# Curvature based remeshing for phase field based topology optimization

Tim Burg

A thesis presented for the degree of
Master of Science

Supervised by:
Professor Dietmar Hömberg
Hang Si


Technische Universität Berlin, Germany
November 2019

*I, Tim Burg confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.*

# Abstract

In this work I implement and apply a relatively new triangular mesh remeshing approach by (Dassi et al. 2016) that facilitates a surface interpolation by Radial Basis Fuctions (RBFs).

While serving a representation of the surface for remeshing this interpolation allows to obtain normalvectors to the surface. These are used in a higher-dimensional embedding sheme to yield a curvature adapted mesh i.e. smaller triangle-sizes where the curvature is larger.

As an application, I wrote a script that incorporates my version of the algorithm to remesh isosurfaces that I obtained from phase-field topology optimization calculations.

These topology optimizations were calculated on the servers of the Weierstraß Institute with their Finite-Element library pdelib. To extract the isosurfaces I programmed an stl-export function for this library.

# Acknowledgements

First and foremost I want to thank my Grandma who helped me out financially during the course of writing this thesis but who is unfortunately not around anymore to witness its submission.

Then I want to thank the researchers of the Weierstraß Institute and in particular Patricio Farrell, Hang Si and Moritz Ebeling-Rump who always helped me out with my queries.

I would also like to thank Timo Streckenbach who helped me with pdelib and the former WIAS researcher Thomas Petzold for their patience.

# Table of Contents

# Chapter 1

# Introduction & Overview

Triangular meshes are the most prominent representation of 3D surfaces in computer graphics and the go-to format for computer-aided design. Aside from the topological aspects of the described surfaces or non-manifold errors (which are an issue of their own in remeshing and additive manufacturing), these meshes, being piecewise linear, can only approximate curved surfaces. Often such approximations are inefficient in terms of their required data storage because the grid is isotropic and uses the same number of datapoints in areas of high curvature variation as in flat areas. This problem arises for example when the datapoints are obtained from some form of measurements (for example laser scanners) or are constructed from local algorithms like marching cubes or marching tetrahedra.

One such case which is the one treated here concerns the extraction of an isosurface-mesh of a scalar-valued function defined on a finite-element mesh. The isovalue-intersections may cut the finite-element simplices close to corners and edges resulting in very small and possibly distorted i.e. non-equilateral triangles. Variance of the surface is not accounted for in the usual isosurface algorithms and hence a mesh that is as dense as the fe-mesh is extracted. Subsequently various remeshing techniques may be used to yield an adaptive mesh. The method used for this in this work is a pliant remeshing algorithm with local mesh modification as defined in (Bossen and Heckbert n.d.).

The application that this fe-mesh is obtained from is a topology optimization using the phase-field method. Topology optimization is a form of optimization of a mechanical

structure that can adapt not only the shape but also the topology of the structure to yield a structure that is optimal with respect to a certain requirement.

Often a certain minimum stiffness or maximum give in a structure is required for it to perform its target application. Topology optimization can then yield designs that need less material or give better stiffness and hence a better performing part for the same material.

The phase-field is integral in this method and distinguishes material from void but (here) has a continuous range from 0 to 1 where 0 is void and 1 is material. Usually then the isosurface to the value 0.5 is used to determine the boundary of the solid-body.

## 1.1 Common approaches for remeshing

There are different approaches for remeshing a given triangular mesh that often involve some form of interpolation or approximation to find new vertices for the mesh or generate a whole new mesh altogether. There are also methods that try to find representations of meshes in frequency domains and then work from there.

Oftentimes the original piecewise-linear complex i.e. the original mesh is used to find the new vertices which means linear interpolation. For applications where a smoother output mesh is desired this is not suitable and smooth interpolation schemes need to be used.

However, all schemes that do use interpolation do respect the original mesh in the sense that the vertices of the mesh are still points on the surface. This is not the case if the original mesh is approximated or more precisely fitted with a functional description.

## 1.2 Radial basis function surface interpolation

For remeshing a surface one has to have a representation that can be queried for new on-surface points. Here implicit surface descriptions $S(x) = 0$ for an interpolating space-function $S : D \subseteq \mathbb{R}^3 \to \mathbb{R}$ where $D$ is he space of the interpolation data have practical benefits. Radial-basis-function(RBF) interpolation falls into this category and it is,

besides polynomial interpolation, among the most widely used interpolation schemes today. Radial-basis-functions are especially suited for multivariate data because the dimensionality of the data is only incorporated through the vectornorm of the associated space and as such is easy to implement.

To see this, consider the simplest case where the interpolant is given as a sum over scaled basis funtions centered at N data points $x_i$:

$$S(x) = \sum_i \alpha_i \varphi(\|x - x_i\|_2)$$

### 1.2.1 Higher dimensional embedding for curvature adaption

The RBF-interpolant can be easily differentiated analytically to obtain the gradient formula. This gradient, standing perpendicular on the 0-level-set describes the curvature of the mesh and is used for curvature adaptive remeshing. This is accomplished with an extension that incorporates the gradient in a 6D formulation: $(x, y, z, \sigma n_x, \sigma n_y, \sigma n_z)$ The Euclidean norm of this 6D vector is then used in the remeshing process. This essentially gives longer edges where the endpoints normals are different and induces a refinement.

Final thoughts on the reliability of this method for a general addaptive mesh can be found in the conclusive statement.

