

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 Chalmovanský, 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 implicitly defined surfaces

**Annotation:** The student proposes an algorithm for triangulation of an implicitly defined surface. Given the singularities of the surface, one starts to triangulate from the singular points. The regular parts should be triangulated adaptively 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 Chalmovanský, PhD.

**Department:** FMFI.KAG - Department of Algebra and Geometry

**Head of department:** doc. RNDr. Pavel Chalmovanský, 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



48237954

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

<b>Introduction</b>	<b>1</b>
<b>1 Theoretical background</b>	<b>3</b>
1.1 Implicit surfaces . . . . .	3
1.1.1 Curvature of a surface . . . . .	4
1.1.2 Curvature formulas for implicit surface . . . . .	7
1.2 ADE singularities . . . . .	8
1.2.1 Correspondence between $SO(3, \mathbb{R})$ group and ADE singularities	9
1.3 CSG modelling for implicit surfaces . . . . .	9
1.3.1 Constructive solid geometry (CSG) . . . . .	10
1.4 Non-isolated surface singularities . . . . .	12
<b>2 Our contributions</b>	<b>15</b>
2.1 Triangulation adaptive to the local curvature . . . . .	15
2.2 Triangulation of ADE singularities . . . . .	18
2.2.1 Analysis of the geometry of ADE singularities . . . . .	18
2.2.2 Analytical calculation of local triangulation of some ADE singularities . . . . .	22
2.2.3 Numerical calculation of local triangulation of ADE singularities	28
2.2.4 Triangulation of a plane with multiple $A_{n--}$ singularities . . . . .	34
2.3 Triangulation of non-isolated singularities . . . . .	39
2.3.1 Creating the local mesh around the singular curves . . . . .	40
<b>3 Implementation</b>	<b>45</b>
3.1 Triangulation of regular implicit surfaces . . . . .	45
3.2 Data structures for triangulation algorithm . . . . .	46
3.2.1 Half-edge data structure . . . . .	46
3.2.2 Range tree . . . . .	49
3.2.3 Mesh structure . . . . .	49
3.2.4 Algorithm runtime . . . . .	50

<b>4 Results</b>	<b>51</b>
4.1 Quality criteria . . . . .	51
4.2 Comparison with SingSurf . . . . .	53
4.2.1 SingSurf algorithm . . . . .	53
4.2.2 Evaluated quality criteria of the SingSurf meshes . . . . .	53
4.3 Computational speed comparison of the reimplemented solution . . . . .	54
<b>5 Future work</b>	<b>57</b>
<b>Conclusion</b>	<b>59</b>
<b>Appendix A</b>	<b>63</b>

# List of Figures

1.1	Implicit surfaces with corresponding equations . . . . .	3
1.2	Isolated and non-isolated singularity . . . . .	4
1.3	Normal cut . . . . .	5
1.4	Categorization of the surface points based on Gaussian curvature . . . . .	6
1.5	Visualisation of the curvatures of the double-torus . . . . .	7
1.6	Shapes representing the $SO(3, \mathbb{R})$ group . . . . .	10
1.7	Example of CSG tree . . . . .	11
1.8	Boolean operations on implicit curves . . . . .	11
1.9	Intersection of a sphere and a plane . . . . .	12
1.10	Intersection of two spheres . . . . .	13
1.11	Projecting a point to the implicit curve . . . . .	13
2.1	Adaptive height of the new triangle . . . . .	15
2.2	Creating the point $P_E$ . . . . .	17
2.3	Adaptive triangulation of a torus . . . . .	17
2.4	$A_{n--}$ singularities . . . . .	19
2.5	$A_{n+-}$ singularities . . . . .	19
2.6	$D_{n+-}$ singularities TODO the coordinates are wrong!! . . . . .	20
2.7	$D_{n--}$ singularities TODO the coordinates are wrong!! . . . . .	20
2.8	Intersection of $D_{n--}$ singularities with plane $z = 0$ . . . . .	21
2.9	Triangulation vectors for two branches of $D_{n--}$ singularities. . . . .	21
2.10	$E_n$ singularities. . . . .	22
2.11	Intersection of $E_{7++}$ and $E_{8++}$ singularities with plane $z = 0$ . . . . .	22
2.12	Triangulation of $A_{n--}$ singularity. . . . .	23
2.13	Resulting uniform triangulation of the $A_{5--}$ singularity . . . . .	24
2.14	Three layers of triangles for the $A_{1--}$ singularity . . . . .	24
2.15	Local mesh with layers for the $A_{5--}$ singularity . . . . .	24
2.16	Resulting mesh for the $A_{5--}$ singularity . . . . .	25
2.17	Equidistant points on ellipse. . . . .	26
2.18	Calculating the point $P_h$ . . . . .	26

2.19 Notation used in the proof . . . . .	27
2.20 1. Category singularities . . . . .	28
2.21 2. Category singularities . . . . .	29
2.22 Binary search for the point on the surface . . . . .	30
2.23 Double binary search for the point on the surface . . . . .	30
2.24 Rotating the plane about the vector . . . . .	31
2.25 Triangulation of 2. category singularities . . . . .	32
2.26 Insufficient and sufficient local approximation . . . . .	33
2.27 Incorrect and correct local mesh . . . . .	33
2.28 Uneven triangles in the local mesh around $D_{4+-}$ singularity . . . . .	34
2.29 Optimized local mesh around $D_{4+-}$ singularity . . . . .	34
2.30 Plane with singularities. . . . .	35
2.31 $C^1$ cosine bump function. . . . .	36
2.32 Construction of the cosine bump function using CSG . . . . .	37
2.33 Attaching the singularity to a plane using the cosine bump function . .	38
2.34 Attaching the singularity to a plane using CSG . . . . .	39
2.35 Plane with multiple attached singularities . . . . .	40
2.36 Approximation of the implicit curve by polyline . . . . .	41
2.37 Definition of the points . . . . .	41
2.38 Definition of the points . . . . .	42
2.39 Local mesh around the singular curve . . . . .	43
 3.1 TODO . . . . .	46
3.2 Example of winged edge representation . . . . .	47
3.3 Visualisation of the half-edge data structure . . . . .	48
3.4 Example of half-edge representation . . . . .	48
 4.1 Resulting uniform triangulation of $A_{n--}$ singularities . . . . .	53
4.2 Resulting triangulation of $A_{n--}$ singularities by SingSurf . . . . .	54
 5.1 Attaching the singularity to a plane using CSG . . . . .	64

# List of Tables

2.1	Implicit equations for bump function modelling . . . . .	37
3.1	Edge table of a winged edge data structure for the Figure 3.2. . . . .	47
3.2	Vertex table of a winged edge data structure for the Figure 3.2. . . . .	47
3.3	Face table of a winged edge data structure for the Figure 3.2. . . . .	47
3.4	Edge table of a half-edge data structure for the Figure 3.4. . . . .	48
3.5	Vertex table of a half-edge data structure for the Figure 3.4. . . . .	48
3.6	Face table of a half-edge data structure for the Figure 3.4. . . . .	49
4.1	Quality criteria – $A_{n--}$ singularities . . . . .	55
4.2	Quality criteria – $A_{n+-}$ singularities . . . . .	55
4.3	Quality criteria – $D_n$ singularities . . . . .	56
4.4	Quality criteria – $E_n$ singularities . . . . .	56



# Introduction

In mathematics, implicit surface is a set of points where the implicit function  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$  is zero. Thanks 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.

In this thesis we follow up on our effort in bachelor's thesis [11] 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 this algorithm to include the triangulation of certain types of isolated singularities - ADE singularities. We analyze the geometry of these singularities and propose an approach to triangulate these singularities.

Lastly, we improve 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, they can be used for modelling of complex surfaces using boolean operations, realistic animations, rendering and other.

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  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ , one can define an implicit surface 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 normalized gradient of the implicit function in that point, provided it is non-zero.

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

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

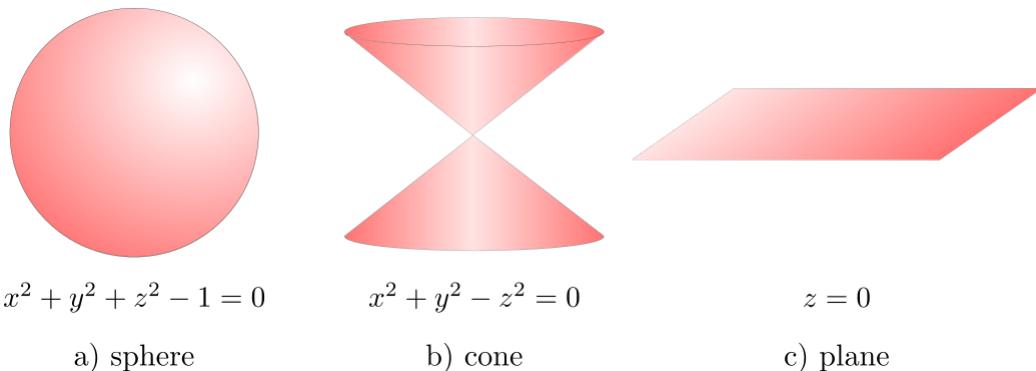


Figure 1.1: Implicit surfaces with corresponding equations.

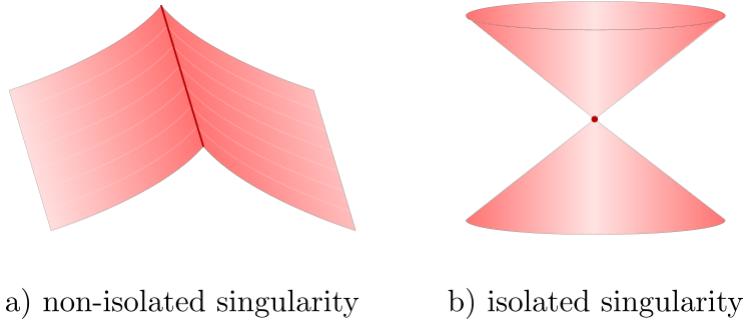


Figure 1.2: Isolated and non-isolated singularity.

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

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

Points lying on the implicit surface can be classified as regular or singular based on the value of the gradient vector in that point.

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

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.

On 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 case of curves, curvature is a measure of how much does the curve differ from a straight line. It is defined as the inverse of the radius of the osculating circle, which is the second order approximation of the curve.

For surfaces, curvature is a measure of how much does the surface differ from a plane. The definition of the curvature of a surface is not as straightforward as 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.

The idea of measuring the curvature of a surface has a long history in mathematics. One of the first contributors was a mathematician Carl Friedrich Gauss, who developed

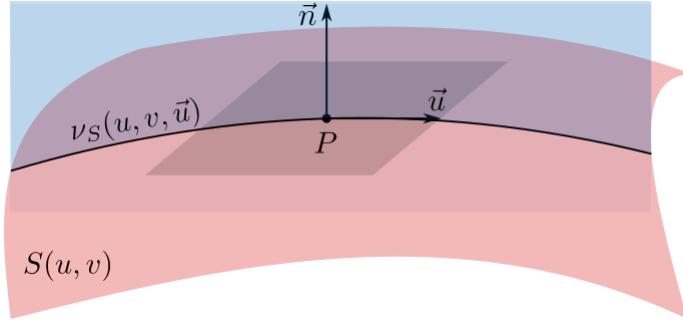


Figure 1.3: Normal cut of the parametric surface  $S(u, v)$ .

the idea of the Gaussian curvature of surfaces. In this subsection, we are drawing from the summary presented by Tiago Novello et al.[15].

### Normal curvature of the surface

Let  $S$  be a parametric surface

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

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

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  $\vec{u}$ .

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

The visualisation of the normal cut is shown on 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  $\nu_S(u, v, \vec{u})$ .

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

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

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

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

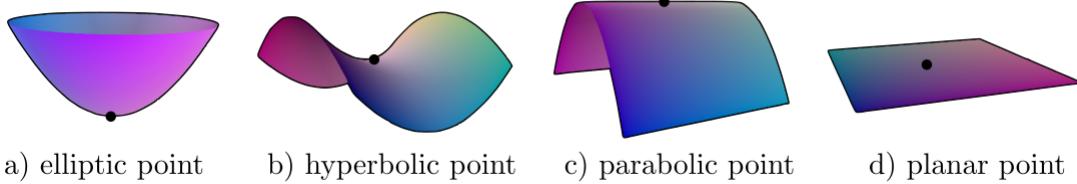


Figure 1.4: Categorization of the surface points based on Gaussian curvature [14].

where  $T_{PS}$  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 neighborhood 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  $\kappa_{\min}$  and  $\kappa_{\max}$  are zero, are called planar. The shape of the surface in the local neighborhoods of the points, visualized on the Figure 1.4, is as follows:

- elliptic points → surface is curved like an ellipsoid,
- hyperbolic points → surface is curved like a saddle,
- parabolic points → surface is curved like a parabolic cylinder,
- planar points → surface is flat.

