

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

TRIANGULATION OF IMPLICITLY DEFINED
SURFACES
MASTER'S THESIS

2023
BC. KRISTÍNA KORECOVÁ

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

TRIANGULATION OF IMPLICITLY DEFINED
SURFACES
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, 2023
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



Univerzita Komenského v Bratislavе
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

- Meno a priezvisko študenta:** Bc. Kristína Korecová
Študijný program: počítačová grafika a geometria (Jednoodborové štúdium, magisterský II. st., denná forma)
- Študijný odbor:** matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský
- Názov:** Triangulation of implicitly defined surfaces
Triangulácia implicitne definovaných ploch
- Anotácia:** Študent navrhuje algoritmus triangulácie implicitne definovanej plochy. Začína sa s trianguláciou zo singulárnych bodov. Reguľárne časti plochy sa triangulujú adaptívne a rovnomerne. Výsledná plocha môže byť ďalej lokálne optimalizovaná. Študentka by mala navrhnúť spôsob numerického výpočtu singulárnych bodov aspoň v prípade algebraických plôch nízkeho stupňa.
- Ciel:** Vypočítať trianguláciu implicitne definovanej plochy a porovnať výsledky s existujúcimi algoritmami z hľadiska kvality a výpočtovej sily navrhovaného algoritmu.
- Literatúra:** 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
- Kľúčové slová:** implicitne definovaná plocha, triangulácia, numerický algoritmus
- Vedúci:** doc. RNDr. Pavel Chalmovianský, PhD.
Katedra: FMFI.KAG - Katedra algebry a geometrie
Vedúci katedry: doc. RNDr. Pavel Chalmovianský, PhD.
- Spôsob sprístupnenia elektronickej verzie práce:**
 prípustná pre vlastnú VŠ
- Dátum zadania:** 19.10.2021
- Dátum schválenia:** 20.10.2021
- prof. RNDr. Július Korbaš, CSc.
 garant študijného programu



Univerzita Komenského v Bratislavě
Fakulta matematiky, fyziky a informatiky

študent

vedúci práce

Acknowledgements: In the first place, I want to thank my supervisor, doc. RNDr. Pavel Chalmovianský, PhD. for answering my questions and providing me with valuable feedback. I also want to thank my partner, Andrej, for always being my (not only technical) support and for the involuntary listening to my geometry lectures. Next, I want to thank my family, especially my mom and grandma, for the support and the best meals. Last but not least, I want to thank my classmates - Lucka, Bea and Mima for listening to my never-ending complaints.

Abstrakt

Implicitne zadané plochy sú plochy definované ako nulová hladina funkcie troch premenných. Aproximácia plochy trojuholníkovým meshom – triangulácia je častý spôsob digitálnej reprezentácie plochy. Pre množstvo algoritmov, ktoré vytvárajú trianguláciu plochy, vznikajú problémy ak plocha obsahuje singularity – body v ktorých je gradient definujúcej funkcie nulový vektor. Niektoré algoritmy ignorujú singulárne body, čo vedie k nekorektnej aproximácií plochy. V tejto práci vytvárame algoritmus na trianguláciu plôch zadaných implicitne obsahujúcich niektoré typy singularít. Pritom sa zameriavame na korektnosť triangulácie a kvalitu meshu. Rozširujeme algoritmus implementovaný v našej bakalárskej práci založený na algoritme Marching Triangles tak, aby umožnil trianguláciu implicitne zadaných plôch obsahujúcich ADE singularity a krivkové singularity na prieniku dvoch regulárnych plôch. Navyše navrhujeme postup na vytvorenie triangulácie adaptívnej na lokálnu krivost' plochy. Nakoniec reimplementujeme algoritmus pre zrýchlenie času behu algoritmu.

Kľúčové slová: implicitne definovaná plocha, triangulácia, numerický algoritmus

Abstract

Implicit surfaces are surfaces defined as a zero set of a function. Approximation of a surface by a triangular mesh – triangulation is a common way of digital surface representation. For most algorithms constructing the triangulation of a surface a problem arises if the surface contains singularities – the points at which the gradient of the defining function vanishes. Some algorithms ignore the singular points, which results in the incorrect approximation of the surface. In this thesis, we create an algorithm for the triangulation of the surfaces with certain types of singularities. We focus on the correctness of the triangulation and the quality of a mesh. We are extending the algorithm implemented in our bachelor's thesis based on the Marching Triangles to allow triangulation of the implicit surfaces containing ADE singularities and curve singularities on the intersection of two regular surfaces. Moreover, we propose an approach for creating a triangulation adaptive to the local curvature of a surface. Lastly, we reimplement the algorithm to speed up its runtime.

Keywords: implicitly defined surface, triangulation, computational approach

Contents

Introduction	1
1 Theoretical background	3
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 $SO(3)$ group and ADE singularities . .	9
1.3 CSG modelling for implicit surfaces	10
1.3.1 Constructive solid geometry (CSG)	10
1.4 Non-isolated surface singularities	12
2 Mathematical model of proposed triangulation	15
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	23
2.2.3 Numerical calculation of local triangulation of ADE singularities	29
2.2.4 Triangulation of a plane with multiple A_{n--} singularities	35
2.3 Triangulation of non-isolated singularities	39
2.3.1 Creating the local mesh around the singular curves	41
2.3.2 Modification of triangulation for union and difference	43
3 Implementation	45
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	48
3.2.3 Mesh structure	48
3.3 Algorithm runtime	49

4 Results	51
4.1 Quality criteria	51
4.2 Comparison with SingSurf	53
4.2.1 SingSurf algorithm	53
4.2.2 Evaluated quality criteria	54
4.3 Curve singularities	61
4.3.1 Intersection	61
4.3.2 Union	63
4.3.3 Difference	64
4.4 Triangulation of a plane with A_{n--} singularities	67
4.5 Runtime comparison of the reimplemented solution	68
Conclusion	71
Appendix A	75
Appendix B	77

List of Figures

1.1	Implicit surfaces with corresponding equations	3
1.2	Isolated and non-isolated singularities of an implicitly defined surfaces	4
1.3	Normal cut	5
1.4	Categorization of the surface points based on the Gaussian curvature	6
1.5	Visualisation of the curvatures of the double-torus	7
1.6	Shapes representing the finite subgroups of $SO(3)$ group	10
1.7	Example of CSG tree	11
1.8	Boolean operations on implicit curves	12
1.9	Intersection of a sphere and a plane	12
1.10	Intersection of two balls	13
1.11	Projecting a point to the implicit curve	13
2.1	Adaptive height of the new triangle	16
2.2	Creating the point C	17
2.3	Adaptive triangulation of a torus	18
2.4	A_{n--} singularities	19
2.5	A_{n+-} singularities	20
2.6	D_{n+-} singularities	21
2.7	D_{n--} singularities	21
2.8	Intersection of D_{n--} singularities with the plane $z = 0$	21
2.9	Triangulation vectors for two branches of D_{n--} singularities	22
2.10	E_n singularities	22
2.11	Intersection of E_{7++} and E_{8++} singularities with the plane $z = 0$	23
2.12	Triangulation of A_{n--} singularity	24
2.13	Resulting uniform triangulation of the A_{5--} singularity	24
2.14	Three layers of triangles for the A_{1--} singularity	25
2.15	Local mesh with layers for the A_{5--} singularity	25
2.16	Resulting mesh for the A_{5--} singularity	26
2.17	Equilateral octagon inscribed inside an ellipse	27
2.18	Calculating the point P_h	27

2.19 Notation used in the proof	28
2.20 1st category singularities	29
2.21 2nd category singularities	30
2.22 Binary search for the point on the surface	30
2.23 Double binary search for the point on the surface	31
2.24 Rotating the plane about the vector	32
2.25 Triangulation of 2nd category singularities	33
2.26 Insufficient and sufficient local approximation	34
2.27 Incorrect and correct local mesh	34
2.28 Uneven triangles in the local mesh around D_{4+-} singularity	35
2.29 Optimized local mesh around D_{4+-} singularity	35
2.30 Plane with singularities	36
2.31 C^1 – continuous cosine bump function – B_{\cos}	37
2.32 Construction of the function B_{\cos} using CSG	38
2.33 Attaching the singularity to a plane using the function B_{\cos}	39
2.34 Attaching the singularity to a plane using CSG	40
2.35 Plane with multiple attached singularities	40
2.36 Approximation of the implicit curve by polyline	42
2.37 Definition of the points M_i , M_i^1 and M_i^2	42
2.38 Definition of the points R_i^1 and R_i^2	43
2.39 Local mesh around the singular curve	43
2.40 Definition of the points for union	44
2.41 Definition of the points for difference	44
3.1 Projecting the point on the surface	46
3.2 Visualisation of the half-edge data structure	47
3.3 Example of half-edge representation	47
4.1 Resulting uniform triangulation of A_{n--} singularities	54
4.2 Resulting adaptive triangulation of A_{n--} singularities	54
4.3 Resulting triangulation of A_{n--} singularities by SingSurf	55
4.4 Resulting uniform triangulation of A_{n+-} singularities	56
4.5 Resulting adaptive triangulation of A_{n+-} singularities	56
4.6 Resulting triangulation of A_{n+-} singularities by SingSurf	56
4.7 Resulting uniform triangulation of D_n singularities	58
4.8 Resulting adaptive triangulation of D_n singularities	58
4.9 Resulting triangulation of D_n singularities by SingSurf	58
4.10 Resulting uniform triangulation of E_n singularities	60
4.11 Resulting adaptive triangulation of E_n singularities	60
4.12 Resulting triangulation of E_n singularities by SingSurf	60

4.13	The surface given by the intersection of a sphere and a hyperboloid	62
4.14	The surface given by the intersection of a sphere and a tangle-cube	62
4.15	The surface given by the intersection of a blobby object and a plane	62
4.16	The surface given by the intersection of a sphere and a plane	63
4.17	The surface given by the union of a sphere and a hyperboloid	63
4.18	The surface given by the union of two spheres	64
4.19	The surface given by the union of a sphere and a tangle-cube	64
4.20	The surface given by the difference of two spheres	65
4.21	The surface given by the difference of a sphere and a tangle-cube	65
4.22	The surface given by the difference of a tangle-cube and a sphere	66
4.23	The triangulation of the A_{2--} singularity	67
4.24	The triangulation of the two A_{2--} singularities	67
4.25	The triangulation of A_{2--} and A_{4--} singularities	67
4.26	The triangulation of A_{1--} , A_{2--} , A_{3--} and A_{4--} singularities	68
4.27	The dependence of the runtime on the input edge length for the unit sphere	69
4.28	The dependence of the runtime on the input edge length for the unit sphere (logarithmic scale)	69
4.29	The dependence of the runtime on the number of generated triangles	70
4.30	The dependence of the runtime on the number of generated triangles (logarithmic scale)	70
4.31	Attaching the singularity to a plane using CSG	76
4.32	Triangulated surfaces containing ADE singularities	77

List of Tables

2.1	Implicit equations for bump function modelling	37
3.1	Edge table of a half-edge data structure for the Figure 3.3.	47
3.2	Vertex table of a half-edge data structure for the Figure 3.3.	48
3.3	Face table of a half-edge data structure for the Figure 3.3.	48
4.1	Quality criteria – A_{n--} singularities	55
4.2	Quality criteria – A_{n+-} singularities	57
4.3	Quality criteria – D_n singularities	59
4.4	Quality criteria – E_n singularities	61

Introduction

In mathematics, an implicit surface in \mathbb{R}^3 is a set of points where a given function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ is zero. Due to a unification of constructive solid geometry modelling and implicit surfaces, one can 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 than calculating the intersection of a ray and an implicit surface.

Many surfaces contain singular points where all three partial derivatives vanish. These points may cause problems for meshing algorithms. Some meshing algorithms, such as Marching Cubes [15] ignore these points, which leads to insufficient surface representation.

In this thesis, we follow up on our effort in our previous work [14] to present an algorithm for creating the triangular mesh of regular implicit surfaces.

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

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

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

In the chapter 4 we compare the quality of the meshes with the SingSurf software [17]. We also compare the runtime of the reimplemented algorithm with the implementation presented in our previous work [14].

Chapter 1

Theoretical background

1.1 Implicit surfaces

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

Implicit functions do not define the surface explicitly, i.e. parametrically. Instead, the surface is defined as a zero set of a function and the parametrization is guaranteed in a neighbourhood of a regular point.

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

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

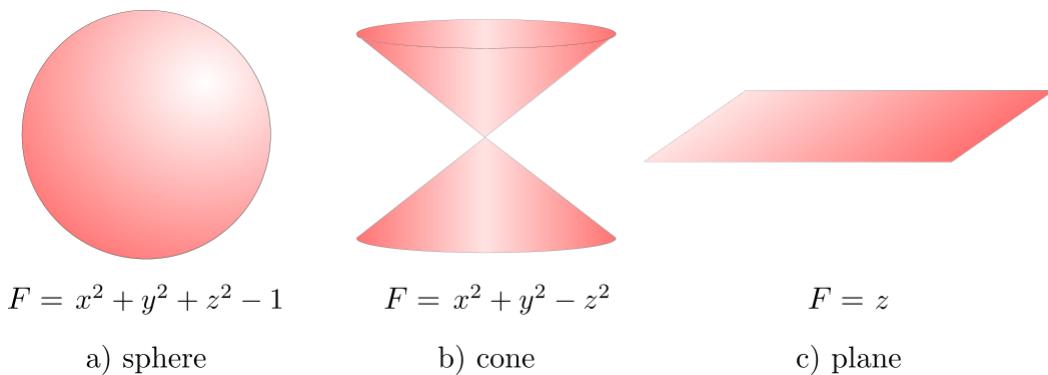


Figure 1.1: Implicit surfaces with corresponding equations.

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

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

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

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

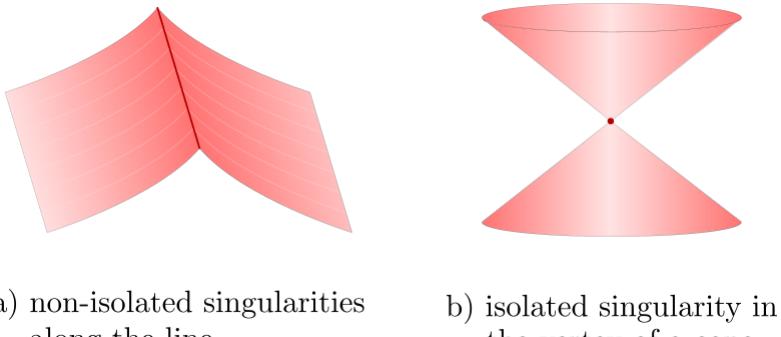
The points lying on an implicit surface $Z(F)$ can be classified as regular or singular based on the value of the gradient vector $\nabla F(x, y, z)$ at that point.

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

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

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

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



a) non-isolated singularities along the line b) isolated singularity in the vertex of a cone

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

1.1.1 Curvature of a surface

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

For surfaces, the curvature is a measure of how much does the surface differ from a plane. The definition of the curvature of a surface is more complex than in the case

of curves. The curvature of a curve on a surface depends on the choice of the tangent direction of a measured point.

The idea of measuring the curvature of a surface has a long history in mathematics. One of the first contributors was the mathematician Carl Friedrich Gauss, who developed the idea of the Gaussian curvature of surfaces. In this subsection, we draw from the summary presented by Tiago Novello et al.[18].

Normal curvature of the surface

Let S be a parametric surface defined on the domain $D \subseteq \mathbb{R}^2$

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

$$S(u, v) = (S_x(u, v), S_y(u, v), S_z(u, v)), \quad (u, v) \in D.$$

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 P on the surface given by parameters u and v and the unit tangent vector in that point \vec{t} .

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

The visualization of the normal cut is shown in the Figure 1.3. It is clear that the normal cut is a plane curve lying on the surface.

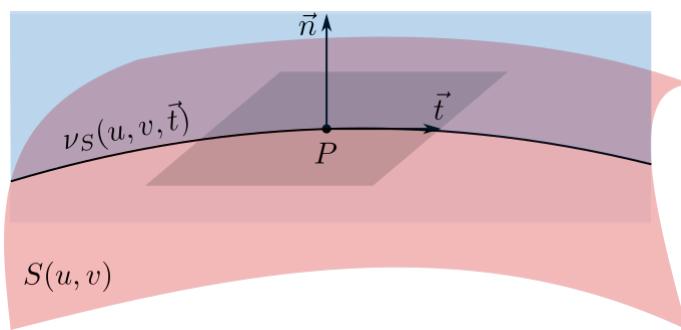


Figure 1.3: Normal cut of the parametric surface $S(u, v)$.

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

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

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

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

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

Minimal and maximal curvatures are called principal curvatures.

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

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

Gaussian curvature describes the shape of the surface in the local neighbourhood of the point.

Points of the surface S , where the Gaussian curvature is positive, are called elliptic.

Points of the surface S , where the Gaussian curvature is negative, are called hyperbolic.

Points of the surface S , where only one of $\kappa_{\min}, \kappa_{\max}$ is zero, are called parabolic.

Points, where both κ_{\min} and κ_{\max} are zero, are called planar.

The shape of the surface in the local neighbourhoods of the points, visualized in the Figure 1.4, is as follows:

- elliptic points – surface is locally curved like an ellipsoid,
- hyperbolic points – surface is not locally convex,
- parabolic points – surface is locally curved like a parabolic cylinder,
- planar points – surface is locally flat like a plane.

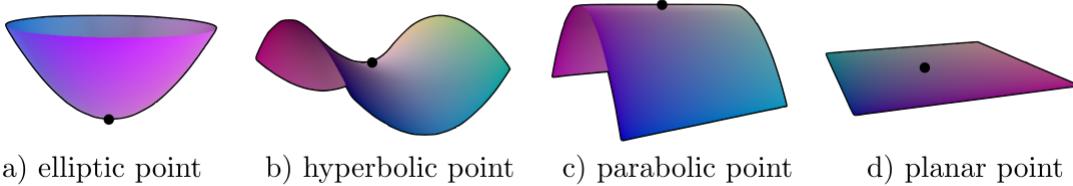


Figure 1.4: Categorization of the surface points based on the Gaussian curvature [17].

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

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

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

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

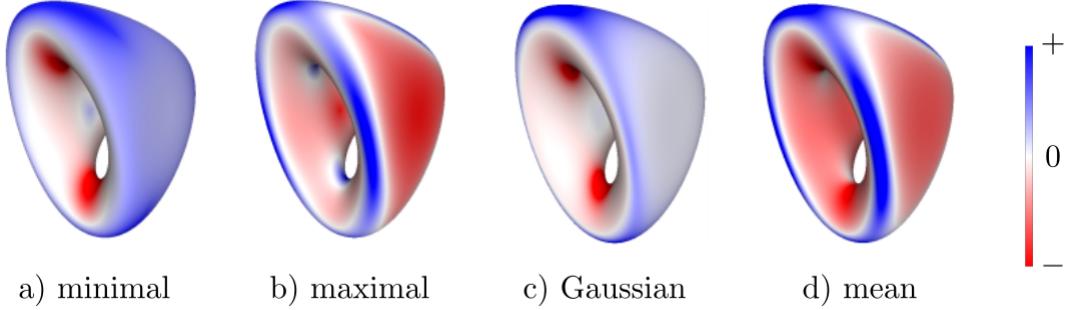


Figure 1.5: Visualization of minimal, maximal, Gaussian and mean curvatures of the double-torus [18]. Blue/white/red colour corresponds to the positive/zero/negative value of the corresponding curvature.