# Chapter 2

# Theoretical backgrounds

First I present the theory for the topology optimization that was used. This includes a brief recapitulation of linear elasticity and compliance minimization and then a short derivation of the optimality conditions and the discretizations that are applied to iterate towards them numerically. After that the extration of the isosurface, the radial basis function based surface interpolation and the remeshing operations are introduced.

## 2.1 Linear elasticity

### 2.1.1 The equations of static elasticity

Mathematical elasticity can be considered a branch of continuum dynamics whose research reaches as far back as the late 16th century. I only give a short outline of the stepstones to linear elasticity that are mostly based on
the book mathematical elasticity by Ciarlet which is a comprehensive standard piece on the topic.

Continuum dynamics deals with a body occupying a lipschitz-continuous reference configuration $\overline{\Omega} \subset \mathbb{R}^3$ under rest which is deformed to a configuration $\Omega \subset \mathbb{R}^3$ by applied forces. The deformation is described by an injective mapping $\varphi$ via a displacement field $u : \overline{\Omega} \mapsto \Omega$:

$$\varphi : \overline{\Omega} \mapsto \Omega \qquad \varphi = id + u$$

For the static case treated here, the deformation is time independent. The deformation and displacement mappings are required to be two times continuously differentiable but this reqirement can be relaxed in the variational formulation of the equations. I denote the coordinates in the reference configuration with $x$ and and those in the deformed configuration with $x^\varphi = \varphi(x)$. In engineering textbooks those coordinates are sometimes referred to as Lagrange- and Euler-coordinates respectively.

The elasticity theory is then build on the following two contributions from Cauchy of which the second is fundamental to continuum dynamics:

1. Axiom of force balance:

   Given volume- and surface-force-densities as $f^\varphi$ and $g^\varphi$ in the deformed configuration then for every subset $A^\varphi \subset \Omega$ the following equality holds:

   $$\int_{A^\varphi} f^\varphi(x^\varphi) dx^\varphi + \int_{\partial A^\varphi} t^\varphi(x^\varphi, n^\varphi) da^\varphi = 0$$

   Here, $dx^\varphi$ and $da^\varphi$ are the volume and surface elements in the deformed configuration, $n^\varphi$ is the surface-unit-normal and $t^\varphi$ is the Cauchy stress vectorfield defined as:

   $$t^\varphi : \Omega \times \mathbb{S}_1 \mapsto \mathbb{R}^3 \quad where \quad \mathbb{S}_1 := \{v \in \mathbb{R}^3 \mid \|v\| = 1\}$$

   Note that the Cauchy stress vector $t^\varphi$ depends on the given volume $A$ only through the normal vector at a surface point and that any surface-force dictated on part of $\partial A \cap \partial\Omega$ must be dispersed through the remaining part of $\partial A$.

2. Stress Tensor theorem:

   Assuming that $f^\varphi$ is continuous and $t^\varphi \in \mathrm{C}^1(\Omega) \cap \mathrm{C}(\mathbb{S}_1)$, then $t^\varphi$ is linear w.r.t. to the surface normal i.e.:

$$t^\varphi(x^\varphi, n) = T^\varphi(x^\varphi)n \qquad \forall x^\varphi \in \Omega, \qquad \forall n \in \mathbb{S}_1 \tag{2.1a}$$

and additionally

$$-\mathrm{div}^\varphi T^\varphi(x^\varphi) = f^\varphi \qquad \forall x^\varphi \in \Omega \tag{2.1b}$$

$$T^\varphi(x^\varphi) = T^\varphi(x^\varphi)^T \qquad \forall x^\varphi \in \Omega \tag{2.1c}$$

$$T^\varphi(x^\varphi)n^\varphi = g^\varphi(x^\varphi) \qquad \forall x^\varphi \in \Gamma_g^\varphi \tag{2.1d}$$

where $\Gamma_g^\varphi$ is the part of $\partial\Omega$ where the boundary condition $g$ is prescribed and $T$ is the Cauchy stress tensor with $\mathrm{div}T = \partial_j T_{ij}, e_i$ (See Ciarlet 1990 pp. 63–65 for the proof).

Notice that the formulation above uses the stress tensor in the deformed configuration where it is symmetric. The pullback of the tensor onto the reference configuration is achieved with the Piola-transformation after which it needs to be symmetrized again. This then yields the so-called first and second Piola-Kirchhoff-Stress-Tensors of which the latter is denoted with $\Sigma$. A change in the force densities due to the deformation is often ignored. Such forces are then called dead loads (see Ciarlet 1990 chap. 2.7).

The equilibrium equations for them are omitted here for brevity but the second Piola-Kirchhoff-Stress is the stress tensor to be determined in the next chapter.

Another thing that is important is that the boundary condition can and will only be prescribed on a part of the boundary.

### 2.1.2 Stess, strain and the equations of equilibrium in the linear case

So far the theory is valid for all continuums but there are also nine unknown functions, namely the three components of the deformation and the six components of the stress tensor. However, several simplifications can be made in the case of isotropic and homogeneous media that lead to a remarkably simple form of the tensor.