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 an 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 blue color indicates positive value of the corresponding visualized curvature, red color indicates negative value of the corresponding visualized curvature and white color indicates zero value of the corresponding visualized curvature.

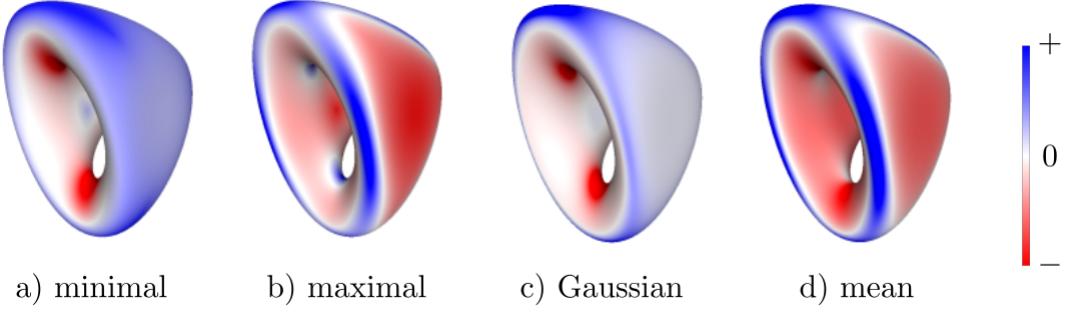


Figure 1.5: Visualisation of minimal, maximal, Gaussian and mean curvatures of the double-torus [15]. Blue/white/red color correspond to positive/zero/negative value of the corresponding curvature.

### 1.1.2 Curvature formulas for implicit surface

A version of curvature formulas for implicit surfaces appeared in [18] and were reformulated, summarized and proved by Ron Goldman [6]. In this subsection, we point out these formulas.

Let  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$  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.

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

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

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

Mean curvature of the implicit surface defined by function  $F$  is given by

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

The principal curvatures  $\kappa_{min}$  and  $\kappa_{max}$  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 [2] according to ADE classification [7] based on correspondence of these singularities to simply laced Dynkin diagrams [4]. We know infinitely many  $A$  singularities –  $A_1, A_2, \dots$ , infinitely many  $D$  singularities –  $D_4, D_5, \dots$  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:

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

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:

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

The most common example of a surface with ADE singularity is an ordinary cone. 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 special orthogonal group over the field of real numbers in three dimensions. It is also called 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  $3 \times 3$  orthogonal matrices of real numbers with determinant 1.*

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

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).

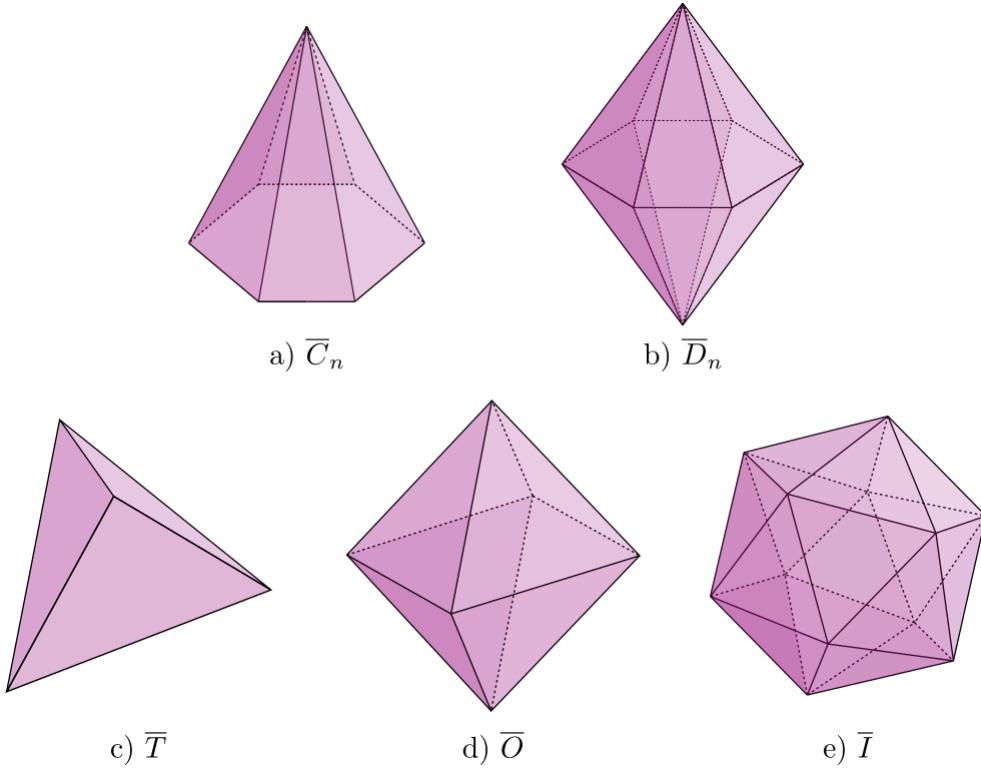
The correspondence is as follows:

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

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

## 1.3 CSG modelling for implicit surfaces

In this section we will discuss how constructive solid geometry can be used for modelling complex implicit surfaces. First major publication regarding constructive solid geometry was published in 1977 by Requicha and Voelcker [17]. More detailed mathematical foundations were published a year later, in 1978 by Requicha and Tilove [16].

Figure 1.6: Shapes representing the  $SO(3, \mathbb{R})$  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 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, which contains geometrical primitives in its leaves and boolean operations in its internal nodes [5]. An example of such CSG tree can be seen on 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 implicit function  $H = \min(F, G)$ . Similarly, the surface of the union of the interiors can be defined by implicit function  $H = \max(F, G)$  and lastly, the surface of the difference of the interiors can be defined by implicit function  $H = \max(F, -G)$ . Two dimensional example of this idea can be seen on the Figure 1.8.

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

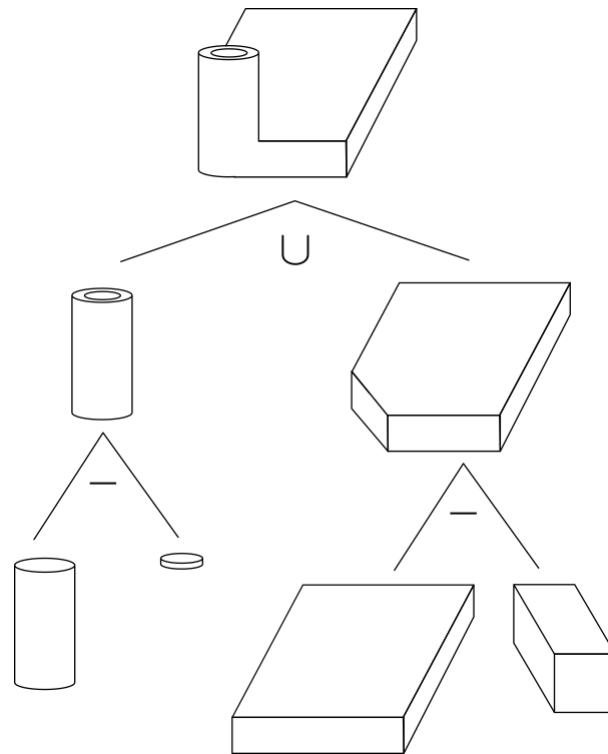


Figure 1.7: Example of CSG tree [5].

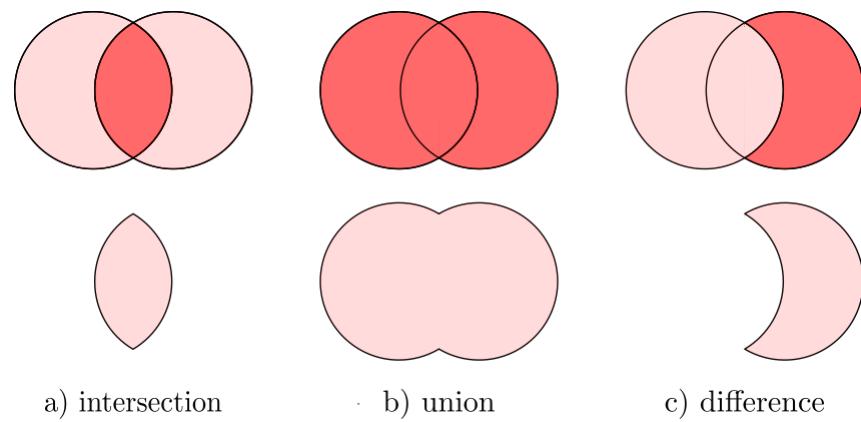


Figure 1.8: Boolean operations on implicit curves.

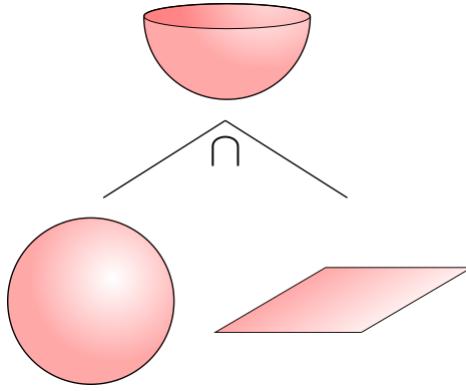


Figure 1.9: Intersection of a sphere and a plane.

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

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

. The maximum function  $\max(F, G)$  of two functions  $F : \mathbb{R}^3 \rightarrow \mathbb{R}$  and  $G : \mathbb{R}^3 \rightarrow \mathbb{R}$  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 finite number of operations union, intersection and difference on arbitrary implicit functions.

An easy example is creating an implicit equation representing a 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

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

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

## 1.4 Non-isolated surface singularities

As we already mentioned, 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

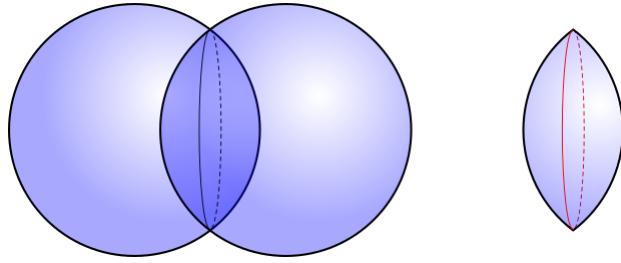


Figure 1.10: Intersection of two spheres.

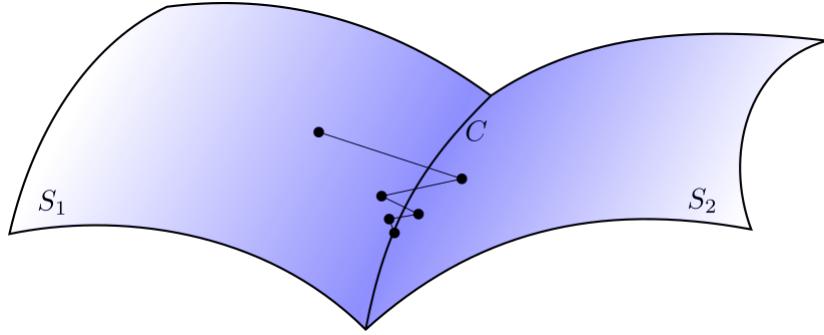


Figure 1.11: Projecting a point to the implicit curve.

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 on the Figure 1.10, the resulting singular curve given by an intersection of the two spheres is displayed by the red color.

### 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-Raphson 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 required precision. A visualization of this approach can be seen on the image 1.11. We denote the projection function  $\text{proj}_C$ .

### Differential geometry of implicit curves

An implicit curve  $C$  is given by two implicit equations  $F_1, F_2 : \mathbb{R}^3 \rightarrow \mathbb{R}$  representing two implicit surfaces  $S_1$  and  $S_2$ , respectively which are intersecting 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  $S_2$

$$\vec{t}_C(P) = \vec{n}_{S_1}(P) \times \vec{n}_{S_2}(P).$$



# Chapter 2

## Our contributions

### 2.1 Triangulation adaptive to the local curvature

As we explained in the begining of chapter 1.1, curvature of the surface is a measure of how much the surface bends.