1.1.2 Curvature formulas for implicit surface

A version of curvature formulas for implicit surfaces appeared in [21] and were reformulated, summarized and proved by Ron Goldman in [9]. In this subsection, we point out these formulas.

Let $F : \mathbb{R}^3 \rightarrow R$ 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 $H(F)$ – 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 $H^*(F)$ 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 $Z(F)$ at a regular point.

Gaussian curvature κ_G is given by

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

The mean curvature κ_M 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 κ_{min} and κ_{max} can be calculated from the Gaussian curvature κ_G and the mean curvature κ_M 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 [4] according to ADE classification [10] based on correspondence of these singularities to simply laced Dynkin diagrams [7]. 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:

- $A_{n\pm\pm} \quad F(x, y, z) = x^{n+1} \pm y^2 \pm z^2,$

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

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

1.2.1 Correspondence between $SO(3)$ group and ADE singularities

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

Definition 11 $SO(3)$ is a group of 3×3 orthogonal matrices of real numbers with determinant 1.

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

Simply laced Dynkin diagrams correspond to all finite subgroups of $SO(3)$. Finite subgroups of $SO(3)$ 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:

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

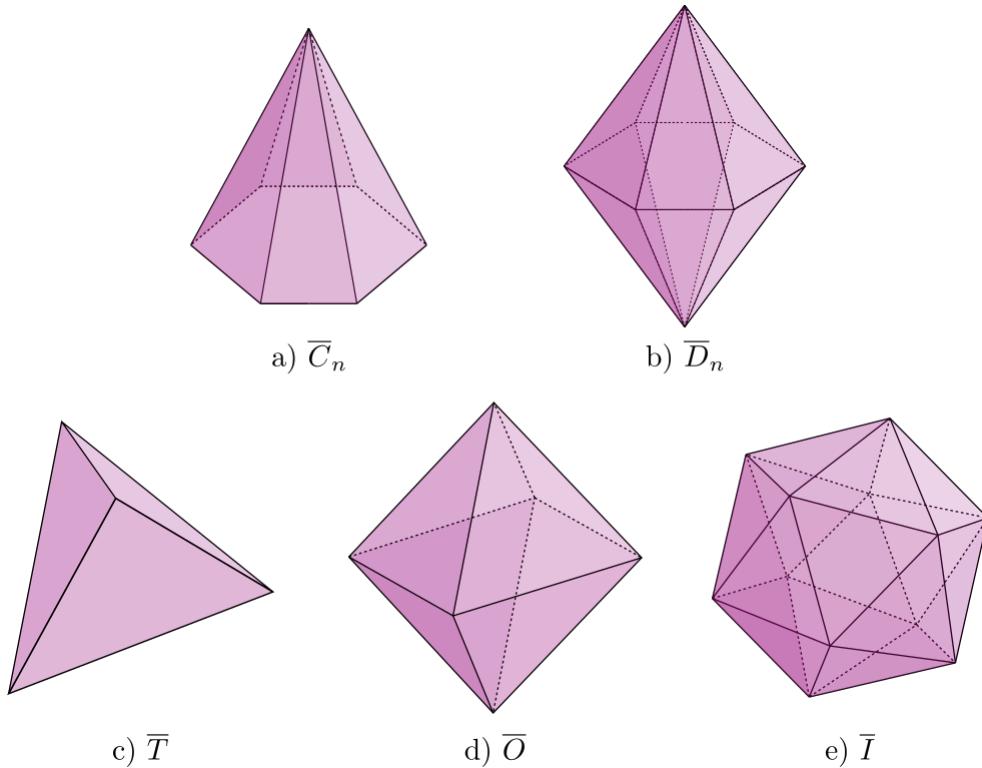


Figure 1.6: Shapes representing the finite subgroups of $SO(3)$ group.

- $E_8 \iff \overline{I}$.

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

1.3 CSG modelling for implicit surfaces

This section will discuss how constructive solid geometry can be used for modelling complex implicit surfaces. The first major publication regarding constructive solid geometry was published in 1977 by Requicha and Voelcker [20]. More detailed mathematical foundations were published a year later, in 1978, by Requicha and Tilove [19].

1.3.1 Constructive solid geometry (CSG)

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

boolean operations in its internal nodes [8]. An example of such a CSG tree can be seen in the Figure 1.7.

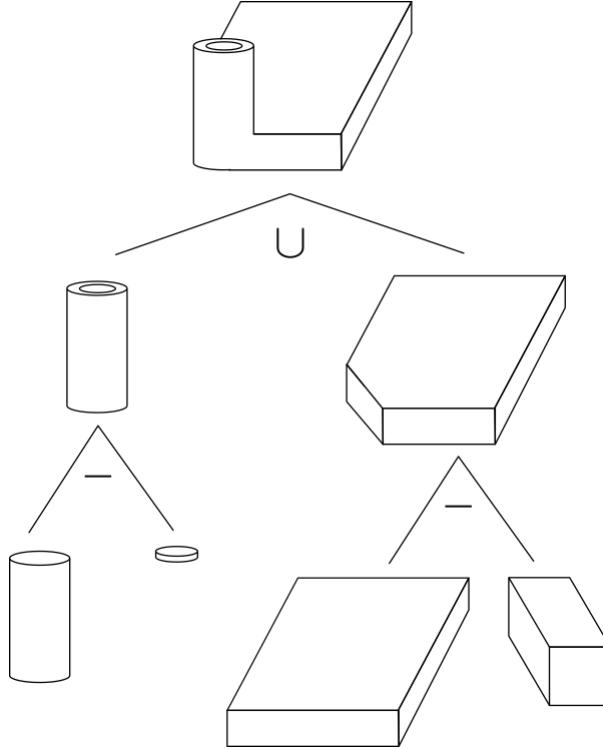


Figure 1.7: Example of CSG tree [8].

CSG for implicit surfaces

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

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

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

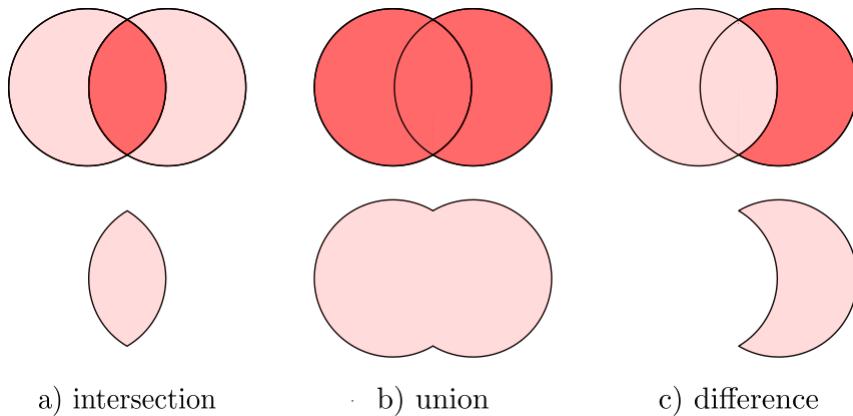


Figure 1.8: Boolean operations on implicit curves.

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

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

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

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

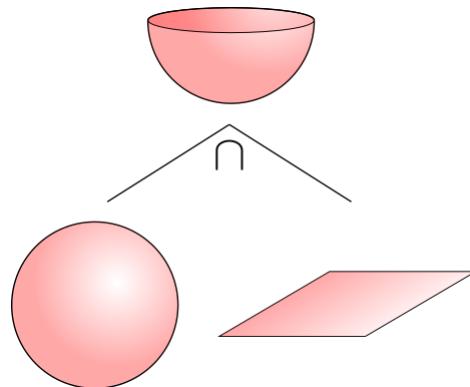


Figure 1.9: Intersection of a sphere and a plane.

1.4 Non-isolated surface singularities

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

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

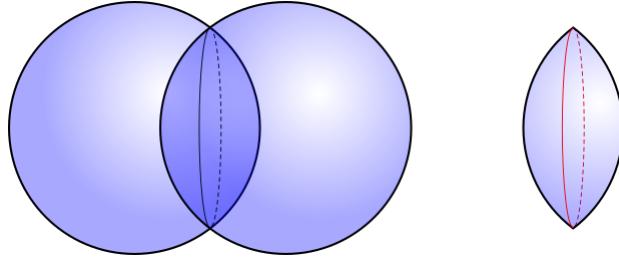


Figure 1.10: Intersection of two balls.

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 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 the required precision. A visualization of this approach can be seen in the Figure 1.11. The projection of a point to the curve C is denoted proj_C .

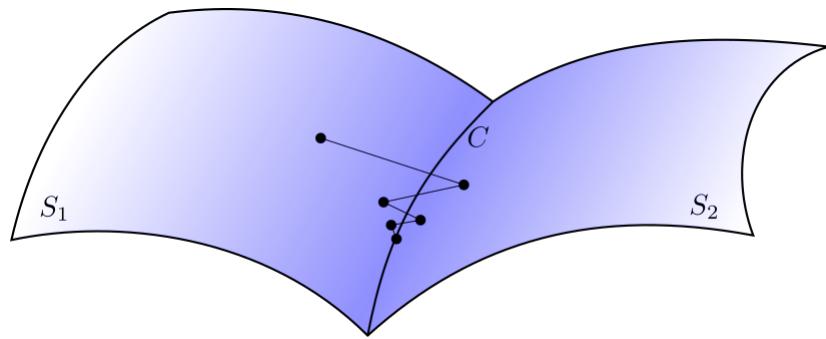


Figure 1.11: Projecting a point to the implicit curve.

Tangent vector of the implicit curve

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 intersect in the implicit curve C .

The unit tangent vector \vec{t}_C of the curve C in the point $P \in C$ is given by the cross product of unit normal vectors of the surfaces S_1 and S_2

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

Chapter 2

Mathematical model of proposed triangulation

Ordered triplet of sets (V, E, F) , where V is the set of vertices, E is the set of edges given by two vertices, and F is a set of faces given by three vertices is called triangular mesh.

The triangulation of a surface is an approximation of the surface by a triangular mesh.

A correct triangulation should approximate the surface precisely enough and be topologically equivalent to the surface. Moreover, a quality triangulation should have triangles as equilateral as possible and adapt the size of the triangles to the local curvature of the surface.

Two of the best-known approaches when triangulating the implicit surface are Marching Cubes [15] and Marching Triangles [11]. While Marching Cubes is a fast algorithm producing non-quality mesh, the Marching Triangles produces quality mesh for the cost of slower algorithm runtime.

In [14], we implemented an algorithm based on the Marching Triangles approach. In this chapter, we introduce the modification of the algorithm to enable adaptive triangulation of surfaces with some singular points and curves.

2.1 Triangulation adaptive to the local curvature

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

A triangulation of a 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 where the surface is curved and bigger triangles where the surface is flatter.

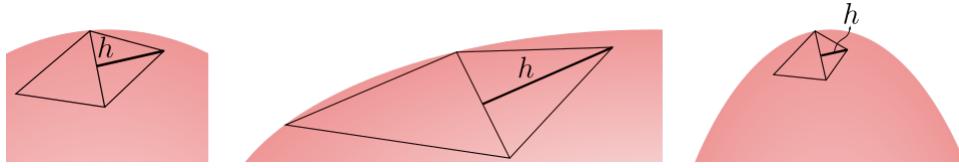


Figure 2.1: Adaptive height of the new triangle.

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

In the original algorithm, the height h of the triangle which is projected to the surface is set to the constant value $\frac{\sqrt{3}}{2}e$, where e is the required length of the side of the triangle. To achieve the adaptivity of the size of the triangles, we set the height h 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 Kendering and Doorn in 1992 [13]. As the authors explained, the Gaussian curvature and the mean curvature are not very descriptive of the local shape of the surface. Principal curvatures taken as a pair are more informative. The curvedness was introduced to measure the curvature in the given point by a single number instead of a pair of numbers. The authors defined curvedness as

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

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

Curvedness has some obvious properties:

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

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

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

- **Sphere:** As the principal curvatures of a sphere are both constant and equal to the inverse of the radius of the sphere, curvedness c 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 c is therefore equal to half of the inverse of the radius of the cylinder.

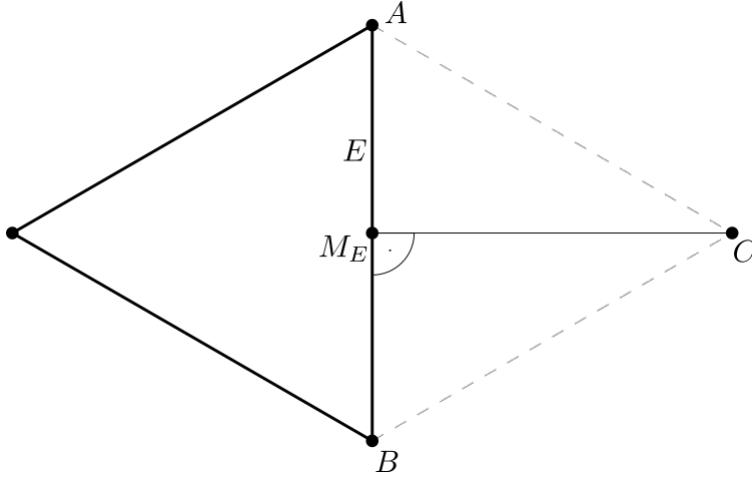


Figure 2.2: Creating the point C for the given edge E .

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

For the triangulation adaptive to the local curvature of the surface, an input size of the edge e is not perceived as the approximate size of the triangulation triangle. Instead, it is perceived as a measure of the level of detail.

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 the curvedness radius r_c in that point. A maximum edge size is set to avoid the edge case in the planar point.

During the run of the algorithm, for a given edge E with the length l_E , the point C is created as displayed in the Figure 2.2. The point C lies in the plane of the incident triangle of the edge E on the line perpendicular to E and passing through M_E – the midpoint of the edge E . The distance between M_E and C is the height of the equilateral triangle with the edge $E \rightarrow |M_E C| = \frac{\sqrt{3}}{2} |E|$.

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

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

We calculate the edge length for the given constant of detail k_d as $\frac{r_c}{k_d}$ which is equal to $\frac{e}{c}$, where e is the size of the edge on the input and c is curvedness in the point P_E .

The new point is then created on the same line perpendicular to the edge E in the distance $\frac{\sqrt{3}e}{2c}$ from the point M_E .

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

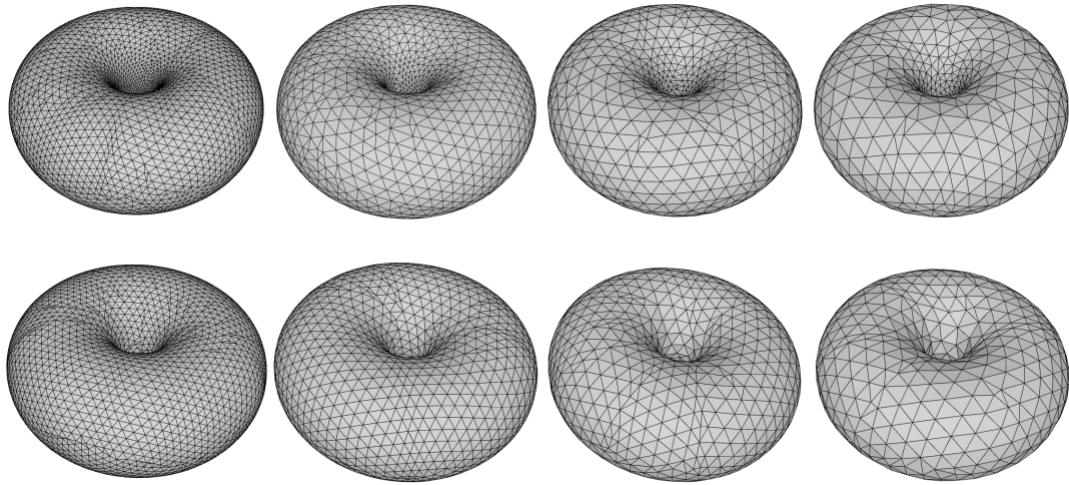


Figure 2.3: Adaptive triangulation of a torus with four different constants of detail (top row). Uniform triangulation of a torus with four different constants of detail (bottom row).

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

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 $A_{1\dots}$ singularity is locally represented as a cone. This section discusses the geometric structure of other ADE surface singularities.

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

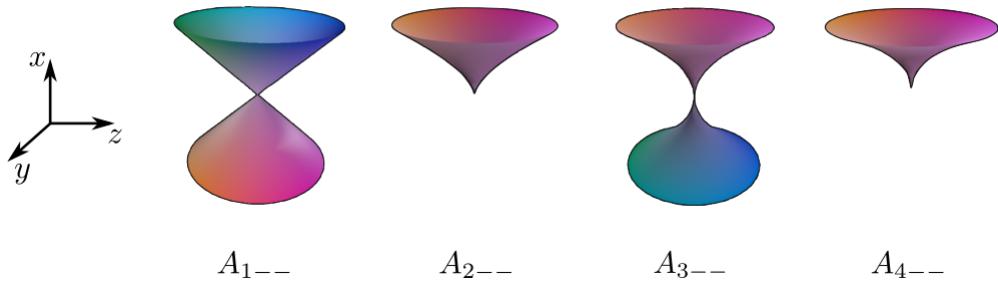


Figure 2.4: A_{n--} singularities. For more examples, see [17].

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

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

A_n singularities

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

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

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

If n is even, A_{n+-} singularities are topologically equivalent to a plane with a shape similar to a hyperbolic paraboloid, therefore, they have a single branch. The first four A_{n+-} singularities can be seen in the Figure 2.5. For this case, we pick the vector

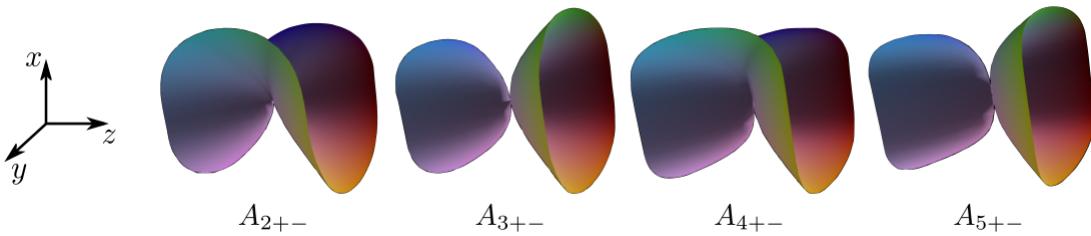


Figure 2.5: A_{n+-} singularities. For more examples, see [17].

$(1, 0, 0)$ 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 eight categories locally represented by the following equations:

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

Now, we look at some equivalences between these eight categories.

- D_{4++} singularity is reflected D_{4+-} singularity.
- D_{5++} singularity is reflected D_{5--} singularity.
- D_{5-+} singularity is reflected D_{5+-} singularity.
- D_{4-+} singularity is reflected D_{4--} singularity.

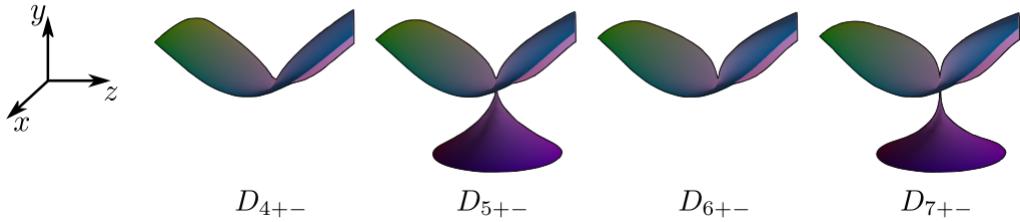


Figure 2.6: D_{n+-} singularities. For more examples, see [17].

