangulation vector for each branch of each ADE singularity. This triangulation vector is a normalized vector either in the direction of the rotation symmetry axis or an intersection of reflection symmetry planes of the corresponding branch. If the branch has only one reflection symmetry plane, the triangulation vector is picked to lie in the reflection symmetry plane.

In the general case, triangulation vectors serve us as partial information about the orientation of a singularity with respect to its normal form.

A_n singularities

As we can see from the equations $F(x, y, z) = x^{n+1} \pm y^2 \pm z^2$, A_{n-+} singularities are just rotated A_{n+-} singularities and A_{n++} singularities are a single point if n is odd and

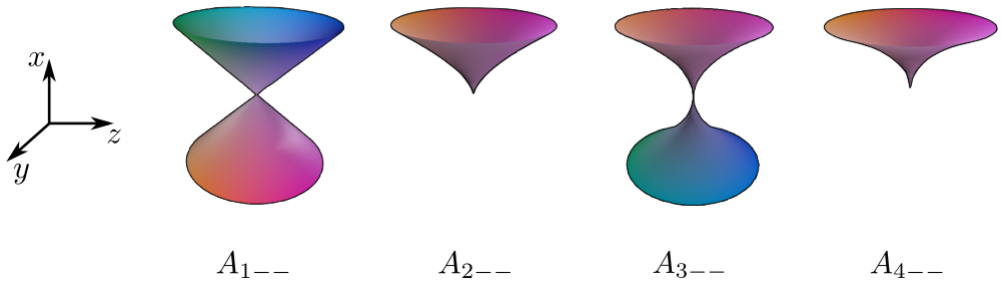


Figure 2.4: A_{n--} singularities. [15]

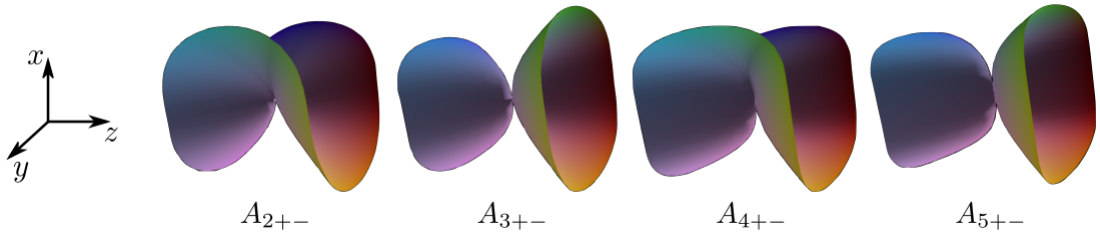


Figure 2.5: A_{n+-} singularities. [15]

reflected A_{n--} singularities if n is even. We therefore only discuss geometry of A_{n--} and A_{n+-} singularities.

A_{n--} singularities are topologically equivalent to a cone if n is odd, therefore they have two branches. If n is even, they are topologically equivalent to a half cone or a plane, therefore they have a single branch. As n gets bigger, the tip of the cone gets sharper. As A_{n--} singularities are rotationally symmetrical, we pick the direction of the axis of symmetry as a triangulation vector. For a normal form, the triangulation vectors are $(1, 0, 0)$ (and $(-1, 0, 0)$ if n is odd). The first four A_{n--} singularities can be seen in the Figure 2.4.

A_{n+-} singularities are topologically equivalent to a cone if n is odd, therefore they have two branches. On the contrary with the previous singularities, as n gets bigger, the tip of the cone gets less sharp and flatter. Branches of these singularities have reflection symmetry planes $x = 0$ and $y = 0$. Therefore we pick the vectors $(0, 0, 1)$ and $(0, 0, -1)$ as the triangulation vectors.

If n is even, A_{n+-} singularities are topologically equivalent to a plane with a shape similar to a hyperbolic paraboloid, therefore, they have a single branch. The first four A_{n+-} singularities can be seen in the Figure 2.5. For this case, we pick the vector $(1, 0, 0)$ as a triangulation vector as these singularities have reflection symmetry planes $y = 0$ and $z = 0$.

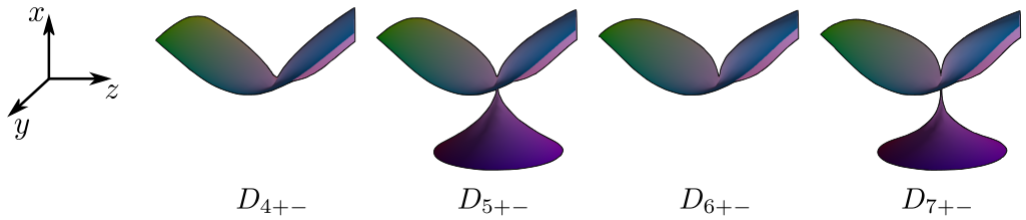


Figure 2.6: D_{n+-} singularities. [15]

D_n singularities

Given by equations $F(x, y, z) = yx^2 \pm y^{n-1} \pm z^2$, we consider 8 categories. For given sign combination and parity of n, the singularities are topologically equivalent, with sharper(or flatter) features around the singularities for increasing value of n similar to A_n singularities.

We can therefore say that D_n singularities can be classified into eight categories locally represented by the following equations:

- D_{4++} $yx^2 + y^3 + z^2$
- D_{5++} $yx^2 + y^4 + z^2$
- D_{4+-} $yx^2 + y^3 - z^2$
- D_{5+-} $yx^2 + y^4 - z^2$
- D_{4-+} $yx^2 - y^3 + z^2$
- D_{5-+} $yx^2 - y^4 + z^2$
- D_{4--} $yx^2 - y^3 - z^2$
- D_{5--} $yx^2 - y^4 - z^2$.

Now we look at some equivalences between these eight categories. D_{4++} singularity is reflected D_{4+-} singularity. D_{5++} singularity is reflected D_{5--} singularity. D_{5-+} singularity is reflected D_{5+-} singularity. D_{4-+} singularity is reflected D_{4--} singularity.

We therefore only analyze geometry of D_{n+-} singularities and D_{n--} singularities.

D_{n+-} singularities are topologically equivalent to a plane when n is even and to a cone when n is odd. Again, as n gets bigger, the features around singularities get sharper. Symmetry planes of these singularities are $x = 0$ and $z = 0$, therefore we pick $(0, 1, 0)$ (and $(0, -1, 0)$ when n is odd) as triangulation vectors. The first four D_{n+-} singularities can be seen in the Figure 2.6.

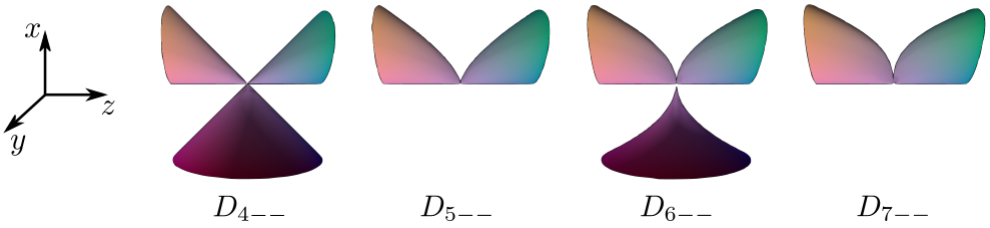


Figure 2.7: D_{n--} singularities. [15]

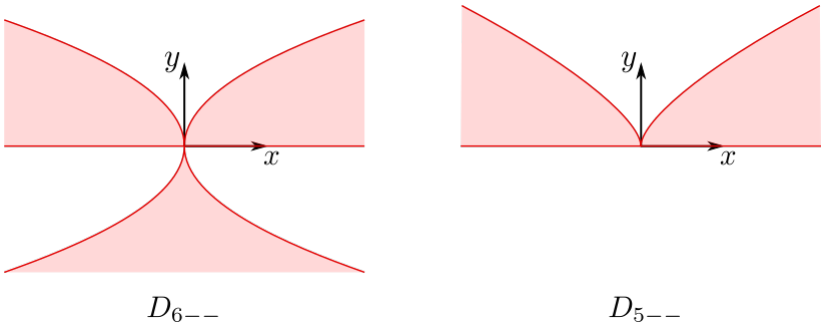


Figure 2.8: Intersection of D_{n--} singularities with plane $z = 0$.

D_{n--} singularities are topologically equivalent to a cone when n is odd and to 3 half-cones connected in the singular point when n is even. First four D_{n--} singularities can be seen in the Figure 2.7.

The symmetry plane for all branches of these singularities is $z = 0$. the intersection of the surface and plane $z = 0$ is displayed on the Figure 2.8.

For D_{n--} singularity, the intersections of the two branches where $y \geq 0$ are bounded by curves $y = 0$ and $x^2 = y^{n-2}$. For given r, we pick the triangulation vectors as $(r, \frac{1}{2}r^{\frac{2}{n-2}}, 0)$ and $(-r, \frac{1}{2}r^{\frac{2}{n-2}}, 0)$. The resulting vectors are displayed in the Figure 2.9 by the blue arrow. Parameter r is changed based on the length of the edge of the triangulation triangle. displayed on the Figure 2.9 by blue arrow. Parameter r is changed based on the length of the edge of the triangulation triangle.

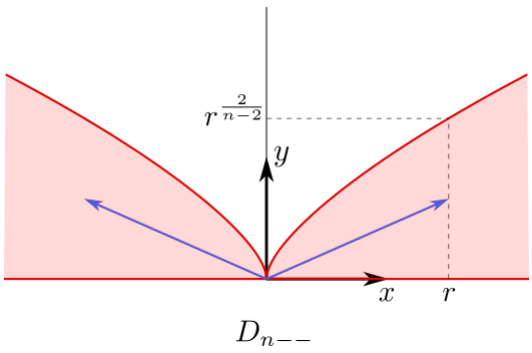


Figure 2.9: Triangulation vectors for two branches of D_{n--} singularities.

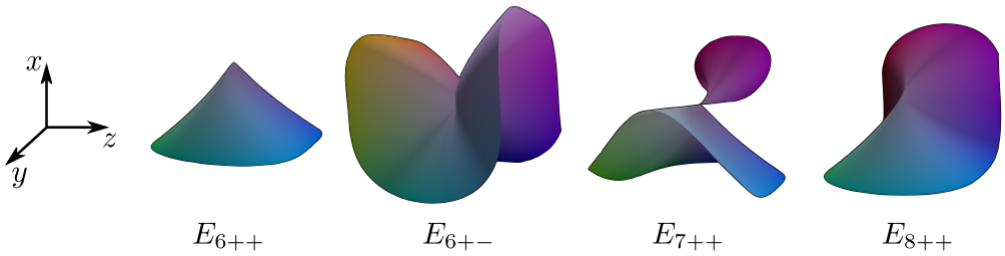


Figure 2.10: E_n singularities [15].

The third branch where $y \leq 0$ has has another plane of symmetry $x = 0$, therefore triangulation vector for this branch is chosen as $(0, -1, 0)$.

E_6, E_7 and E_8 singularities

Given by equations $F(x, y, z) = x^3 \pm y^4 \pm z^2$, $F(x, y, z) = x^3 \pm xy^3 \pm z^2$ and $F(x, y, z) = x^3 \pm y^5 \pm z^2$, we can see the following equivalences: E_{6++} singularity is reflected E_{6--} singularity. E_{6+-} singularity is reflected E_{6-+} singularity. E_{7+-}, E_{7-+} and E_{7--} are all reflected E_{7++} singularity. E_{8+-}, E_{8-+} and E_{8--} are all reflected E_{8++} singularity.

We only analyze geometry of E_{6++}, E_{6+-}, E_{7++} and E_{8++} singularities. These singularities are displayed in the Figure 2.10.

Both E_{6++} and E_{6+-} are topologically equivalent to a plane, thus they each have only one branch. The planes of symmetry of both of these branches are $y = 0$ and $z = 0$, therefore we pick $(-1, 0, 0)$ as the triangulation vector.

E_{7++} singularity is topologically equivalent to a cone, therefore, it has two branches. The plane of symmetry of this singularity is $z = 0$.

E_{8++} singularity is also topologically equivalent to a plane, therefore, it has only one branch. This branch has only one plane of symmetry $z = 0$.

We again look at the intersection of the surfaces with the plane of symmetry, this is displayed in the Figure 2.11.

For E_{7++} singularity, we pick $(-1, 0, 0)$ and $(\frac{1}{2}r^{\frac{3}{2}}, -r, 0)$ as triangulation vectors. For E_{8++} singularity, we pick $(-1, -1, 0)$ as a triangulation vector. These vectors are displayed as blue arrows in the Figure 2.11.

2.2.2 Analytical calculation of local triangulation of some ADE singularities

For given edge size e, we calculate the local triangulation of ADE singularities, such that edges on the border of the local triangulation have length e.

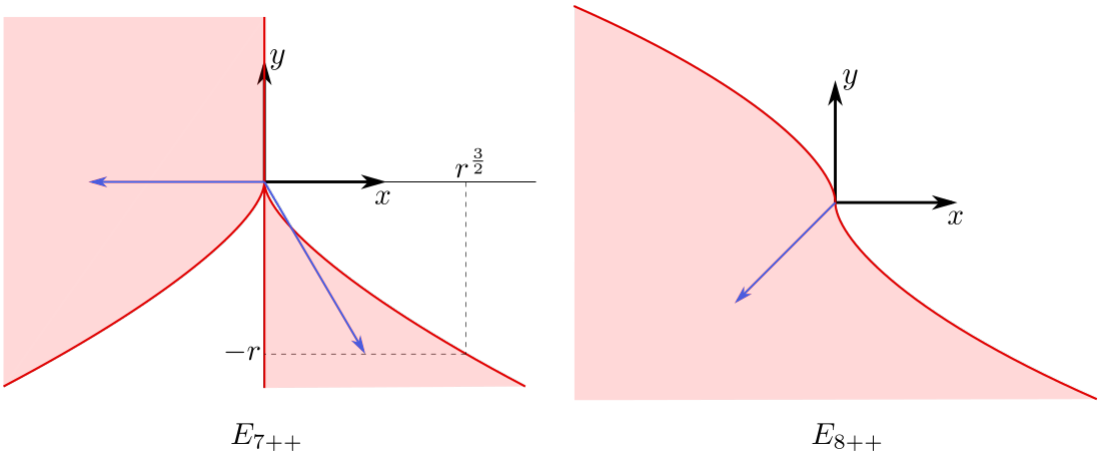


Figure 2.11: Intersection of E_{7++} and E_{8++} singularities with plane $z = 0$.

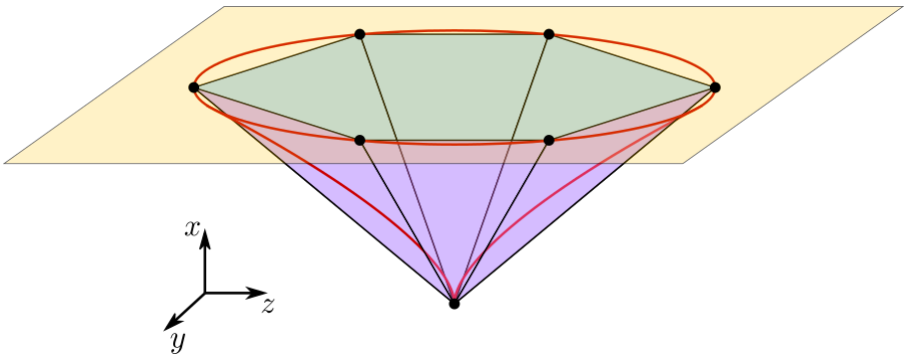


Figure 2.12: Triangulation of A_{n--} singularity.

A_{n--} singularities

For A_{n--} singularities, we create a disc of six isosceles triangles with a vertex in the singular point. The bases of these triangles create regular hexagon in the plane P parallel to the plane $x = 0$, as showed on the Figure 2.12. Given by equation $x^{n+1} - y^2 - z^2 = 0$, we find the distance of the plane P from the plane $x = 0$ for the given length e of the sides of the hexagon.