The triangulation of the surface should be accurate enough, but also memory efficient. This can be achieved by creating a triangulation which is locally adaptive to the curvature of the surface. Therefore having smaller triangles in the places where the surface is curved and having bigger triangles where 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 triangles size, 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 [10]. As the authors explained, the Gaussian curvature and the mean curvature, are not very descriptive about the local shape of the surface. Principal curvatures taken as a pair are more informative. The curvedness was introduced to allow to measure the amount of curvature in the given point by a single number,

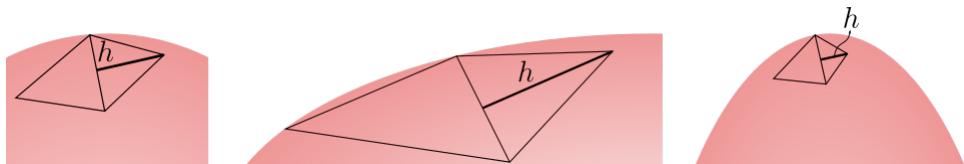


Figure 2.1: Adaptive height of the new triangle.

instead of a pair of numbers. The authors defined the curvedness as

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

where  $\kappa_1$  and  $\kappa_2$  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 in that point.

As an example, we describe the curvedness of 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 a 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 side 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 into  $r_c$ . To avoid the edge case in the planar point, a maximum edge size is set.

During the algorithm, for a given edge  $E$  with the length  $l_E$ , the point  $P_E$  is created as displayed on the Figure 2.2, the point  $P_E$  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

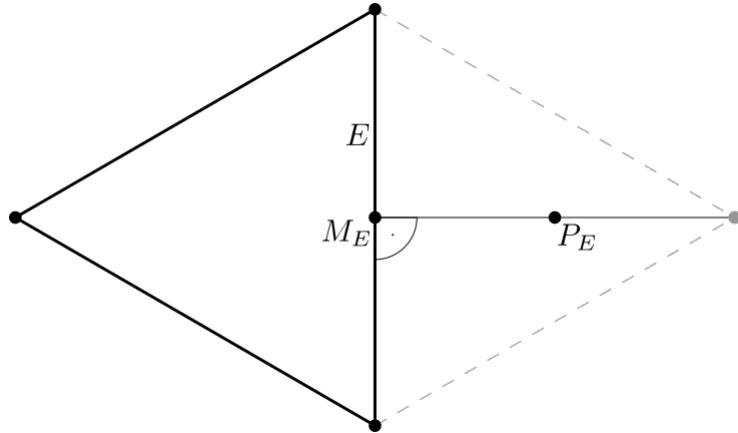


Figure 2.2: Creating the point  $P_E$  for the given edge  $E$ .

the edge  $E$ . The distance between  $M_E$  and  $P_E$  is half of the height of the equilateral triangle with the edge  $E$ .

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 torus and minimal in the outer part of torus. The resulting mesh for 4 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 in the outer part of the torus.

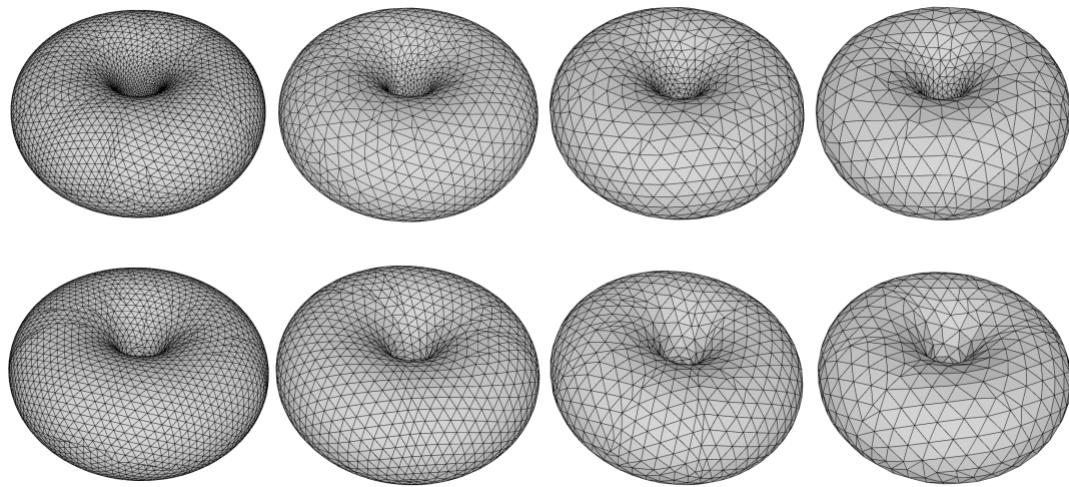


Figure 2.3: Adaptive triangulation of a torus with 4 different constants of detail (top row). Uniform triangulation of a torus with 4 different constants of detail (bottom row).

For the contrast, the uniform triangulation of the same torus for 4 different edge lengths is displayed on the Figure 2.3 in the bottom row. One may note that the triangles in the inner part of the torus are approximately of the same size as the triangles in the outer part of the torus.

## 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. In this section we discuss geometric structure of other ADE surface singularities.

**Definition 12** (*TODO rewrite*) *Let us define branch of ADE singularity as the part of the surface, which is connected to the rest only by the singular point.*

For our needs, we pick one triangulation vector for each branch of each ADE singularity. This triangulation vector is normalized vector either in the direction of 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.

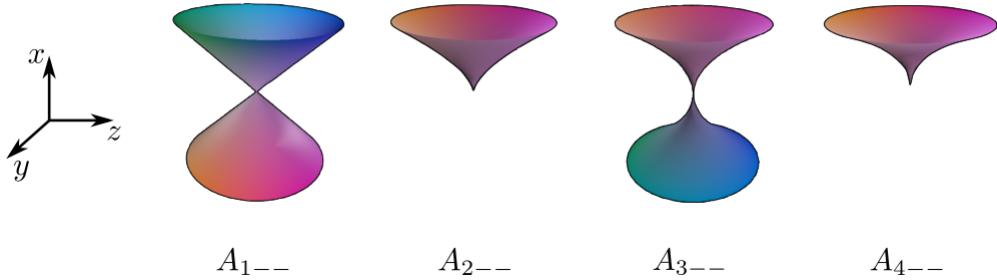
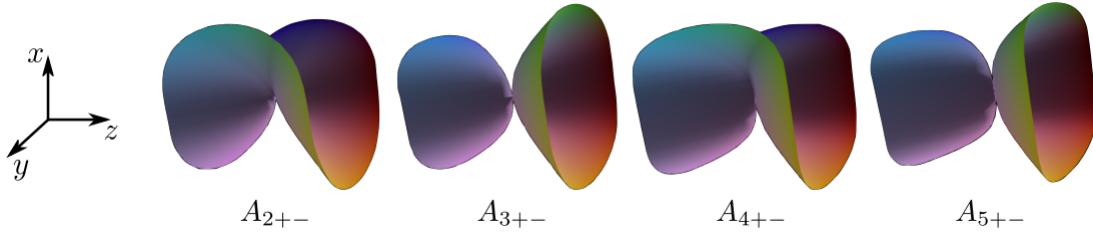
In the general case, triangulation vectors serve us as a partial information about the orientation of a singularity with respect to its normal form.

#### $A_n$ 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 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 axis of symmetry as triangulation vector. For a normal form, the triangulation vectors are  $(1, 0, 0)$  (and  $(-1, 0, 0)$  if  $n$  is odd). First four  $A_{n--}$  singularities can be seen on the Figure 2.4.

$A_{n+-}$  singularities are topologically equivalent to a cone if  $n$  is odd, therefore they have two branches. In the contrary with the previous singularities, as  $n$  gets bigger,

Figure 2.4:  $A_{n--}$  singularities. [14]Figure 2.5:  $A_{n+-}$  singularities. [14]

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.

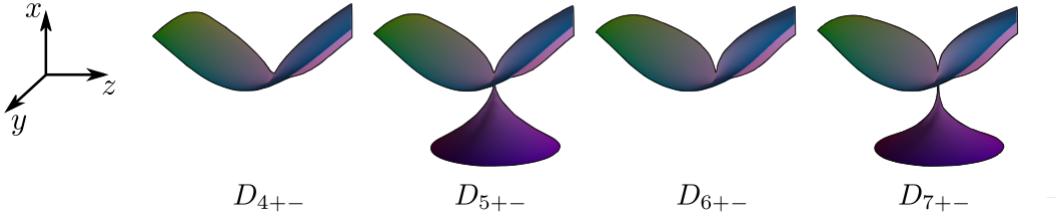
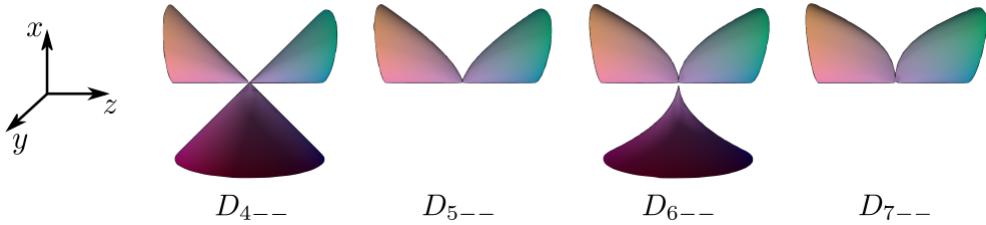
If  $n$  is even,  $A_{n+-}$  singularities are topologically equivalent to a plane with shape similar to hyperbolic paraboloid, therefore they have a single branch. First four  $A_{n+-}$  singularities can be seen on the Figure 2.5. For this case, we pick the vector  $(1, 0, 0)$  as a triangulation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

### $D_n$ 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 8 categories locally represented by the following equations:

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

Figure 2.6:  $D_{n+-}$  singularities. [14]Figure 2.7:  $D_{n--}$  singularities. [14]

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

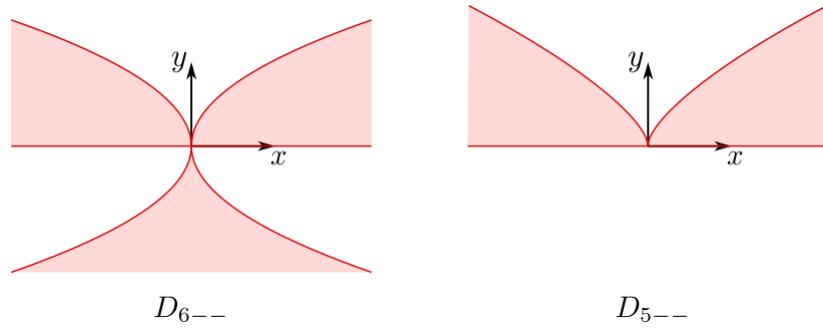
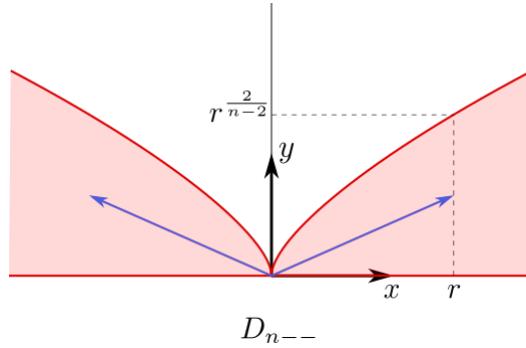
Now we look at some equivalences between these 8 categories.  $D_{4++}$  singularity is reflected  $D_{4+-}$  singularity.  $D_{5++}$  singularity is reflected  $D_{5--}$  singularity.  $D_{5-+}$  singularity is reflected  $D_{5+-}$  singularity.  $D_{4-+}$  singularity is reflected  $D_{4--}$  singularity.

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. First four  $D_{n+-}$  singularities can be seen on the Figure 2.6.

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

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.

Figure 2.8: Intersection of  $D_{n--}$  singularities with plane  $z = 0$ .Figure 2.9: Triangulation vectors for two branches of  $D_{n--}$  singularities.

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 on the Figure 2.9 by blue arrow. Parameter  $r$  is changed based on the length of the edge of triangulation triangle. displayed on the Figure 2.9 by blue arrow. Parameter  $r$  is changed based on the length of the edge of triangulation triangle.