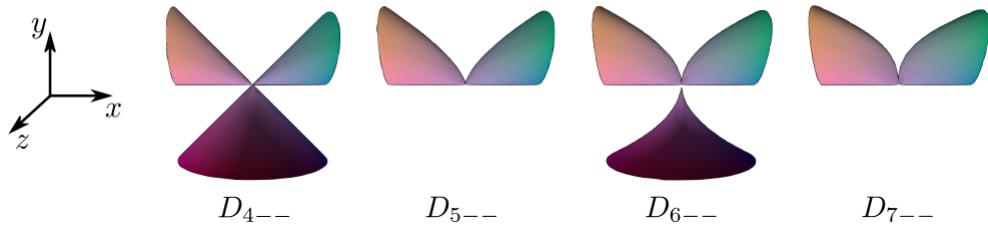


Figure 2.7: D_{n--} singularities. For more examples, see [17].

All of these reflections are the reflections by the xz plane. We therefore only analyze geometry of D_{n+-} singularities and D_{n--} singularities.

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

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

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

For D_{n--} singularity, the intersections of the two branches where $y \geq 0$ are bounded

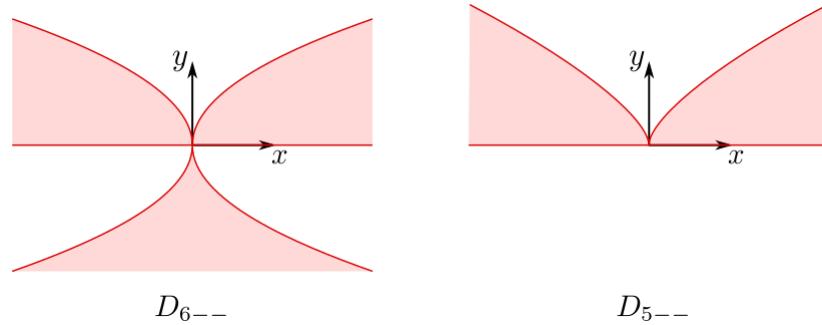


Figure 2.8: Intersection of D_{n--} singularities with the plane $z = 0$.

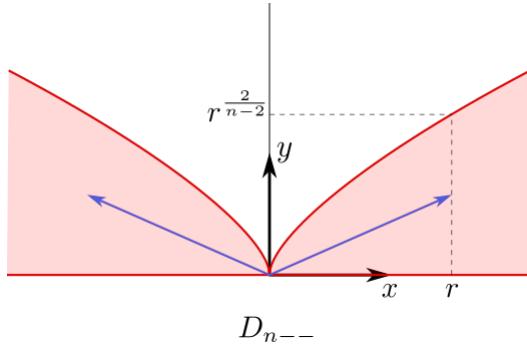


Figure 2.9: Triangulation vectors for two branches of D_{n--} singularities.

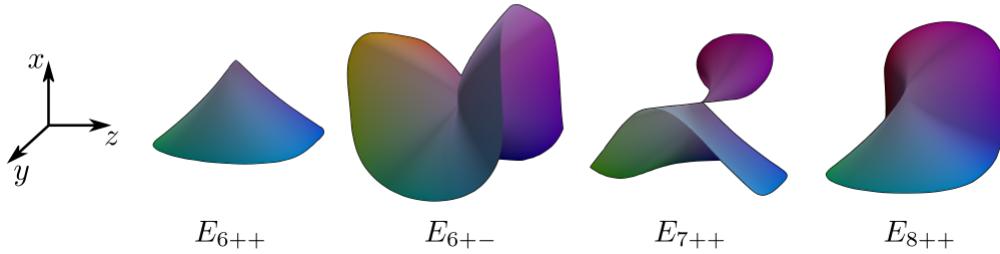


Figure 2.10: E_n singularities. For more examples, see [17].

by curves $y = 0$ and $x^2 = y^{n-2}$. For given r , we pick the triangulation vectors as $(r, \frac{1}{2}r^{\frac{2}{n-2}}, 0)$ and $(-r, \frac{1}{2}r^{\frac{2}{n-2}}, 0)$. The resulting vectors are displayed in the Figure 2.9 by blue arrows. The parameter r is picked based on the length of the edge of the 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 $F(x, y, z) = x^3 \pm y^5 \pm z^2$, one can see the following equivalences:

- E_{6++} singularity is reflected E_{6--} singularity.
- E_{6+-} singularity is reflected E_{6-+} singularity.
- E_{7+-}, E_{7-+} and E_{7--} are all reflected E_{7++} singularity.
- E_{8+-}, E_{8-+} and E_{8--} are all reflected E_{8++} singularity.

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

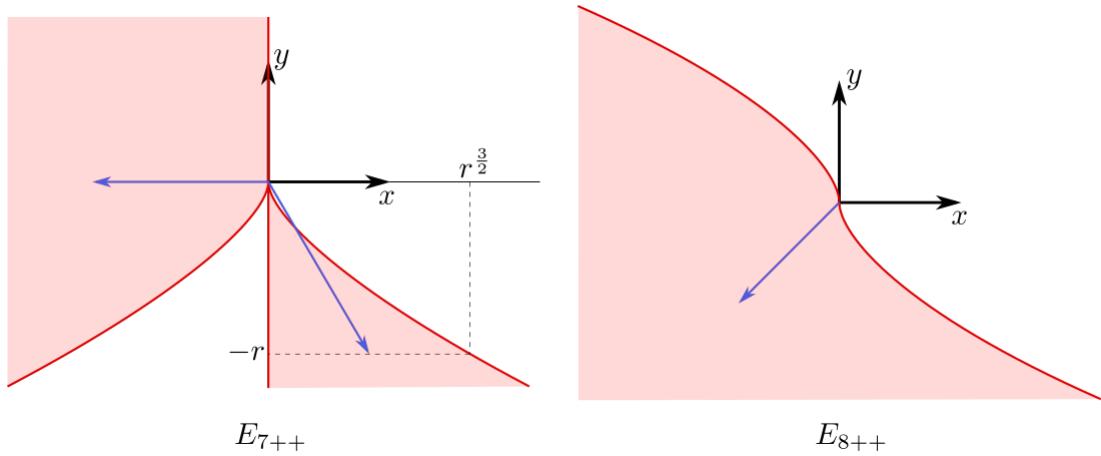


Figure 2.11: Intersection of E_{7++} and E_{8++} singularities with the plane $z = 0$.

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

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

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

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

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

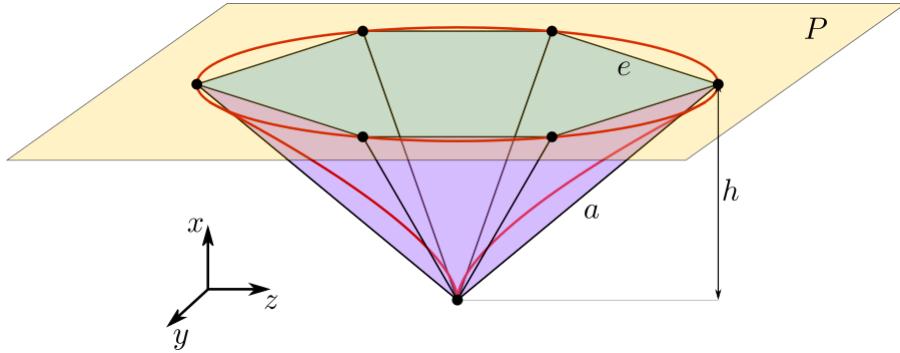
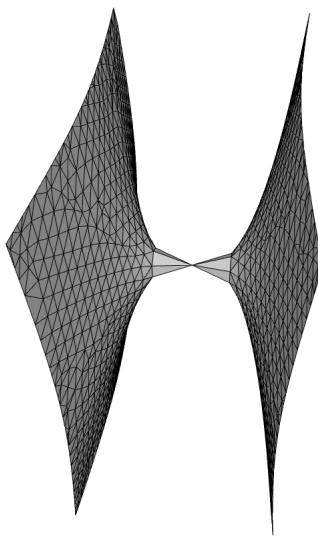
2.2.2 Analytical calculation of local triangulation of some ADE singularities

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

A_{n--} singularities

For A_{n--} singularities, we create a disc of six isosceles triangles with a vertex in the singular point. The bases of these triangles create regular hexagon in the plane P parallel to the plane $x = 0$, as showed in the Figure 2.12. Given the 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

Figure 2.12: Triangulation of A_{n--} singularity.Figure 2.13: Resulting uniform triangulation of the A_{5--} singularity.

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

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

The resulting mesh with uniform triangles for A_{5--} singularity is displayed in the Figure 2.13.

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

Given the edge length e and number of layers, we calculate the height h_e at which the bases of the triangles have the length e . We divide the height by the number of

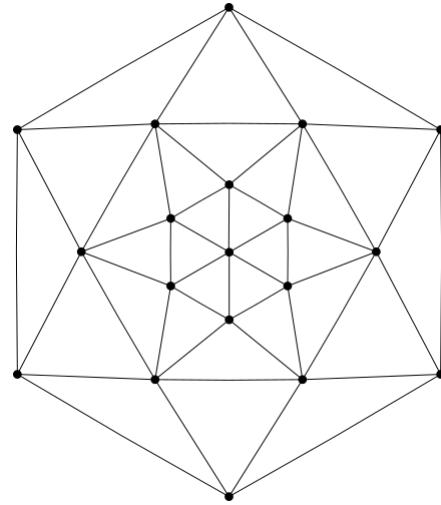


Figure 2.14: Three layers of triangles for the A_{1--} singularity.

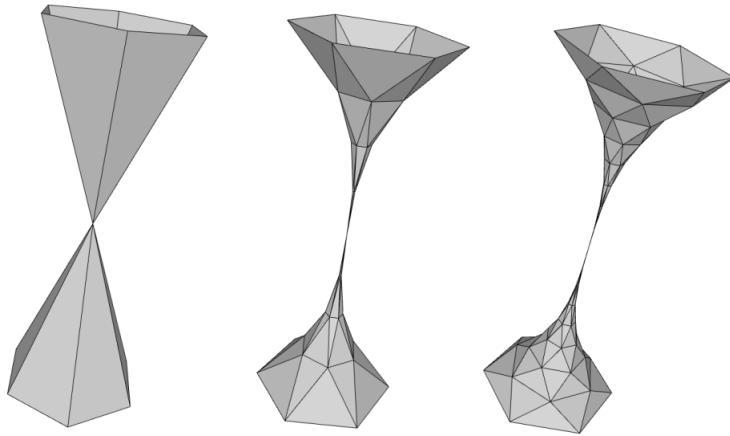


Figure 2.15: Local mesh with one layer, four layers and eight layers (from left to right) for the A_{5--} singularity.

layers to obtain the layer height h_l . Now, i -th layer of points is obtained using the height $h = i \cdot h_l$, $1 \leq i \leq l$. To connect these points into triangles, every second layer is rotated by $\frac{\pi}{6}$. The resulting 3 layers of triangles for the A_{1--} singularity as viewed from the point $(1, 0, 0)$ is displayed in the Figure 2.14.

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

The resulting mesh with four layers is displayed in the Figure 2.16.

D_n singularities

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

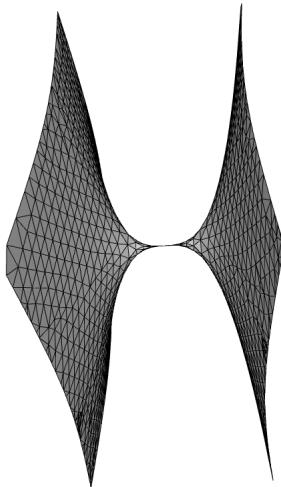


Figure 2.16: Resulting mesh for the A_{5--} singularity – four layers.

lie on the ellipse, and they have the same base length.

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

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

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

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

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 , i.e.

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

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

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

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

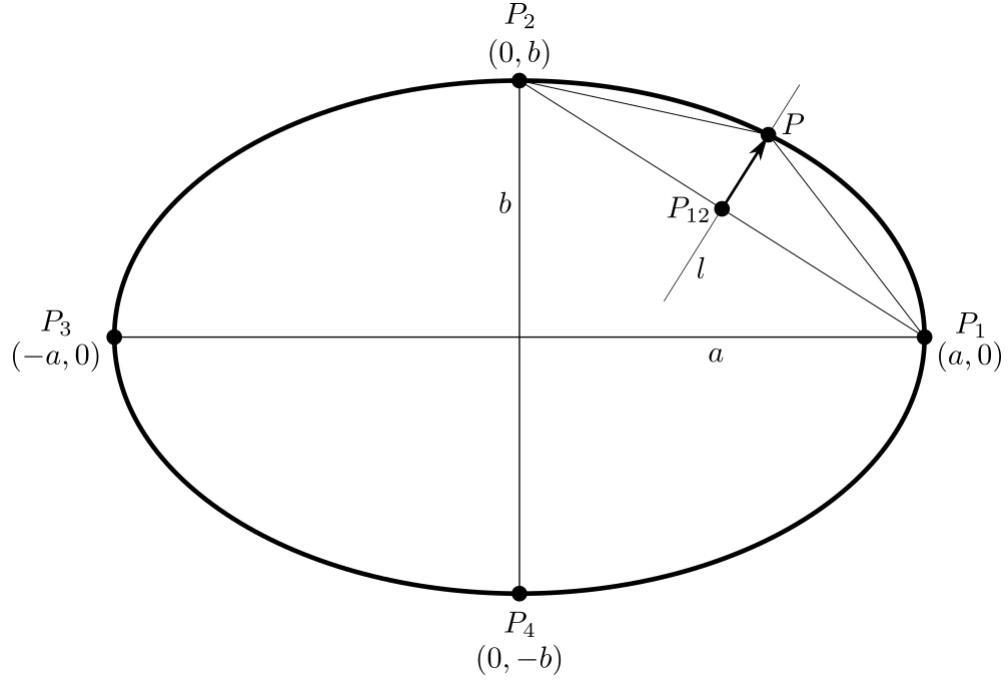
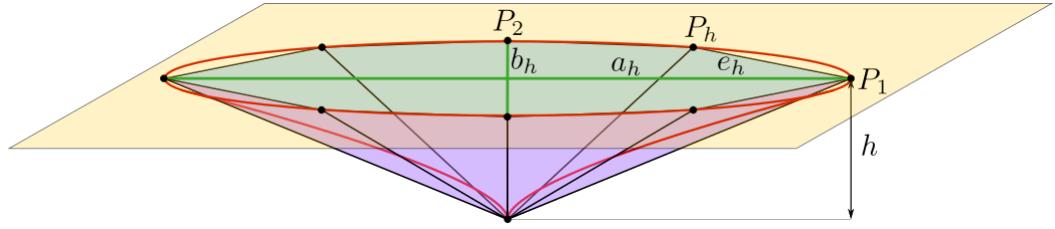


Figure 2.17: Equilateral octagon inscribed inside an ellipse.

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

therefore

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

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 in the Figure 2.18.

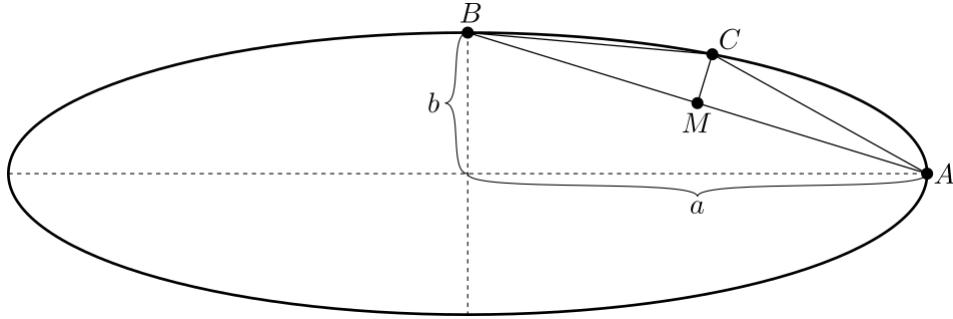
We use binary search to find such height. Given the height and the singularity class, we can calculate the semi-major axis and semi-minor axis as

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

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

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

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

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

$$\begin{aligned} D_{n--} : \quad & -hx^2 - h^{n-1} - z^2 = 0, \quad h > 0, \\ 2|n, x^2 + \frac{z^2}{h} = -h^{n-2} \implies & x^2 + \frac{z^2}{h} = h^{n-2}. \end{aligned}$$

Then by substituting for a and b in (1)

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

and simplifying

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

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

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

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

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

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

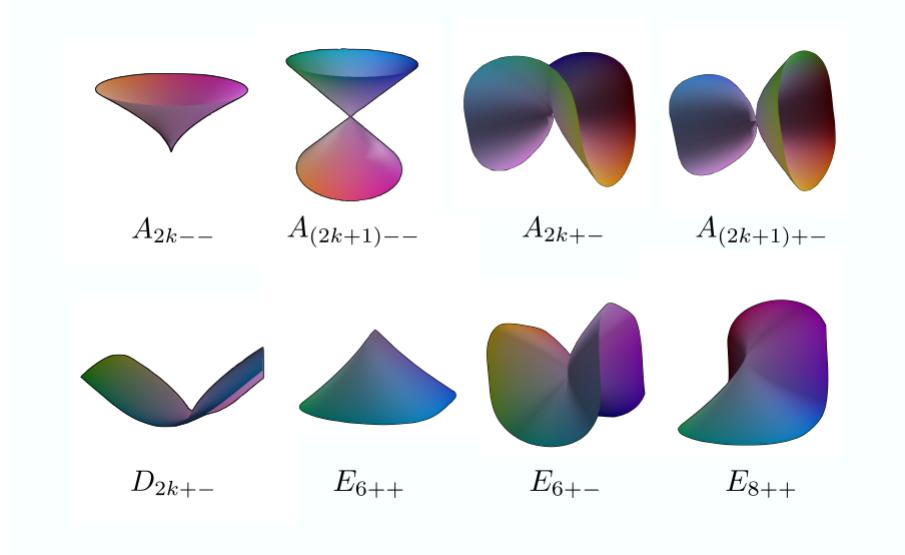


Figure 2.20: 1st category singularities. For more examples, see [17].

2.2.3 Numerical calculation of local triangulation of ADE singularities

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

We begin by dividing ADE singularities into two categories.

- 1st category singularities – displayed in the Figure 2.20:

- A_{n--} ,
- A_{n+-} ,
- D_{n+-} , where $n = 2k$,
- E_{6++} ,
- E_{6+-} ,
- E_{8++} .

- 2nd category singularities – displayed in the Figure 2.21:

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

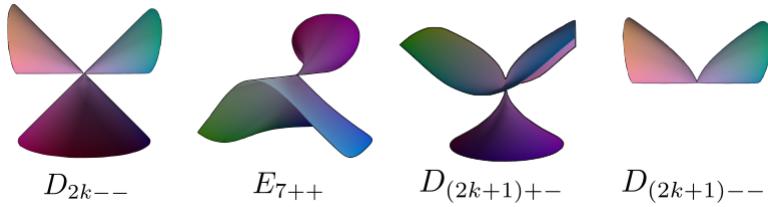


Figure 2.21: 2nd category singularities. For more examples, see [17].