Let e be the length of the side of the hexagon, then the circumscribed circle has radius e. This circle is identical with the intersection of the surface and the plane $x = h$. The equation of the intersecting circle is $y^2 + z^2 = h^{n+1}$ therefore, the radius can be also expressed as $r = h^{\frac{n+1}{2}}$, which emerges $h = e^{\frac{2}{n+1}}$. Knowing the distance of the plane, one can easily calculate the length of the arms of the triangles using the Pythagorean theorem:

$$a^2 = h^2 + e^2 \implies a = \sqrt{e^{\frac{4}{n+1}} + e^2}$$

Layers for A_{n--} singularities: The resulting mesh with uniform triangles for A_{5--} singularity is displayed on the Figure 2.13.

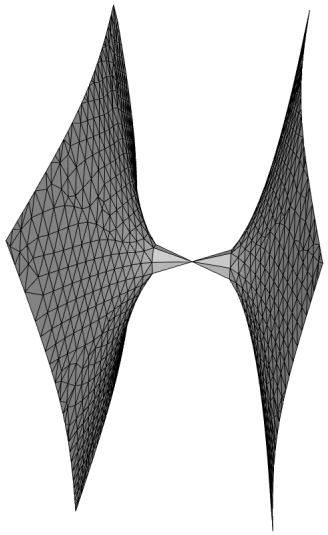


Figure 2.13: Resulting uniform triangulation of the A_{5--} singularity.

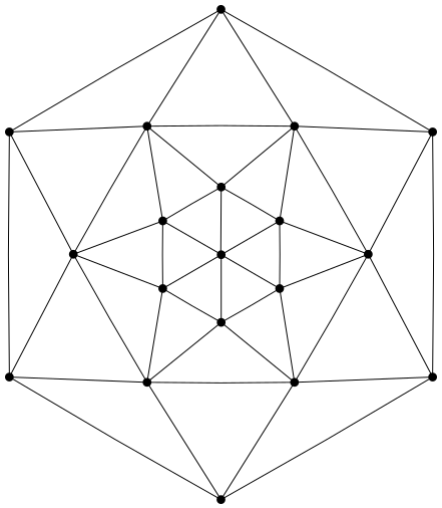


Figure 2.14: Three layers of triangles for the A_{1--} singularity.

As one may see, the approximation around the singular point is less accurate than for the rest of the surface. For this problem, we introduce layers – the surroundings of singularities are triangulated in multiple layers of triangles, and only after that, the algorithm for regular parts is used.

Given the edge length e and number of layers, we calculate the height h_e at which the bases of the triangles have the length e. We divide the height by the number of layers to obtain the layer height h_l. Now, i–th layer of points is obtained using the height $h = i \cdot h_l$. To connect these points into triangles, every second layer is rotated by $\frac{\pi}{6}$. The resulting 3 layers of triangles for the A_{1--} singularity as viewed from the point $(1, 0, 0)$ is displayed on the Figure 2.14.

The local mesh with one layer, four layers and eight layers (from left to right) for A_{5--} singularity is displayed on the Figure 2.15.

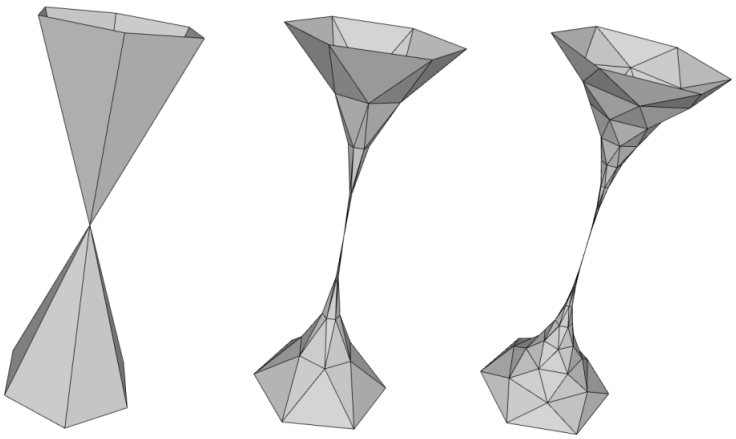


Figure 2.15: Local mesh with one layer, four layers and eight layers (from left to right) for the A_{5--} singularity.

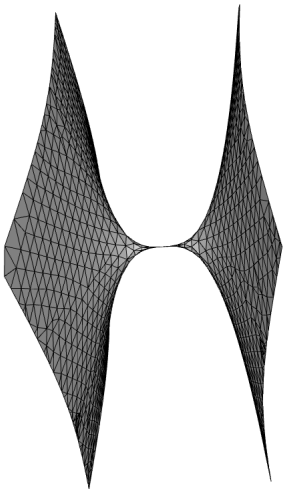


Figure 2.16: Resulting mesh for the A_{5--} singularity - four layers.

The resulting mesh with four layers is displayed on the Figure **??**.

D_n singularities

Some D_n singularities have branches with elliptical intersection with a plane parallel to the plane $y = 0$. As ellipses have two axes of symmetry, we create eight triangles with an apex in the singular point for these branches. The other points of the triangles lie on the ellipse, and they have the same base length.

Let us have an ellipse E with semi-major axis a, semi-minor axis b and the center in the point $(0, 0)$.

$$E : \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

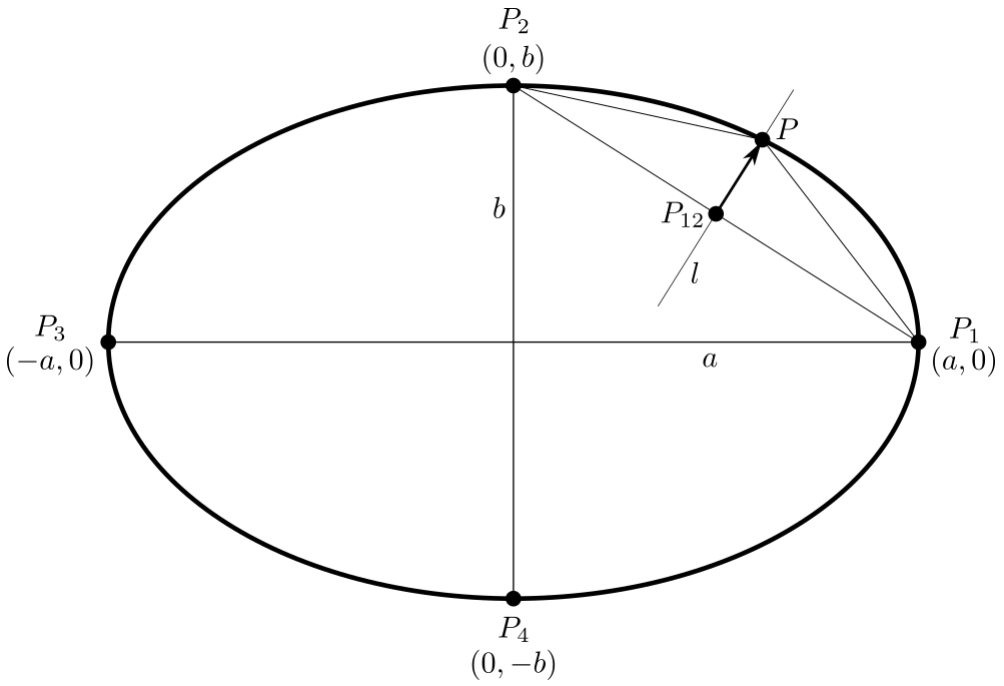


Figure 2.17: Equidistant points on the ellipse.

As displayed in the Figure 2.17, we pick the leftmost, the rightmost, the top and the bottom points. As shown on the figure 2.17, the coordinates of these points are $P_1 = (a, 0)$, $P_2 = (0, b)$, $P_3 = (-a, 0)$, $P_4 = (0, -b)$. Then we can calculate the point P on the ellipse equidistant from points P_1 and P_2. We calculate this point by taking the point P_{12} in the middle of a line segment P_1P_2.

$$P_{12} = \frac{1}{2}(a, b)$$

Then, the point P is lying on the intersection of the ellipse and a line l passing through the point P_{12}, perpendicular to the line segment P_1P_2.

$$l : \frac{1}{2}(a, b) + \frac{t}{2}(b, a), \quad t \in \mathbb{R}$$

Given the ellipse with semi-major axis a, semi-minor axis b and the center in the point $(0, 0)$, the point P can be calculated as follows

$$P \in l \cap E \implies \frac{(a + tb)^2}{4a^2} + \frac{(b + ta)^2}{4b^2} = 1$$

$$t = \frac{ab(\sqrt{3a^4 + 2a^2b^2 + 3b^4} - a^2 - b^2)}{a^4 + b^4},$$

therefore

$$P = \frac{1}{2}(a, b) + \frac{ab(\sqrt{3a^4 + 2a^2b^2 + 3b^4} - a^2 - b^2)}{2(a^4 + b^4)}(b, a).$$

Given edge length e, we are not able to calculate the height in which the distance between points $P1$ and P is e. The visualization showing this is in the Figure 2.18.

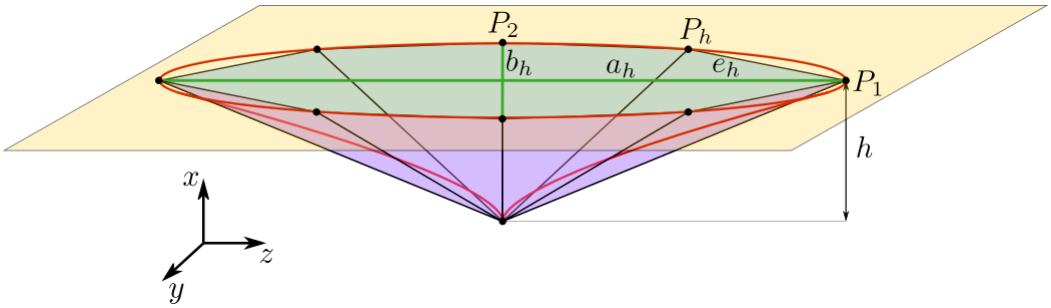


Figure 2.18: Calculating the point P_h – point on the ellipse equidistant from P_1 and P_2.

We use binary search to find such height. Given the height and the singularity class, we can calculate the semi-major axis and semi-minor axis as

$$D_{n+-} \quad : \quad -hx^2 + h^{n-1} - z^2 = 0 \quad h > 0$$

$$x^2 + \frac{z^2}{h} = h^{n-2}$$

$$\frac{x^2}{h^{n-2}} + \frac{z^2}{h^{n-1}} = 1 \implies a_h = max(h^{\frac{n-2}{2}}, h^{\frac{n-1}{2}}) \wedge b_h = min(h^{\frac{n-2}{2}}, h^{\frac{n-1}{2}}).$$

As we can see, we get the same ellipse for D_{n--} singularities:

$$D_{n--} \quad : \quad -hx^2 - h^{n-1} - z^2 = 0 \quad h > 0$$

$$2|n \wedge x^2 + \frac{z^2}{h} = -h^{n-2} \implies x^2 + \frac{z^2}{h} = h^{n-2}.$$

Then

$$P_h = \frac{1}{2}(h^{\frac{n-2}{2}}, h^{\frac{n-1}{2}}) + \frac{h^{\frac{2n-3}{2}}(\sqrt{3h^{2n-4} + 2h^{2n-3} + 3h^{2n-2}} - h^{n-2} - h^{n-1})}{2(h^{2n-4} + h^{2n-2})}(h^{\frac{n-1}{2}}, h^{\frac{n-2}{2}})$$

$$P_h = \frac{1}{2}(h^{\frac{n-2}{2}}, h^{\frac{n-1}{2}}) + \frac{h^{\frac{1}{2}}(\sqrt{3 + 2h + 3h^2} - 1 - h)}{2(1 + h^2)}(h^{\frac{n-1}{2}}, h^{\frac{n-2}{2}})$$

and we can calculate $e_h = ||P_h - P_1||$.

As $e \leq a_h$, we can start the binary search on the interval $\langle 0, a^{\frac{2}{n-2}} \rangle$ or $\langle 0, a^{\frac{2}{n-1}} \rangle$ and finish when the required precision is reached.

Proof that e_h is monotone in h: Let us consider a general ellipse with the semi-major axis a, semi-minor axis b and center in the point $(0, 0)$. Let us denote the points as follows: $A = (a, 0)$, $B = (0, b)$, $M = (\frac{a}{2}, \frac{b}{2})$ and C - point in the first quadrant of the ellipse equidistant from A and B, the points are displayed on the Figure 2.19.

As we calculated, it holds that

$$C = M + \frac{t}{2} \cdot (b, a) \quad \text{for} \quad t = \frac{ab(\sqrt{3a^4 + 2a^2b^2 + 3b^4} - a^2 - b^2)}{a^4 + b^4}.$$

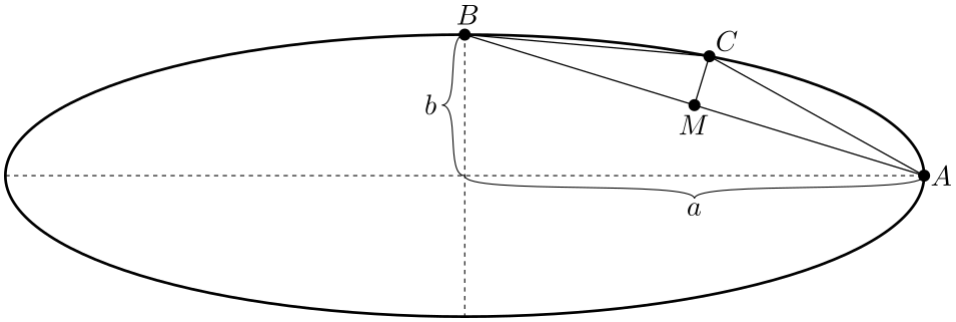


Figure 2.19: Notation used in the proof that e_h is monotone in h.

As length of the vector (b, a) is $\sqrt{a^2+b^2}$, the distance $|MC|$ is $\frac{t}{2} \cdot \sqrt{a^2+b^2}$.

After substituing for t, one gets

$$|MC| = \sqrt{a^2+b^2} \cdot \frac{ab(\sqrt{3a^4+2a^2b^2+3b^4} - a^2 - b^2)}{2(a^4+b^4)},$$

the derivative $\frac{\partial |MC|}{\partial a}$ is positive for all $b > 0$ and $\frac{\partial |MC|}{\partial b}$ is positive for all $a > 0$.

This means that $|MC|$ is increasing in both a and b. As the distance $|MA| = |MB|$ is also increasing in a and b, it follows from Pythagorean theorem, that $|AC| = |BC|$ is increasing in a and b.

When the height h increases, both the semi-major and semi-minor axis of the ellipse increase, thus e_h is increasing in h.

2.2.3 Numerical calculation of local triangulation of ADE singularities

The exact analytical calculations become more complicated for other types of ADE singularities. In this section, we present an approach for triangulation of all types of ADE singularities using iterative numerical algorithms based on the binary search.

We begin by dividing ADE singularities into two categories.

1. Category singularities – displayed on the Figure 2.20:
   - A_{n--},
   - A_{n+-},
   - D_{n+-}, where $n = 2k$,
   - E_{6++},
   - E_{6+-},
   - E_{8++}.
2. Category singularities – displayed on the Figure 2.21:

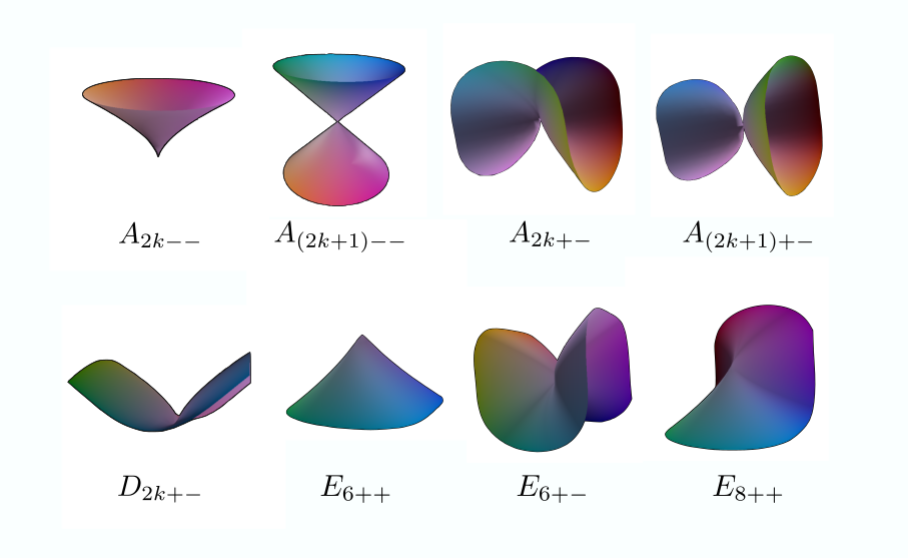


Figure 2.20: 1. Category singularities [15].