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

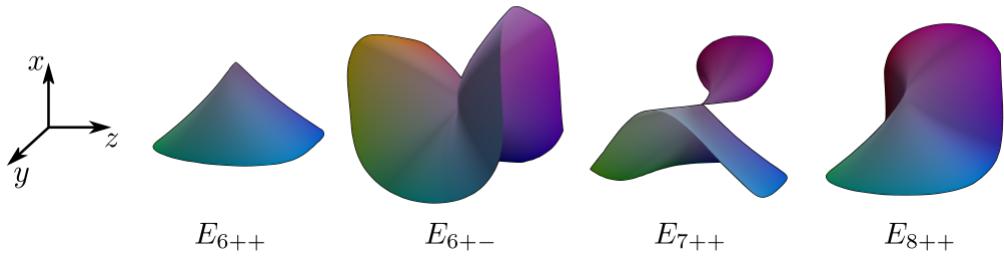
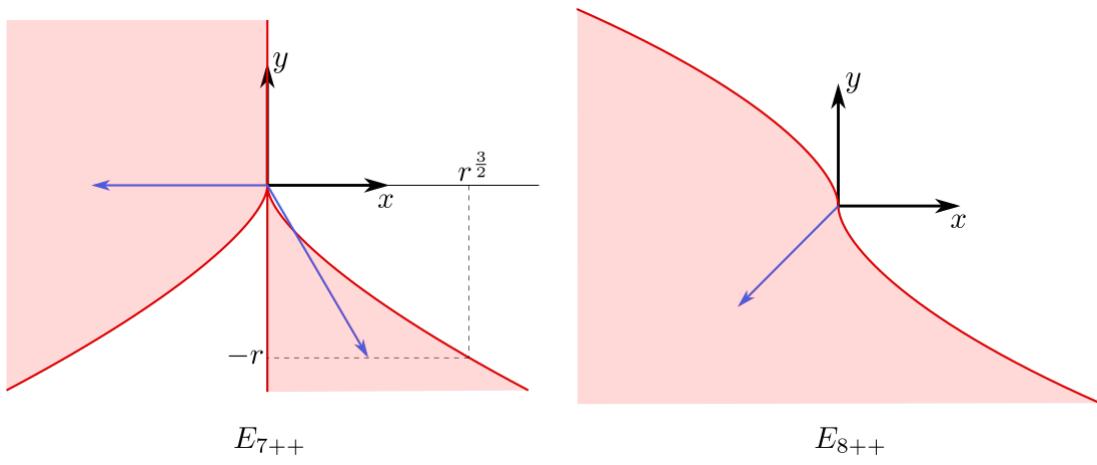
### $E_6, E_7$ and $E_8$ singularities

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 on the 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.

Figure 2.10:  $E_n$  singularities [14].Figure 2.11: Intersection of  $E_{7++}$  and  $E_{8++}$  singularities with plane  $z = 0$ .

The plane of symmetry of this singularity is  $z = 0$ .

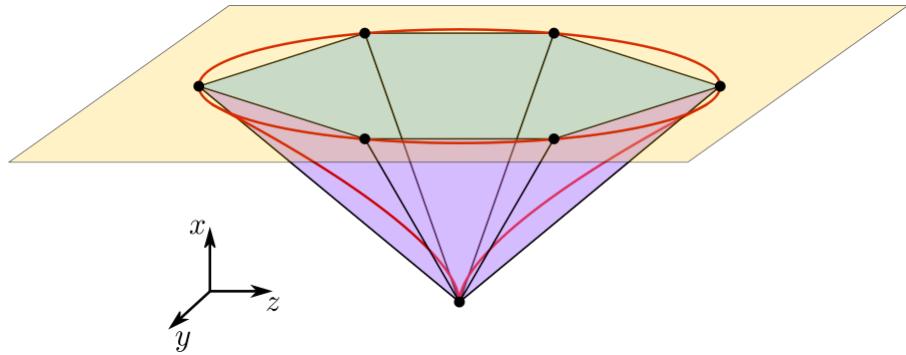
$E_{8++}$  singularity is also topologically equivalent to a plane, therefore it has only one branch. This branch has only one plane of symmetry  $z = 0$ .

We again look at the intersection of the surfaces with the plane of symmetry, this is displayed on 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 on the Figure 2.11 as blue arrows.

## 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 2.12: Triangulation of  $A_{n--}$  singularity.

### $A_{n--}$ singularities

For  $A_{n--}$  singularities, we create a disc of six isosceles triangles with 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 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.

As one may see, the approximation around the singular point is not as accurate as 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 the 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 by using the height  $h = i \cdot h_l$ . As we want 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 1 layer, 4 layers and 8 layers (from left to right) for  $A_{5--}$  singularity is displayed on the Figure 2.15.

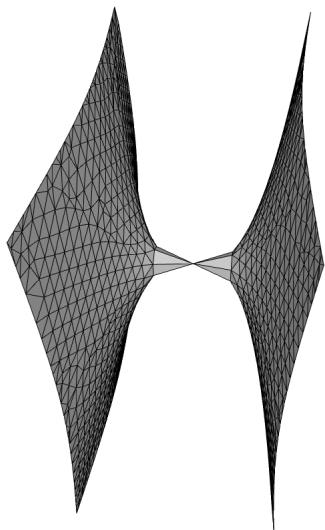


Figure 2.13: Resulting uniform triangulation of the  $A_{5--}$  singularity.

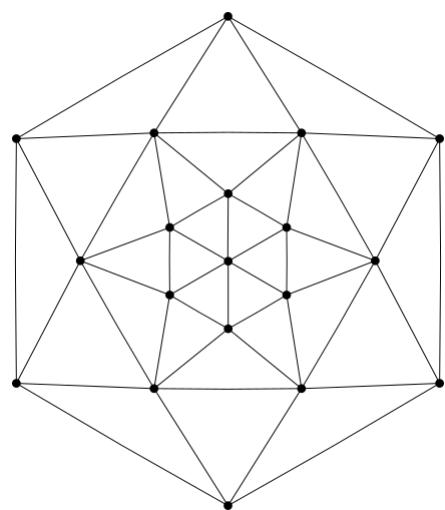


Figure 2.14: Three layers of triangles for the  $A_{1--}$  singularity.

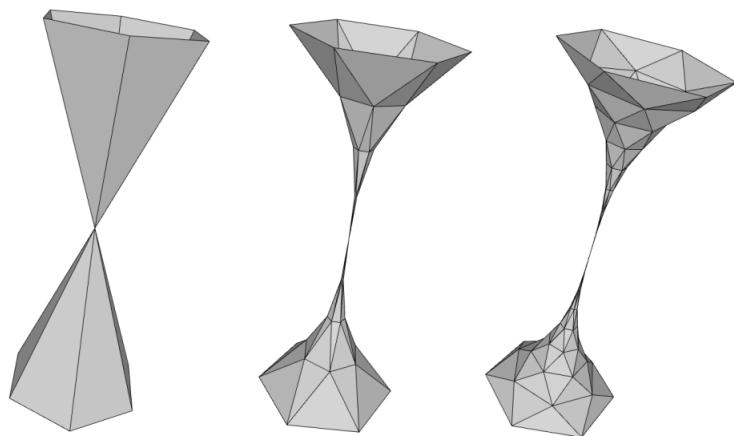


Figure 2.15: Local mesh with 1 layer, 4 layers and 8 layers (from left to right) for the  $A_{5--}$  singularity.

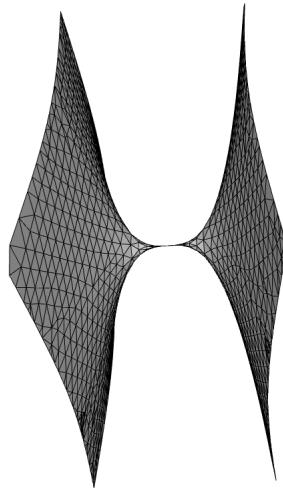


Figure 2.16: Resulting mesh for the  $A_{5--}$  singularity - 5 layers.

The resulting mesh with 4 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 apex in the singular point for these branches. The other points of the triangles lie on the ellipse and they have the same length of the base.

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

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

As displayed on 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 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:

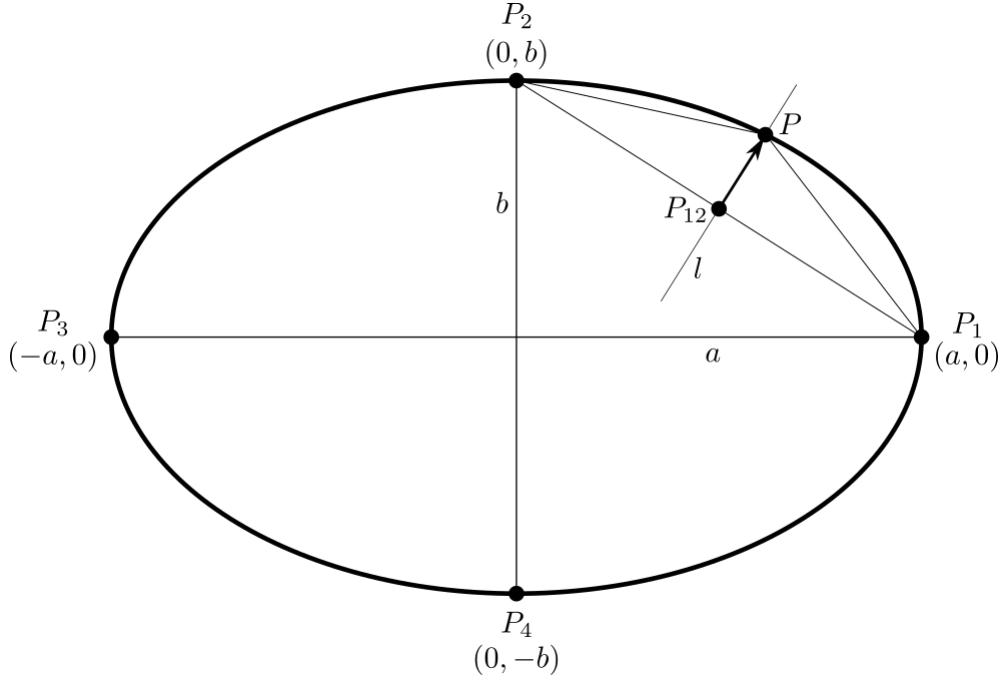
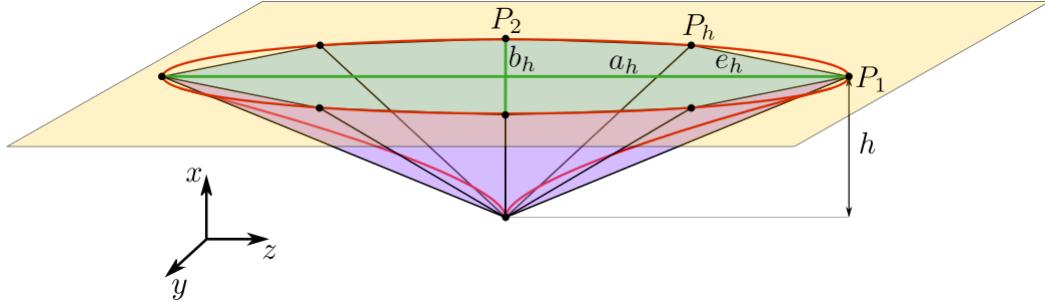


Figure 2.17: Equidistant points on ellipse.

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

$$\begin{aligned} P \in l \cap E \implies \frac{(a+tb)^2}{4a^2} + \frac{(b+ta)^2}{4b^2} = 1 \\ t = \frac{ab(\sqrt{3a^4 + 2a^2b^2 + 3b^4} - a^2 - b^2)}{a^4 + b^4}, \end{aligned}$$

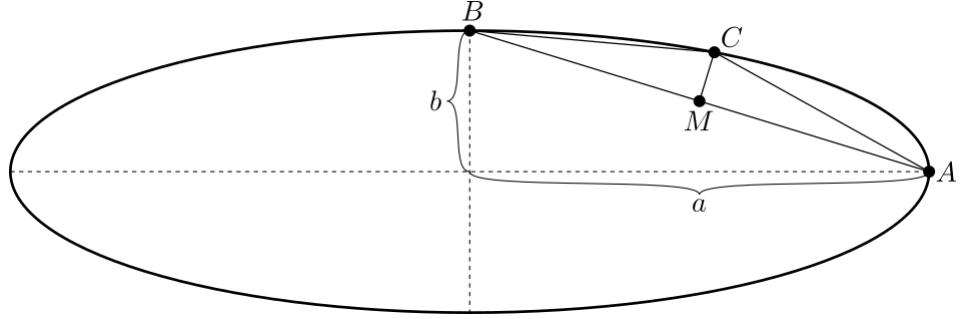
therefore

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

Given edge length  $e$ , we are not able to calculate the height in which the distance between points  $P_1$  and  $P$  is  $e$ . The visualization showing this is on the Figure 2.18.