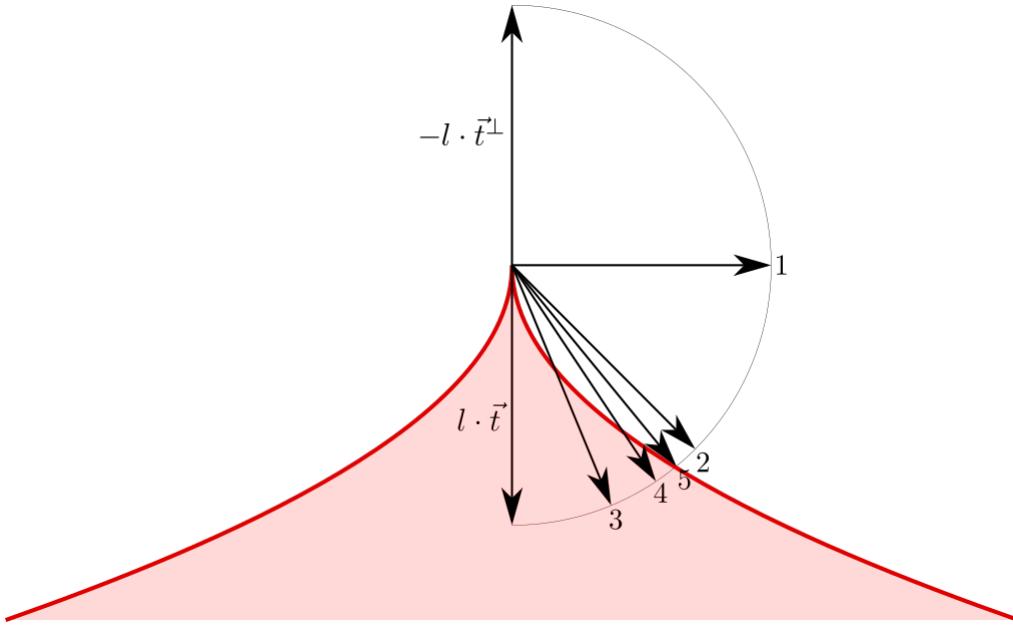


Figure 2.22: Binary search for the point on the surface.

Triangulation of 1st 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 1st category, topologically equivalent to a plane, given by the implicit equation $F(x, y, z) = 0$ it holds, that $F(O + \vec{t}) \cdot F(O - \vec{t}) < 0$. For the singularities from the 2nd category, topologically equivalent to a cone, it holds, that $F(O + \vec{t}_j) \cdot F(O + \vec{t}_j^\perp) < 0$ for $j = 0, 1$. This means that in both cases, we can perform a binary search to find a point on the surface. Given the required length l – the length of the leg of the isosceles triangle created around the singularity, we perform a binary search to find an angle, for which the endpoint of the rotated vector $l \cdot \vec{t}$ lies on the surface with given precision as displayed in the Figure 2.22.

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

The binary search is performed in the half-planes passing through the point O and

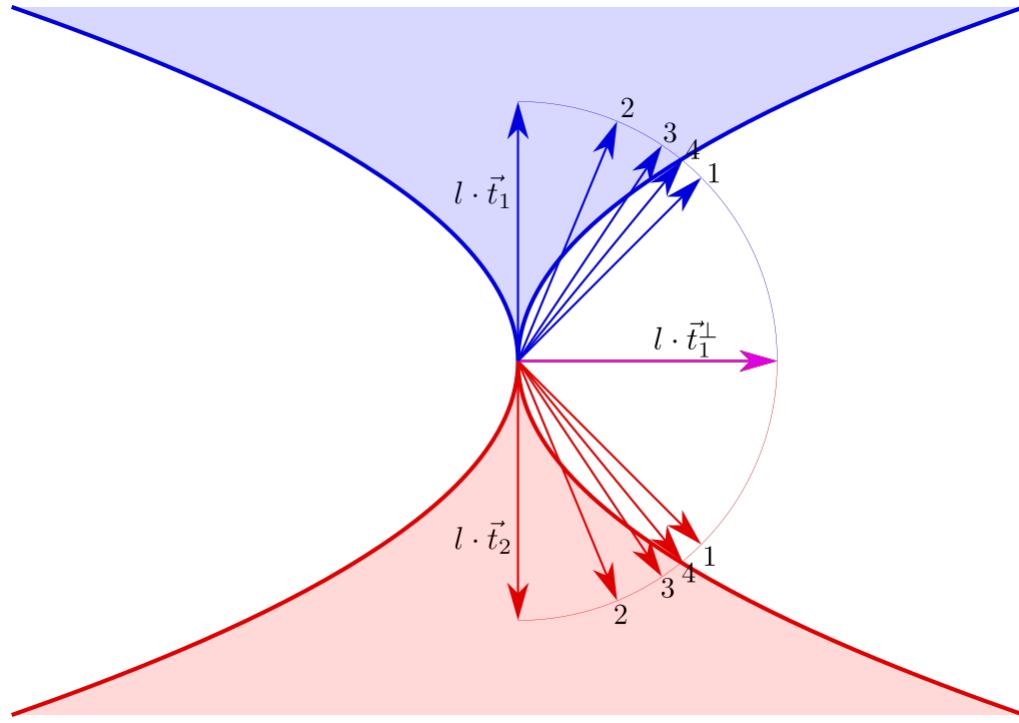


Figure 2.23: Double binary search for the point on the surface.

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 in the Figure 2.24.

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

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

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

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

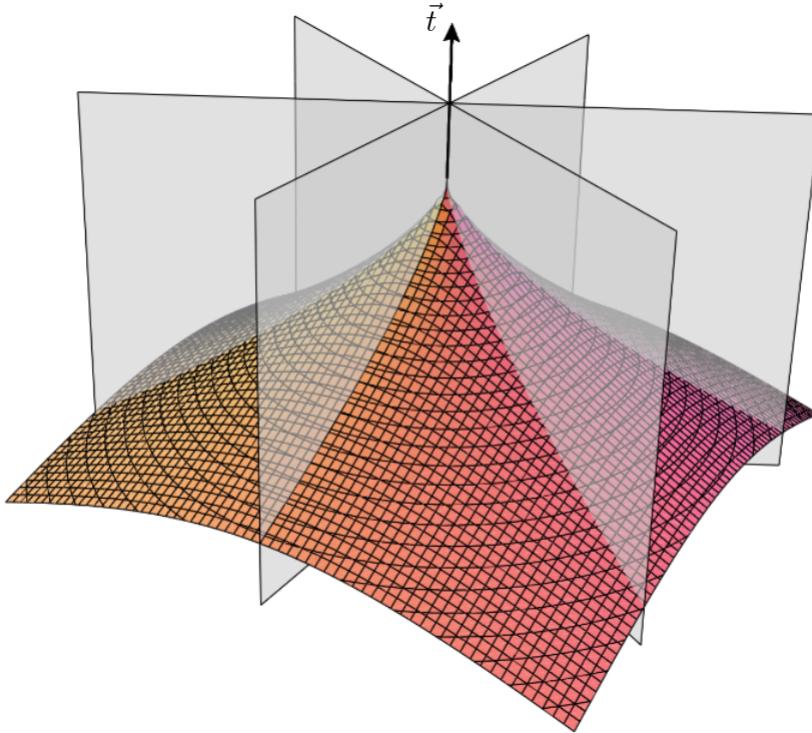


Figure 2.24: Rotating the plane about the vector \vec{t} [1].

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

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

Triangulation of 2nd category singularities

We create a more general, slower solution for the singularities in the 2nd category. Again, we find the points on the surface lying in the rotated half-planes as displayed in the Figure 2.24. We do not have the general rule for the start and end angle to begin the binary search for these singularities. We have the triangulation vector, whose endpoint lies inside the space volume bounded by the respective branch. We consider this triangulation vector a start vector to start the binary search. We find the end vector by rotating the triangulation vector \vec{t} around the vector \vec{p} – vector perpendicular to \vec{t} , by small increments, until the endpoint of the rotated vector is outside of the space volume bounded by the branch, let us denote this vector \vec{r} . To find the point of the surface, we perform an angular binary search between \vec{t} and \vec{r} . We proceed to find other points by rotating \vec{p} around \vec{t} by $\frac{2\pi}{k}$. By using the small

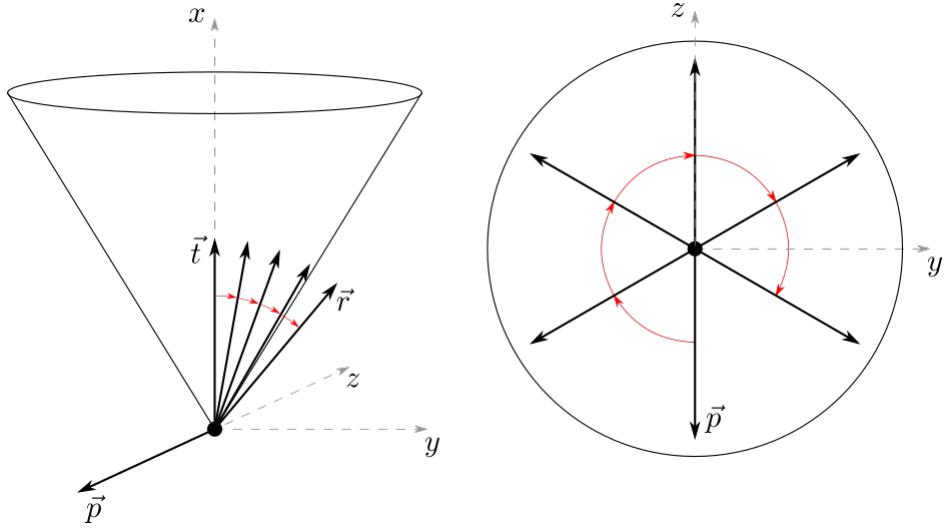


Figure 2.25: Iterative process of finding the vector \vec{r} by rotating \vec{t} around \vec{p} (left). Rotating the vector \vec{p} around \vec{t} by $\frac{2\pi}{k}$ (right).

increments, we ensure that the rotated vector does not end up in the space volume bounded by another branch, but it slows down the process of creating the local mesh. For the illustration, in the Figure 2.25 on the left, one can see the iterative process to find the vector \vec{r} by rotating \vec{t} around \vec{p} . On the right, one can see the rotation of \vec{p} around \vec{t} by $\frac{2\pi}{k}$ to find the remaining points on the surface.

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

Layers for ADE singularities

We have already introduced the concept of creating multiple layers of triangles around A_{n--} singularities. In this section, we explain the extension of this concept to remaining ADE singularities. We create l layers of points for each branch for a given singularity. In contrast with A_{n--} singularities, we do not rotate every second layer by $\frac{\pi}{k}$. The reason is that the midpoints of the edges connecting a particular layer of points do not lie on the surface. This results in a poor approximation of the curves lying in the intersection of the surface and coordinate planes. Sometimes, it even results in incorrect local mesh with overlapping triangles.

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

In the Figure 2.27 on the left, one may see the incorrect local mesh of A_{2+-} singu-

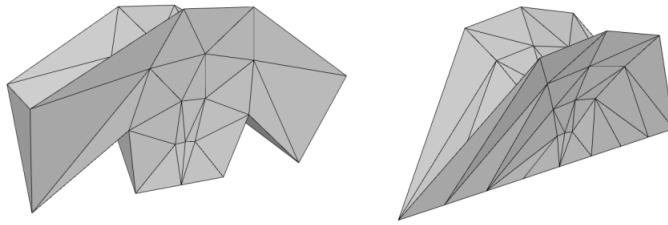


Figure 2.26: Insufficient (left) and sufficient (right) local approximation of D_{4+} -singularity.

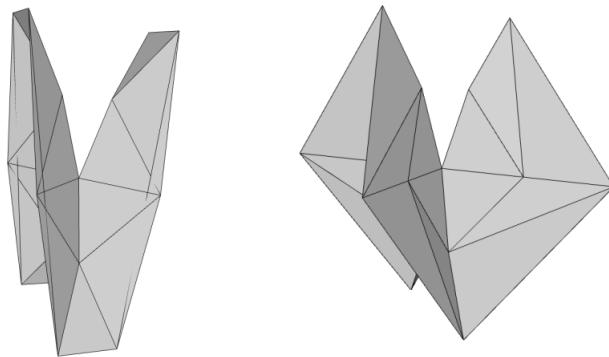


Figure 2.27: Incorrect (left) and correct (right) local mesh for A_{2+} -singularity.

larity 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 in the Figure 2.24. An example of such singularities are D_{n+} -singularities. The local mesh created with eight triangles using the described approach is displayed in the Figure 2.28. One may see that the outer edges of the triangles have a significant length difference.

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

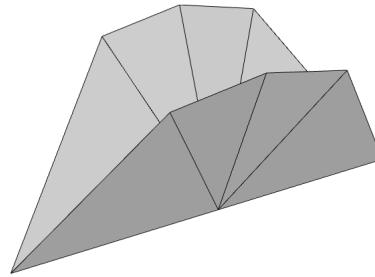


Figure 2.28: Uneven triangles in the local mesh around D_{4+} - singularity.

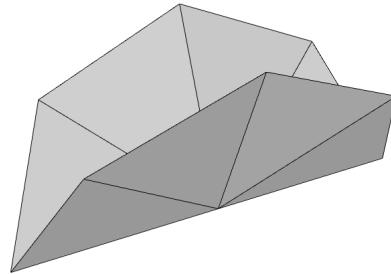


Figure 2.29: Optimized local mesh around D_{4+} - singularity.

between the two angles. The new point is then calculated using the binary search to find a point on the surface, as shown in the Figure 2.22.

As all singularities using this optimization are symmetrical by two coordinate planes, we only need to perform this binary search once for each layer of points and use mirror symmetry to obtain the remaining points.

The result of the binary search of an angle for the D_{4+} - singularity is displayed in 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 —continuously connected to this plane.

Input and output

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

1. the number of singularities — m ,
2. m discrete points on a plane — $(x_1, y_1), \dots, (x_m, y_m)$,

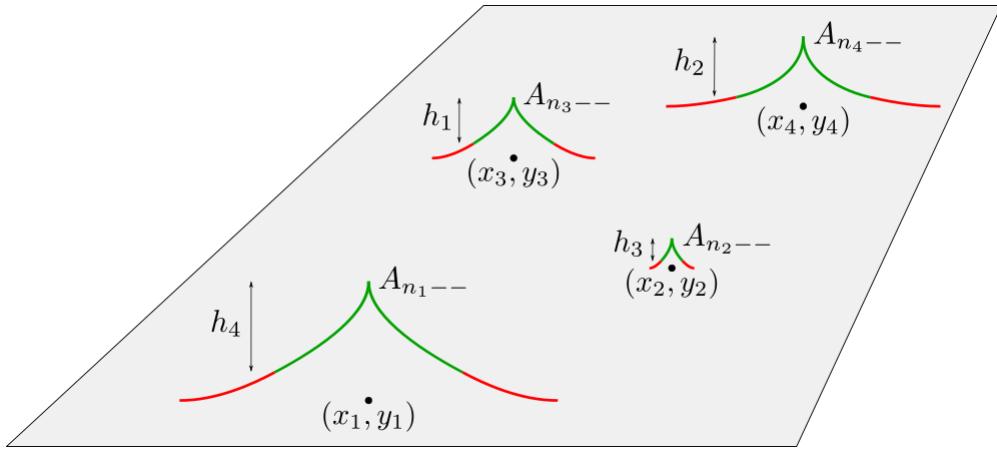


Figure 2.30: Plane with singularities.

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

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^∞) and compactly supported (the closed support of the function f is a compact subset of \mathbb{R}^n).

The most common example of such bump function is the function

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

which is both C^∞ and compactly supported.

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

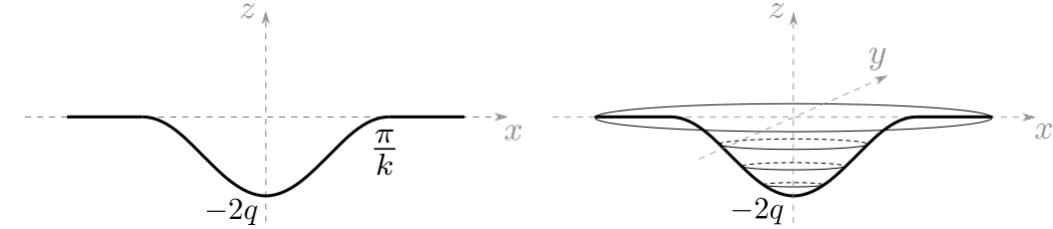


Figure 2.31: C^1 – continuous cosine bump function – B_{\cos} .

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 cosine bump function – B_{\cos}

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

These functions can be seen in the Figure 2.31.

Implicit equation of the function B_{\cos}

To construct the implicit equation of the function B_{\cos} , we use CSG – constructive solid geometry, described in the section 1.3.

First, we cut out the part of the rotated cosine, where $x^2 + y^2 < (\frac{\pi}{k})^2$ using cylinder and intersection operation. Next, we use a plane and the union operation to *glue* the bump to the plane. The described process is displayed in the Figure 2.32 in the form of a CSG tree. We use the equations of the surfaces in the Table 2.1 to model the function B_{\cos} .

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

Table 2.1: Implicit equations for bump function modelling.

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) .

Attaching singularities to the plane using the function B_{\cos}

Given the type of the singularity – n and given height – h , we calculate the constants of the function B_{\cos} to connect C^1 smoothly to the singularity.

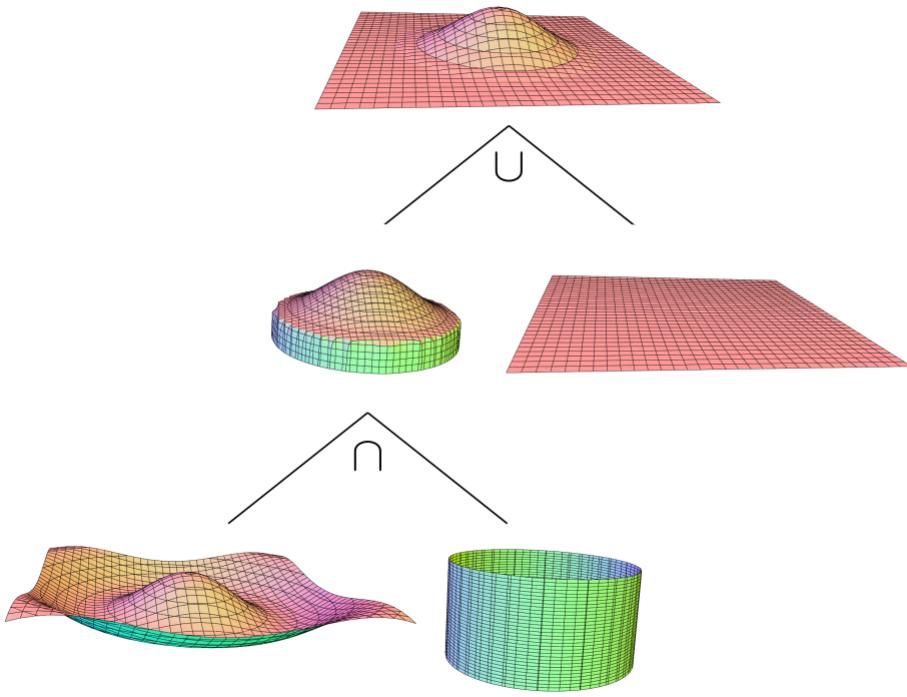


Figure 2.32: Construction of the function B_{\cos} using CSG [1].

The singularity given by the function $F(x, y, z) = x^{n+1} - y^2 - z^2$ intersected with the plane $x = h$ produces a circle with the radius $r = \sqrt{h^{n+1}}$. The function B_{\cos} is scaled using q and k to smoothly connect the singularity in the middle of the function B_{\cos} . This approach is displayed in the Figure 2.33. The singularity is shifted such that it connects to the plane $x = 0$.