To this end the (right-) Chauchy (Green) strain tensor $C$ and its difference from unity $E$ is introduced. I will refer to $E$ simply as the strain tensor. They describe the first

order change in local length-scale under a deformation and are defined via the Fréchet derivative of the mapping $\varphi$ called the deformation gradient $\nabla\varphi$:

$$\nabla\varphi = \begin{pmatrix} \partial_1 u_1 & \partial_2 u_1 & \partial_3 u_1 \\ \partial_1 u_2 & \partial_2 u_2 & \partial_3 u_2 \\ \partial_1 u_3 & \partial_2 u_3 & \partial_3 u_3 \end{pmatrix}$$

$$C = \nabla\varphi^T \nabla\varphi = I + \nabla u^T + \nabla u + \nabla u^T \nabla u = I + 2E$$

Viewed in a different light, the deformed state can be considered a manifold with $C$ as the metric-tensor.

The simplification of the second Piola-Kichhoff-Stress-tensor follows these steps:

1. The stress tensor can only depend on $\varphi$ through its derivative $\nabla\varphi$ (Elasticity)
2. Material-Frame Indifference (invariance under changes of coordinates)
3. Isotropy of the material
4. Rivlin-Ericksen representation theorem for matrices
5. Homogeneity of the material

Details on these steps can again be found in (Ciarlet 1990 chap. 3). After following these steps, $\Sigma$ takes on the following form:

$$\Sigma(C) = \lambda(\mathrm{tr}E)I + 2\mu E + o(\|E\|)$$

Here, $\lambda$ and $\mu$ are the Lamé coefficients of the material and $I$ is the identity tensor. In the linear theory that is used as the basis for the topology optimization, the strain $E$ is replaced with its linearized version $\epsilon$:

$$\varepsilon = \frac{1}{2}(\nabla u^T + \nabla u)$$

this yields the following even simpler form of the tensor which is referred to as $\sigma$:

$$\sigma = \lambda(\nabla u)I + \mu\,(\nabla u + \nabla u^T)$$

The more prominent form of which is called Hooks-law and written with the 4th order stiffness tensor that is unfortunately also named $C$ in some literature:

$$\sigma = C : \epsilon \qquad \text{or} \qquad \sigma = C\epsilon \tag{2.2}$$

Here, the second form is written in vector notation for the components of the tensors and the following tensoroperation for rank 2 tensors is introduced that will be used in the coming sections:

$$G : V := \sum_{i,j} G_{ij} V_{ij}$$

Furthermore, in the following sections $\cdot$ denotes the Euclidean scalar product.

### 2.1.3 Variational formulation

For finite-element simulations and reduced smoothness requirements of the displacement, a variational formulation of the equilibrium equations (2.1) must be formulated.

For this we first define the space

$$H^1_D = \{\theta \in H^1(\Omega, \mathbb{R}^3) \mid \theta = 0 \text{ on } \Gamma_D = \partial\bar{\Omega} - \Gamma_g\},$$

to exclude the boundary on $\Gamma_D$ from contributing its force terms.

Multiplying equation (2.1b) with a test function $\theta$ from this space on both sides and integrating yields ($\cdot$ denotes the scalar product):

$$\int_{\Omega^\varphi} \mathrm{div}^\varphi T^\varphi \cdot \theta^\varphi \, dx^\varphi = -\int_{\Omega^\varphi} f^\varphi \cdot \theta^\varphi \, dx^\varphi + \int_{\Gamma_g^\varphi} g^\varphi \cdot \theta^\varphi \, da^\varphi ,$$

which has to hold for all test functions $\theta \in H^1_D$.

Using the Greens-formula for Tensor fields:

$$\int_{\bar{\Omega}} \mathrm{div} H \cdot \theta \, dx = -\int_{\bar{\Omega}} H : \nabla\theta \, dx + \int_{\Gamma} Hn \cdot \theta \, da$$

and applying the pullback to the reference configuration with the second Piola-Kirchhoff-Stress-tensor then gives:

$$\int_{\bar{\Omega}} \nabla\varphi\Sigma : \nabla\theta \, dx = \int_{\bar{\Omega}} f \cdot \theta \, dx + \int_{\Gamma} g \cdot \theta \, da \qquad \forall\theta \in H^1_D$$

For All vector fields $\theta : \bar{\Omega} \mapsto \mathbb{R}^3$ from $H^1_D$. This is also called the 'principle of virtual work' (in the reference configuration).

For very small strains the deformation gradient in front of $\Sigma$ can be dropped since the diplacement gradient multiplied with the stress-tensor is of second order in $\nabla u$. (see Braess 2013 p. 280)

Furthermore, since the product of a symmetric tensor with an antisymmetric one is zero, we split $\nabla\theta$ into: $\left[\frac{1}{2}(\nabla\theta + \nabla\theta^T) + \frac{1}{2}(\nabla\theta - \nabla\theta^T)\right]$

And thus we write the equilibrium equation in the variational form as:

$$\int_{\bar{\Omega}} \sigma \frac{1}{2}(\nabla\theta + \nabla\theta^T) = \int_{\bar{\Omega}} \sigma\varepsilon(\theta) = \int_{\bar{\Omega}} f \cdot \theta dx + \int_{\Gamma} g \cdot \theta da \quad \forall\theta \in H^1_D \tag{2.3}$$

For the following sections I denote this as the following shorthand notation using the inner product and the stiffness tensor from (2.2): $\langle A, B\rangle_C = \int_{\bar{\Omega}} A : CB :$

$$\langle\varepsilon(u), \varepsilon(\theta)\rangle_{C(\varphi)} = \int_{\bar{\Omega}} f \cdot \theta \, dx + \int_{\Gamma} g \cdot \theta \, da =: F(\theta) \qquad \forall\theta \in H^1_D \tag{2.4}$$

Notice that I sneaked a dependence of the stiffness tensor on a parameter $\varphi$ into the equation that will be made concrete in the following chapter. The well posedness of this weak formulation is proved using the Lax-Milgram Lemma and Korns Inequality in (Blank et al. 2014 Theorem 3.1)

For an actual deformation rather than a virtual one, the functional $F$ in (2.4) represents the compliance of the structure. This compliance is what is to be optimized in the following section.