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

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

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

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

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

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

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

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

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 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

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

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 substituting 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  $a > 0$ .

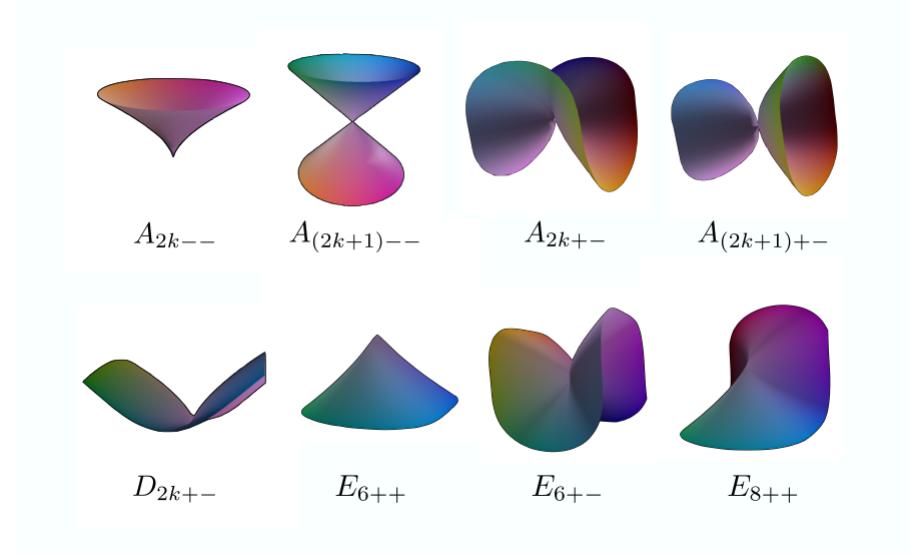


Figure 2.20: 1. Category singularities [14].

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 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

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

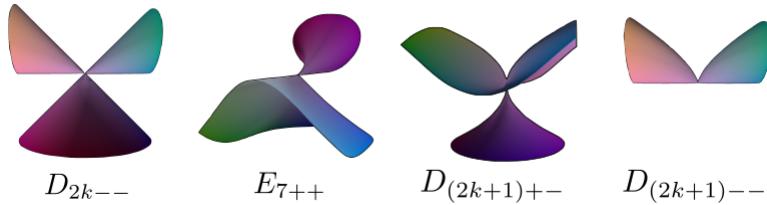


Figure 2.21: 2. Category singularities [14].

- $D_{n+-}$ , 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.  $\vec{t}_1, \dots, \vec{t}_i$  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 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 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 binary search to find an angle, for which the endpoint of the rotated vector  $l \cdot \vec{t}$  lies on the surface with given precision as displayed on 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 case of other singularities, we create different number of triangles:

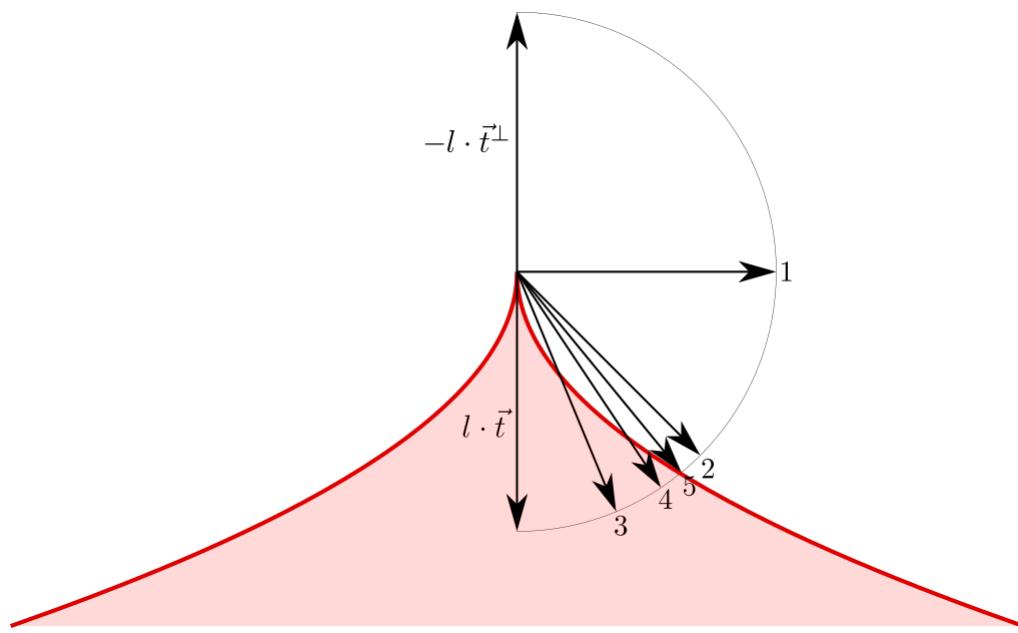


Figure 2.22: Binary search for the point on the surface.

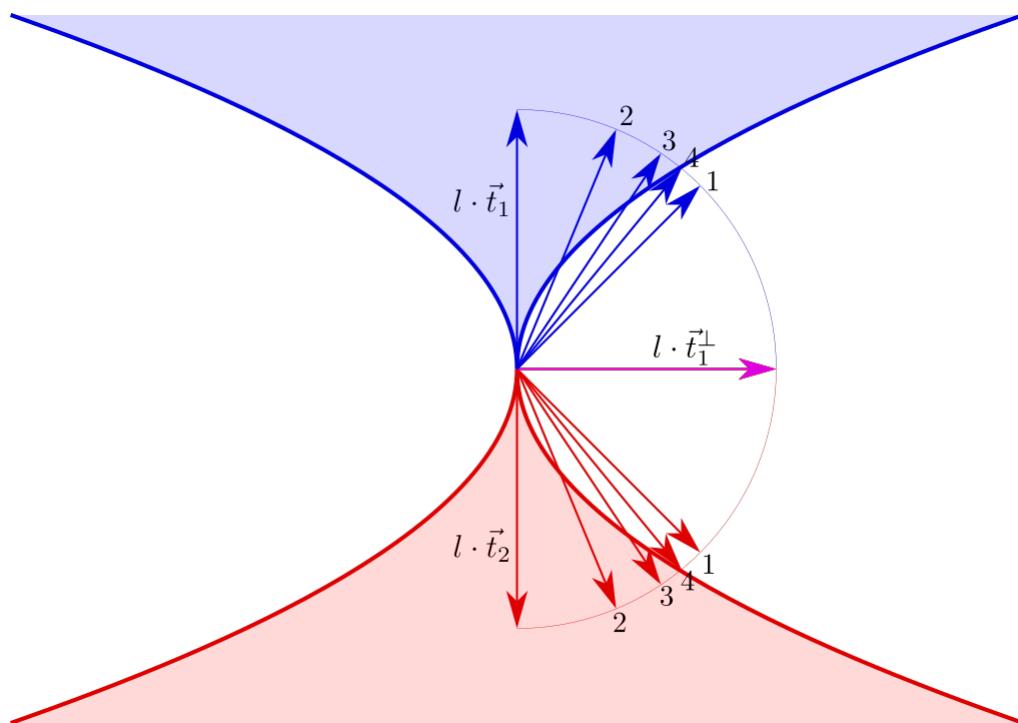


Figure 2.23: Double binary search for the point on the surface.

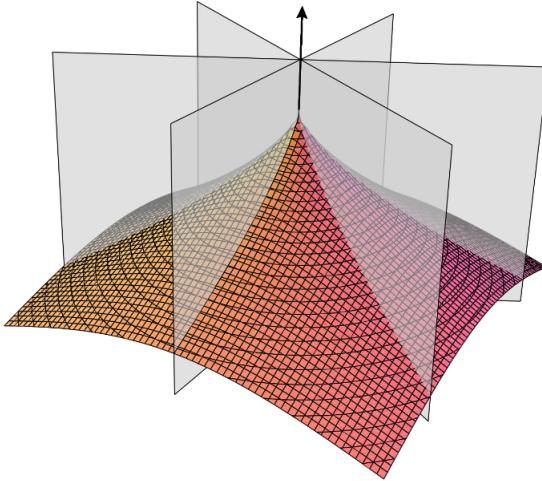


Figure 2.24: Rotating the plane about the vector  $\vec{t}$ .

- $A_{2k+-}$  – 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,
- $E_{6++}$  – 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 less times and find the remaining points as the mirror symmetries of the points found by binary search.

### Triangulation of 2. category singularities

For the singularities in the 2. category, we create more general, slower solution. Again, we find the points on the surface, lying in the rotated half-planes as displayed on the Figure 2.24. For these singularities, we do not have the general rule for the start and end angle to begin the binary search with. We have the triangulation vector, whose endpoint

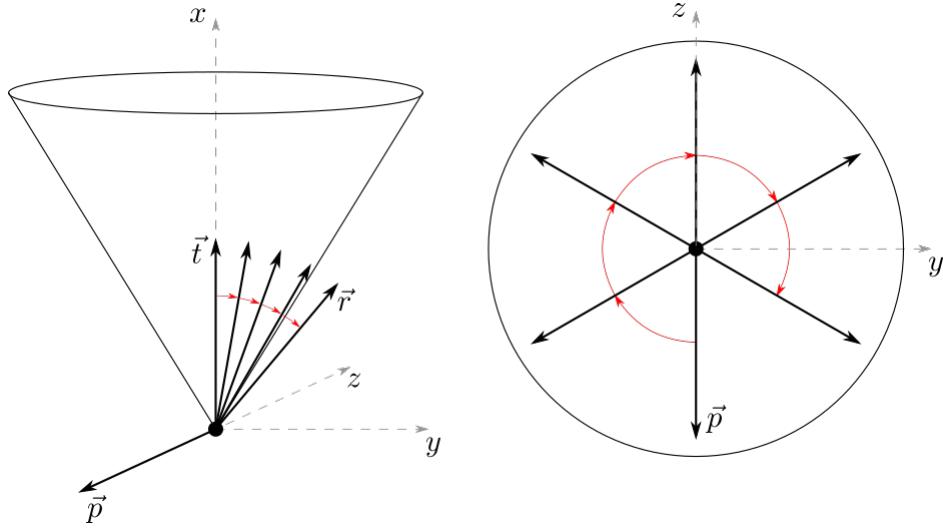


Figure 2.25: Iterative process of finding the vector  $\vec{r}$  by rotating  $\vec{t}'$  around  $\vec{p}$  (left). Rotating the vector  $\vec{p}$  around  $\vec{t}'$  by  $\frac{2\pi}{k}$  (right).

is lying inside of the space volume bounded by the respective branch. We consider this triangulation vector as 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 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, on 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 remaining points on the surface.

TODO rewrite this odsek For the singularities with symmetrical branches, we only need to perform binary search for one of the branches. In case of the singularities with rotation symmetry, we again create the remaining points by using the rotation symmetry. In case of other singularities, we again make use of the potential planes of symmetry.

### Layers for ADE singularities

We 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. For a given singularity, we create  $l$  layers of points for each branch. In contrast with  $A_{n--}$  singularities, we do not rotate every second layer by  $\frac{\pi}{k}$ . The reason is, that the midpoints of the edges connecting a particular layer of points do

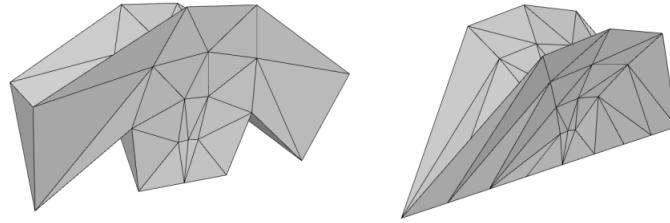


Figure 2.26: Insufficient (left) and sufficient (right) local approximation of  $D_{4+-}$  singularity.

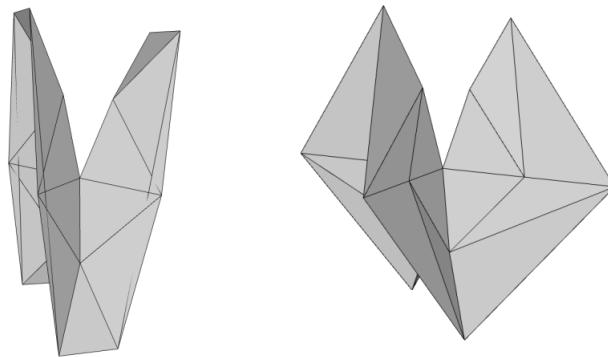


Figure 2.27: Incorrect (left) and correct (right) local mesh for  $A_{2+-}$  singularity.

not lie on the surface. This results in insufficient approximation of the curves lying in the intersection of the surface and coordinate planes. In some cases, it even results in incorrect local mesh with overlapping triangles.