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

$$\begin{aligned} G(x, y) &= (x + h + q)^{n+1} - y^2 \implies \nabla G(x, y) = [(n+1)(x + h + q)^n, -2y] \\ \nabla G \left(-q, \frac{\pi}{2k} \right) &= \left[(n+1)h^n, -\frac{\pi}{k} \right] \end{aligned}$$

The function $H(x, y)$ denotes the function B_{\cos} intersected with the plane $z = 0$.

$$\begin{aligned} H(x, y) &= x + q \cdot \cos(ky) + q \implies \nabla H(x, y) = [1, -qk \cdot \sin(ky)] \\ \nabla H \left(-q, \frac{\pi}{2k} \right) &= [1, -qk] \end{aligned}$$

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

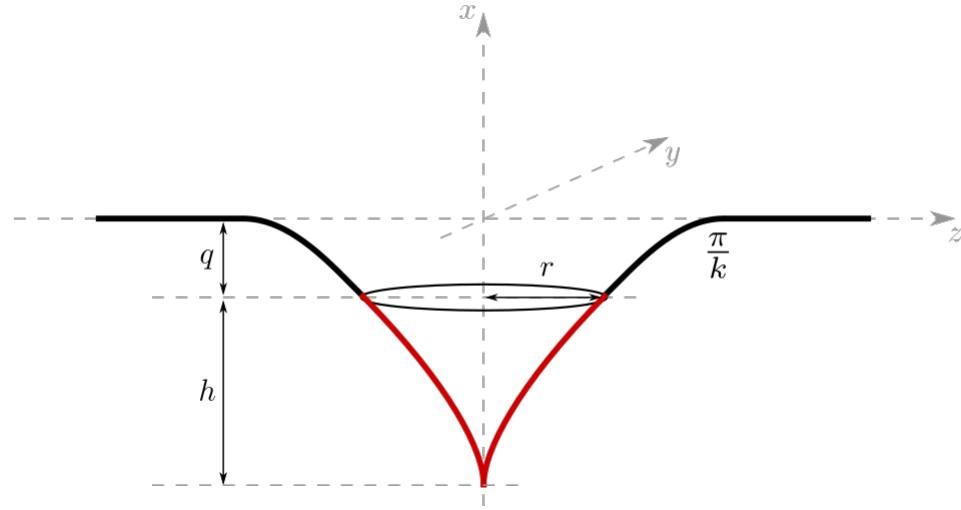


Figure 2.33: Attaching the singularity to a plane using the function B_{\cos} .

After calculating the parameters q and k of the function B_{\cos} , we proceed to connect the singularity to the bump function. We use intersection with the plane $x = -q$ to get the sections of the singularity and the section of the bump function, and lastly, we use the union of these two surfaces. The described process is displayed in the Figure 2.34. The information about the detailed calculation of the implicit equation can be found in the appendix 4.5.

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

Limitations on the input data

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

Let l be the number of layers used for the triangulation and h_e be the height calculated from the input edge length e as shown in the chapter 2.2.2. Then, the height h displayed on the image 2.33 must be bigger than the height h_e .

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, union or difference operation on the interiors of two regular implicit surfaces.

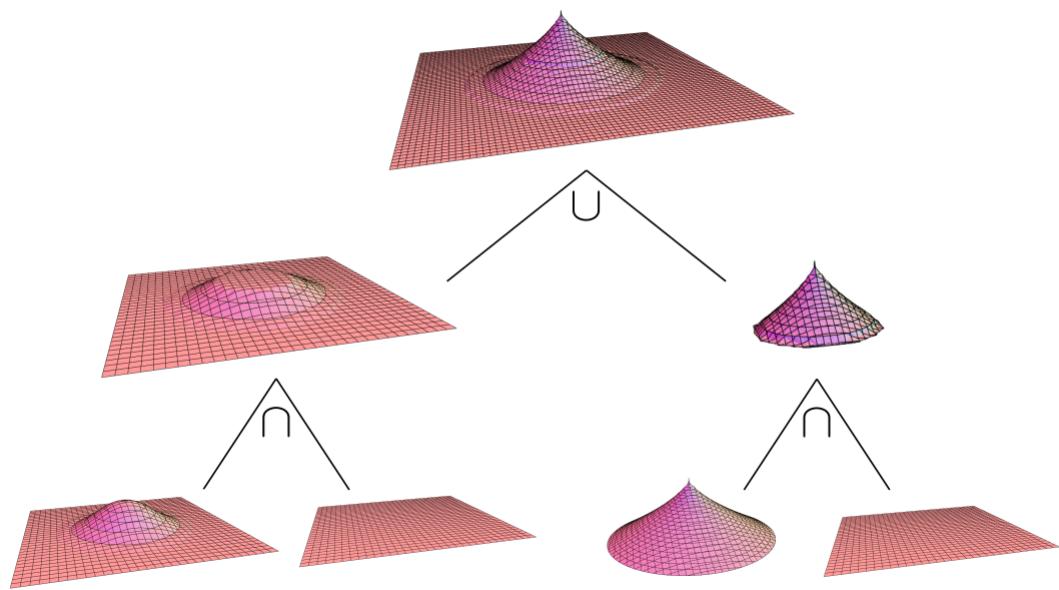


Figure 2.34: Attaching the singularity to a plane using CSG [1].

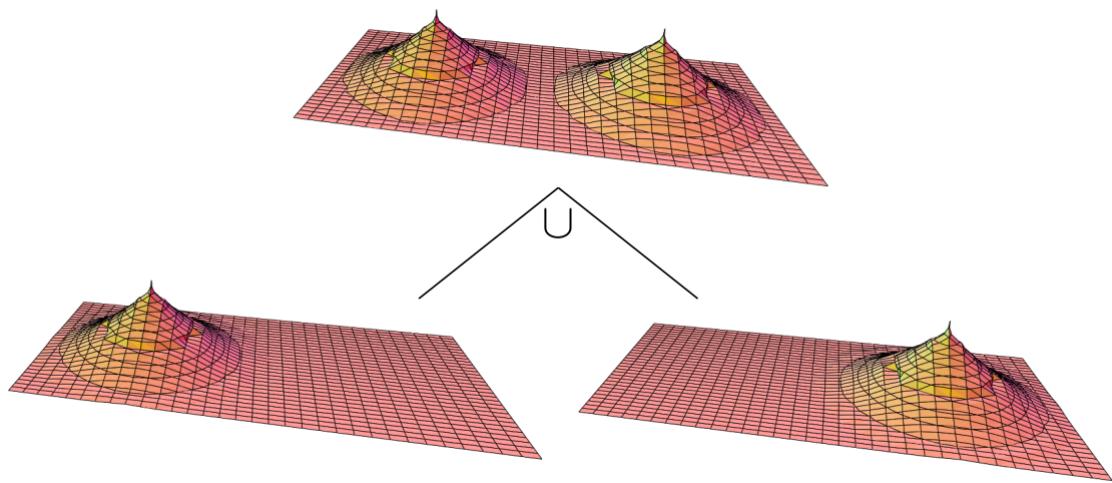


Figure 2.35: Plane with multiple attached singularities [1].

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

2.3.1 Creating the local mesh around the singular curves

We begin with 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 the case of the open curve, to complete the whole polyline, we also approximate the second part of the singular curve by iteratively creating points in the opposite direction, starting from point P_0 . Let m be the number of points in the polyline. Then 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

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

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

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

After obtaining the polyline approximation of the curve on the intersection of the two surfaces, one may create the local mesh. Let us rename the polyline points to P_0, P_1, \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 a closed curve).

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

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

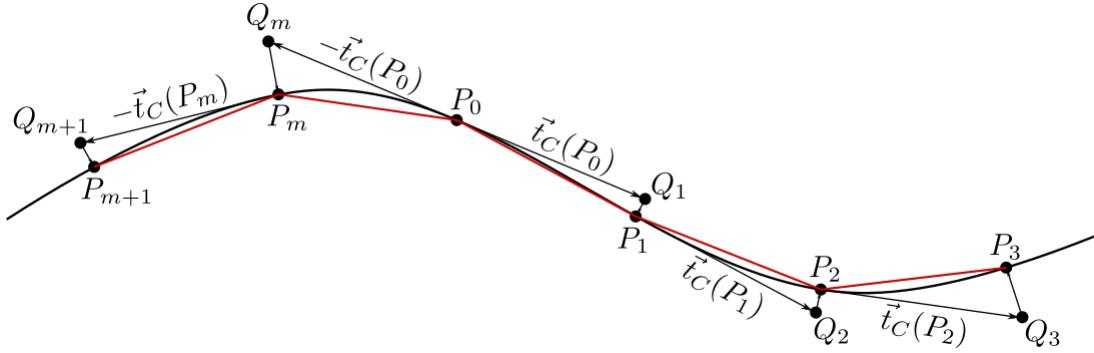
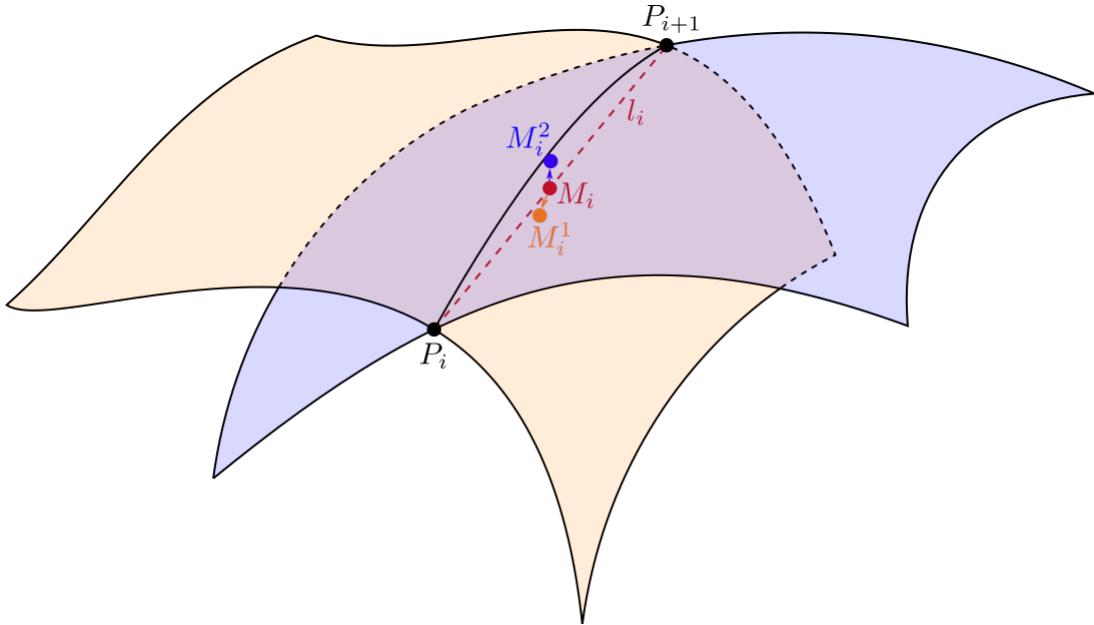


Figure 2.36: Approximation of the implicit curve by a polyline.

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

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

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

The point R_i^1 is obtained by moving from the point M_i in the direction perpendicular to both $\overrightarrow{P_iP_{i+1}}$ and $\nabla F_1(M_1)$ by the given height $h_e = \frac{\sqrt{3}}{2}e$ associated with the edge length e .

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

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

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

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

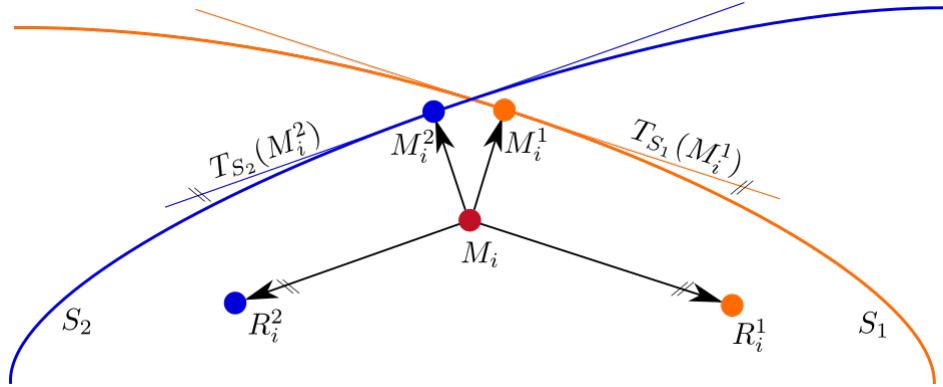


Figure 2.38: Definition of the points R_i^1 and R_i^2 in the plane H_i .

The points R_i^1 and R_i^2 are projected to the surface given by the intersection of the interiors of the two surfaces. The equation defining the surface is

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

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

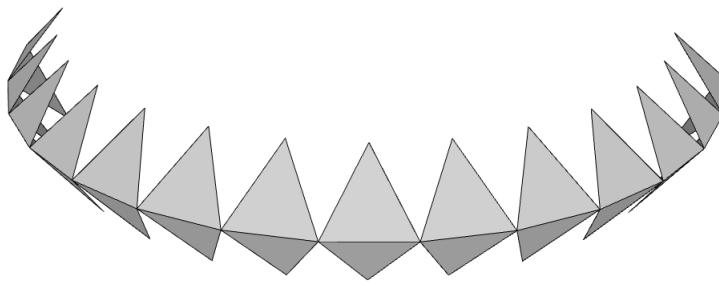


Figure 2.39: Local mesh around the singular curve.

2.3.2 Modification of triangulation for union and difference

For the intersection, the points R_i^1 and R_i^2 were picked in such a way, that the point R_i^1 does not lie outside of the S_2 and the point R_i^2 does not lie outside of the surface

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

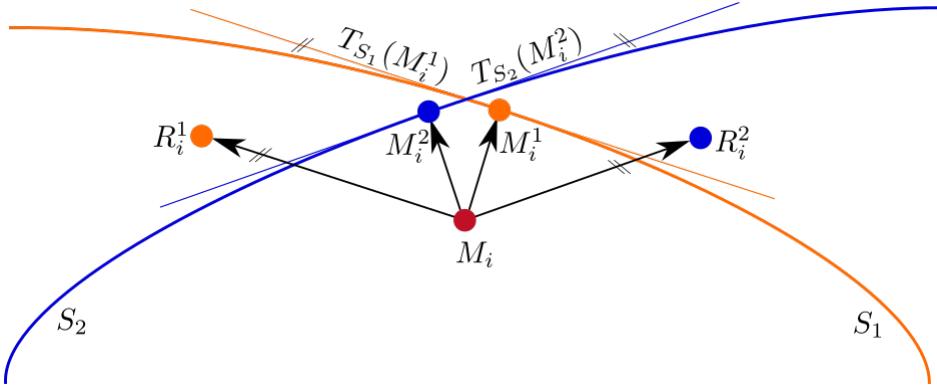


Figure 2.40: Definition of the points M_i , M_i^1 and M_i^2 for union.

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

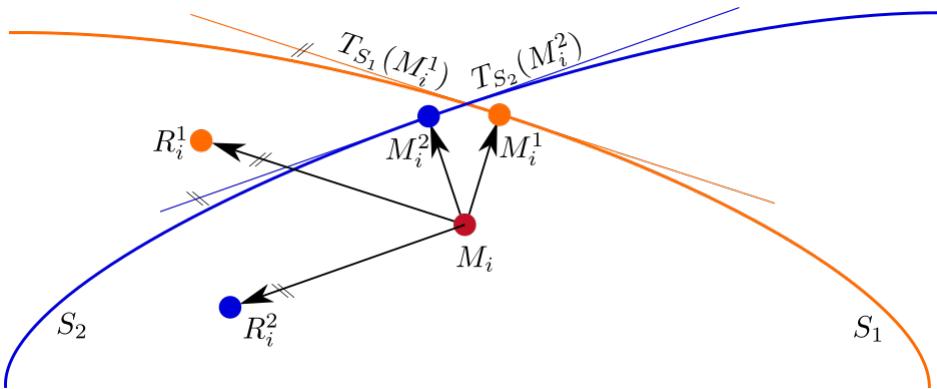


Figure 2.41: Definition of the points M_i , M_i^1 and M_i^2 for difference.

Chapter 3

Implementation

3.1 Triangulation of regular implicit surfaces

In this section, we shortly describe the algorithm for triangulation of the regular parts of implicit surfaces introduced in bachelor's thesis [14].

The algorithm creates triangles iteratively, one at a time. The new triangle is created at the edge of the existing triangle. The new point is projected on the surface using the well-known iterative Newton-Raphson method, the fast-converging method for root finding of a function $y = f(x)$. The condition for convergence of this method is to provide a point on the function sufficiently close to the root. We use this method to find the root of the implicit function lying on the line, which is in the direction of the gradient of the implicit function. This approach is displayed in 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 [11]. The Delaunay condition checks if some other points are in the proximity of the circumcenter of the new triangle. These conditions minimize the chances of a triangle intersection. If the new triangle intersects with some existing triangles, the algorithm tries to connect the new triangle to the existing points of the mesh in its proximity.

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

The algorithm was implemented using brute force. The queue of the edges on the mesh border – *active edges*, was being iteratively updated. In each step, an edge from the active edges queue was removed, and the creation of a new triangle near this edge was attempted. If the new triangle was successfully created, the edge has become *inside edge*, in case of failure, the edge has become *checked edge*. The last characterization of an edge is for the edges with both end-points lying on the bounding box. These edges

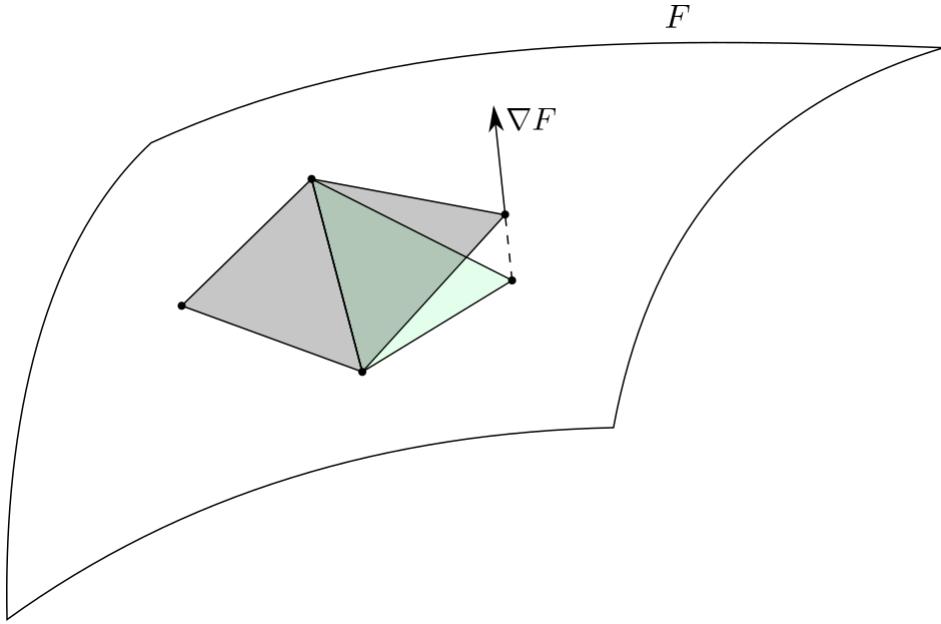


Figure 3.1: Projecting the point on the surface in the direction of the gradient.

are called *bounding edges*.