## 2.2 Topology optimization

The topology optimization procedure used in this work is a direct implementation of the works of (Blank et al. 2014). As a result of this I frequently refer to this paper and those

relating to it and only strive to give a compact summary of the steps. A phase field based topology optimization was first introduced by (Bourdin and Chambolle 2003). It has has similarities to the SIMP-method[1] introduced by Bendsøe over a decade earlier (see (Bendsoe and Sigmund 2004) for a reference thereof).

Figure 2.1: Example of a topology optimised structure in 2D. source:(Ebeling-Rump et al. 2019)

The term topology optimization was coined in the context of optimizing mechanical structures. It is not bound to a certain implementation but to the requirement that a structure under load may change its topology under the optimization procedure. For this the facility of nucleation of holes in a previously filled material is generally needed.

The method used throughout this work which accomplishes this is based on minimizing a functional with gradient based optimization. The functional is constructed from a compliance term and a regularization term for the phasefield. In effect, compliance minimization maximizes the stiffness of the structure while a mass- or volume constraint formulated via the phase field controls where material is placed in the domain.

The use of a phase-function description allows to incorporate a computationally cheap perimeter regularization via an additional term in the optimization functional. This is needed since the compliance minimization in itself is not well posed and allows high variation in the microstructure of a part that cannot be manufactured and is not numerically stable. This can manifest itself as a checkerboard like pattern in the phasefield.

[1]Solid Isotropic Material with Penalization

### 2.2.1 Compliance minimization

Compliance is a very common goal function for topology optimization and defined as in equation (2.4):

$$F(u) = \int_{\bar{\Omega}} f \cdot u \, dx + \int_{\Gamma} g \cdot u \, da$$

Here, $u$ is the displacement solution of the mechanical system in the left-hand side of equation (2.4)

However, it is still open at this time how the compliance depends on the structure of the part. This dependence is actually encoded in the Stiffness-tensor $C$ of the mechanical system that will be constructed from the phase field after it is defined in the next section.

### 2.2.2 The phasefield description and regularization energy

For the modeling of structures either a level set method or a phase field description is viable. In case of the phasefield a continuous function is chosen that here can take on values in the range from 0 to 1 where 0 represents the void and 1 the material:

$$0 \leq \varphi \leq 1$$

A penalty term is then added to the optimization to force the phasefield to condensate to either 0 or 1 depending on a forcing term from the compliance minimization. Consequently an interface between the two phases forms whose change is subject to the resulting Allen-Cahn type equation which can drive the interface in the direction of its normal. This falls into the category of advancing-front algorithms. For details see (Barles et al. 1993) or (Blank et al. 2010). For an understanding of the convergence of the induced dynamics I consider the optimality conditions of the system.

Since integrating the phase field over a region gives the volume, a volume constraint is imposed by requiring the integral to be equal to a parameter $m$ which then dictates

how much of the design domain is allowed to be material:

$$\int_\Omega \varphi \, dx = m \, \mathrm{vol}(\Omega)$$

Bear in mind that values in the intermediate range of $\varphi$ distort this relationship. However, since they only occur in the interfacial region that will be forced to occupy a neglible portion of the domain, this is can be neglected.

For the reading convenience I stipulate the requirements on $\varphi$ in the following space:

$$\mathcal{G}^m = \left\{ \varphi \in H^1(\Omega, \mathbb{R})) \quad \middle| \quad 0 \leq \varphi \leq 1 \quad \text{and} \quad \int_\Omega \varphi dx = m \, \mathrm{vol}(\Omega) \right\}$$

These requirements are later taken care of by terms from the Karush-Kuhn-Tucker theory, namely the complementary slackness and a Lagrange multiplier.

As stated, an additional term has to be added to the compliance functional as to enforce a condensation of the phasefield and regularize the occurence of jumps. The term used is due to (Takezawa et al. 2010) and is called the Ginzburg-Landau Energy:

$$E^\epsilon = \int_\Omega \frac{\epsilon}{2} |\nabla \varphi|^2 + \frac{1}{\epsilon} \Psi(\varphi) dx$$

The potential $\Psi(\varphi)$ serves as the condensation potential that forces the phase field to either 0 or 1. Here, an obstacle potential is used while for the analysis, a differentiable double-well potential is considered:

$$\Psi_{\mathrm{dw}}(\varphi) = \frac{1}{4}(\varphi^2 - \varphi)^2$$

$$\Psi_{\mathrm{obs}} = \begin{cases} (1-\varphi)\varphi, & \text{if } \varphi \in [0,1] \\ \infty & \text{else} \end{cases}$$

The two potentials are displayed in figure fig. 2.2.