On 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  $\frac{\pi}{k}$ .

### 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 on the Figure 2.24. An example of such singularities are  $D_{n+-}$  singularities. The local mesh created with 8 triangles using the

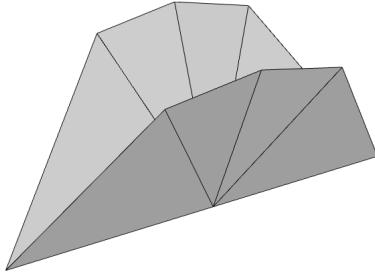


Figure 2.28: Uneven triangles in the local mesh around  $D_{4+}$ - singularity.

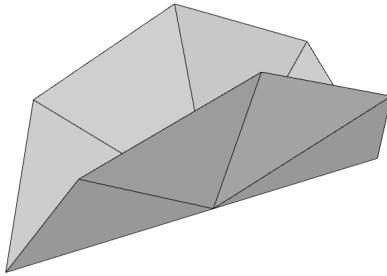


Figure 2.29: Optimized local mesh around  $D_{4+}$ - singularity.

described approach is displayed on the Figure 2.28. One may see that the outer edges of the triangles have a significant length difference.

The problem is significant on the singularities with 8 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 in between these points are iteratively changed until the triangles are isosceles with 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 on 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 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 with 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

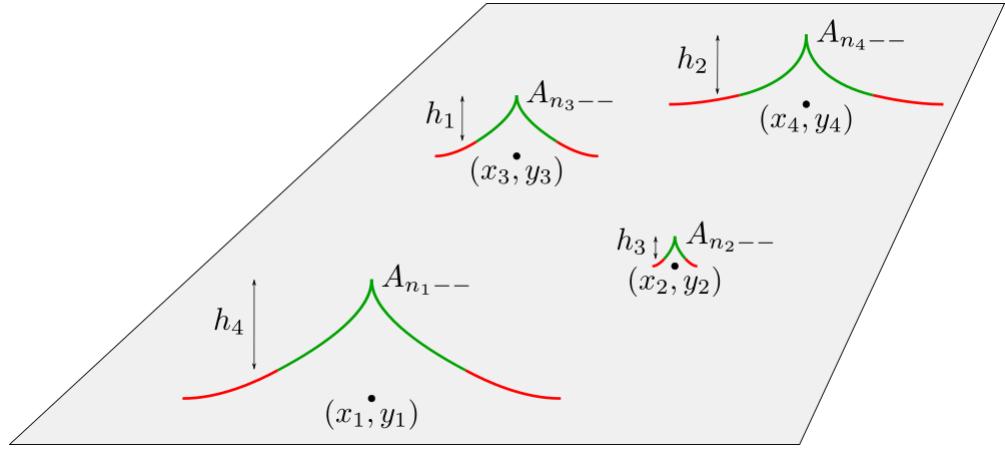


Figure 2.30: Plane with singularities.

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 -  $(x_1, y_1), \dots, (x_m, y_m)$ ,
3.  $m$  degrees of the singularities -  $n_1, \dots, n_m$ ,
4.  $m$  heights at which each singularity is connected -  $h_1, \dots, h_m$ .

The visualisation of desired output function can be seen on the Figure 2.30. On this figure, the singularity is displayed by green color, the red color is used to display 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 value of  $d_{ij}$  in the section TODO.

### Bump function

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

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

*The closed support of the function  $f$  is defined as a closure of  $\text{supp}(f)$ .*

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

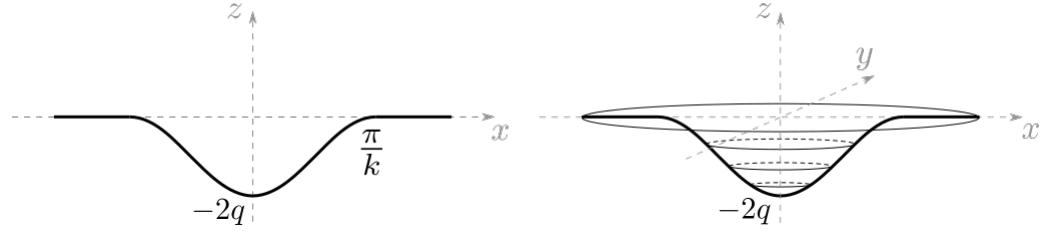


Figure 2.31:  $C^1$  cosine bump function.

The most common example of such bump function is the function

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

which is both  $C^\infty$  and compactly supported.

The bump function can be used to smoothly connect a curve to a line or a surface to a plane in 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

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

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

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

Both of these functions can be seen on the Figure 2.31.

### Implicit equation of the cosine bump function

To construct the implicit equation of the cosine bump function, we use CSG - constructive solid geometry, which is 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 on the Figure 2.32 in the form of 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  $(p_y, p_z)$ .

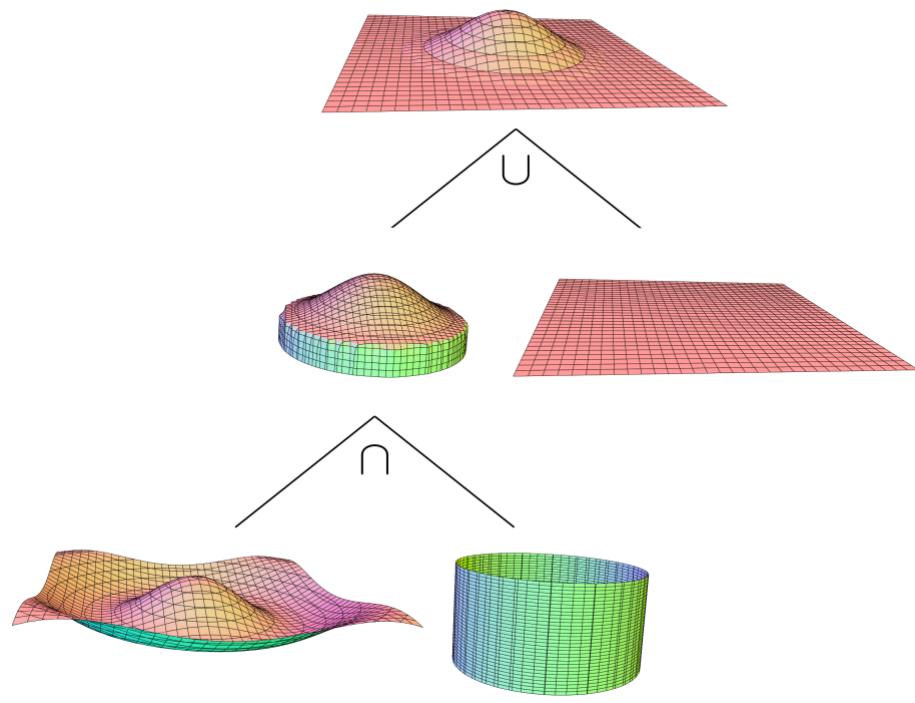


Figure 2.32: Construction of the cosine bump function using CSG.

Funcion 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.

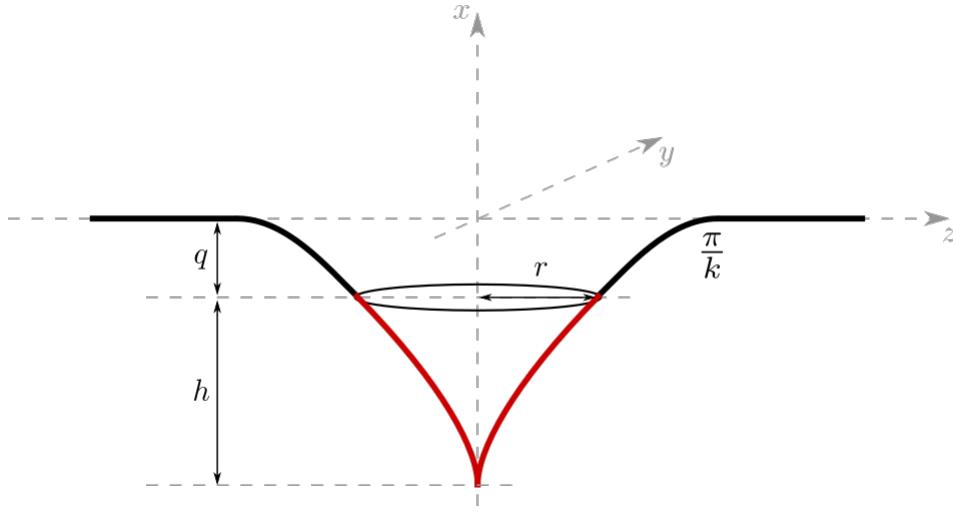


Figure 2.33: Attaching the singularity to a plane using the cosine bump function.

### 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 on 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  $C^1$  continuity requirement. We require the gradients to be linearly dependent in the points of connection. Due to the rotation symmetry, we check it only for the intersection with the plane  $z = 0$  for the point  $(-q, \frac{\pi}{2k})$ .

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

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

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

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

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 union of these two surfaces. The described process is displayed on the

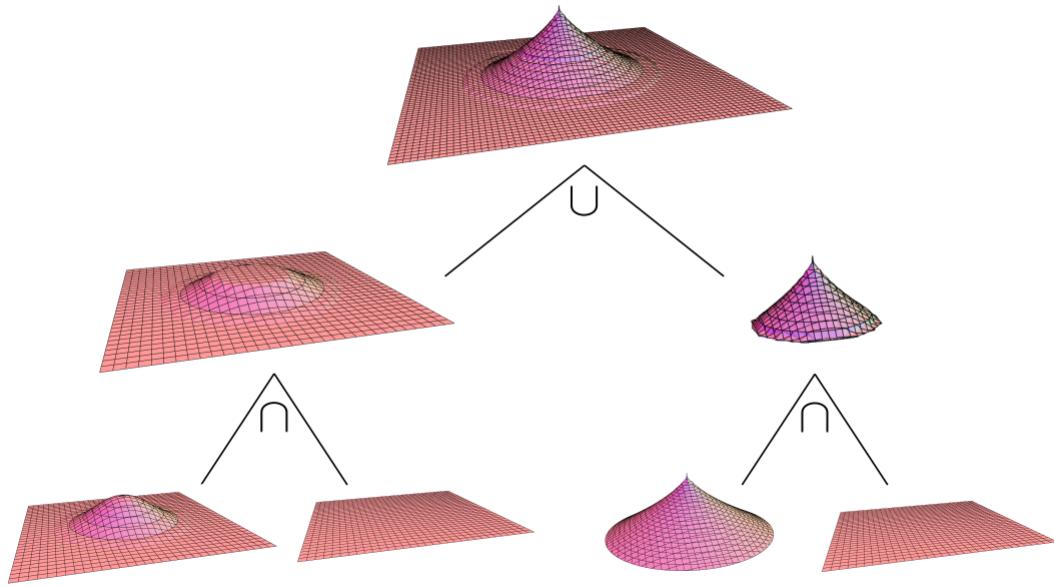


Figure 2.34: Attaching the singularity to a plane using CSG.

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 of these singularities and then use the union operation to create a surface with multiple singularities. This procedure is displayed on the Figure 2.35.

### Limitations on the input data

As we already 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.

### 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.

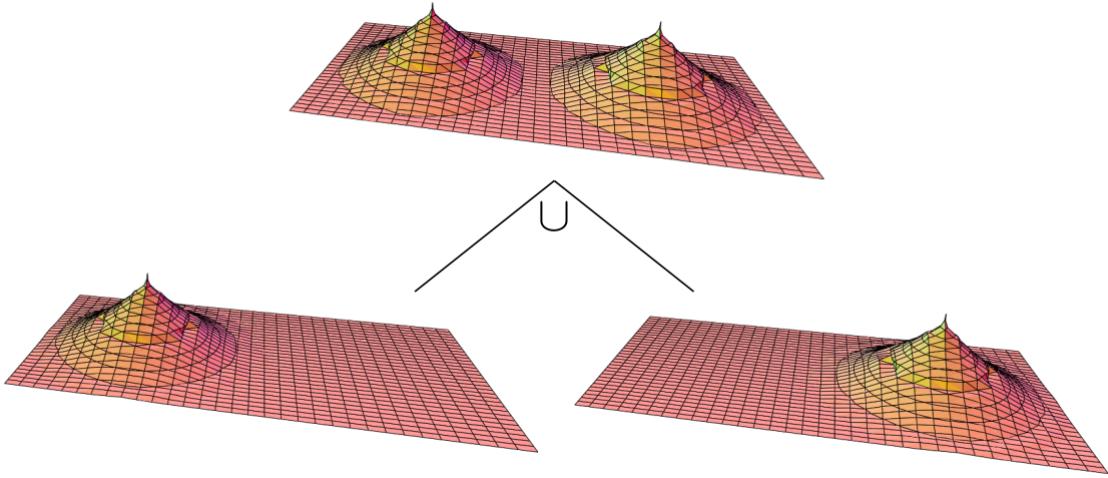


Figure 2.35: Plane with multiple attached singularities.