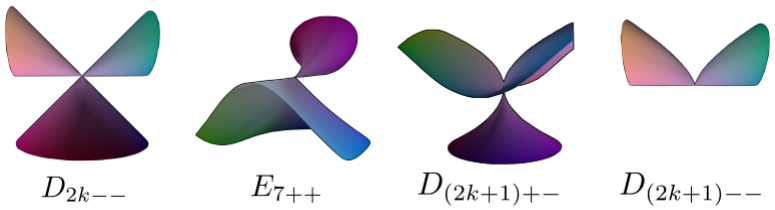


Figure 2.21: 2. Category singularities [15].

- D_{n+-}, where $n = 2k + 1$,
- D_{n--}„
- E_{7++}.

Triangulation of 1. category singularities

For all of the singularities, we have defined the triangulation vector, let us denote the normalized triangulation vector of the singularity as $\vec{t}$, resp. $\vec{t}_1, .., \vec{t}_i$ if the singularity has i triangulation vectors. Let us assume that the singular point is located in the point $O = (0, 0, 0)$. For the singularities from 1. category, topologically equivalent to a plane, given by the implicit equation $F(x, y, z) = 0$ it holds, that $F(O+\vec{t}) \cdot F(O-\vec{t}) < 0$. For the singularities from the 2. category, topologically equivalent to a cone, it holds, that $F(O+\vec{t}_j) \cdot F(O+\vec{t}_j^{\perp}) < 0$ for $j = 0, 1$. This means that in both cases, we can perform a binary search to find a point on the surface. Given the required length l – the length of the leg of the isosceles triangle created around the singularity, we perform a binary search to find an angle, for which the endpoint of the rotated vector $l \cdot \vec{t}$ lies on the

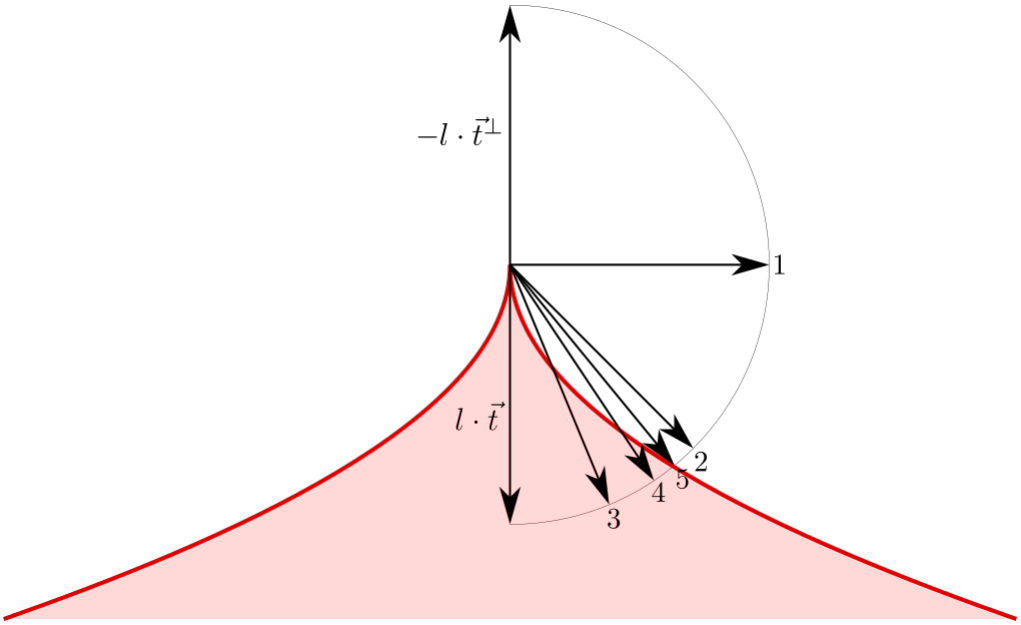


Figure 2.22: Binary search for the point on the surface.

surface with given precision as displayed in the Figure 2.22.

The binary search for singularities topologically equivalent to a cone is displayed on the Figure 2.23.

The binary search is performed in the half-planes passing through the point O and with the vector $\vec{t}$ lying in the half-planes. Using this approach, we create k points on the surface, while each time rotating the half-plane in which the binary search is performed by $\frac{2\pi}{k}$ about the axis given by $\vec{t}$ as displayed on the Figure 2.24.

For A_{n--} singularities, we create six triangles near the singular point. We only need to perform the binary search once, other points are obtained by rotating the point on the surface about the vector $\vec{t}$ by $\frac{2\pi}{6}$.

For A_{n--} singularities with two branches, the same approach is used, we perform the binary search only once and use rotation and reflection to create the remaining points on the surface.

In the case of other singularities, we create different numbers of triangles:

- A_{2k+-} – 8 triangles,
- $A_{(2k+1)+-}$ – 6 triangles for each branch,
- D_{2k+-} – 8 triangles for the branch given by the triangulation vector $(0, 1, 0)$ and 4 triangles for the branch given by the triangulation vector $(0, -1, 0)$,
- $D_{(2k+1)+-}$ – 8 triangles,
- D_{2k--} – 4 triangles for each branch,

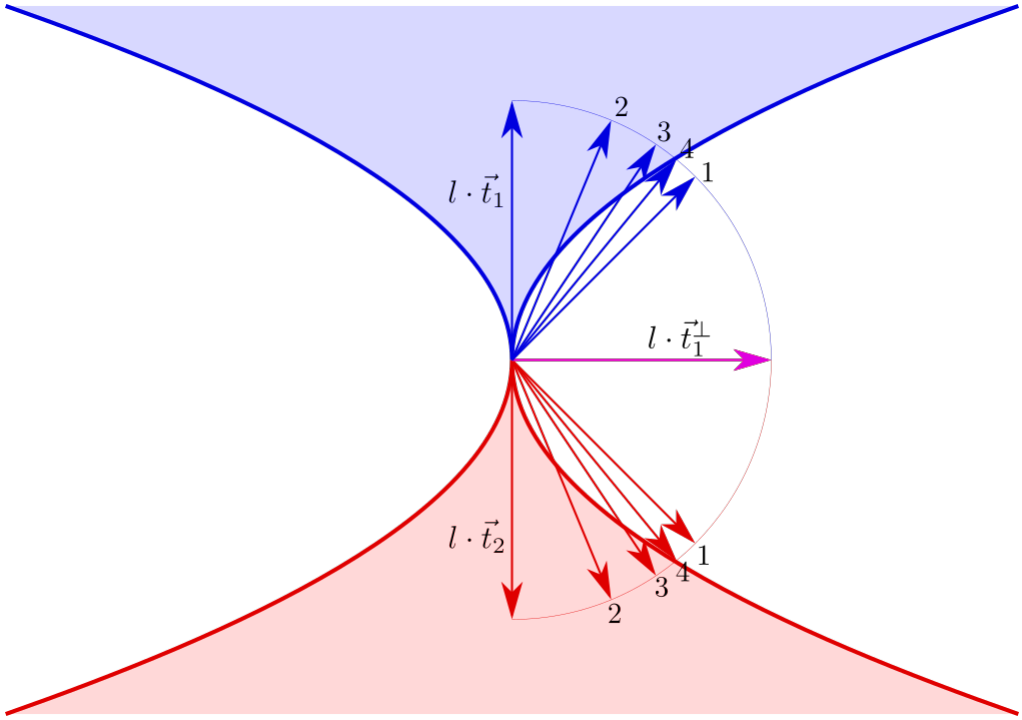


Figure 2.23: Double binary search for the point on the surface.

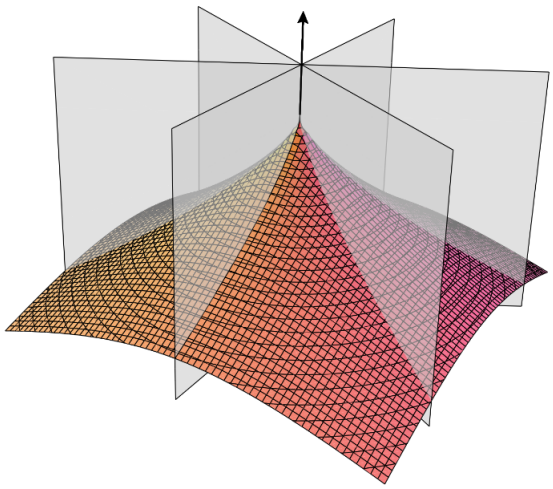


Figure 2.24: Rotating the plane about the vector $\vec{t}$ [1].

- E_{6++} – 6 triangles,
- E_{6+-} – 8 triangles,
- E_{7++} – 6 triangles for the branch, given by the triangulation vector $(-1, 0, 0)$ and 4 triangles for the other branch,
- E_{8++} – 6 triangles.

As some singularities have symmetries, we can use these symmetries to perform the binary search fewer times and find the remaining points as the mirror symmetries of the points found by binary search.

Triangulation of 2. category singularities

We create a more general, slower solution for the singularities in the 2. category. Again, we find the points on the surface lying in the rotated half-planes as displayed on the Figure 2.24. We do not have the general rule for the start and end angle to begin the binary search for these singularities. We have the triangulation vector, whose endpoint lies inside the space volume bounded by the respective branch. We consider this triangulation vector a start vector to start the binary search. We find the end vector by rotating the triangulation vector $\vec{t}$ around the vector $\vec{p}$ – vector perpendicular to $\vec{t}$, by small increments, until the endpoint of the rotated vector is outside of the space volume bounded by the branch, let us denote this vector $\vec{r}$. To find the point of the surface, we perform an angular binary search between $\vec{t}$ and $\vec{r}$. We proceed to find other points by rotating $\vec{p}$ around $\vec{t}$ by $\frac{2\pi}{k}$. By using the small increments, we ensure that the rotated vector does not end up in the space volume bounded by another branch, but it slows down the process of creating the local mesh. For the illustration, in the Figure 2.25 on the left, one can see the iterative process to find the vector $\vec{r}$ by rotating $\vec{t}$ around $\vec{p}$. On the right, one can see the rotation of $\vec{p}$ around $\vec{t}$ by $\frac{2\pi}{k}$ to find the remainding points on the surface.

For the singularities with rotation symmetry, we only need to perform the binary search once and create the remaining points using rotation symmetry. In the case of other singularities with different symmetries, we again make use of the potential planes of symmetry to find the points on the surface.

Layers for ADE singularities

We have already introduced the concept of creating multiple layers of triangles around A_{n--} singularities. In this section, we explain the extension of this concept to remaining ADE singularities. We create l layers of points for each branch for a given singularity. In contrast with A_{n--} singularities, we do not rotate every second layer by $\frac{\pi}{k}$. The

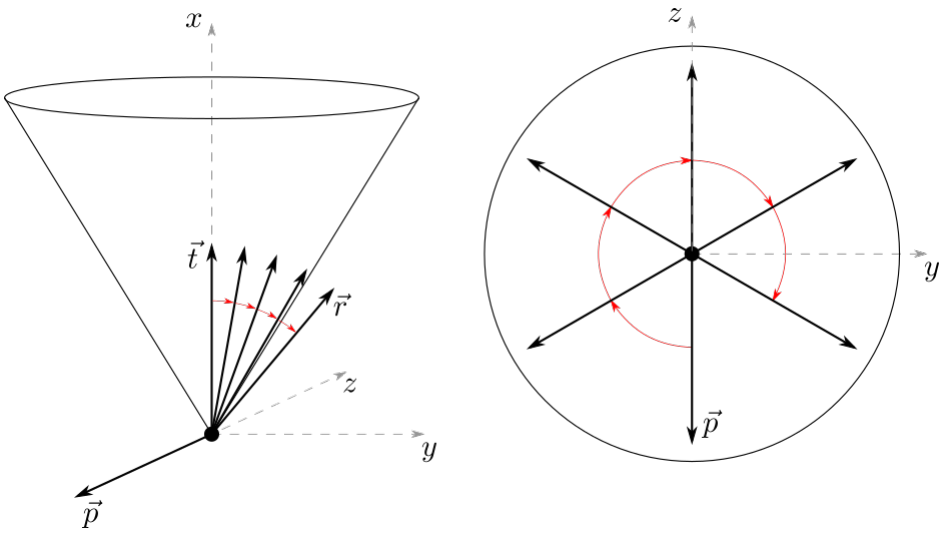


Figure 2.25: Iterative process of finding the vector $\vec{r}$ by rotating $\vec{t}$ around $\vec{p}$ (left). Rotating the vector $\vec{p}$ around $\vec{t}$ by $\frac{2\pi}{k}$ (right).

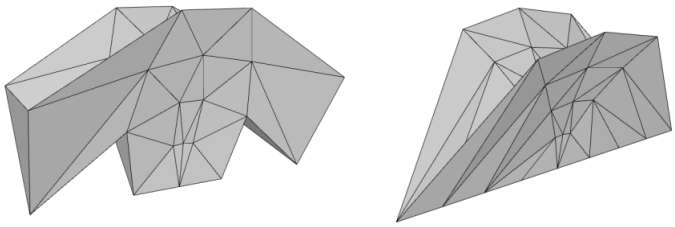


Figure 2.26: Insufficient (left) and sufficient (right) local approximation of D_{4+-} singularity.

reason is that the midpoints of the edges connecting a particular layer of points do not lie on the surface. This results in a poor approximation of the curves lying in the intersection of the surface and coordinate planes. Sometimes, it even results in incorrect local mesh with overlapping triangles.

In the Figure 2.26 on the left, one may see the insufficient approximation of D_{4+-} singularity with four layers of triangles and rotation by $\frac{\pi}{k}$. On the right, one may see the local mesh of the same singularity with four layers of triangles without rotation by the angle $\frac{\pi}{k}$.

On the Figure 2.27 on the left, one may see the incorrect local mesh of A_{2+-} singularity with two layers of triangles and rotation by $\frac{\pi}{k}$. On the right, one may see the local mesh of the same singularity with two layers of triangles without rotation by the angle $\frac{\pi}{k}$.

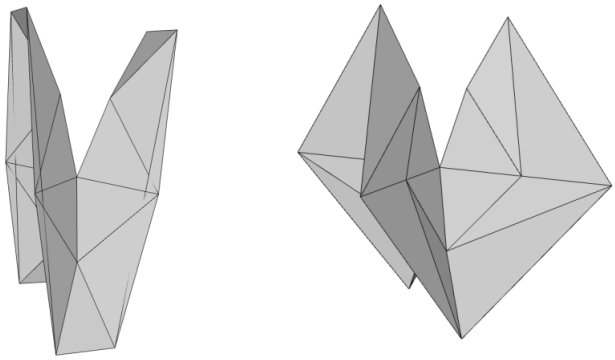


Figure 2.27: Incorrect (left) and correct (right) local mesh for A_{2+-} singularity.

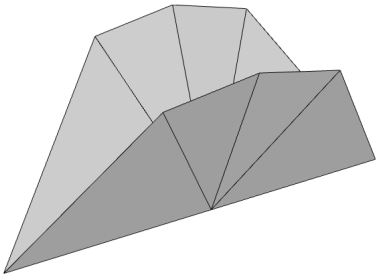


Figure 2.28: Uneven triangles in the local mesh around D_{4+-} singularity.

Optimization of the quality of the local mesh for some singularities

For some of the singularities, it is not convenient to look for the points on the surface in the planes rotated by identical angles as seen in the Figure 2.24. An example of such singularities are D_{n+-} singularities. The local mesh created with eight triangles using the described approach is displayed in the Figure 2.28. One may see that the outer edges of the triangles have a significant length difference.