As a part of our work, we reimplement the algorithm to be more effective by using advanced data structures, such as the half-edge data structure [12] and the range tree [16].

3.2 Data structures for triangulation algorithm

3.2.1 Half-edge data structure

Triangular mesh is given by 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 [6] presented a representation using winged edges, which was further improved in 1985 by Weiler [22] who presented the modification called a 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 information about the surrounding vertices, edges and faces.

In the half-edge representation, edges are split into two halves of directed edges. Each half-edge stores reference to its initial vertex, left face, opposite half-edge and left traverse – preceding and succeeding half-edge. A visualization of the half-edge is displayed in the Figure 3.2. An example of half-edge representation is displayed in

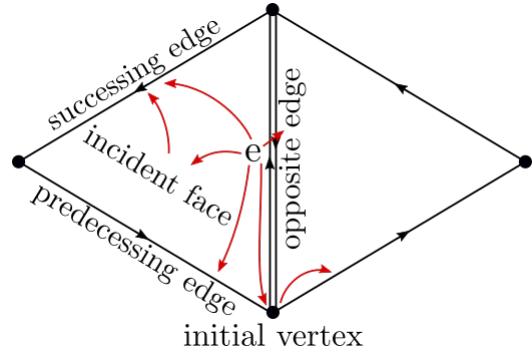


Figure 3.2: Visualization of the half-edge data structure.

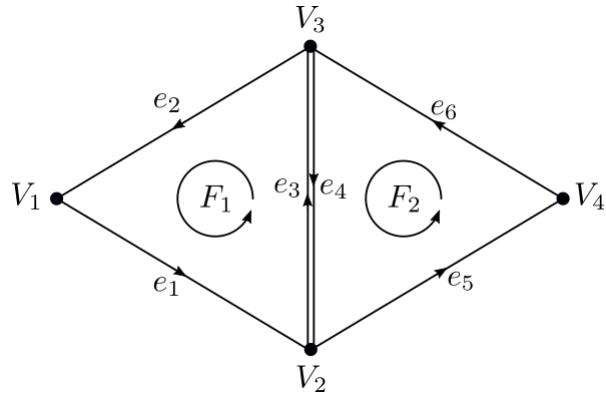


Figure 3.3: Example of half-edge representation.

the Figure 3.3 and tables 3.1, 3.2, 3.3.

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

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

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

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_3
V_3	x_3	y_3	z_3	e_2
V_4	x_4	y_4	z_4	e_6

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

Face	Edge
F_1	e_1
F_2	e_4

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

3.2.2 Range tree

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

3.2.3 Mesh structure

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

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

Vertex consists of

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

Half-edge consists of

- index of itself in the list of half-edges,
- index of the initial vertex in the list of vertices,
- index of the opposite edge in the list of half-edges,
- index of the preceding edge in the list of half-edges,
- index of the succeeding edge in the list of half-edges,
- index of the incident face in the list of faces,
- boolean values stating if the half-edge is active, checked, bounding or inside.

Face consists of

- index of incident half-edge in the list of half-edges.

3.3 Algorithm runtime

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

When checking for the conditions for the newly created triangle, the proximity within the mesh provided by the half-edge data structure is insufficient. Here, the range tree is used. To find the triangles in the proximity of a point P , all mesh points in a three-dimensional interval around this points are found using the range tree. The triangles are then extracted by finding the incident faces of outgoing edges of the points found by the range tree.

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

Chapter 4

Results

In this chapter, we present the results achieved in the triangulation of ADE singularities, surfaces with ADE singularities and surfaces with singular curves which arose from performing CSG operation on regular surfaces. We measure some of the quality criteria proposed in [14] 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 [17]. We compare the computational speed of the algorithm after the reimplementation with the algorithm for the regular parts of the implicit surfaces implemented in [14].

4.1 Quality criteria

The following quality criteria [14] 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 shortest side of the triangle and the longest side of the triangle shows the uniformity of the triangles. For an equilateral triangle, the ratio equals exactly one. The ratio is smaller than one for isosceles and scalene triangles. The further the triangle is from an equilateral triangle, the smaller the ratio. A number close to one indicates triangles close to the equilateral triangles, whereas a number close to zero indicates more non-uniform triangles in the mesh. Let us denote this criterium k_1 .

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 between two sets of points.

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

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

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 the 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. Let us denote this criterium k_2 .

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. Let us denote this criterium k_3 .

4. The mean distance of the neighbour vertices from a 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. A small standard deviation indicates that the values are close to the mean. On the contrary, a 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 σ 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 calculate the mean standard deviation as an arithmetical mean of the standard deviations of the distances of all points from its neighbour points in the mesh. Let us denote this criterium k_4 .

4.2 Comparison with SingSurf

SingSurf [17] is a software used to visualize 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 [17].

4.2.1 SingSurf algorithm

In this subsection, we draw from the article by Richard Morris [17].

The server takes the defining equation $F(x, y, z)$ of an algebraic surface and produces a polygonization of the surface inside the given bounding box. The algorithm used by the SingSurf surface begins with a recursive subdivision of the bounding box to smaller boxes with the half of the size of the edge of the bigger box. They use the test based on Bernstein polynomials to determine whether the subdivided box contains a part of the surface. Only boxes which contain a surface are further subdivided. After three or four levels of subdivision, the smaller boxes are examined in greater detail.

They define and find three types of points.

1. Points on the edges of the box where $F = 0$.
2. Points on the faces of the box where $F = 0$ and at least one of the partial derivatives, $\frac{\partial F}{\partial x}$, $\frac{\partial F}{\partial y}$ or $\frac{\partial F}{\partial z}$ is zero. They call these points 2-nodes.
3. Points in the interior of the box where $F = 0$ and at least two of the partial derivatives are zero. They call these points 3-nodes.

These points are found in every box, and they are connected into triangles to form a mesh.

Limitations for the input data for the SingSurf software

To avoid degenerate cases, the input to the software is required to satisfy the following conditions.

1. The surface does not intersect the corner points of the bounding box.
2. None of the partial derivatives vanish at the solutions on the edges of the box.
3. The 2-nodes on the faces of the box are isolated.

This usually means that the bounding box needs to be constructed with unequal bounds so that the origin, which is usually the singular point, does not lie at a corner of a box.

4.2.2 Evaluated quality criteria

We compared the quality of 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 lying on a line) produced by the SingSurf software are not considered.

A_{n--} singularities

We created meshes of A_{1--} , A_{2--} , A_{3--} and A_{4--} singularities.

The resulting uniform meshes from our algorithm can be seen in the Figure 4.1.

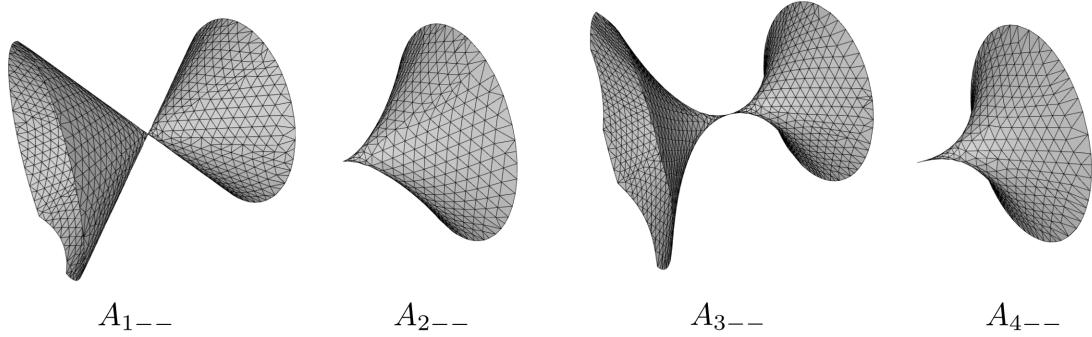


Figure 4.1: Resulting uniform triangulation of A_{n--} singularities with layers.

The resulting adaptive meshes from our algorithm can be seen in the Figure 4.2.

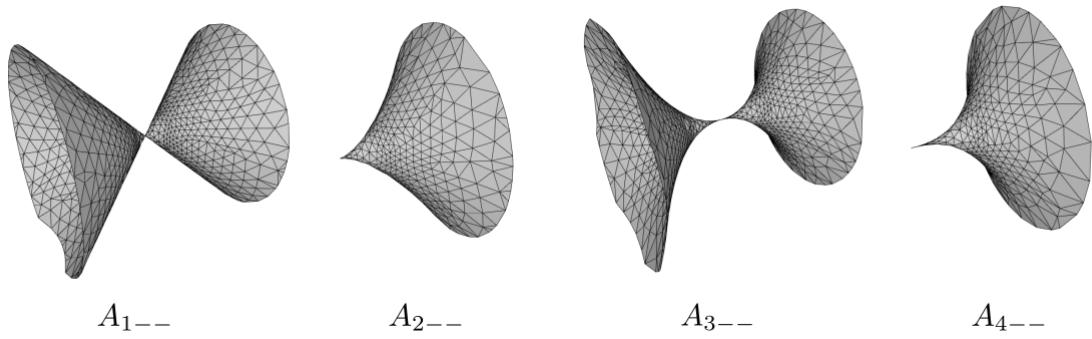


Figure 4.2: Resulting adaptive triangulation of A_{n--} singularities with layers.

The resulting meshes generated by SingSurf can be seen in the Figure 4.3.

The comparison of the quality criteria measured on these meshes is in the Table 4.1. The better values are visualized by the green color, and the worse values are visualized by the red color.

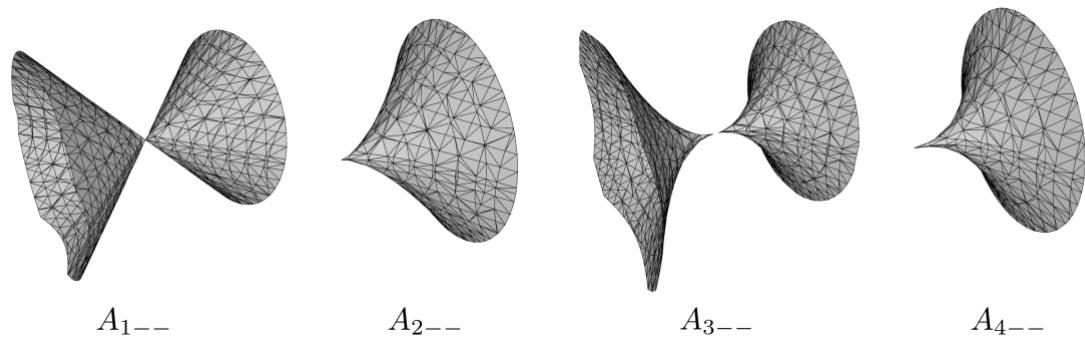


Figure 4.3: Resulting triangulation of A_{n--} singularities by SingSurf [17].

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

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

A_{n+-} singularities

We created meshes of A_{2+-} , A_{3+-} and A_{4+-} singularities.

The resulting uniform meshes from our algorithm can be seen in the Figure 4.4.

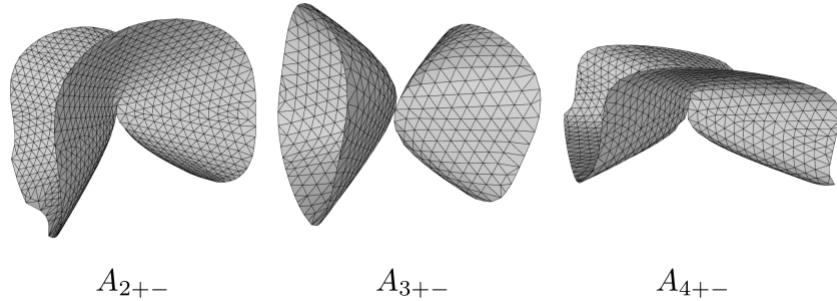


Figure 4.4: Resulting uniform triangulation of A_{n+-} singularities with layers.

The resulting adaptive meshes from our algorithm can be seen in the Figure 4.5.

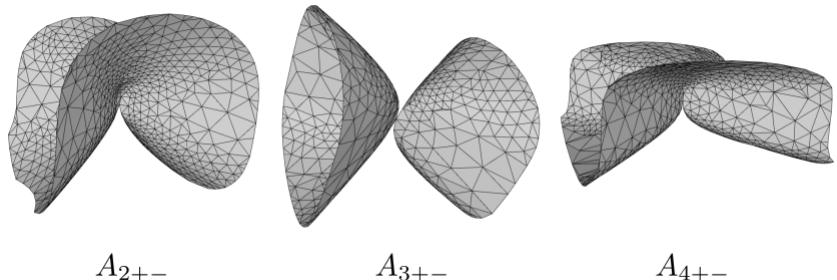


Figure 4.5: Resulting adaptive triangulation of A_{n+-} singularities with layers.

The resulting meshes generated by SingSurf can be seen in the Figure 4.3.

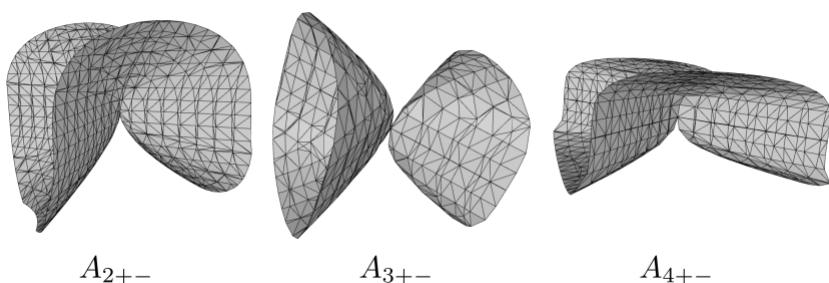


Figure 4.6: Resulting triangulation of A_{n+-} singularities by SingSurf [17].

The comparison of the quality criteria measured on these meshes is in the Table 4.2. The better values are visualized by the green color, and the worse values are visualized by the red color.

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

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

D_n singularities

We created meshes of D_{4+-} , D_{4--} , D_{5+-} and D_{5--} singularities.

The resulting uniform meshes from our algorithm can be seen in the Figure 4.7.

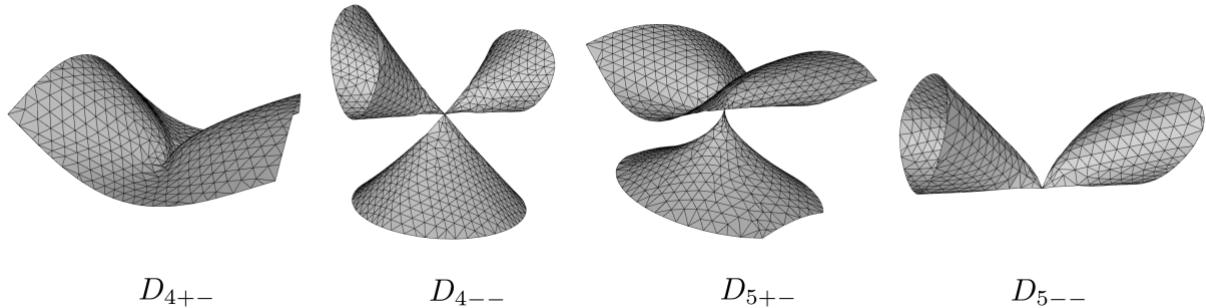


Figure 4.7: Resulting uniform triangulation of D_n singularities with layers.

The resulting adaptive meshes from our algorithm can be seen in the Figure 4.8.

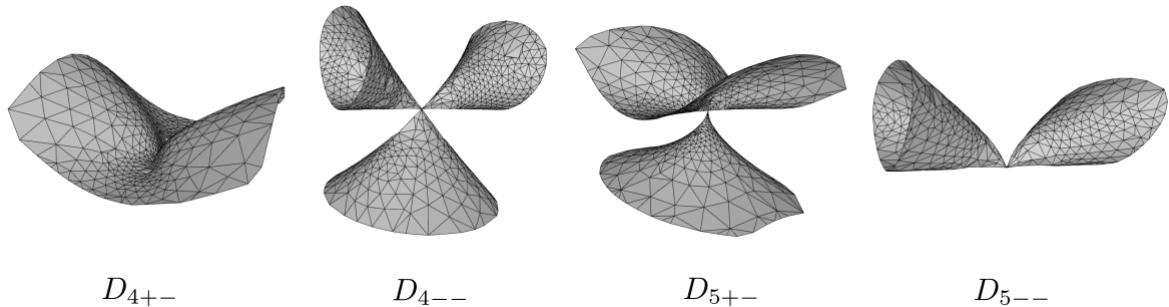


Figure 4.8: Resulting adaptive triangulation of A_n singularities with layers.

The resulting meshes generated by SingSurf can be seen in the Figure 4.9.

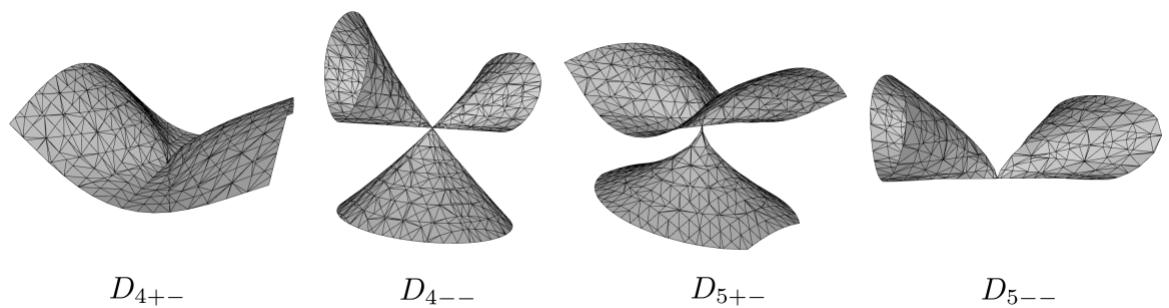


Figure 4.9: Resulting triangulation of D_n singularities by SingSurf [17].

The comparison of 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.

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

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

E_n singularities

We created meshes of E_{4++} , E_{6+-} , E_{7++} and E_{8++} singularities.

The resulting uniform meshes from our algorithm can be seen in the Figure 4.10.

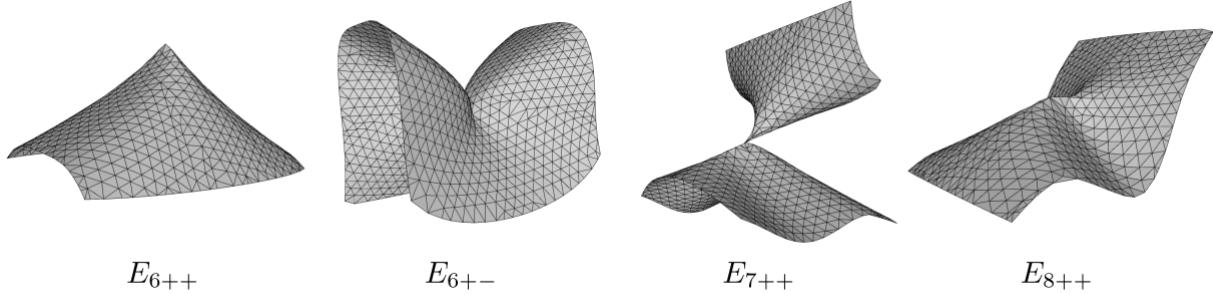


Figure 4.10: Resulting uniform triangulation of E_n singularities with layers.

The resulting adaptive meshes from our algorithm can be seen in the Figure 4.11.