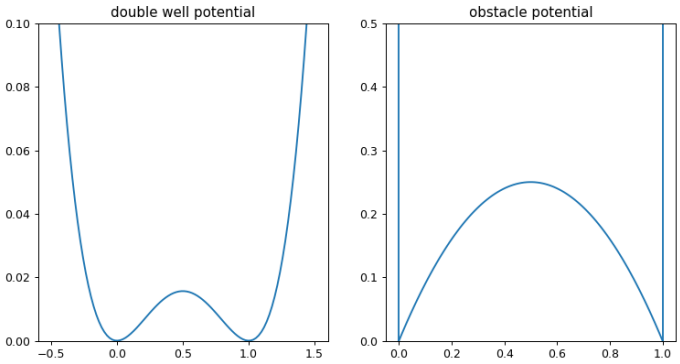



Figure 2.2: The two considered potentials for the Ginzburg-Landau energy term

### 2.2.3 Interpolation of the stiffness tensor and gravitational force

Using this phasefield, an interpolating stiffness tensor can be constructed. The stiffness tensor in the void is modeled by a very soft material $C_{\text{void}}$. For the interpolation in the interfacial region towards the material tensor $C_{\text{mat}}$ a linear interpolation with a superimposed transition function $t(\varphi) = \varphi^3$ is used:

$$C(\varphi) = C_{\text{mat}} t(\varphi) + C_{void}(1 - t(\varphi))$$

Also, the force $f$ occuring in the mechanical system can now be made concrete. Namely the phasefield acts as a direct scaling factor for the mass density $\rho_{\text{mat}}$ and excludes the void from contributing any forces. The gravitational force is here exemplary taken to act in the z-direction. This may change depening on the coordinates of the domain.

$$f = \varphi \, \rho_{\text{mat}} \, G_z \quad \text{with } G_z = 9.81 \, [0, 0, 1]^T$$

and the units of $G_z$ are in Newton('N').

$$F(u, \varphi) = \int_{\bar{\Omega}} \varphi \, \rho_{\text{mat}} \, G_z \cdot u \, dx + \int_{\Gamma_g} g \cdot u \, da$$

### 2.2.4 The optimal control problem

I now state the first order necessary optimality conditions for a minimum which are provided by the Karush-Kuhn-Tucker theory.

As indicdated before, the goal is to minimize the functional made up of the compliance and the Ginzburg-Landau term:

$$\min J(u,\varphi) := \gamma E(\varphi) + F(u,\varphi) \quad \text{with } \varphi \in \mathcal{G}^m \text{ and } u \text{ fulfills eq (2.4)}$$

Here $\gamma$ is a parameter that controls the influence of the regularization term and it's will value will determine the porosity/filigreeness of the obtained structure.

To write down the optimality conditions in a concise form, consider the control-to-state operator $S(\varphi) = u$ defined implicitly by equation (2.4). Its directional derivative in the direction $h \in H^1(\Omega, \mathbb{R})$, $S'(\varphi)h = p$ is given by the solution to:

$$\langle \varepsilon(p), \varepsilon(\eta) \rangle_{C(\varphi)} = \langle \varepsilon(u), \varepsilon(\eta) \rangle_{C'(\varphi)h} - \int_{\bar{\Omega}} h\, f \cdot \eta \, dx \quad \forall \eta \in H^1_D \tag{2.5}$$

where $p$ is also a function from $H^1_D$.(see Blank et al. 2014 theorem 3.3).

It follows from the definition of the total differential and the chain rule that the reduced functional $\tilde{J}(\varphi) = J(S(\varphi), \varphi)$ is Fréchet-differentiable with the derivative:

$$\tilde{J}'(\varphi)h = \frac{\partial}{\partial u} J(u,\varphi)p + \frac{\partial}{\partial \varphi} J(u,\varphi)h$$

Since $E(\varphi)$ is independent of $u$, the partial derivative with respect to $u$ is just the right-hand side of the state equation 2.4:

$$\frac{\partial}{\partial u} J(u,\varphi)p = F(p,\varphi)$$

Here I have used the fact that the Fréchet derivative of a linear functional in a direction $p$ is the functional applied to that direction.

This is, since $p \in H^1_D$ is also an admissible test function in the state equation, equal to the left hand side of the state equation. Note that if this was not the case, an auxillary state $q \in H^1_D$ could have been introduced which would solve a system sometimes called

the adjoint system as to make the following equality hold.

Now using equation (2.5) with $u$ as a test function this can be written as:

$$\frac{\partial}{\partial u}J(u,\varphi)p = F(p,\varphi) = \langle\varepsilon(u),\varepsilon(p)\rangle_{C(\varphi)} \quad = -\langle\varepsilon(u),\varepsilon(u)\rangle_{C'(\varphi)h} - \int_{\bar{\Omega}} h\, f\cdot u\, dx \tag{2.6}$$

The calculation of the partial derivative of $J$ in the direction $\xi \in H^1(\Omega,\mathbb{R})$ with respect to $\varphi$ is straightforward:

$$\frac{\partial}{\partial\varphi}J(u,\varphi)\xi = \gamma\epsilon\int_\Omega \nabla\varphi\cdot\nabla\xi\; dx + \gamma\int_\Omega \frac{1}{\epsilon}\Psi'(\varphi)\xi\; dx + F(u,\xi)$$

Summing up and using $\xi = h$ as the direction in which to derive, the reduced functional has the following directional derivative:

$$\frac{d}{d\varphi}\tilde{J}(\varphi)\xi = 2\,F(u,\xi) + \gamma\int_\Omega \epsilon\nabla\varphi\cdot\nabla\xi + \frac{1}{\epsilon}\Psi'(\varphi)\xi\; dx - \langle\varepsilon(u),\varepsilon(u)\rangle_{C'(\varphi)\xi} \tag{2.7}$$

To incorporate the constraints on $\varphi$ we now follow the Karush-Kush-Tucker theory. For this to work we have to make sure a constraint qualification is satisfied. Here I consider the slater condition that, for a homogeneous itermediate density distribution in the whole domain that satisfies the volume constraint, is obviously satisfied. Thus we can assume strong duality and the complentarity follows.