### 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:

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

$$P_{n+1} = \text{proj}_C(Q_{n+1}),$$

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 curve is a closed curve.

In 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 number of points in the polyline, the second part of the polyline consists of the points  $P_m, P_{m+1}, \dots$ , where

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

$$P_m = \text{proj}_C(Q_m),$$

and again, iteratively

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

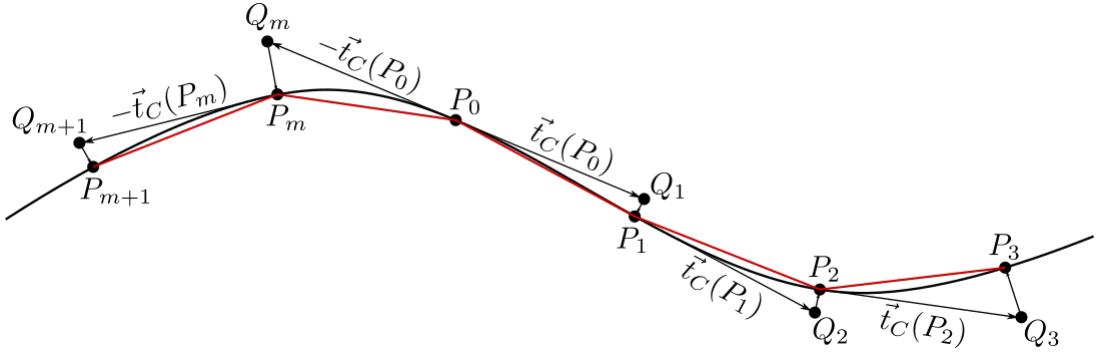
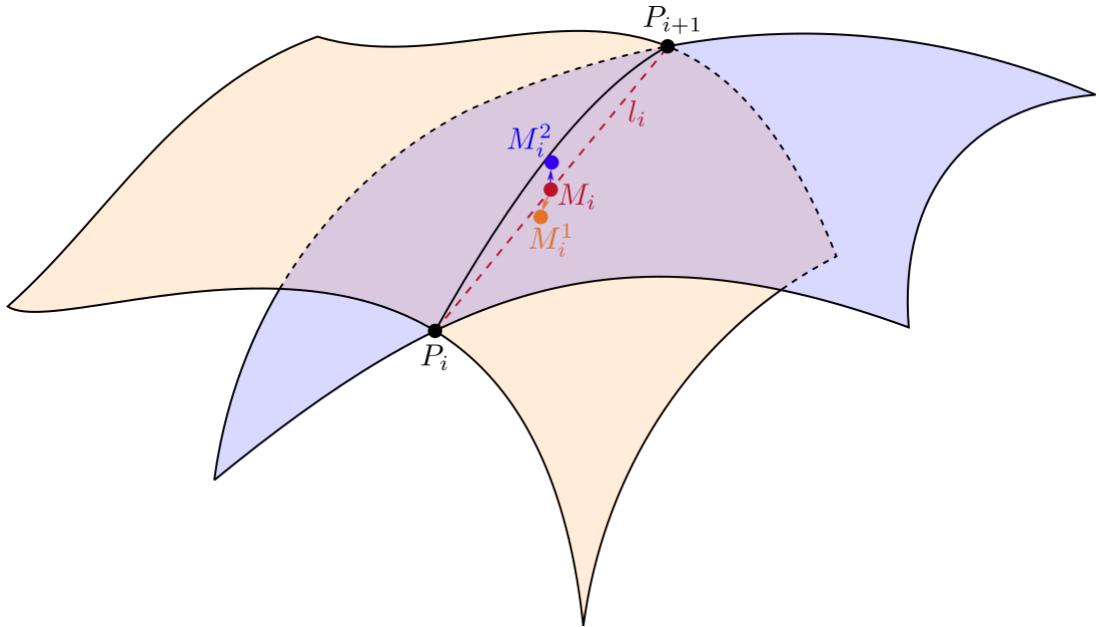


Figure 2.36: Approximation of the implicit curve by polyline.

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

$$P_{n+1} = \text{proj}_C(Q_{n+1}), \quad n = m, m+1, \dots$$

The presented approach is visualized in two dimensions on 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  $P_0, P_1, \dots, P_k$ , such that  $l_i = \overline{P_i P_{i+1}}$  is a line segment of the polyline for  $i = 0, \dots, k-1$  (and  $l_k = \overline{P_k P_0}$  is a line segment of the polyline for 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 on 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  $S_2$ .

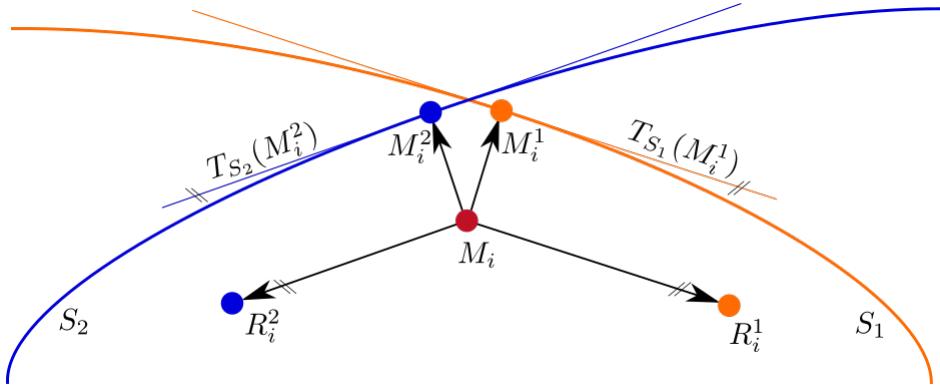


Figure 2.38: Definition of the points  $R_i^1$  and  $R_i^2$  in the plane  $H_i$ .

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  $e$ .

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

The described situation is displayed on 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 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 on 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.

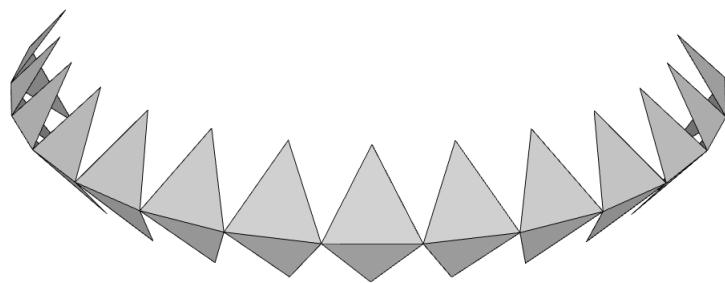


Figure 2.39: Local mesh around the singular curve.



# Chapter 3

## Implementation

### 3.1 Triangulation of regular implicit surfaces

In this section, we shortly describe the algorithm for triangulation of the regular parts of implicit surfaces introduced in bachelor's thesis [11].

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 Newton-Raphson method. This method finds 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 [8]. 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 triangles in its proximity.

The algorithm is enriched with the possibility to triangulate adaptively to the curvature of the surface using approach presented by Akkouche [1]. 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 axes.

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 [9] and the range tree [12].

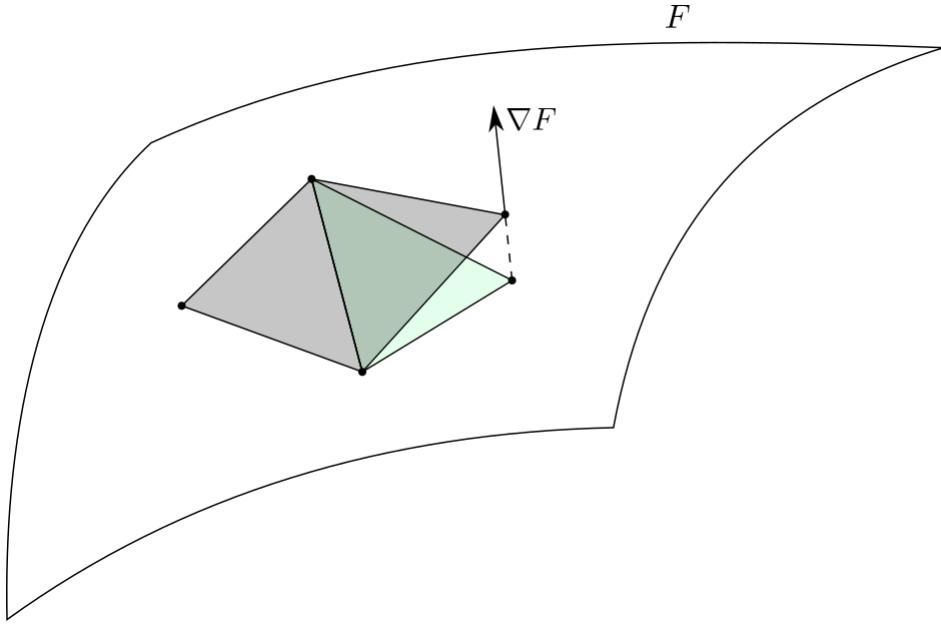


Figure 3.1: TODO

## 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 faces 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 [3] presented a representation using winged edges, which was further improved in 1985 by Weiler [19] 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 winged edge representation, each edge is represented as oriented line segment and stores references to its vertices, both incident faces, left and right traverse. Each vertex stores a reference to one of the outgoing edges and each face stores a reference to one of its incident edge. An example of such representation is shown on the Figure 3.2 and tables 3.1, 3.2, 3.3.

In the half-edge representation, edges are split into two halves. Each half-edge stores reference to its initial vertex, left face, opposite half-edge and left traverse - preceding and succeeding half-edge. A visualisation of the half-edge is displayed on the Figure 3.3. An example of half-edge representation is shown on the Figure 3.4 and tables 3.4, 3.5, 3.6.

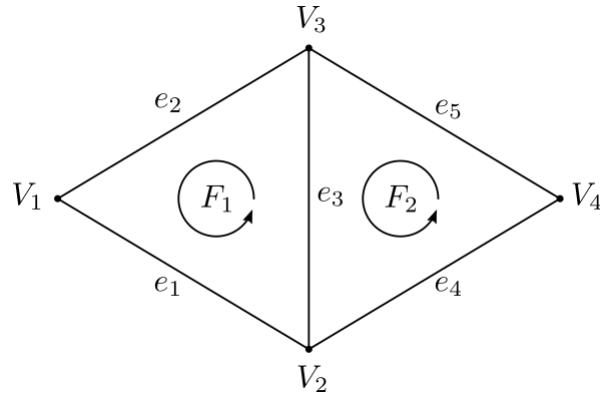


Figure 3.2: Example of winged edge representation.

Edge	Vertex		Face		Left traverse		Right traverse	
	Initial	Terminal	Left	Right	Pred.	Succ.	Pred.	Succ.
$e_1$	$V_1$	$V_2$	$F_1$		$e_2$	$e_3$		
$e_2$	$V_3$	$V_1$	$F_1$		$e_3$	$e_1$		
$e_3$	$V_2$	$V_3$	$F_1$	$F_2$	$e_1$	$e_2$	$e_5$	$e_4$
$e_4$	$V_2$	$V_4$	$F_2$		$e_3$	$e_5$		
$e_5$	$V_4$	$V_3$	$F_2$		$e_4$	$e_3$		

Table 3.1: Edge table of a winged edge data structure for the Figure 3.2.

Vertex	Coordinates			Edge
	x	y	z	
$V_1$	$x_1$	$y_1$	$z_1$	$e_1$
$V_2$	$x_2$	$y_2$	$z_2$	$e_4$
$V_3$	$x_3$	$y_3$	$z_3$	$e_2$
$V_4$	$x_4$	$y_4$	$z_4$	$e_5$

Table 3.2: Vertex table of a winged edge data structure for the Figure 3.2.

Face	Edge
$F_1$	$e_1$
$F_2$	$e_4$

Table 3.3: Face table of a winged edge data structure for the Figure 3.2.