The problem is significant in the singularities with eight triangles near the singular point. Our solution is to use binary search to find the angle where the triangles are close to isosceles triangles. The points found in rotated half-spaces given by the angles $0, \frac{\pi}{2}, \pi$ and $\frac{3\pi}{4}$ are unchanged. The points between these points are iteratively changed until the triangles are isosceles with the required precision. Starting on the interval of angles $(0, \frac{\pi}{2})$, in each step, the new angle is calculated as an angle in the middle between the two angles. The new point is then calculated using the binary search to find a point on the surface as displayed in the Figure 2.22.

As all singularities using this optimization are symmetrical by two coordinate planes, we only need to perform this binary search once for each layer of points and

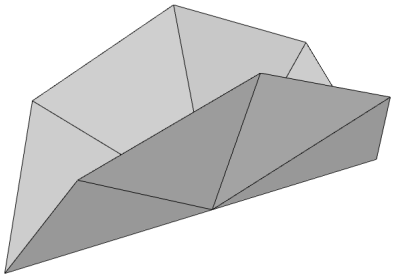


Figure 2.29: Optimized local mesh around D_{4+-} singularity.

use mirror symmetry to obtain the remaining points.

The result of the binary search of an angle for the D_{4+-} singularity is displayed on the Figure 2.29. The triangles around the singularity are closer to isosceles triangles in comparison to the Figure 2.28.

2.2.4 Triangulation of a plane with multiple A_{n--} singularities

In this section, we present an approach for creating an implicit equation of a surface, which consists of a plane and arbitrary many A_{n--} singularities C^1 smoothly connected to this plane.

Input and output

In this section, the following data are provided on the input:

1. the number of singularities - m,
2. m discrete points on a plane - $(x_1, y_1), ..., (x_m, y_m)$,
3. m degrees of the singularities - $n_1, ..., n_m$,
4. m heights at which each singularity is connected - $h_1, ..., h_m$.

The visualization of desired output function can be seen in the Figure 2.30. In this figure, the singularity is displayed in green colour. The red colour displays the function which connects the singularity to a plane - the bump function. There are some limitations on the input data. As we do not want the singularities or the bump functions to intersect, we require that each pair of input points is distanced d_{ij} from each other. We specify the d_{ij} value in the section 2.2.4.

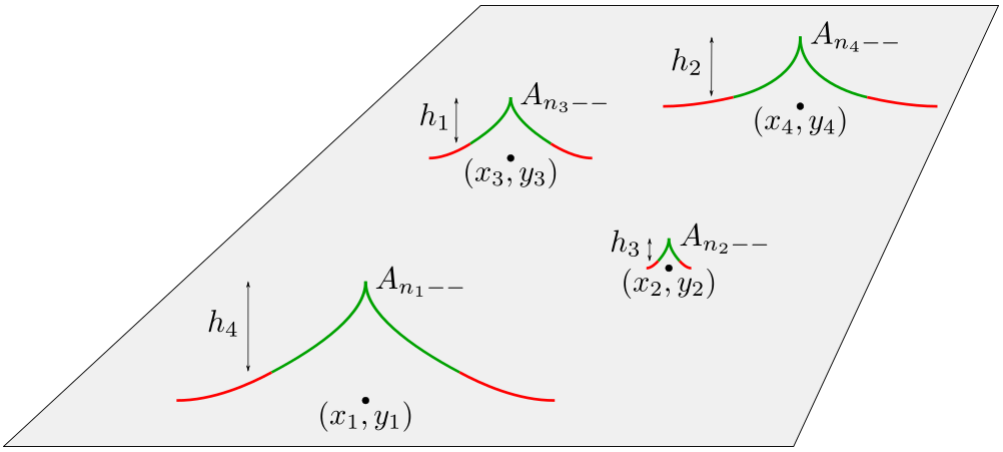


Figure 2.30: Plane with singularities.

Bump function

Definition 13 *The support $supp(f)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is a set of points where f is not zero*

$$supp(f) = \{x \in \mathbb{R}^n : f(x) \neq 0\}.$$

The closed support of the function f is defined as a closure of $supp(f)$.

Bump function if a function $f : \mathbb{R}^n \to \mathbb{R}$ which is smooth (C^∞) and compactly supported (the closed support of the function f is a compact subset of $\mathbb{R}^n$).

The most common example of such bump function is the function

$$f(x) = \begin{cases} e^{-\frac{1}{1-x^2}}, & x \in (-1, 1) \\ 0, & otherwise, \end{cases}$$

which is both C^∞ and compactly supported.

The bump function can smoothly connect a curve to a line or a surface to a plane in a higher dimension. If we only need to connect a curve and a line C^n smoothly, we only need a bump function which connects C^n smoothly to a line.

In our work, we connect two surfaces with C^1 continuity and for this purpose we use the function

$$f(x) = \begin{cases} -q \cdot cos(k \cdot x) - q, & x \in (-\frac{\pi}{k}, \frac{\pi}{k}) \\ 0, & otherwise, \end{cases}$$

rotated about z-axis. The result of the rotation is the following function:

$$f(x, y) = \begin{cases} -q \cdot cos(k \cdot (x^2 + y^2)) - q, & x^2 + y^2 <= (\frac{\pi}{k})^2 \\ 0, & otherwise. \end{cases}$$

These functions can be seen in the Figure 2.31.

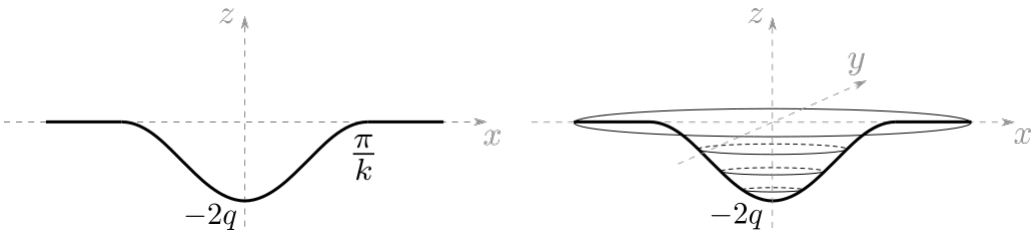


Figure 2.31: C^1 cosine bump function.

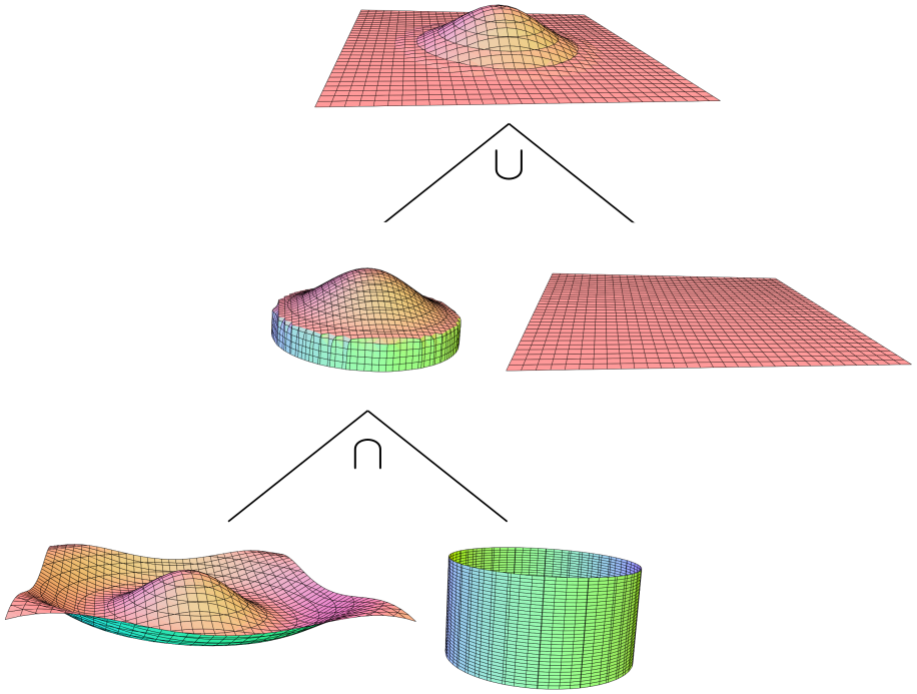


Figure 2.32: Construction of the cosine bump function using CSG [1].

Implicit equation of the cosine bump function

To construct the implicit equation of the cosine bump function, we use CSG - constructive solid geometry, described in the section 1.3.

First, we cut out the part of the rotated cosine, where $x^2+y^2 < (\frac{\pi}{k})^2$ using cylinder and intersection operation. Next, we use a plane and the union operation to *glue* the bump to the plane. The described process is displayed in the Figure 2.32 in the form of a CSG tree. We use the following equations of the surfaces to model the cosine bump function:

Parameters q and k allow us to change the amplitude and the frequency of the cosine function, parameters p_y and p_z are used to move the bump function to the given point (p_y, p_z).

Function name	Implicit equation
Rotated cosine function	$x + q \cdot cos(k \cdot \sqrt{(y-p_y)^2 + (z-p_z)^2}) + q = 0$
Cylinder	$(y-p_y)^2 + (z-p_z)^2 - (\frac{\pi}{k})^2 = 0$
Plane	x=0

Table 2.1: Implicit equations for bump function modelling.

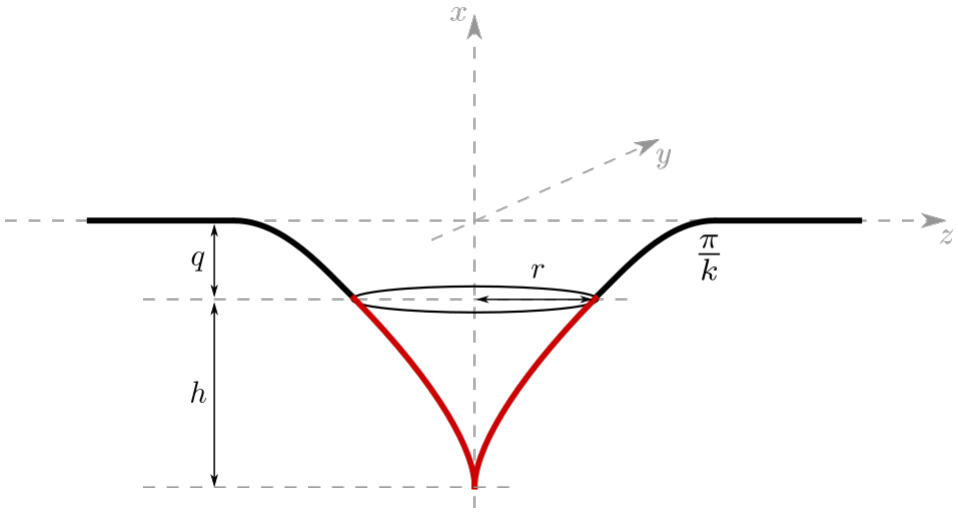


Figure 2.33: Attaching the singularity to a plane using the cosine bump function.

Attaching singularities to the plane using the cosine bump function

Given the type of the singularity – n and given height – h, we calculate the constants of the cosine bump function to connect C^1 smoothly to the given singularity.

The singularity given by the implicit equation $x^{n+1} - y^2 - z^2$ intersected with the plane $x = h$ produces a circle with the radius $r = \sqrt{h^{n+1}}$. The cosine bump function is scaled using q and k to smoothly connect the singularity in the middle of the cosine bump function. This approach is displayed in the Figure 2.33.

As the singularity is attached in the middle of the bump function, we get the equality $r = \frac{\pi}{2k}$ and therefore $\sqrt{h^{n+1}} = \frac{\pi}{2k} \implies k = \pi/(2\sqrt{h^{n+1}})$. The parameter q is calculated from the C^1 continuity requirement. We require the gradients to be linearly dependent on the points of connection. Due to the rotation symmetry, we check it only for the intersection with the plane $z = 0$ for the point $(-q, \frac{\pi}{2k})$.

$$F = (x + h + q)^{n+1} - y^2 \implies \nabla F = [(n+1)(x+h+q)^n, -2y]$$

$$\nabla F\left(-q, \frac{\pi}{2k}\right) = \left[(n+1)h^n, -\frac{\pi}{k}\right]$$

$$G = x + q \cdot cos(ky) + q \implies \nabla G = [1, -qk \cdot sin(ky)]$$

$$\nabla G\left(-q, \frac{\pi}{2k}\right) = [1, -qk]$$

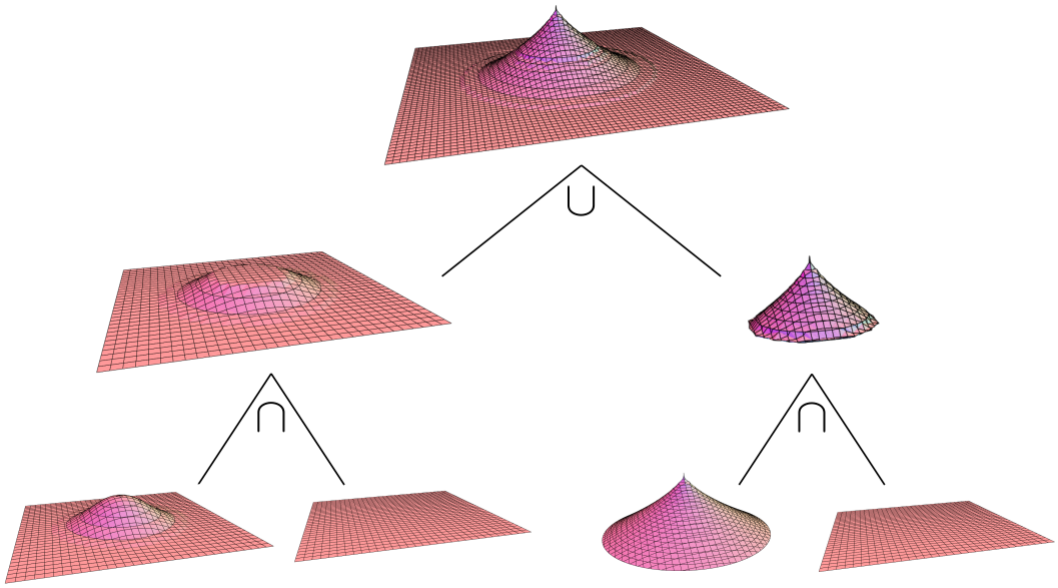


Figure 2.34: Attaching the singularity to a plane using CSG [1].

Requiring $\nabla F(-q, \frac{\pi}{2k}) = s \cdot \nabla G(-q, \frac{\pi}{2k})$ and knowing $k = \pi/(2\sqrt{h^{n+1}})$, we get $s = (n+1)h^n$ and therefore $q = 4h/(\pi(n+1))$.

After calculating the parameters q and k of the cosine bump function, we proceed to connect the singularity to the bump function. We use intersection with the plane $x = -q$ to get the sections of the singularity and the section of the bump function, and lastly, we use the union of these two surfaces. The described process is displayed in the Figure 2.34. The information about the detailed calculation of the implicit equation can be found in the appendix 5.

To connect multiple singularities to the same plane, we construct the implicit equation for each singularity and then use the union operation to create a surface with multiple singularities. This procedure is displayed in the Figure 2.35.

Limitations on the input data

As mentioned, we require that each pair of input points is distanced d_{ij} from each other. As the radius of the closed support of the cosine bump function is $r = \frac{\pi}{k} = 2\sqrt{h^{n+1}}$, the distance between two input points p_j, p_j must be at least $d_{ij} = 2\sqrt{h_i^{n_i+1}} + 2\sqrt{h_j^{n_j+1}}$. This way, both singularities and the corresponding bump functions do not intersect.