To reconsider, we have the following additional requirements

$$\int_\Omega \varphi - m\; dx = 0 \tag{2.8}$$

$$\varphi - 1 \leq 0 \tag{2.9}$$

$$-\varphi \leq 0 \tag{2.10}$$

Introducing Lagrange multipliers $\kappa \in \mathbb{R}$, and $\mu_+, \mu_- \in L^2(\Omega)$ the KKT first-order necessary optimality conditions read:

$$\frac{d}{d\varphi}\tilde{J}(\tilde{\varphi})\omega + \kappa\int_\Omega \omega\ dx + \mu_+\omega - \mu_-\omega = 0 \qquad \forall\omega \in H^1_D \tag{2.11}$$

$$\langle\varepsilon(\tilde{u}), \varepsilon(v)\rangle_{C(\varphi)} = F(\tilde{u}, v) \quad \forall v \in H^1_D \tag{2.12}$$

$$\int_\Omega \tilde{\varphi} - m\ dx = 0 \tag{2.13}$$

$$0 \leq \varphi \leq 1 \qquad \text{a.e. in } \Omega \tag{2.14}$$

$$\mu_+ \geq 0, \quad \mu_- \geq 0 \qquad \text{a.e. in } \Omega \tag{2.15}$$

$$(\mu_+, \tilde{\varphi} - 1) = 0 \qquad \text{a.e. in } \Omega \tag{2.16}$$

$$(\mu_-, -\tilde{\varphi}) = 0 \qquad \text{a.e. in } \Omega \tag{2.17}$$

Where the last three conditions arise due to complementarity. The existence of a minimizer to this problem is shown in (Blank et al. 2014, Theorem 4.1).

### 2.2.5 Numerial soution

#### 2.2.5.1 Pseudo time stepping

Suppose for a moment that $\kappa, \mu_+$ and $\mu_-$ were given. This is reasonable because in the next section, iterates for those multipliers are given by means of a Primal-Dual Active Set strategy. We then we could solve for $\tilde{\varphi}$ in the optimality conditions as follows. Equation (2.11) defines a linear functional on $H^1_D$ which I refer to as $\nabla\mathcal{L}(\omega)$. Using a scalar product this functional can be identified with a funtion on $L^2$ that we loosely call the gradient. This approach is called a gradient flow. Consequently a gradient descent in conjunction with a semi-implicit stepping scheme is employed.

$$(\partial_t\varphi, \omega) = \nabla\mathcal{L}(\omega)$$

Expanding the functional gives:

$$
\begin{aligned}
(\partial_t \varphi, \omega) =& \gamma \int_\Omega \epsilon \nabla \varphi \nabla \omega + \frac{1}{\epsilon} \Psi'(\varphi)\omega \, dx + 2\, F(u, \omega) \\
& - \langle \varepsilon(u), \varepsilon(u) \rangle_{C'(\varphi)\omega} + \kappa \int_\Omega \omega \, dx + \mu_+ \omega - \mu_- \omega
\end{aligned}
\tag{2.18}
$$

We start with some initial function $\varphi^k,\ k = 0$. Inserting for $\partial_t \varphi$ its approximation $\frac{\varphi^{k+1} - \varphi^k}{\tau}$ and using $\nabla \varphi^{k+1}$ for $\nabla \varphi$ we end up with:

$$
\begin{aligned}
\frac{1}{\tau} \int_\Omega \varphi^{k+1} \omega dx + \gamma \epsilon \int_\Omega \nabla \varphi^{k+1} \nabla \omega dx =& \frac{1}{\tau} \int_\Omega \varphi^k \omega dx \\
& + \frac{\gamma}{\epsilon} \int_\Omega \Psi'(\varphi^k)\omega dx + \langle \varepsilon(u), \varepsilon(u) \rangle_{C'(\varphi^k)\omega} \\
& - 2 \int_\Omega \omega \rho_0 g u dx + \kappa \int_\Omega \omega \, dx + \mu_+ \omega - \mu_- \omega
\end{aligned}
\tag{2.19}
$$

Where, as before, $u$ solves the mechanical system (2.4) and the primal feasibility (2.13) and (2.14) need to be cared for. This defines an iterative scheme for a descent.

#### 2.2.5.2 Primal-dual active set strategy

A sophisticated method to alleviate the Lagrange multipliers $\mu_+$ and $\mu_-$ from this equation is the Primal-Dual Active Set strategy (PDAS). The theory to this approach was developed in (Blank et al. 2013).

In essence, PDAS maintains a set of active constraints for every point.
A constraint is inactive if the corresponding Lagrange multiplier is zero and active if $\varphi$ takes on the corresponding bound - one of which has to hold due to the equations (2.16) and (2.17).