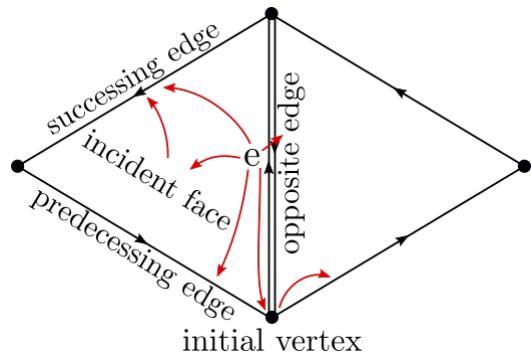


Figure 3.3: Visualisation of the half-edge data structure.

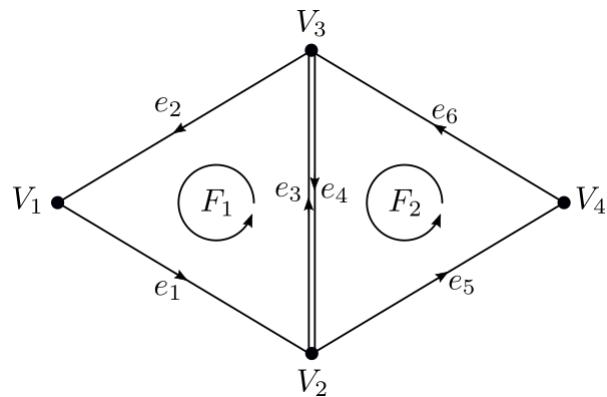


Figure 3.4: Example of half-edge representation.

Edge	Vertex	Face	Edges		
	Initial	Incident	Pred.	Succ.	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 3.4: Edge table of a half-edge data structure for the Figure 3.4.

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 3.5: Vertex table of a half-edge data structure for the Figure 3.4.

Face	Edge
$F_1$	$e_1$
$F_2$	$e_4$

Table 3.6: Face table of a half-edge data structure for the Figure 3.4.

### 3.2.2 Range tree

### 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 vertices 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,
- set of active edges,
- set of checked edges,
- set of bounding edges,
- 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 preceding edge in the list of half-edges,
- index of succeeding edge in the list of half-edges,
- index of incident face in the list of faces.

Face consists of

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

### 3.2.4 Algorithm runtime

TODO

# Chapter 4

## Results

In this chapter, we present the results achieved in triangulation of ADE singularities, surfaces with ADE singularities and surfaces with singular curves arised from CSG operations on the regular surfaces. We measure some of the quality criteria proposed in [11] 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 [14]. We compare the computational speed of the algorithm after the reimplementation with the algorithm for the regular parts of the implicit surfaces implemented in [11].

### 4.1 Quality criteria

The following quality criteria [11] 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  $\delta$  of the point  $a \in \mathbb{R}^3$  and the set  $M \subset \mathbb{R}^3$  is defined as*

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

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*

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

The Hausdorff distance is numerically approximated 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 distribution 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 –  $x_1, \dots, x_N$ , let us denote the arithmetical mean of the values as  $\bar{x}$ . The standard deviation  $\sigma$  can be calculated as*

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

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.

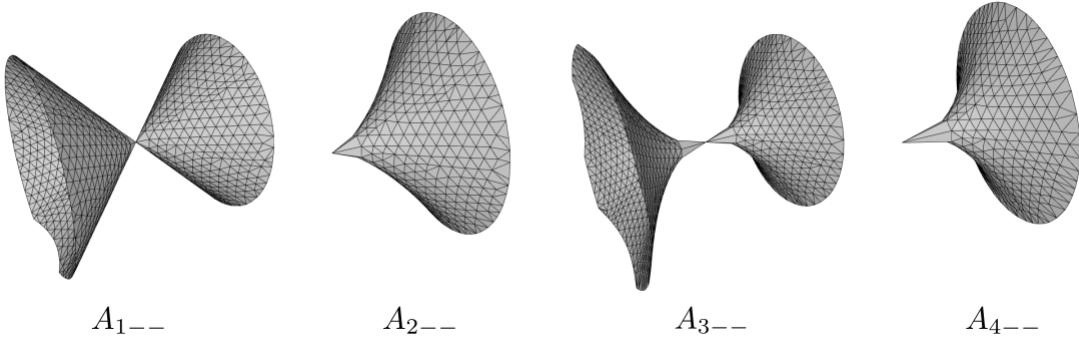


Figure 4.1: Resulting uniform triangulation of  $A_{n--}$  singularities without layers.

## 4.2 Comparison with SingSurf

SingSurf [13] 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 [14].

### 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 meshes from our algorithm can be seen on the Figure 4.1. The resulting meshes generated by SingSurf can be seen on the Figure 4.2. By the look, one can tell, that the meshes produced by us have more equilateral triangles. On the other hand, the meshes produced by SingSurf approximate the surface better around the singular points.

The comparison on the quality criteria 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.

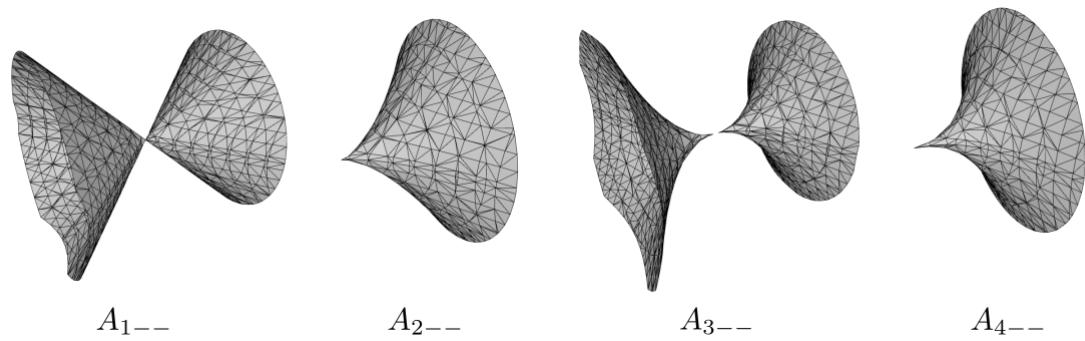


Figure 4.2: Resulting triangulation of  $A_{n--}$  singularities by SingSurf [13].

### 4.3 Computational speed comparison of the reimplemented solution

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	0.840	0.007	0.001	0.010
	Adaptive algorithm	0.000	0.000	0.000	0.000
$A_{2--}$	SingSurf	0.077	0.008	0.001	0.051
	Uniform algorithm	0.840	0.007	0.001	0.010
	Adaptive algorithm	0.000	0.000	0.000	0.000
$A_{3--}$	SingSurf	0.049	0.012	0.001	0.048
	Uniform algorithm	0.833	0.013	0.001	0.011
	Adaptive algorithm	0.000	0.000	0.000	0.000
$A_{4--}$	SingSurf	0.325	0.020	0.001	0.045
	Uniform algorithm	0.810	0.021	0.001	0.014
	Adaptive algorithm	0.000	0.000	0.000	0.000

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	0.835	0.008	0.002	0.011
	Adaptive algorithm	0.000	0.000	0.000	0.000
$A_{3+-}$	SingSurf	0.001	0.042	0.005	0.073
	Uniform algorithm	0.824	0.013	0.002	0.015
	Adaptive algorithm	0.000	0.000	0.000	0.000
$A_{4+-}$	SingSurf	0.001	0.061	0.006	0.118
	Uniform algorithm	0.820	0.033	0.006	0.031
	Adaptive algorithm	0.000	0.000	0.000	0.000

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	0.794	0.024	0.002	0.009
	Adaptive algorithm	0.000	0.000	0.000	0.000
$D_{4+-}$	SingSurf	0.094	0.028	0.003	0.078
	Uniform algorithm	0.855	0.022	0.002	0.019
	Adaptive algorithm	0.000	0.000	0.000	0.000
$D_{5--}$	SingSurf	0.001	0.032	0.006	0.076
	Uniform algorithm	0.789	0.029	0.005	0.022
	Adaptive algorithm	0.000	0.000	0.000	0.000
$D_{5+-}$	SingSurf	0.132	0.029	0.003	0.076
	Uniform algorithm	0.831	0.021	0.002	0.020
	Adaptive algorithm	0.000	0.000	0.000	0.000

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	0.858	0.017	0.002	0.014
	Adaptive algorithm	0.000	0.000	0.000	0.000
$E_{6+-}$	SingSurf	0.011	0.040	0.003	0.077
	Uniform algorithm	0.403	0.021	0.002	0.019
	Adaptive algorithm	0.000	0.000	0.000	0.000
$E_{7++}$	SingSurf	0.198	0.024	0.004	0.110
	Uniform algorithm	0.803	0.059	0.004	0.030
	Adaptive algorithm	0.000	0.000	0.000	0.000
$E_{8++}$	SingSurf	0.260	0.045	0.004	0.100
	Uniform algorithm	0.828	0.044	0.004	0.029
	Adaptive algorithm	0.000	0.000	0.000	0.000

# Chapter 5

## Future work



# Conclusion

TODO Conclusion



# Bibliography

- [1] 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.
- [2] Vladimir Igorevich Arnol'd. Normal forms for functions near degenerate critical points, the weyl groups of a k, d k, e k and lagrangian singularities. *Functional Analysis and its applications*, 6:254–272, 1972.
- [3] 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.
- [4] Evgenii Borisovich Dynkin. The structure of semi-simple algebras. *Uspekhi Matematicheskikh Nauk*, 2(4):59–127, 1947.
- [5] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, and John F Hughes. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996.
- [6] Ron Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005.
- [7] M Hazewinkel, W Hesselink, D Siersma, and FD Veldkamp. The ubiquity of coxeter-dynkin diagrams. *Nieuw Arch. Wisk.*, 25:257–307, 1977.
- [8] 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.
- [9] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1):65–90, 1999.
- [10] Jan J Koenderink and Andrea J Van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992.

- [11] PhD. Kristína Korecová, doc. RNDr. Pavel Chalmovianský. Algoritmy triangulácie implicitne definovanej plochy. 2021.
- [12] 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.
- [13] Dr. Richard Morris. 3d viewer-generator.
- [14] 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.
- [15] TIAGO NOVELLO, VINÍCIUS DA SILVA, GUILHERME SCHARDONG, LUIZ SCHIRMER, HÉLIO LOPES, and LUIZ VELHO. Differential geometry of implicit surfaces. 2021.
- [16] Aristides Requicha and Robert Tilove. Mathematical foundations of constructive solid geometry: General topology of closed regular sets. 1978.
- [17] Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. 1977.
- [18] Michael Spivak. *A comprehensive introduction to differential geometry*, volume 3. Publish or Perish, Incorporated, 1975.
- [19] 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

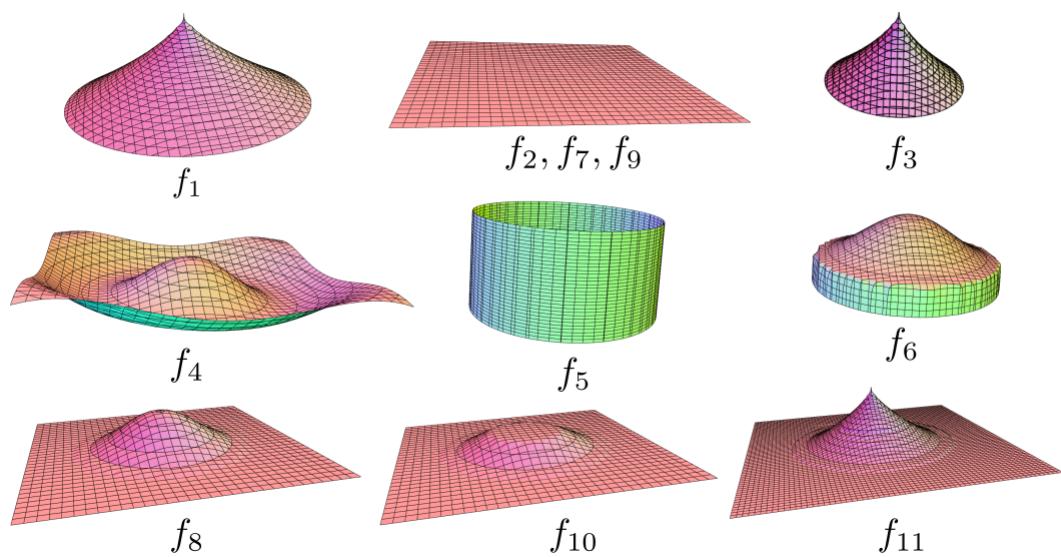


Figure 5.1: Attaching the singularity to a plane using CSG.