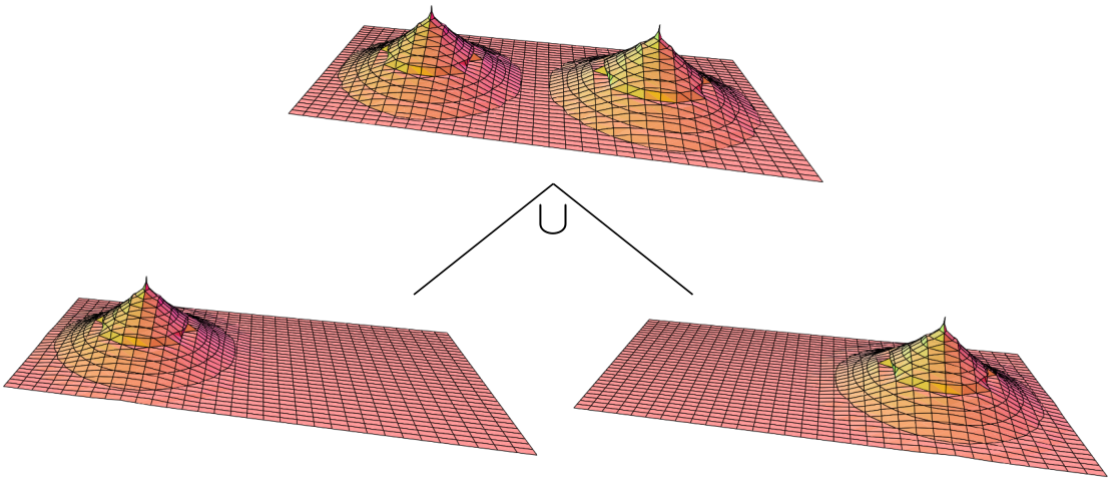


Figure 2.35: Plane with multiple attached singularities [1].

Visualization of results of the triangulated surfaces

2.3 Triangulation of non-isolated singularities

We present an approach for triangulation of implicit surfaces with singular curves which are a result of performing intersection operation on the interiors of two regular implicit surfaces.

We start by creating a local mesh for the surroundings of these singular curves and finish the mesh in the regular parts.

2.3.1 Creating the local mesh around the singular curves

We start by approximating the singular curve by a polyline. On the input, one point close to the singular curve is given. This point serves as a starting point P_0. The tangent vector $\vec{t}_C(P_0)$ of the curve is computed as the cross product of the unit normal vectors of the two surfaces S_1 and S_2 given by the implicit equations $F_1 = 0$ and $F_2 = 0$, respectively. Given the required approximate edge length e, a point Q_1 is computed as $Q_1 = P_0 + e \cdot \vec{t}_C(P_0)$. We obtain the point P_1 by projecting the point Q_1 to the curve C using an approach described in the section 1.4. Other points are then created iteratively by the same approach:

$$Q_{n+1} = P_n + e \cdot \vec{t}_C(P_n),$$

$$P_{n+1} = proj_C(Q_{n+1}),$$

while checking if the new point is inside the axis-aligned bounding box. We stop once the new point is outside of the axis-aligned bounding box or if the new point is close to the starting point P_0, which means the approximated the curve is a closed curve.

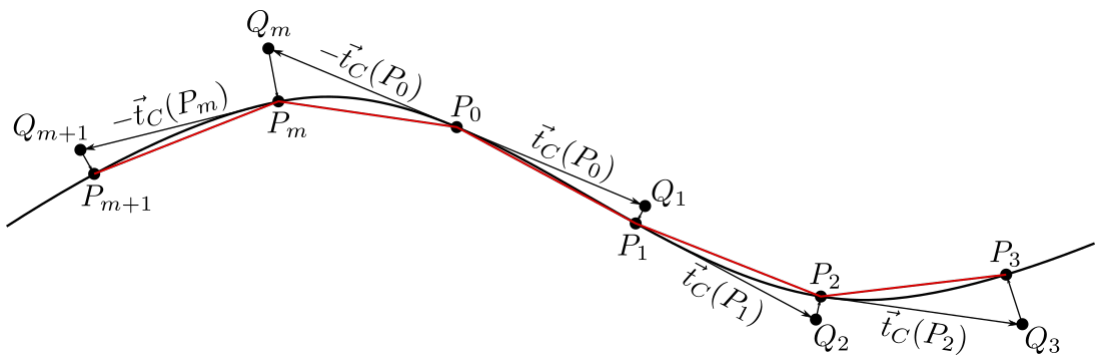


Figure 2.36: Approximation of the implicit curve by a polyline.

In the case of the open curve, to complete the whole polyline, we also approximate the second part of the singular curve by iteratively creating points in the opposite direction, starting from point P_0. Let m be the number of points in the polyline the second part of the polyline consists of the points $P_m, P_{m+1}, ...$, where

$$Q_m = P_0 - e \cdot \vec{t}_C(P_0),$$

$$P_m = proj_C(Q_m),$$

and again, iteratively

$$Q_{n+1} = P_n - e \cdot \vec{t}_C(P_n), \quad n = m, m+1, ...$$

$$P_{n+1} = proj_C(Q_{n+1}), \quad n = m, m+1, ...$$

The presented approach is visualized in two dimensions in the Figure 2.36.

After obtaining the polyline approximation of the curve on the intersection of the two surfaces, one may create the local mesh. Let us rename the polyline points to $P_0, P_1, ..., P_k$, such that $l_i = \overline{P_i P_{i+1}}$ is a line segment of the polyline for $i = 0, ..., k-1$ (and $l_k = \overline{P_k P_0}$ is a a line segment of the polyline for a closed curve).

For each line segment l_i, we create two adjacent triangles containing l_i.

We start by calculating the midpoint $M_i = \frac{P_i + P_{i+1}}{2}$. By projecting the point M_i to the surface S_1, we obtain the point M_i^1. By projecting it to the surface S_2, we obtain the point M_i^2. The defined points are visualized in the Figure 2.37.

As the point M_i^1 is lying on the surface, the tangent plane $T_{S_1}(M_i^1)$ of the surface S_1 in the point M_i^1 is well defined. The same holds for the point M_i^2 lying on the surface S_2.

We first define points R_i^1 and R_i^2 close to the surface. After projecting these points on the surface, we obtain the third point for each of the two adjacent triangles.

The point R_i^1 is obtained by moving from the point M_i in the direction perpendicular to both $\overrightarrow{P_i P_{i+1}}$ and $\nabla F_1(M_1)$ by the given edge length e.

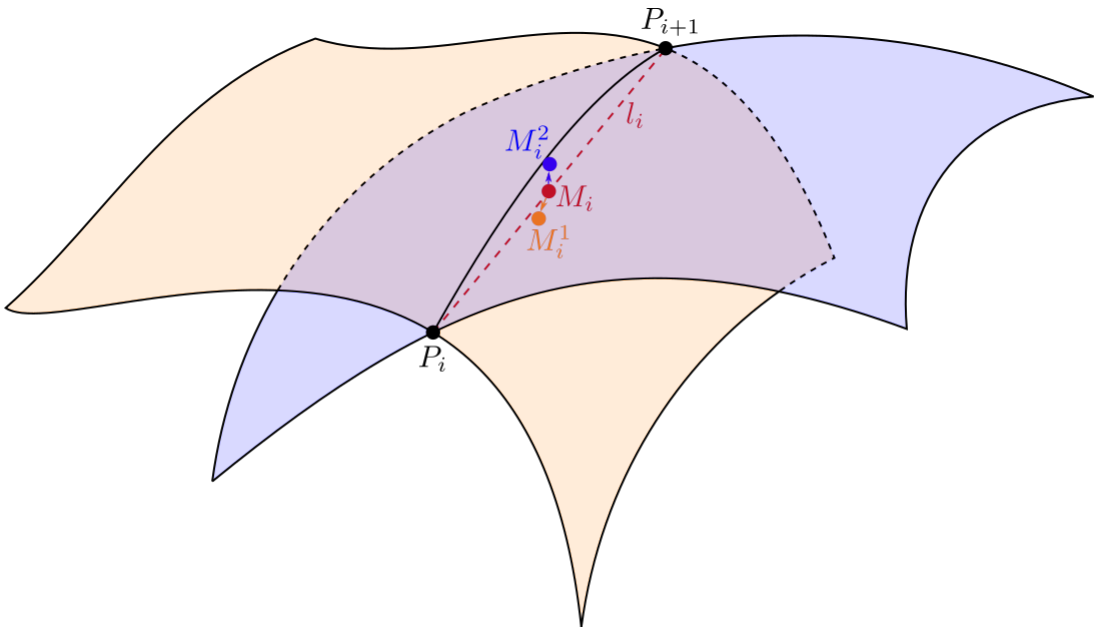


Figure 2.37: Definition of the points M_i, M_i^1 and M_i^2.

$$R_i^1 = M_i + e \cdot \frac{\nabla F_1(M_1) \times \overrightarrow{P_i P_{i+1}}}{||\nabla F_1(M_1) \times \overrightarrow{P_i P_{i+1}}||}.$$

The described situation is displayed in the Figure 2.38. For the point M_i, the plane H_i passing through M_i, perpendicular to the line segment l_i is defined:

$$H_i : \overrightarrow{P_i P_{i+1}} \cdot (X - M_i) = 0.$$

For better understanding, the Figure 2.38 is the projection of the surroundings of the point M_i to the plane H_i. Generally, points M_i^1 and M_i^2 do not lie in the plane H_i. The points R_i^1 and R_i^2 are picked in such way, that the point R_i^1 does not lie outside of the S_2 and the point R_i^2 does not lie outside of the surface S_1.

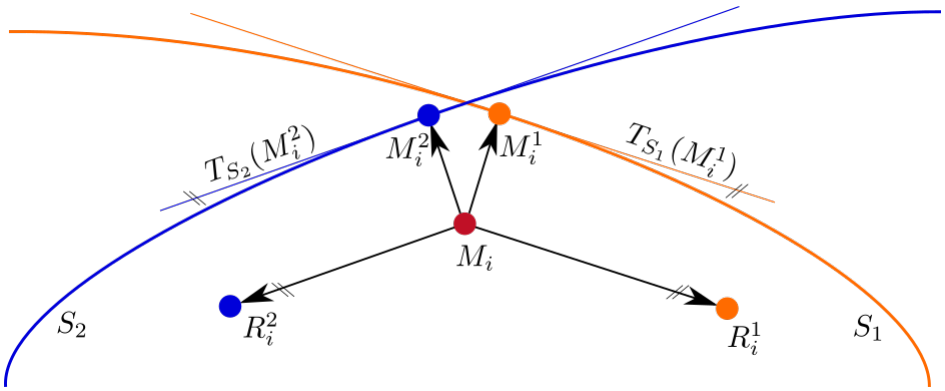


Figure 2.38: Definition of the points R_i^1 and R_i^2 in the plane H_i.

The points R_i^1 and R_i^2 are projected to the surface given by the intersection of the

interiors of the two surfaces. The equation defining the surface is

$$F_{1\cap 2} = F_1 + F_2 + \sqrt{F_1^2 + F_2^2}.$$

An example of the resulting local mesh for an open curve is displayed in the image 2.39. This curve is a part of the singular curve on the surface given by the intersection of the interiors of a sphere and a hyperboloid.

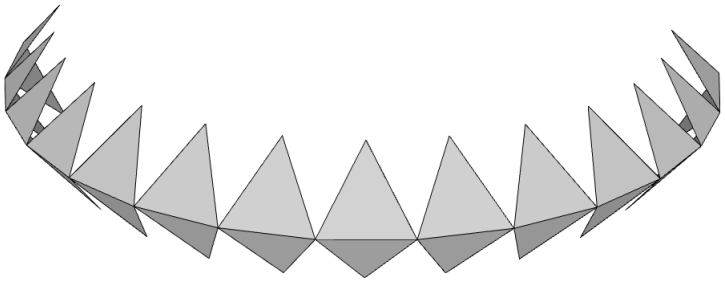


Figure 2.39: Local mesh around the singular curve.

2.3.2 Modification for triangulation of the union and the difference

For the intersection, the points R_i^1 and R_i^2 were picked in such way, that the point R_i^1 does not lie outside of the S_2 and the point R_i^2 does not lie outside of the surface S_1. The local mesh for the union is achieved by picking R_i^1 and R_i^2 in such way, that R_i^1 does not lie inside of S_2 and the point R_i^2 does not lie inside of the surface S_1. The points R_i^1 and R_i^2 for creating local mesh for the union of the interiors is displayed on the Figure 2.40.

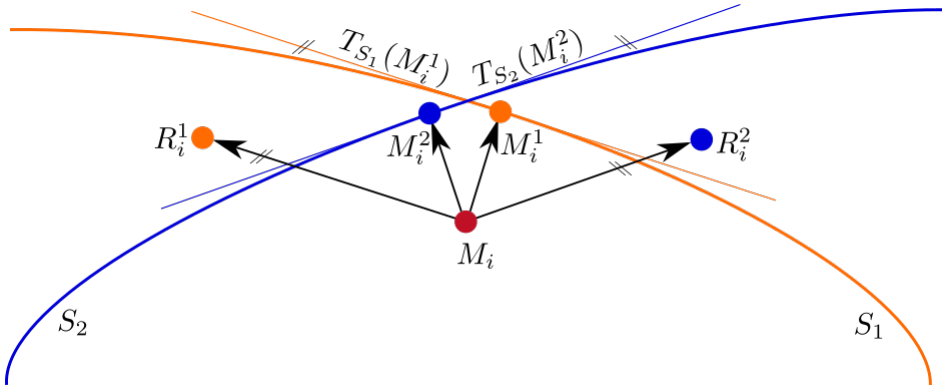


Figure 2.40: Definition of the points M_i, M_i^1 and M_i^2 for union.

For the difference, the points R_i^1 and R_i^2 are picked in such way, that R_i^1 does not lie inside of S_2 and the point R_i^2 does not lie outside of the surface S_1. The points R_i^1 and R_i^2 for creating local mesh for the difference of the interiors is displayed on the Figure 2.40.

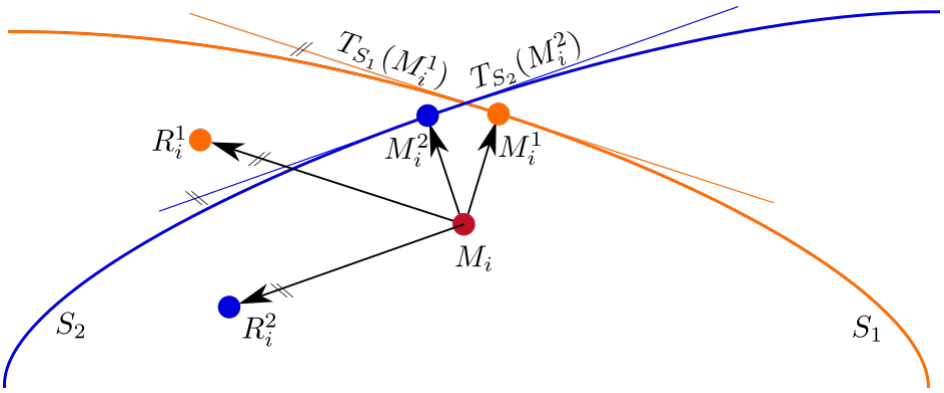


Figure 2.41: Definition of the points M_i, M_i^1 and M_i^2 for difference.

Chapter 3

Implementation

3.1 Triangulation of regular implicit srufaces

In this section, we shortly describe the algorithm for triangulation of the regular parts of implicit surfaces introduced in bachelor's thesis [13].

The algorithm creates triangles iteratively, one at a time. The new triangle is created at the edge of the existing triangle. The new point is projected on the surface using the well known iterative Newton-Rhapson method, which is the fast-converging method for root finding of a function $y = f(x)$. The condition for convergence of this method is to provide a point on the function sufficiently close to the root. We use this methodto find the root of the implicit function lying on the line, which is in the direction of the gradient of the implicit function. This approach is displayed on the Figure 3.1.

After the new point is projected on the surface, conditions are checked. Some of these conditions are based on the Delaunay triangulation introduced by Hilton [10]. The Delaunay condition checks, if some other points are in the proximity of circumcenter of the new triangle. The conditions minimize the chances of triangle intersection. If the new triangle intersects with some existing triangles, the algorithm tries to connect the new triangle to the existing points of the mesh in its proximity.

The algorithm is enriched with the possibility to triangulate adaptively to the curvature of the surface using approach presented by Akkouche [3]. It can also triangulate surfaces in the bounded volume - axis aligned bounding box, given by six numbers - minimal and maximal value for each of the three coordinate axes.