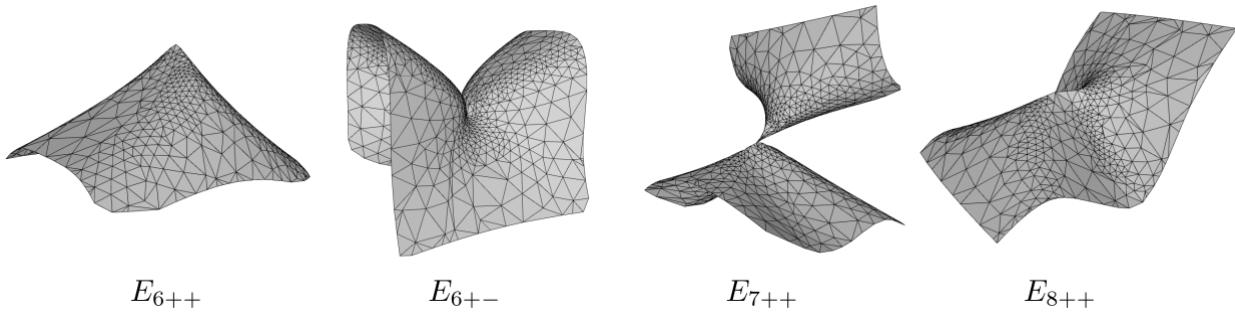


Figure 4.11: Resulting adaptive triangulation of E_n singularities with layers.

The resulting meshes generated by SingSurf can be seen in the Figure 4.12.

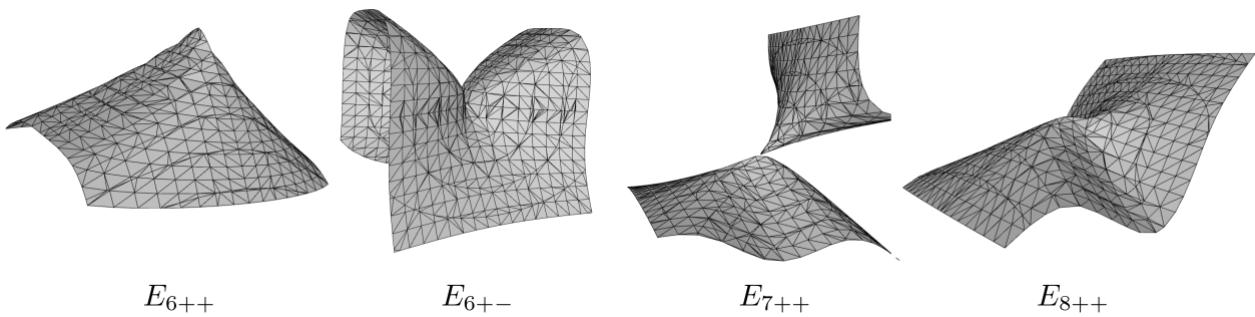


Figure 4.12: Resulting triangulation of E_n singularities by SingSurf [17].

The comparison of the quality criteria measured on these meshes is in the Table 4.4. The better values are visualized by the green color, and the worse values are visualized by the red color.

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

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

4.3 Curve singularities

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

4.3.1 Intersection

The intersection of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 4 = 0$$

and the inside of a hyperboloid given by the equation

$$x^2 - y^2 + z^2 - 1 = 0$$

is displayed in the Figure 4.13.

The intersection of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 9 = 0$$

and the inside of a tangle-cube given by the equation

$$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 11.8 = 0$$

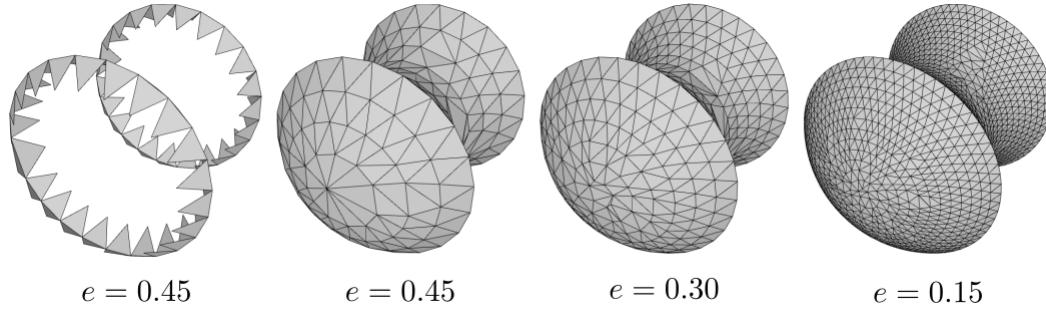


Figure 4.13: The surface given by the intersection of a sphere and a hyperboloid.

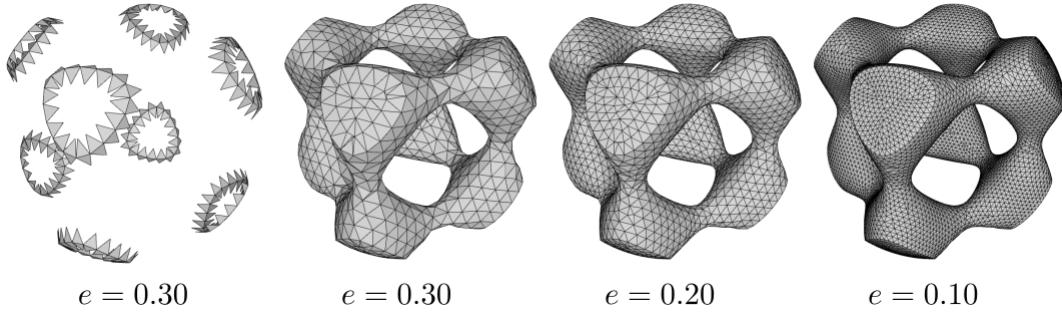


Figure 4.14: The surface given by the intersection of a sphere and a tangle-cube.

is displayed in the Figure 4.14.

The intersection of the inside of a blobby object given by the equation

$$2\sqrt{x^2 + y^2 + z^2 + 1} - 1.1 = 0$$

and the inside of a plane (halfspace) given by the equation

$$z = 0$$

is displayed in the Figure 4.15.

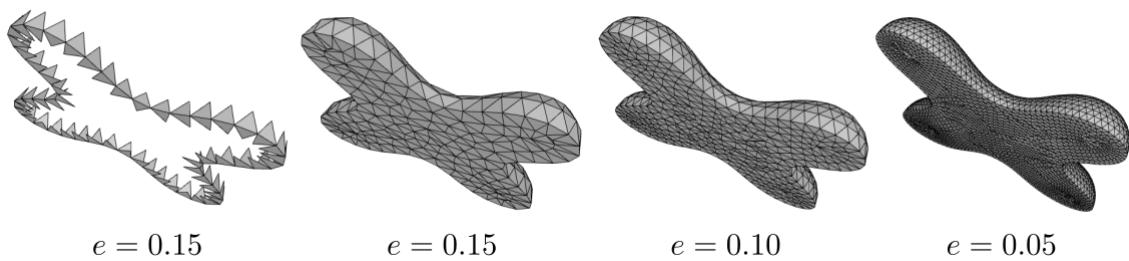


Figure 4.15: The surface given by the intersection of a blobby object and a plane.

The intersection of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 4 = 0$$

and the inside of a plane (halfspace) given by the equation

$$y = -1$$

is displayed in the Figure 4.16.

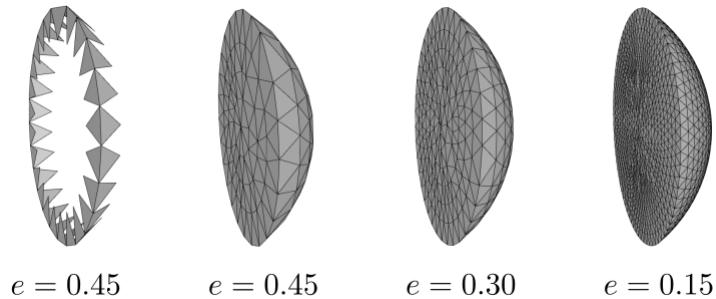


Figure 4.16: The surface given by the intersection of a sphere and a plane.

4.3.2 Union

The union of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 4 = 0$$

and the inside of a hyperboloid given by the equation

$$x^2 - y^2 + z^2 - 1 = 0$$

is displayed in the Figure 4.17.

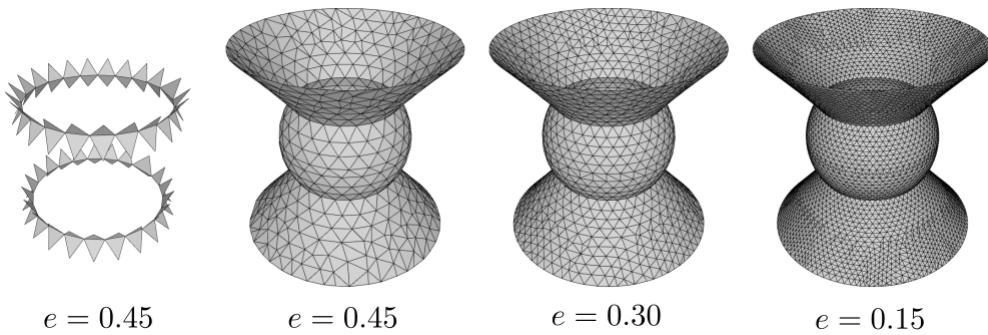


Figure 4.17: The surface given by the union of a sphere and a hyperboloid.

The union of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 4 = 0$$

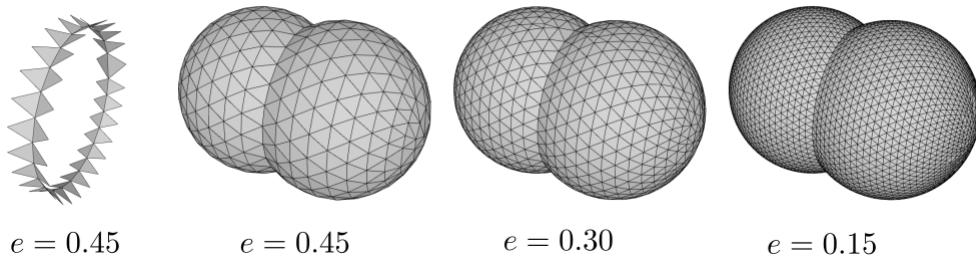


Figure 4.18: The surface given by the union of two spheres.

and the inside of a sphere given by the equation

$$(x - 2)^2 + y^2 + z^2 - 4 = 0$$

is displayed in the Figure 4.18.

The union of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 9 = 0$$

and the inside of a tangle-cube given by the equation

$$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 11.8 = 0$$

is displayed in the Figure 4.19.

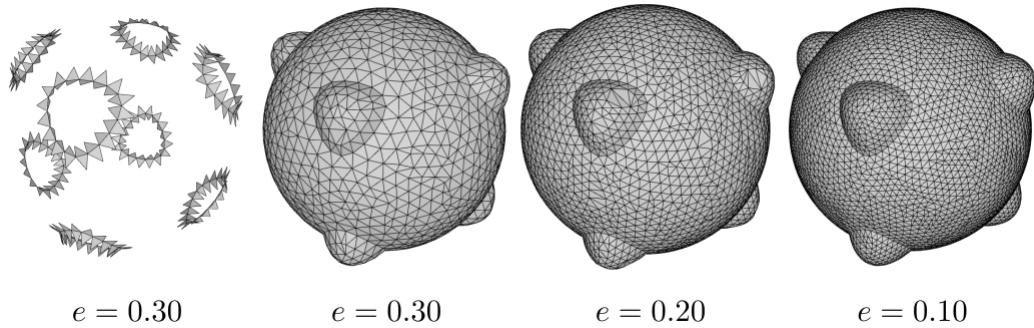


Figure 4.19: The surface given by the union of a sphere and a tangle-cube.

4.3.3 Difference

The difference of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 4 = 0$$

and the inside of a sphere given by the equation

$$(x - 2)^2 + y^2 + z^2 - 4 = 0$$

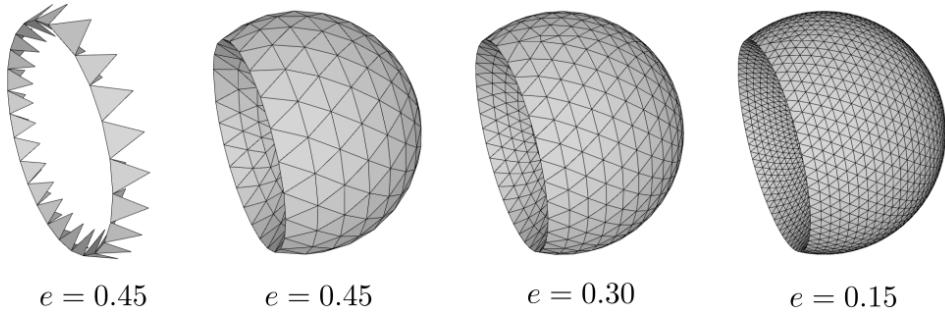


Figure 4.20: The surface given by the difference of two spheres.

is displayed in the Figure 4.20.

The difference of the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 5.5 = 0$$

and the inside of a tangle-cube given by the equation

$$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 11.8 = 0$$

is displayed in the Figure 4.21.

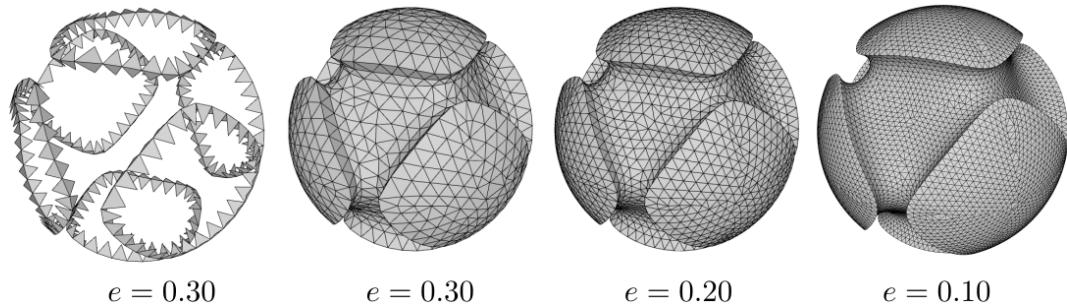


Figure 4.21: The surface given by the difference of a sphere and a tangle-cube.

The difference of the inside of a tangle-cube given by the equation

$$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 11.8 = 0$$

and the inside of a sphere given by the equation

$$x^2 + y^2 + z^2 - 5.5 = 0$$

is displayed in the Figure 4.22.

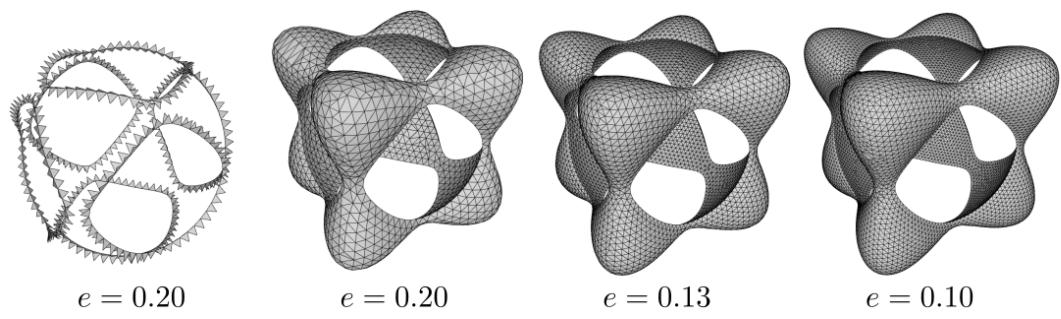


Figure 4.22: The surface given by the difference of a tangle-cube and a sphere.

4.4 Triangulation of a plane with A_{n--} singularities

In this section, we present the meshes¹ generated for the A_{n--} singularities glued to the plane. The presented meshes are uniform with different edge lengths.

The uniform mesh with a single A_{2--} singularity is displayed in the Figure 4.23.

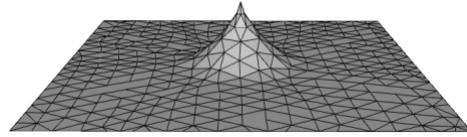


Figure 4.23: The triangulation of the A_{2--} singularity.

The uniform mesh with a two A_{2--} singularities is displayed in the Figure 4.24.

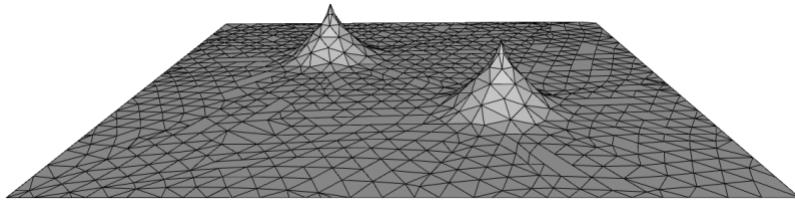


Figure 4.24: The triangulation of the two A_{2--} singularities.

The uniform mesh with one A_{2--} singularity and one A_{4--} singularity is displayed in the Figure 4.25.

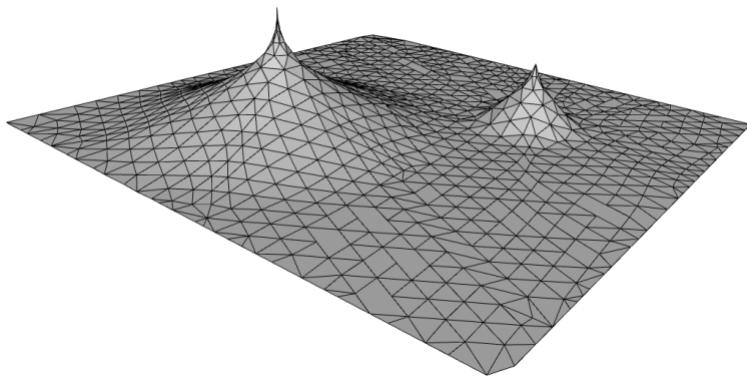


Figure 4.25: The triangulation of A_{2--} and A_{4--} singularities.

The uniform mesh with A_{1--} , A_{2--} , A_{3--} , A_{4--} singularities is displayed in the Figure 4.26.

¹Some mesh edges are not displayed due to zero angle between the incident triangles. This is an error in the used obj viewer.

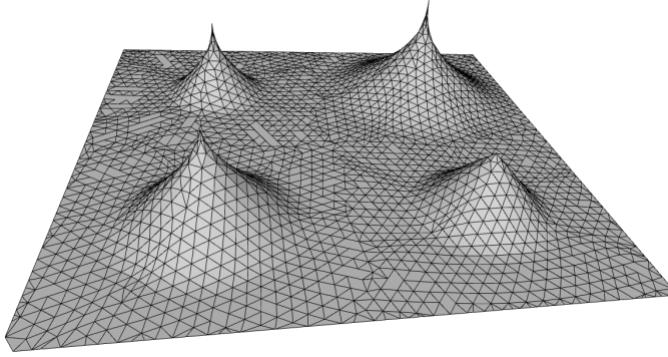


Figure 4.26: The triangulation of A_{1--} , A_{2--} , A_{3--} and A_{4--} singularities.