These sets are then updated by first solving the so-called primal problem which is (2.19) without $\mu_\pm$ but (2.13) explicitly cared for. With the then obtained $\varphi$ and $\kappa$ a dual problem for the $\mu_\pm$ Lagrange multipliers is solved.

More precisely, let $\mu = \mu_+ - \mu_-$ then (2.16) and (2.17) can be equivalently written as

$$c(\varphi(x) - 1) + \mu(x) \geq 0 \quad \text{and} \quad c(-\varphi(x)) + \mu(x) \leq 0$$

respectively for any $c > 0$.

With this, the active sets $\mathcal{A}^+$ and $\mathcal{A}^-$ as well as the inactive set $\mathcal{I}$ are defined as:

$$\begin{aligned} \mathcal{A}^+ &= \{x \in \Omega \mid c(\varphi(x) - 1) + \mu(x) \geq 0\} \\ \mathcal{A}^- &= \{x \in \Omega \mid c(-\varphi(x)) + \mu(x) \leq 0\} \\ \mathcal{I} &= \Omega \setminus (\mathcal{A}^+ \cup \mathcal{A}^-) \end{aligned} \tag{2.20}$$

To iterate, start with a guess of those sets, set $\varphi$ to 1 on $\mathcal{A}^+$ and 0 on $\mathcal{A}^-$. Then solve the unconstrained iteration step (2.19) with $\mu_\pm$ set to 0 and the primal feasibility cared for on the set $\mathcal{I}$. This is referred to as the primal problem. Note that due to the definition, $\mathcal{I}$ is synonymous to the interfacial region that takes up a very small portion of the space thereby making this calculation efficient.

Subsequently the Lagrange multiplier $\mu$ is updated on $\mathcal{A}^\pm$ with a dual formulation given by (Blank et al. 2013 (PDAS-I)) as follows:

$$\mu = \kappa - \frac{1}{\tau}(\varphi^{k+1} - \varphi^k) + \epsilon\gamma\Delta\varphi^{k+1} + \frac{\gamma}{\epsilon}\varphi^{k+1}$$

With this updated $\mu$, the active and inactive sets are recalculated and if no change is detected the descent step is complete. Otherwise reiterate the primal problem (using the same $u$) with the new different sets until a convergence is reached.

### 2.2.6 Isosurface extraction

After a configuration sufficiently close to the minimum has been found, an isofurface for $\varphi = 0.5$ is extracted that represents the surface of the part.

Since the Finite-Element-Mesh is providing a 3D-tesselation of the domain, which for my calculations consists of tetrahedra, the generation of an isosurface is handled as in the marching-tetrahedra algorithm (see Müller and Wehle 1999, also Treece et al. 1999).

Tetrahedra, as opposed to cubes, can only have 3 distinct cases of edge intersections that differ in terms of their makeup of triangular faces. No intersections, intersection at 3 edges(1 triangle) and intersection at 4 edges (2 triangles). See figure 2.3 for an illustration.

For the edge intersections, a linear interpolation of the values between two vertices is used. Possible intersections are then found via simple line intersections. If 3 intersections are found one triangle is created with the vertices of the intersections and if 4 intersections are found 2 triangles are generated. Susequently the ordering of the vertices is checked so that looking from the outside, the vertices are ordered counterclockwise in accordance with the stl-specification[2]. For this, the function values at the tetrahedra-nodes are considered to find a point that is inside (has a value greater than 0). This is especially important since the orientation is used in the remeshing procedure and can only be correctly determined at this step.

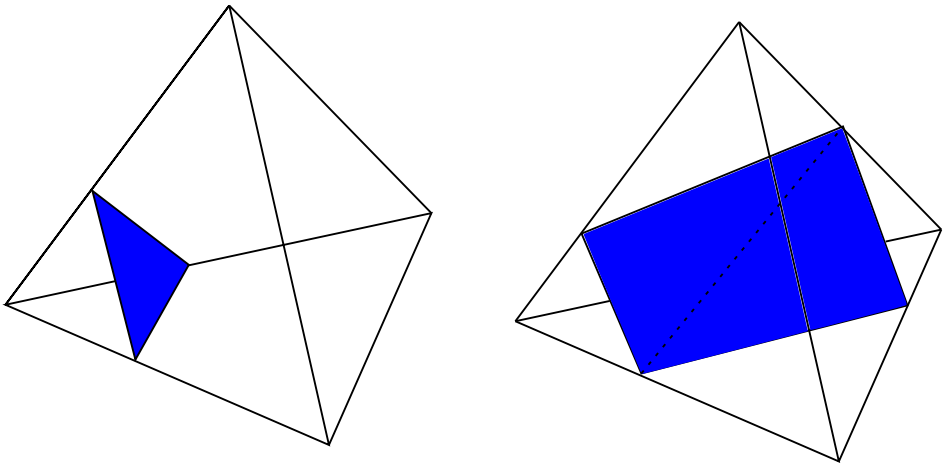


Figure 2.3: In the case of a 3D-Tetrahedra tesselation only 3 distinct cases can appear a)intersection at three edges(left) or b)intersection at 4 edges(right) or c)no intersections(not displayed)

## 2.3 Radial basis function theory

### 2.3.1 RBF Interpolation

Interpolation can be viewed as a special kind of approximation in which, for an approximant $S$ to some function $F$, it is demanded that $S$ reproduces the original functions

[2]stl is one of the primary file formats for triangular meshes

values at special points $x_i$ i.e.:

$$S(x_i) = F(x_i) \quad \forall x_i \in \Xi \tag{2.21}$$

Where $\Xi$ is some finite (possibly scattered) set of pairwise distinct points from $\mathbb{R}^d$ (multivariate) ie. a set $\{x_i \mid x_i \in \mathbb{R}^d, i = 1, .., N, x_i \neq x_j\}$. The functions considered here are scalar valued functions but a vector-valued interpolant may be constructed from scalar-valued-component functions.