The algorithm was implemented using brute force. The queue of the edges on the border of the mesh – *active edges*, was being iteratively updated. In each step, an edge from the active edges queue was removed and the creation of the new triangle near this edge was attempted. If the new triangle was successfully created, the edge has become *inside edge*, in case of failure, the edge has become *checked edge*. The last

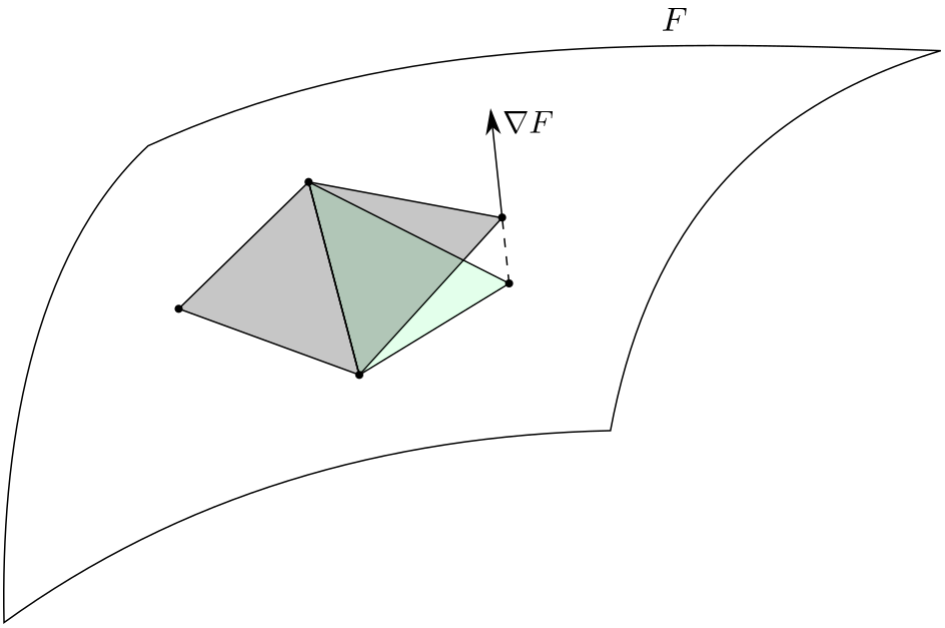


Figure 3.1: Projecting the point on the surface in the direction of the gradient.

characterization of an edge is for the edges with both end-points lying on the bounding box, these edges are called *bounding edges*.

As a part of our work, we reimplement the algorithm to be more effective by using advanced data structures, such as the half-edge data structure [11] and the range tree [14].

3.2 Data structures for triangulation algorithm

3.2.1 Half-edge data structure

Triangular mesh is given by a set of vertices, non-oriented edges and triangular faces. There are multiple methods for mesh representation. The straightforward one - list of vertices, edges and vertices does not provide any information about the local surroundings of the vertices, edges and faces and therefore the searching for incident faces or incident edges is complicated and inefficient.

In 1975, Baumgart [5] presented a representation using winged edges, which was further improved in 1985 by Weiler [20] who presented the modification called half-edge data structure. Both of these representations are edge-based representations, each edge stores references (pointers) to the surrounding vertices, edges and faces. One can easily extract the information of the surrounding vertices, edges and faces.

In the half-edge representation, edges are split into two halves of directed edges. Each half-edge stores reference to its initial vertex, left face, opposite half-edge and left traverse - predecessing and successing half-edge. A visualisation of the half-edge is

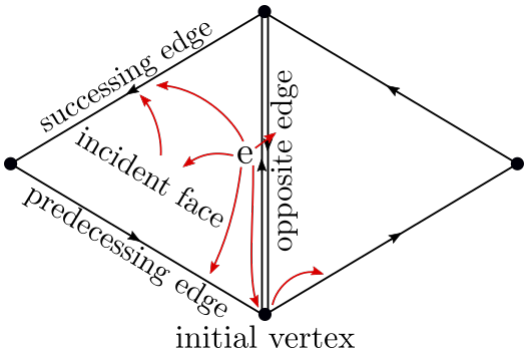


Figure 3.2: Visualisation of the half-edge data structure.

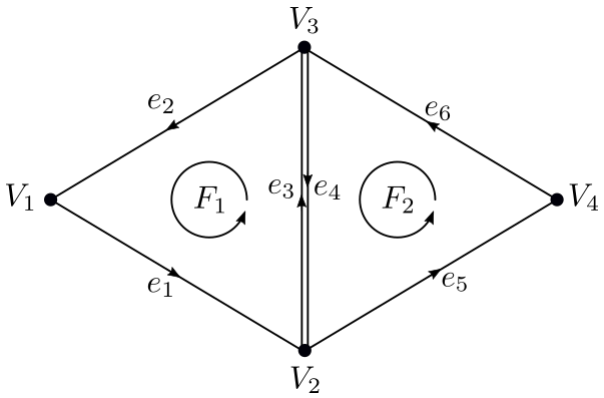


Figure 3.3: Example of half-edge representation.

displayed on the Figure 3.2. An example of half-edge representation is shown on the Figure 3.3 and tables 3.1, 3.2, 3.3.

The half-edge data structure is implemented in a way, which does not provide neighborhood information for non-manifold meshes. As our algorothm creates non-manifold meshes, we implement the half-edge data structure with the modification – each point has references to all outgoing edges instead of just one outgoing edge.

3.2.2 Range tree

The range tree is a tree data structure used for geometric search. The range tree holds a list of points and it answers the queries about the points in the given range. When used in three-dimensional space, the range corresponds to the three-dimensional interval. The advantage of the range tree compared to other tree structures for three-dimensional search, such as quad-tree or k-d tree is a fast query time. The implementation we used [2] answers the query for the set of points in a given three-dimensional interval in $\mathcal{O}(log^3 n + k)$, where n is the number of the points in the tree and k is the number of the points in the three-dimensional interval.

Edge	Vertex	Face	Edges		
	Initial	Incident	Predecessor	Successor	Opposite
e_1	V_1	F_1	e_2	e_3	
e_2	V_3	F_1	e_3	e_1	
e_3	V_2	F_1	e_1	e_2	e_4
e_4	V_3	F_2	e_6	e_5	e_3
e_5	V_2	F_2	e_4	e_6	
e_6	V_4	F_2	e_5	e_4	

Table 3.1: Edge table of a half-edge data structure for the Figure 3.3.

Vertex	Coordinates			Edge
	x	y	z	Outgoing edge
V_1	x_1	y_1	z_1	e_1
V_2	x_2	y_2	z_2	e_3
V_3	x_3	y_3	z_3	e_2
V_4	x_4	y_4	z_4	e_6

Table 3.2: Vertex table of a half-edge data structure for the Figure 3.3.

Face	Edge
F_1	e_1
F_2	e_4

Table 3.3: Face table of a half-edge data structure for the Figure 3.3.

3.2.3 Mesh structure

Triangular mesh consists of vertices, edges and faces. We use the half-edge data structure for mesh representation and the range tree for time efficient search of points in three dimensional interval. The mesh structure used for maintaining and modifying the triangular mesh consists of:

- list of vertices,
- list of half-edges,
- list of faces,
- range tree of all vertices.

Vertex consists of

- three coordinates,
- index of itself in the list of vertices,
- list of indices to all outgoing edges.

Half-edge consists of

- six coordinates,
- index of itself in the list of half-edges,
- index of initial vertex in the list of vertices,
- index of terminal vertex in the list of vertices,
- index of opposite edge in the list of half-edges,
- index of predecessing edge in the list of half-edges,
- index of successing edge in the list of half-edges,
- index of incident face in the list of faces,
- boolean values stating if the halfedge is active, checked, bounding or inside.

Face consists of

- nine coordinates,
- index of incident half-edge in the list of half-edges.

3.2.4 Algorithm runtime

During the algorithm, the new points, edges and triangles are added in at the end of the respective list in the mesh. The algorithm is implemented as a single pass through the list of mesh edges. If the edge is classified as active, the algoorithm proceeds to attempt to create the triangle near the edge. The triangle is being created the same way as in the brute-force algorithm. By this approach, we do not need to maintain the queue of active edges. As the new edges are being added at the end of the list, it behaves as a queue.

When checking for the conditions for the newly created triangle, the proximity within the mesh provided by the half-edge data structure is not enough. Here, the range tree is used. To find the triangles in the proximity of a point P, all meshpoints in a three-dimensional interval around this points are found using range tree. The triangles are then extracted by finding the incident faces of outgoing edges of the points found by range tree.

Time complexity of the algorithm: The number of edges, points and triangles in the mesh is asymptotically equal, let us denote the number n. The query time for the three-dimensional range tree is $\mathcal{O}(log^3 n + k)$, where k is the number of points reported by a given query. In each step of the algorithm, a constant number of operations and constant number of queries to the range tree is performed. Therefore, overall time complexity of producing the mesh with $\mathcal{O}(n)$ faces is $\mathcal{O}(n(log^3 n + k))$

Chapter 4

Results

In this chapter, we present the results achieved in triangulation of ADE singularities, surfaces with ADE singularities and surfaces with sungular curves arised from CSG operations on the regular surfaces. We measure some of the quality criteria proposed in [13] and compare our meshes in terms of quality with the meshes produced by a software for visualization of the implicit surfaces with ADE singularities implemented based on the article by Richard Morris [15]. We compare the computational speed of the algorithm after the reimplementation with the algorithm for the regular parts of the implicit surfaces implemented in [13].

4.1 Quality criteria

The following quality criteria [13] are evaluated for the resulting mesh with uniform edge size:

1. *The mean ratio of the length of the sides of the triangle*

   The mean ratio of the length of the longest side of the triangle and the shortest side of the triangle shows the uniformity of the triangles. For equilateral triangle, the ratio equals exactly one. The ratio is greater than one for isosceles and scalene triangles. The further the triangle is from equilateral triangle, the greater is the ratio. The number close to one indicates triangles close to the equilateral triangles, whereas the number greater than one indicates more non-uniform triangles in the mesh.

2. *Discrete approximation of the Hausdorff distance*

   To define the discrete approximation of the Hausdorff distance of the mesh and the implicit surface, one needs to introduce the notion of the distance of two sets of points.

Definition 14 *The distance δ of the point $a \in \mathbb{R}^3$ and the set $M \subset \mathbb{R}^3$ is defined as*

$$\delta(a, M) = \inf_{b \in M} d(a, b), \tag{4.1}$$

where $d(a, b)$ is the Euclidean distance of two points in $\mathbb{R}^3$.

Definition 15 *The Hausdorff distance of two sets $N \subset \mathbb{R}^3$ and $M \subset \mathbb{R}^3$ is defined as*

$$h(M, N) = \max\left\{ \sup_{a \in M} \delta(a, N), \sup_{b \in N} \delta(M, b) \right\}. \tag{4.2}$$

The Hausdorff distance is numerically aproximated by taking the maximum distance of the gravity center of a triangle of the mesh and perpendicular projection of the gravity center to the triangulated surface. This approximation is based on the assumption, that the point on the surface closest to the gravity center of the triangle is the perpendicular projection of that point to the surface.

The Hausdorff distance measures the accuracy of the triangulation by picking the triangle, which approximates the surface the worst.

3. *The mean distance of the gravity center and its perpendicular projection*

   Measuring the mean distance of the gravity center and its perpendicular projection shows the accuracy of the approximation globally, by taking into account all triangles of the mesh rather than picking out the worst approximating triangle.

4. *The mean distance of the neighbour vertices from the vertex and the standard deviation of the distance from the mean*

   The standard deviation of the values from the mean value says about the distibution of the values around the mean value. Small standard deviation indicates, that the values are close to the mean. On the contrary, bigger standard deviation indicates, that the values are further distributed from the mean value.

Definition 16 *Given N values – $x_1, ..., x_N$, let us denote the arithmetical mean of the values as $\overline{x}$. The standard deviation σ can be calculated as*

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \overline{x})^2}. \tag{4.3}$$

For the whole mesh, we are calculating the mean standard deviation from the standard deviation of all points and its neighbour points in the mesh.

4.2 Comparison with SingSurf

SingSurf [15] is a software used for visualization of two dimensional and three dimensional models. It can work with parametric and implicit surfaces, while allowing to model also some of the surface singularities, including ADE singularities. The software was implemented based on an article by Richard Morris [15].

4.2.1 SingSurf algorithm

4.2.2 Evaluated quality criteria of the SingSurf meshes

We compared in quality on fifteen different models. The models were chosen to capture all categories of ADE singularities. Each pair of models was created to have the same axis aligned bounding box and approximately the same number of faces. In the measurement, the invalid faces (the points are lying on a line) produced by SingSurf are not taken into account.

A_{n--} singularities

We created meshes of $A_{1--}, A_{2--}, A_{3--}$ and A_{4--} singularities. The resulting uniform meshes from our algorithm can be seen on the Figure 4.1. The resulting adaptive meshes

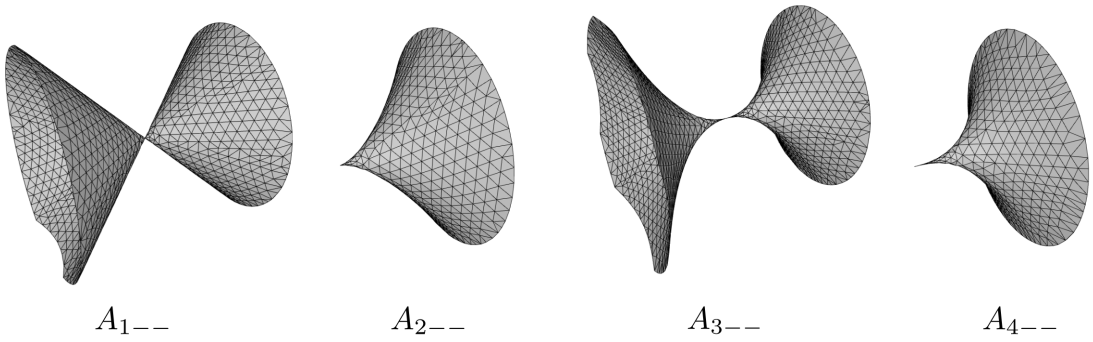


Figure 4.1: Resulting uniform triangulation of A_{n--} singularities with layers.

from our algorithm can be seen on the Figure 4.2. The resulting meshes generated

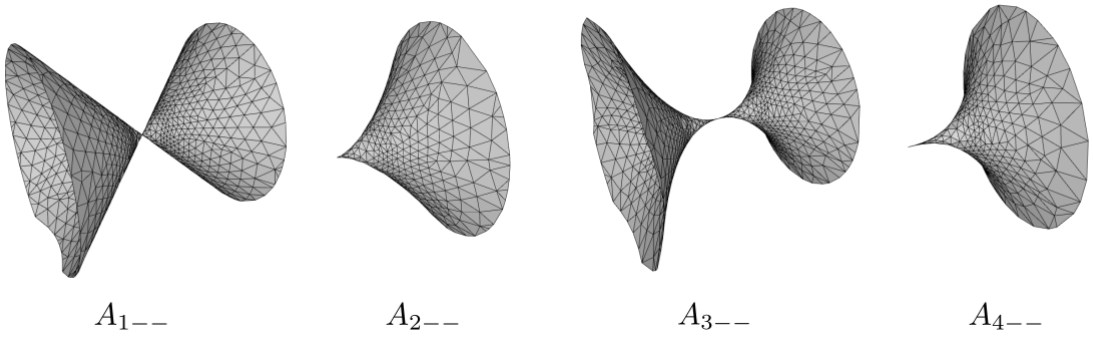


Figure 4.2: Resulting adaptive triangulation of A_{n--} singularities with layers.

by SingSurf can be seen on the Figure 4.3. The comparison on the quality criteria

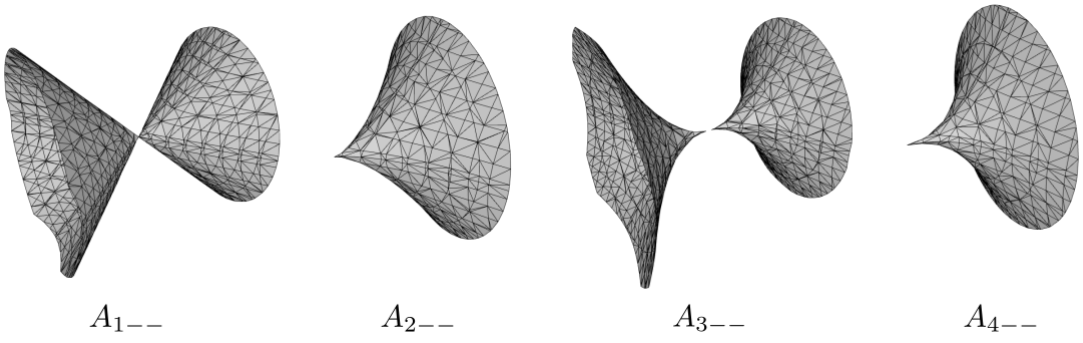


Figure 4.3: Resulting triangulation of A_{n--} singularities by SingSurf [15].

measured on these meshes is in the table 4.1. The better values are visualized by the green color and the worse values are visualized by the red color.