4.5 Runtime comparison of the reimplemented solution

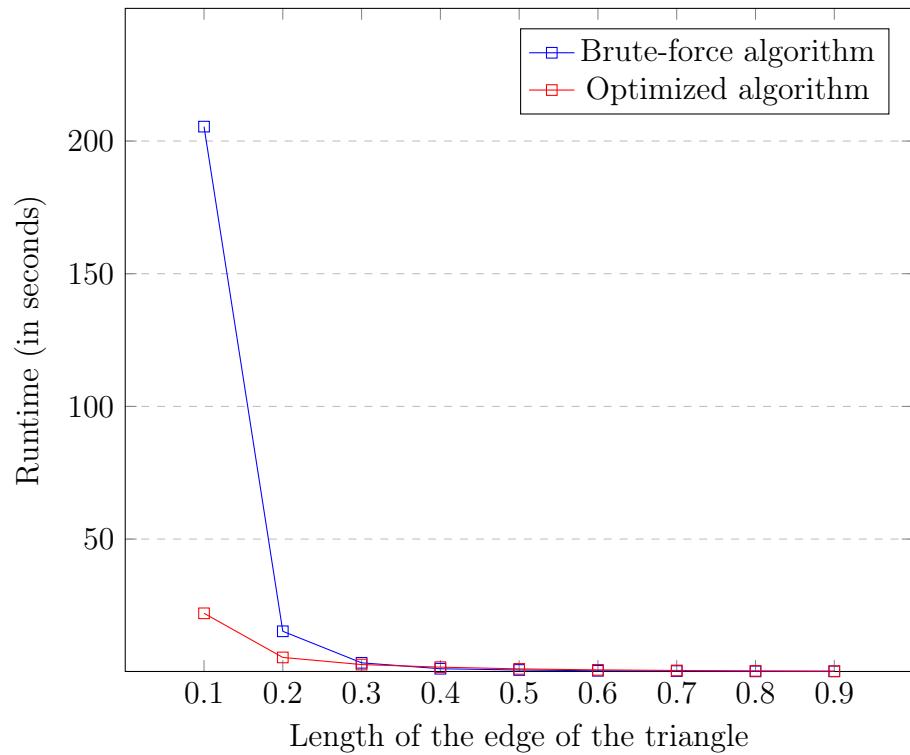
In this section, we compare the runtime of the reimplemented solution using half-edge and range tree to the brute-force solution implemented in [14].

To obtain the results, a laptop with a 6-core AMD Ryzen 5 5600U was used. We measured the runtime of the algorithm with a given edge length on a unit sphere.

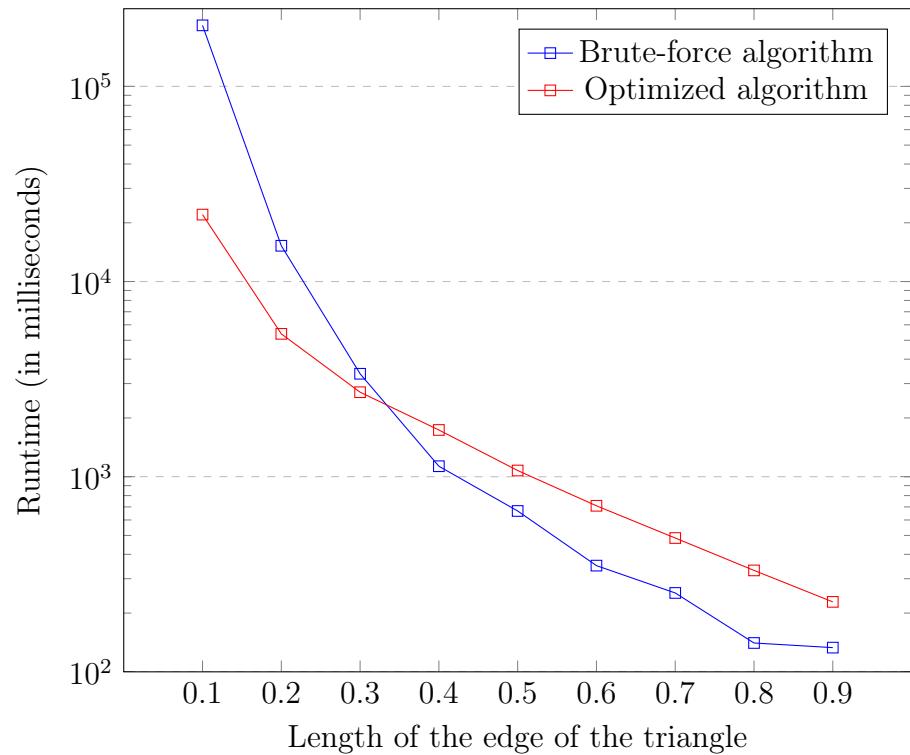
The dependence of the time from the length of the edge of the triangle is displayed on the Graph 4.27. For a better insight into the meshes with bigger triangles, the same values are displayed on the Graph 4.28 with a logarithmic scale.

As the edge size does not fully capture the size of the mesh, we provide the graphs of dependence of the runtime of the the algorithm from the number of triangles in the mesh. The dependence is displayed on the Graph 4.29. The dependence with logarithmic scale is displayed on the Graph 4.30.

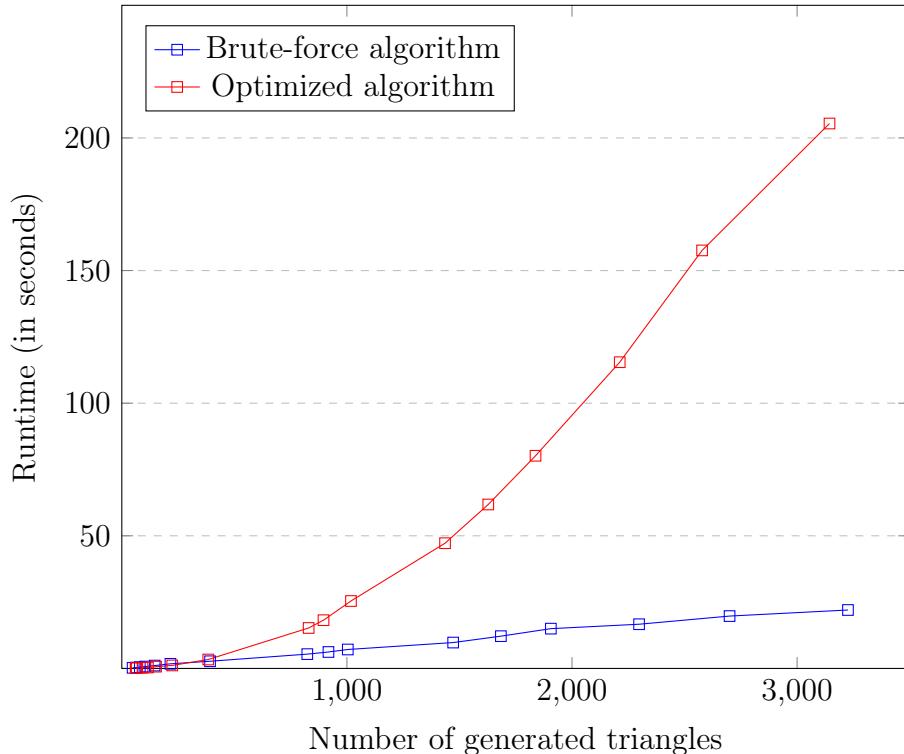
One can see that the brute-force algorithm beats the optimized implementation with the smaller meshes. However, for the bigger meshes, the optimized implementation is more than nine times faster for the biggest model.



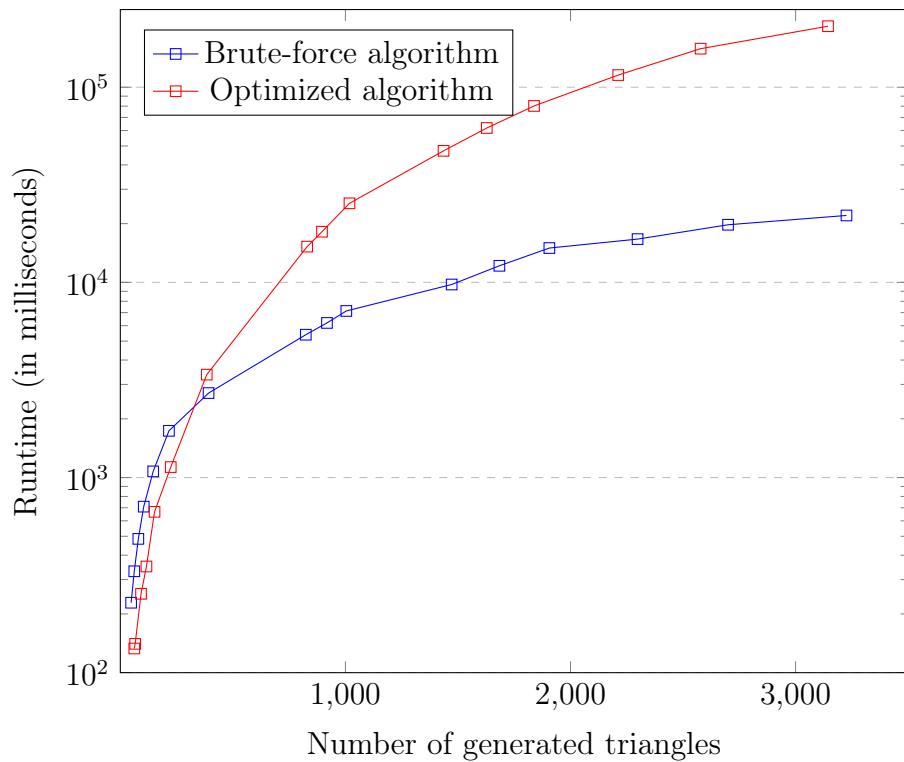
Graph 4.27: The dependence of the runtime on the input edge length for the unit sphere



Graph 4.28: The dependence of the runtime on the input edge length for the unit sphere (logarithmic scale)



Graph 4.29: The dependence of the runtime on the number of generated triangles



Graph 4.30: The dependence of the runtime on the number of generated triangles (logarithmic scale)

Conclusion

The goal of this work was to develop an approach to triangulate surfaces with certain types of singular points and singular curves. We developed an approach for creating a local mesh for ADE singularities and singular curves arising as the curves on the intersection of two regular surfaces. The local mesh could then be finished by using the algorithm for regular parts of the implicit surfaces.

We introduced analytical solutions for the A_{n--} singularities and one branch of the D_{n--} singularities. For other types of singularities, we introduced different numerical solutions based on the local geometry of the ADE singularities. Moreover, we introduced layers to improve the local approximation around the singular points. We optimized the numerical solution to create triangles closer to isosceles triangles to improve the mesh quality.

For the singular curves given by the intersection of two regular surfaces we created an approach for both – closed and open singular curves. The local mesh can be created for intersection, union or difference of the two regular surfaces.

To achieve the triangulation adaptive to the local curvature of the surface, we proposed a solution which determines the length of the edge of the triangle based on the constant of detail. The constant of detail is a parameter, which determines the relation between the size of the triangle and the curvedness in a given point.

Lastly, we reimplemented the algorithm for regular parts of the implicit surfaces using a half-edge data structure and a range-tree. We implement the solution in C++ using the GiNaC library [5] – open framework for symbolic computation within the C++ programming language. The reimplementation sped up the runtime of the algorithm significantly when creating the meshes with a large number of triangles.

In the section 4.2, we presented a comparison with the SingSurf software [17] in terms of the quality of the resulting mesh of ADE singularities.

In the future, one could develop an approach for the analytical triangulation of other ADE singularities. Other types of singularities could also be triangulated. For ADE singularities, we have the triangulation vectors given on the input. In the future, these could be calculated or approximated directly from the equation. For the CSG modelling, one could implement a solution for triangulation of the surfaces resulting from the intersection of multiple surfaces. In these surfaces, the singular curves can

include curve singular points. In the end, improving the algorithm for regular parts is possible, including creating a robust algorithm for fixing holes or postprocessing to improve the mesh quality.

Bibliography

- [1] CalcPlot3D - software for surface visualisation. <https://c3d.libretexts.org/CalcPlot3D/index.html>. [online: 12.5.2023].
- [2] Range tree implementation - GitHub repository. <https://github.com/Aj0SK/range-tree>. [online: 12.5.2023].
- [3] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. In *Computer Graphics Forum*, volume 20, pages 67–80. Wiley Online Library, 2001.
- [4] Vladimir Igorevich Arnold. 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.
- [5] Christian Bauer, Alexander Frink, and Richard Kreckel. Introduction to the GiNaC framework for symbolic computation within the C++ programming language. *Journal of Symbolic Computation*, 33(1):1–12, 2002.
- [6] 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.
- [7] Evgenii Borisovich Dynkin. The structure of semi-simple algebras. *Uspekhi Matematicheskikh Nauk*, 2(4):59–127, 1947.
- [8] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, and John F Hughes. *Computer graphics: principles and practice*, volume 12110. 1996.
- [9] Ron Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005.
- [10] M Hazewinkel, W Hesselink, D Siersma, and FD Veldkamp. The ubiquity of Coxeter-Dynkin diagrams. *Nieuw Arch. Wisk.*, 25:257–307, 1977.

- [11] 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.
- [12] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1):65–90, 1999.
- [13] Jan J Koenderink and Andrea J Van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992.
- [14] Kristína Korecová. Algoritmy triangulácie implicitne definovanej plochy. *Bachelor's thesis*, 2021.
- [15] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.
- [16] 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.
- [17] 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.
- [18] Tiago Novello, Vinícius Da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Differential geometry of implicit surfaces. 2021.
- [19] Aristides Requicha and Robert Tilove. Mathematical foundations of constructive solid geometry: General topology of closed regular sets. *Production Automation Project, University of Rochester, Rochester, New York 14627*, 1978.
- [20] Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. *CUMINCAD*, 1977.
- [21] Michael Spivak. *A comprehensive introduction to differential geometry*, volume 3. Publish or Perish, Incorporated, 1975.
- [22] 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

The process of calculating the C^1 – continuous connecting of a singularity to a plane.

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

$$f_2 = -x - q$$

$$f_3 = f_1 \cap f_2$$

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

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

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

$$f_6 = f_4 \cap f_5$$

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

$$f_7 = x$$

$$f_8 = f_6 \cup f_7$$

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

$$f_9 = x + q$$

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

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

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

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

The corresponding surfaces are displayed in the Figure 4.31.

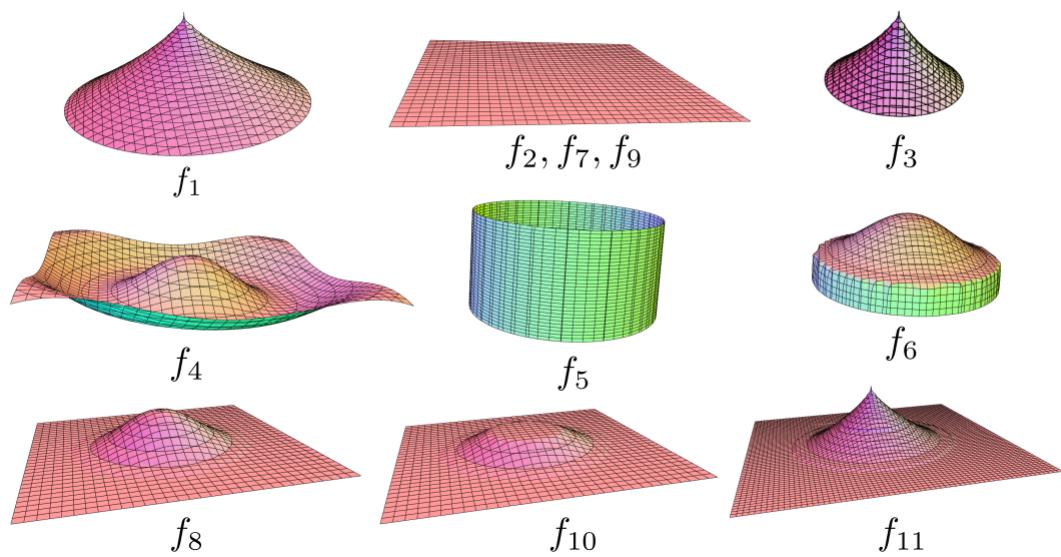


Figure 4.31: Attaching the singularity to a plane using CSG [1].

Appendix B

An examples of a triangulated surfaces containing ADE singularities.

The surface containing A_{2+-} singularity given by the equation $xz + y^2(x+y+z) = 0$ is displayed in the Figure 4.32 at the bottom.

The surface containing A_{2--} singularity given by the equation $x^2 + y^2 + z^3 + 3.2(x^3 - 3xy^2) = 0$ is displayed in the Figure 4.32 at the top.

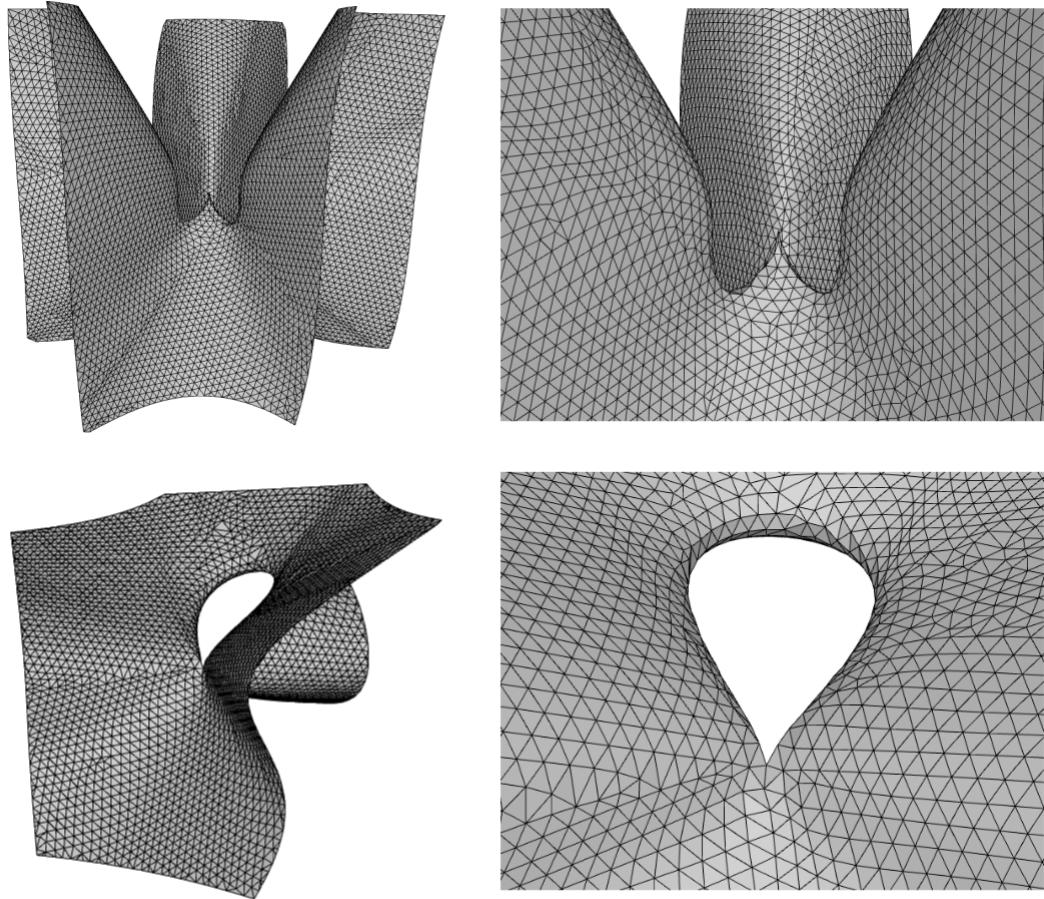


Figure 4.32: Triangulated surfaces containing ADE singularities.