Table 4.1: Comparison of the quality criteria for A_{n--} singularities.

A_{n--} singularities					
type		k_1	k_2	k_3	k_4
A_{1--}	SingSurf	0.113	0.015	0.001	0.052
	Uniform algorithm	0.834	0.007	0.001	0.010
	Adaptive algorithm	0.760	0.011	0.002	0.017
A_{2--}	SingSurf	0.077	0.008	0.001	0.051
	Uniform algorithm	0.793	0.008	0.001	0.010
	Adaptive algorithm	0.729	0.008	0.001	0.017
A_{3--}	SingSurf	0.049	0.012	0.001	0.048
	Uniform algorithm	0.651	0.008	0.001	0.011
	Adaptive algorithm	0.599	0.008	0.001	0.018
A_{4--}	SingSurf	0.325	0.020	0.001	0.045
	Uniform algorithm	0.309	0.008	0.001	0.014
	Adaptive algorithm	0.365	0.008	0.001	0.020

A_{n+-} singularities

We created meshes of A_{2+-}, A_{3+-} and A_{4+-} singularities. The resulting uniform meshes from our algorithm can be seen on the Figure 4.4. The resulting adaptive meshes

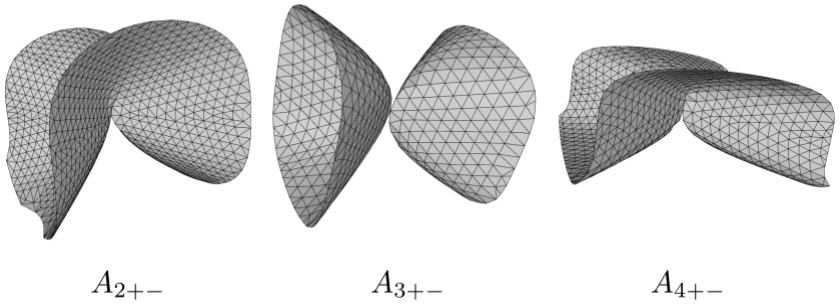


Figure 4.4: Resulting uniform triangulation of A_{n+-} singularities with layers.

from our algorithm can be seen on the Figure 4.5. The resulting meshes generated

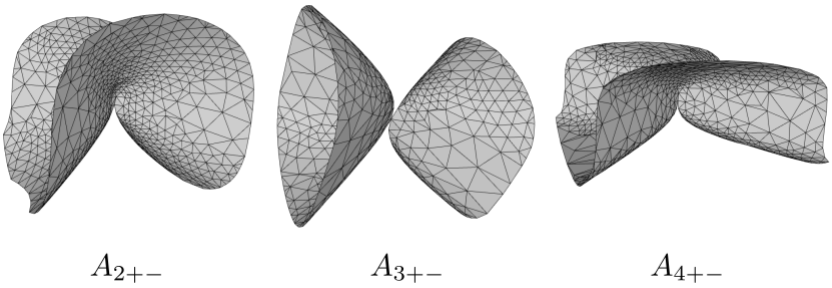


Figure 4.5: Resulting adaptive triangulation of A_{n+-} singularities with layers.

by SingSurf can be seen on the Figure 4.3. The comparison on the quality criteria

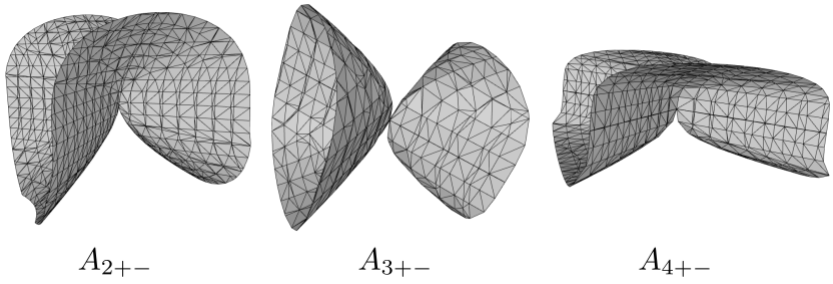


Figure 4.6: Resulting triangulation of A_{n+-} singularities by SingSurf [15].

measured on these meshes is in the table 4.2. The better values are visualized by the green color and the worse values are visualized by the red color.

Table 4.2: Comparison of the quality criteria for A_{n+-} singularities.

A_{n+-} singularities					
type		k_1	k_2	k_3	k_4
A_{2+-}	SingSurf	0.189	0.017	0.002	0.052
	Uniform algorithm	0.839	0.017	0.001	0.010
	Adaptive algorithm	0.739	0.008	0.001	0.019
A_{3+-}	SingSurf	0.001	0.042	0.005	0.073
	Uniform algorithm	0.868	0.019	0.003	0.014
	Adaptive algorithm	0.732	0.017	0.003	0.029
A_{4+-}	SingSurf	0.001	0.061	0.006	0.118
	Uniform algorithm	0.839	0.053	0.006	0.030
	Adaptive algorithm	0.711	0.032	0.006	0.063

D_n singularities

We created meshes of $D_{4+-}, D_{4--}, D_{5+-}$ and D_{5--} singularities. The resulting uniform meshes from our algorithm can be seen on the Figure 4.7. The resulting adaptive meshes

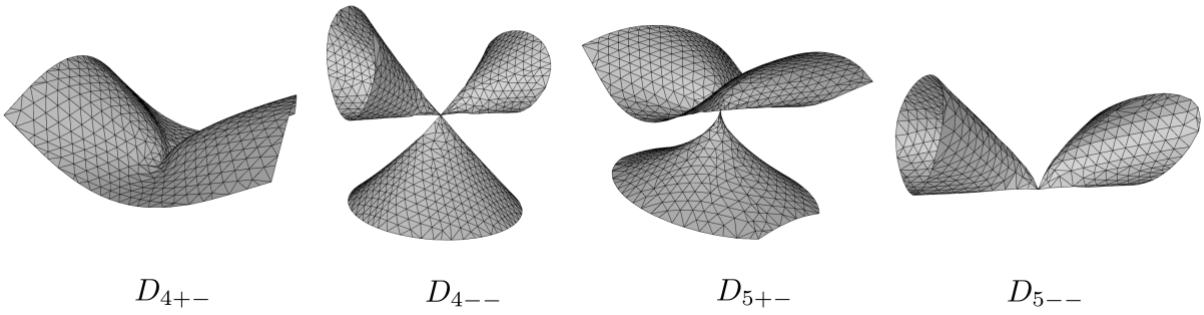


Figure 4.7: Resulting uniform triangulation of D_n singularities with layers.

from our algorithm can be seen on the Figure 4.8. The resulting meshes generated

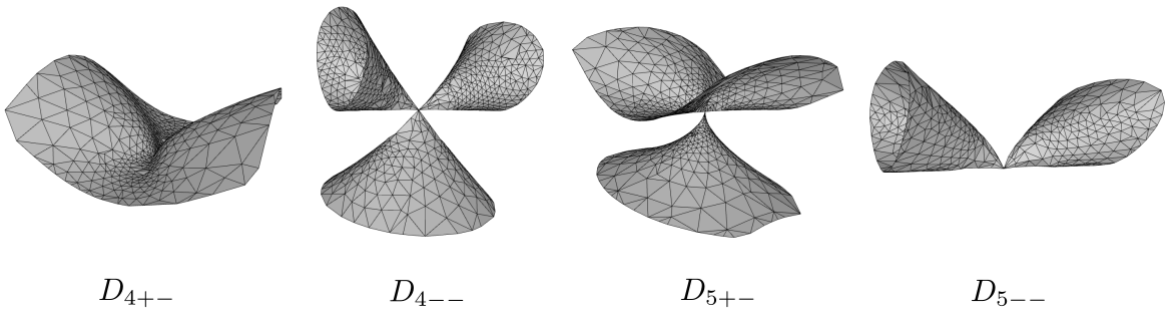


Figure 4.8: Resulting adaptive triangulation of A_n singularities with layers.

by SingSurf can be seen on the Figure 4.9. The comparison on the quality criteria

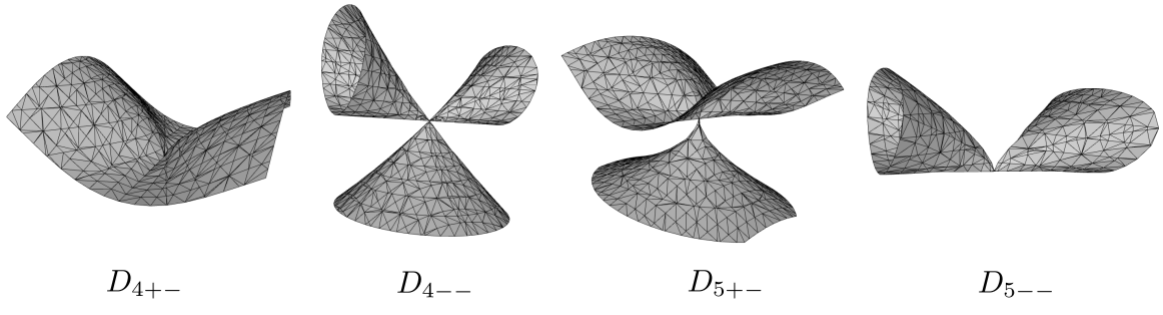


Figure 4.9: Resulting triangulation of D_n singularities by SingSurf [15].

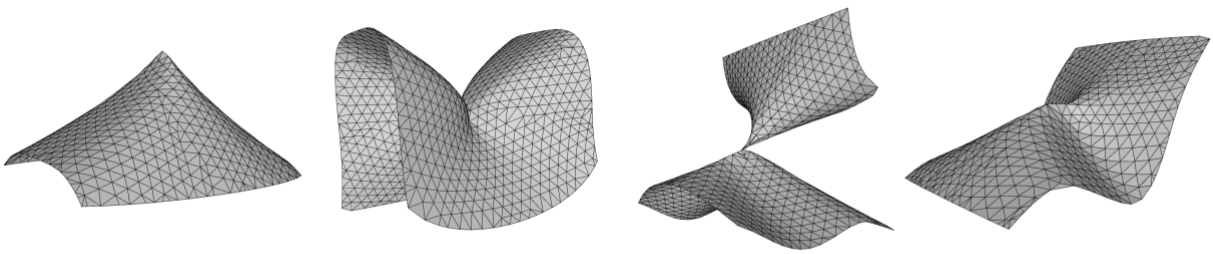
measured on these meshes is in the table 4.3. The better values are visualized by the green color and the worse values are visualized by the red color.

Table 4.3: Comparison of the quality criteria for D_n singularities.

D_n singularities					
type		k_1	k_2	k_3	k_4
D_{4--}	SingSurf	0.091	0.017	0.002	0.052
	Uniform algorithm	0.761	0.023	0.002	0.013
	Adaptive algorithm	0.686	0.025	0.002	0.020
D_{4+-}	SingSurf	0.094	0.028	0.003	0.078
	Uniform algorithm	0.650	0.032	0.003	0.027
	Adaptive algorithm	0.713	0.028	0.003	0.040
D_{5--}	SingSurf	0.001	0.032	0.006	0.076
	Uniform algorithm	0.763	0.059	0.005	0.026
	Adaptive algorithm	0.685	0.061	0.006	0.040
D_{5+-}	SingSurf	0.132	0.029	0.003	0.076
	Uniform algorithm	0.747	0.030	0.003	0.027
	Adaptive algorithm	0.670	0.027	0.003	0.038

E_n singularities

We created meshes of $E_{4++}, E_{6+-}, E_{7++}$ and E_{8++} singularities. The resulting uniform meshes from our algorithm can be seen on the Figure 4.10. The resulting adaptive



E_{6++} E_{6+-} E_{7++} E_{8++}

Figure 4.10: Resulting uniform triangulation of E_n singularities with layers.

meshes from our algorithm can be seen on the Figure 4.11. The resulting meshes generated by SingSurf can be seen on the Figure 4.12. The comparison on the quality criteria measured on these meshes is in the table 4.4. The better values are visualized by the green color and the worse values are visualized by the red color.

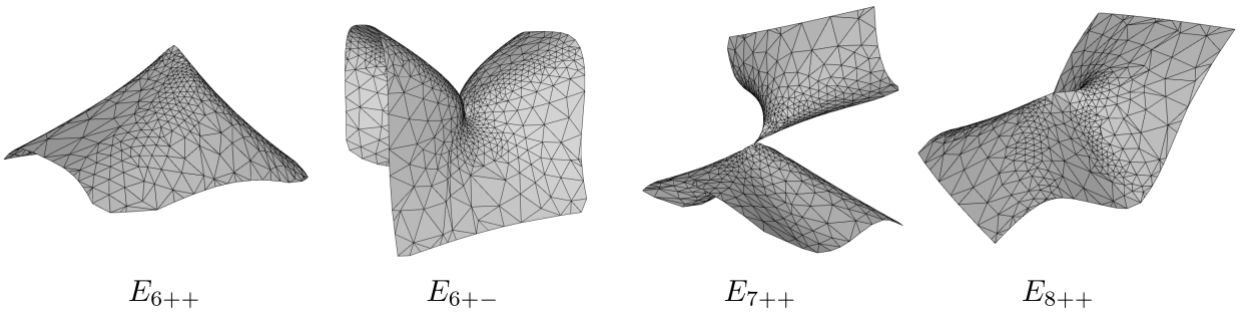


Figure 4.11: Resulting adaptive triangulation of E_n singularities with layers.

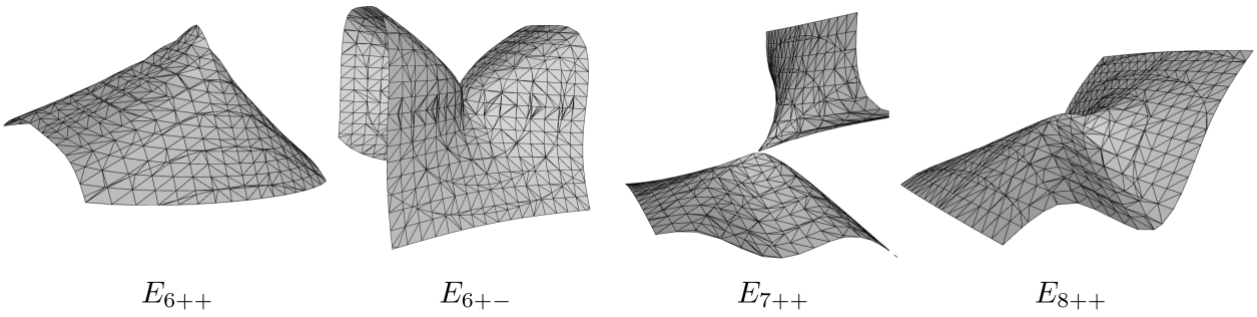


Figure 4.12: Resulting triangulation of E_n singularities by SingSurf [15].

Table 4.4: Comparison of the quality criteria for E_n singularities.

E_n singularities					
type		k_1	k_2	k_3	k_4
E_{6++}	SingSurf	0.006	0.043	0.003	0.069
	Uniform algorithm	0.831	0.013	0.002	0.017
	Adaptive algorithm	0.738	0.021	0.002	0.030
E_{6+-}	SingSurf	0.011	0.040	0.003	0.077
	Uniform algorithm	0.842	0.014	0.002	0.017
	Adaptive algorithm	0.721	0.034	0.002	0.034
E_{7++}	SingSurf	0.198	0.024	0.004	0.110
	Uniform algorithm	0.797	0.027	0.004	0.027
	Adaptive algorithm	0.673	0.028	0.004	0.048
E_{8++}	SingSurf	0.260	0.045	0.004	0.100
	Uniform algorithm	0.803	0.028	0.004	0.032
	Adaptive algorithm	0.703	0.022	0.004	0.056

4.3 Curve singularities

In this section we will present the meshes generated for the intersection, union and difference of the interiors of two regular surfaces. The uniform meshes are generated with multiple different edges length.

4.3.1 Intersection

The intersection of the inside of a sphere given by the equation $x^2 + y^2 + z^2 - 4 = 0$ and the inside of a hyperboloid given by the implicit equation $x^2 - y^2 + z^2 - 1 = 0$ is displayed on the Figure 4.13.

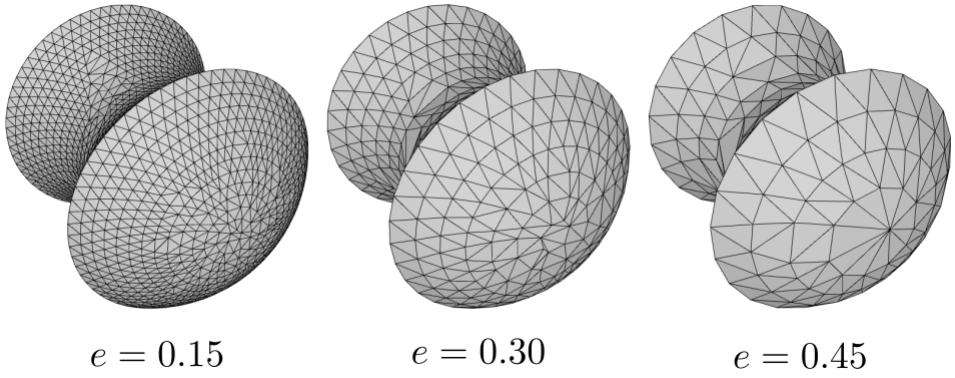


Figure 4.13: The surface given by the intersection of a sphere and a hyperboloid.

The intersection of the inside of a sphere given by the equation $x^2 + y^2 + z^2 - 9 = 0$ and the inside of a tanglecube given by the implicit equation $x^4 - 5 * x^2 + y^4 - 5 * y^2 + z^4 - 5 * z^2 + 11.8 = 0$ is displayed on the Figure 4.14.

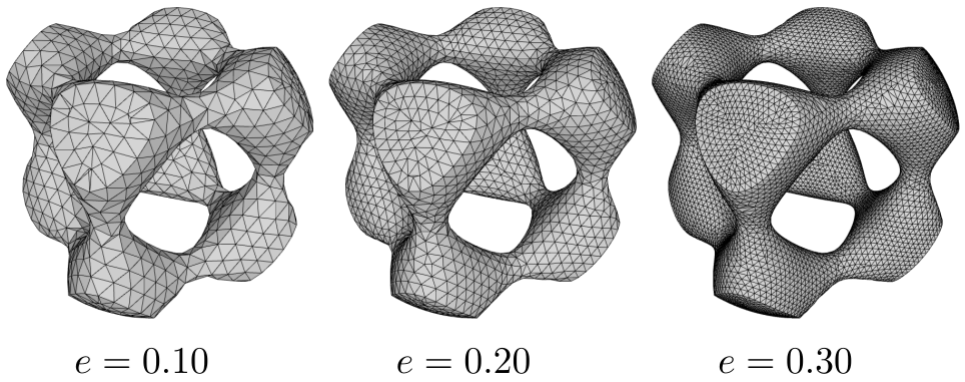
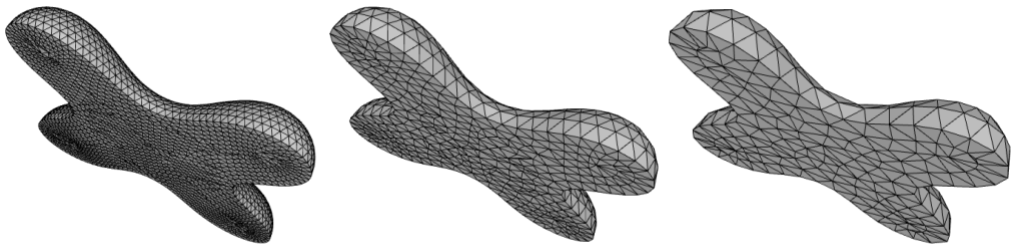


Figure 4.14: The surface given by the intersection of a sphere and a tanglecube.

The intersection of the inside of a blobbly object given by the equation $\sqrt{((x-1)*(x-1)+y*y+z*z)}*\sqrt{((x+1)*(x+1)+y*y+z*z)}*\sqrt{(x*x+(y-1)*(y-}$

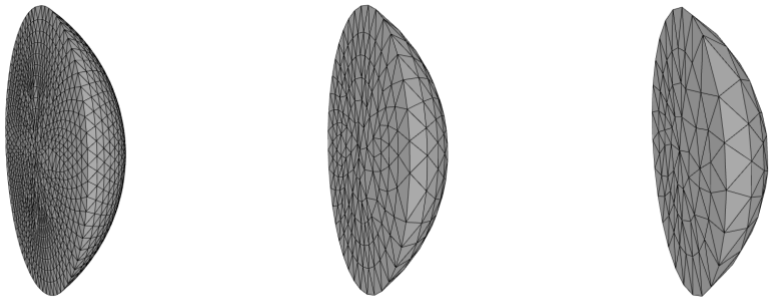
$1) + z * z) * \sqrt{(}x * x + (y + 1) * (y + 1) + z * z) - 1.1 = 0$ and the inside of a plane (halfspace) given by the implicit equation $z = 0$ is displayed on the Figure 4.15.



$e = 0.15$ $e = 0.30$ $e = 0.45$

Figure 4.15: The surface given by the intersection of a blobbly object and a plane.

The intersection of the inside of a sphere given by the equation $x^2 + y^2 + z^2 - 4 = 0$ and the inside of a plane (halfspace) given by the implicit equation $y =$ is displayed on the Figure 4.16.



$e = 0.15$ $e = 0.30$ $e = 0.45$

Figure 4.16: The surface given by the intersection of a sphere and a plane.

4.3.2 Union

TODO The union of the inside of a sphere given by the equation $x^2 + y^2 + z^2 - 4 = 0$ and the inside of a hyperboloid given by the implicit equation $x^2 - y^2 + z^2 - 1 = 0$ is displayed on the Figure 4.13. TODO

4.3.3 Difference

TODO

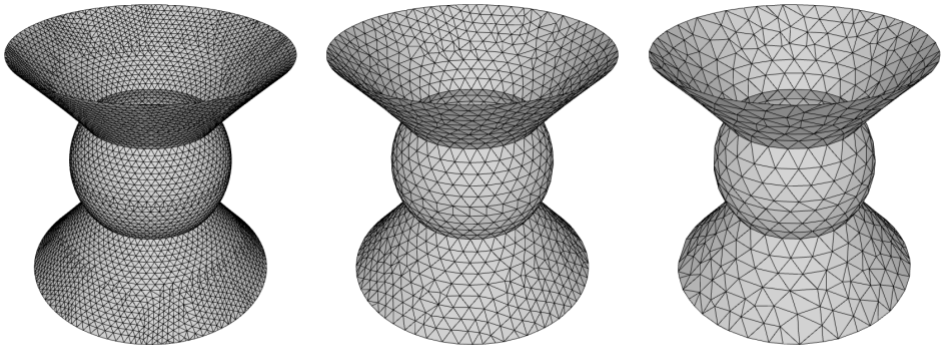


Figure 4.17: The surface given by the intersection of a sphere and a hyperboloid TODO.

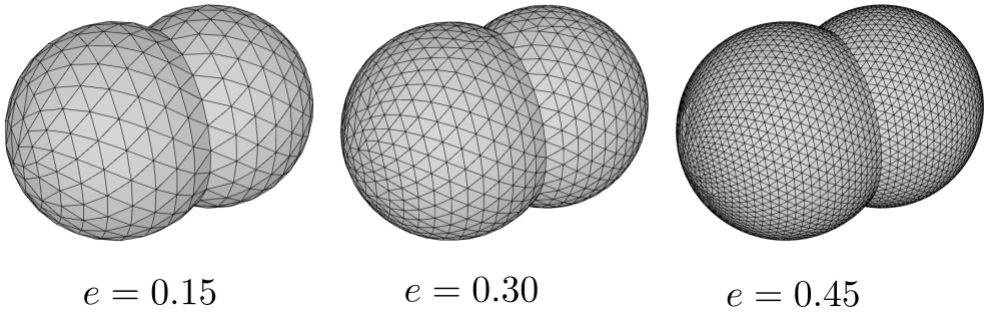


Figure 4.18: The surface given by the intersection of a sphere and a hyperboloid TODO.

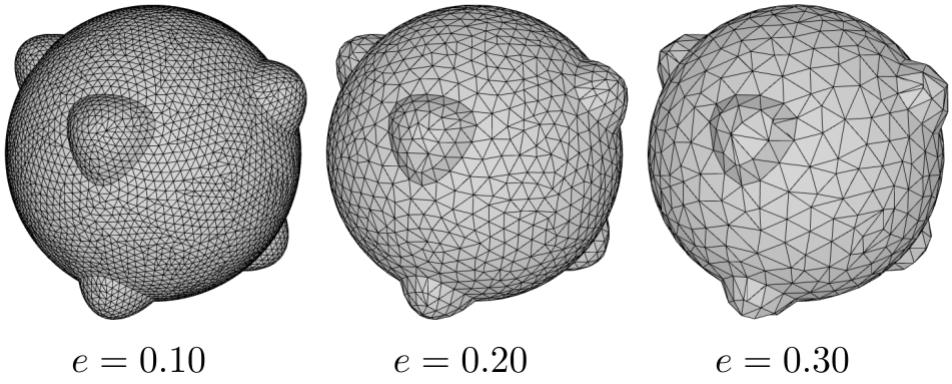


Figure 4.19: The surface given by the intersection of a sphere and a hyperboloid TODO.

4.4 Computational speed comparison of the reimplemented solution

Chapter 5

Future work

Conclusion

TODO Conclusion

Bibliography

[1] CalcPlot3D - software for surface visualisation. `https://c3d.libretexts.org/CalcPlot3D/index.html`. [online: 12.5.2023].

[2] Range tree implementation - GitHub repository. `https://github.com/AjOSK/range-tree`. [online: 12.5.2023].

[3] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. In *Computer Graphics Forum*, volume 20, pages 67–80. Wiley Online Library, 2001.

[4] Vladimir Igorevich Arnol'd. Normal forms for functions near degenerate critical points, the weyl groups of Ak, Dk, Ek and Lagrangian singularities. *Functional Analysis and its applications*, 6:254–272, 1972.

[5] Bruce G Baumgart. A polyhedron representation for computer vision. In *Proceedings of the May 19-22, 1975, national computer conference and exposition*, pages 589–596, 1975.

[6] Evgenii Borisovich Dynkin. The structure of semi-simple algebras. *Uspekhi Matematicheskikh Nauk*, 2(4):59–127, 1947.

[7] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, and John F Hughes. *Computer graphics: principles and practice*, volume 12110. 1996.

[8] Ron Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005.

[9] M Hazewinkel, W Hesselink, D Siersma, and FD Veldkamp. The ubiquity of Coxeter-Dynkin diagrams. *Nieuw Arch. Wisk*, 25:257–307, 1977.

[10] Adrian Hilton, Andrew J Stoddart, John Illingworth, and Terry Windeatt. Marching triangles: range image fusion for complex object modelling. In *Proceedings of 3rd IEEE international conference on image processing*, volume 2, pages 381–384. IEEE, 1996.

[11] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1):65–90, 1999.

[12] Jan J Koenderink and Andrea J Van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992.

[13] doc. RNDr. Pavel Chalmovianský, PhD. Kristína Korecová. Algoritmy triangulácie implicitne definovanej plochy. *Bachelor's thesis*, 2021.

[14] George S Lueker. A data structure for orthogonal range queries. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 28–34. IEEE, 1978.

[15] Richard Morris. A client-server system for the visualisation of algebraic surfaces on the web. In *Algebra, Geometry and Software Systems*, pages 239–253. Springer, 2003.

[16] Tiago Novello, Vinícius Da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Differential geometry of implicit surfaces. 2021.

[17] Aristides Requicha and Robert Tilove. Mathematical foundations of constructive solid geometry: General topology of closed regular sets. *Production Automation Project, University of Rochester, Rochester, New York 14627*, 1978.

[18] Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. *CUMINCAD*, 1977.

[19] Michael Spivak. *A comprehensive introduction to differential geometry*, volume 3. Publish or Perish, Incorporated, 1975.

[20] Kevin Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer graphics and applications*, 5(1):21–40, 1985.

Appendix A

$$f_1 = (x - p_x)^{n+1} - (y - p_y)^2 - (z - p_z)^2$$

$$f_2 = -x - q$$

$$f_3 = f_1 \cap f_2$$

$$f_3 = f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$$

$$f_4 = x + q + q \cdot cos\left(k\sqrt{(y - p_y)^2 + (z - p_z)^2}\right)$$

$$f_5 = 4h^{n+1} - (z - p_z)^2 - (y - p_y)^2$$

$$f_6 = f_4 \cap f_5$$

$$f_6 = f_4 + f_5 - \sqrt{f_4^2 + f_5^2}$$

$$f_7 = x$$

$$f_8 = f_6 \cup f_7$$

$$f_8 = f_6 + f_7 + \sqrt{f_6^2 + f_7^2}$$

$$f_9 = x + q$$

$$f_{10} = f_8 \cap f_9$$

$$f_{10} = f_8 + f_9 - \sqrt{f_8^2 + f_9^2}$$

$$f_{11} = f_{10} \cup f_3$$

$$f_{11} = f_{10} + f_3 + \sqrt{f_{10}^2 + f_3^2}$$

$$f_{11} =$$

The corresponding surfaces are displayed on the figure 5.1

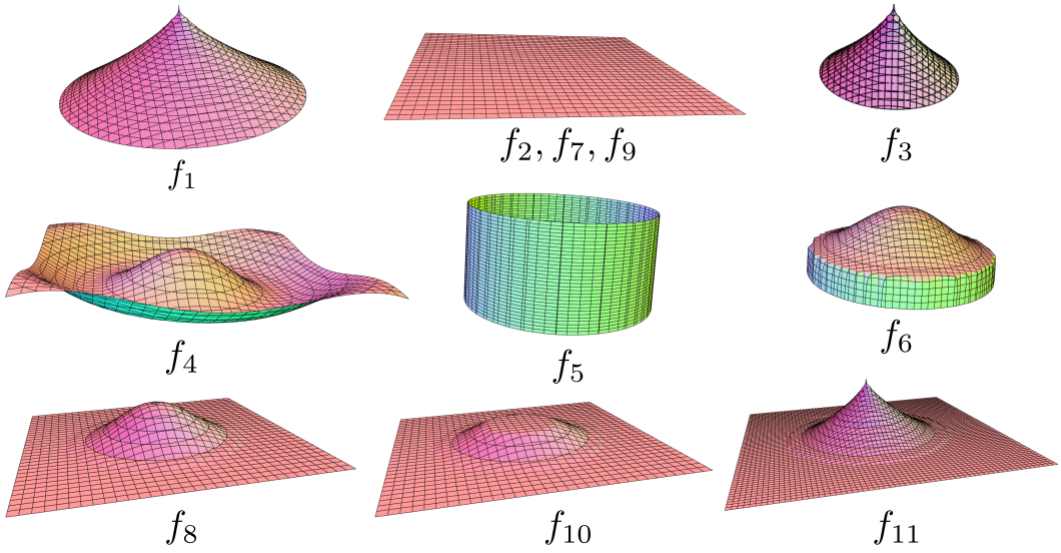


Figure 5.1: Attaching the singularity to a plane using CSG [1].