Interpolants are constructed from some function space which in this case is made up of radial basis functions centered at the interpolation points. These are therefor often called interpolation centers. The dependence on the points means that the RBF function-spaces are individual to each dataset $\Xi$.

Radial basis functions themselves are multivariate functions constructed from univariate functions of the form $\phi : [0, \infty) \mapsto \mathbb{R}$ superimposed over the Euclidean norm:

$$\Phi(x) = \phi(\|x\|_2) \qquad x \in \mathbb{R}^d$$

Special monotonicity properties of those functions lead to unisolvent interpolation (unique solvability) as explained in the next section. Radial basis functions can also be introduced as general multivariate functions $\Phi : \mathbb{R}^d \mapsto \mathbb{R}$ that then need to be even: $\Phi(x) = \Phi(-x)$. The norm usually denotes the standard Euclidean norm which is essential for the convergence results. Some information about other norms is given in (Wendland 2005 pp. 83, 84). In the following $d$ will be 3 as required for the generation of implicit 2-d surfaces.

Notably, radial basis functions are special in that they allow easy interpolation of scattered multivariate data of arbitrary dimension with guaranteed existence and uniqueness results. At the time of writing this is unique to these functions and can not be achieved by for example "classical" polynomial interpolation.

As for the approximation quality, different interpolants do behave differently for the space in between the data sites and are distinguished by their approximation and or convergence properties for special classes of interpolated functions $F$. However, when no such functions (just the values $F(x_i)$) are given, the accuracy of an interpolation can not generally be assessed. Because this is the case here, determining qualities for

the RBF-interpolant in the context of surface interpolation are discussed in sec. 3.2.

The interpolant $S$ is constructed as a linear combination of scaled radial basis functions centered at the data sites:

$$S(x) = \sum_{j}^{N} \alpha_j \phi(\|x - x_j\|)$$

By introducing the interpolation matrix $\mathbf{A}$ as:

$$A_{ij} = \phi(\|x_i - x_j\|)|_{i,j} \tag{2.22}$$

we can write the interpolation condition (2.21) as:

$$\mathbf{A}\alpha = \mathbf{F} \tag{2.23}$$

where $\mathbf{F}$ contains the values $F(x_i)$.

The immediate requirement then is that this system is uniquely solvable which is determined by the invertibility of the interpolation matrix $A$

### 2.3.2 Existence and Uniqueness results

The invertibility of the interpolation matrix $\mathbf{A}$ for all pairwise distinct combinations of centers in $\mathbb{R}^d$ is considered. This property can be guaranteed for certain radial basis functions which satisfiy a montonicity property.

**Definition:** A continuous function $\Phi : \mathbb{R}^d \mapsto \mathbb{R}$ is called positive (semi-)definite if, for all $N \in \mathbb{N}$ and all sets of pairwise distinct points $X = \{x_1, ..., x_N\} \subseteq \mathbb{R}^d$ and all $\alpha \in \mathbb{R}^N$ the following quadratic form is (nonnegative) positive:

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j \alpha_k \Phi(x_j - x_k)$$

Analogously, I call the univariate functions $\phi$ positive (semi-) definite if the induced function $\Phi$ satisfies the definition.

Such induced positive (semi-) definiteness of a function can be shown via a property called complete monotonicity:

$$(-1)^l g^l(t) \geq 0 \quad \forall l \in \mathbb{N} \quad \forall t > 0$$

Where $g^l$ denotes the l-th derivative of a function $g : \mathbb{R} \mapsto \mathbb{R}$. The prototype of a complete monotone function is the exponential function $e^{-\alpha t}$ for some non negative $\alpha$.

**Theorem:** If $\phi(\sqrt{\cdot})$ is completely monotone on $[0, \infty)$ and not constant, then $\phi$ is positive definite.

The proof requires some theory on measure spaces and generalized fourier transforms and rests on the Bernstein-Widder representation of such functions. Since these tools are out of scope I refer to (Wendland 2005 theorem 7.14).

A first formal proof of a positive definite (pd) function was completed for the multiquadratics function by (Micchelli 1986). Multiquadratics are not completely monotone but rather one can show that it actually suffices that the first derivative of $\phi$ is completely monotone (Buhmann 2003 theorem 2.2).

Complete monotonicity of only some derivative motivates a weaker concept called conditionally positive definiteness (cpd) and requires an added polynomial for unisolvence. This is described in the Appendix sec. 5.2.

### 2.3.3 Commonly used Radial basis functions

I now state some of the more often used radial basis functions and give some detail on the ones with local support that were used for the actual implementation. During the course of writing the program it became clear that only local basis functions would be good candidates for a surface interpolation due to the number of vertices used in most triangular meshes and the resulting size of a dense interpolation matrix. But also the conditon number of the interpolation matrix of globally supported functions is problematic for large interpolation sets. To this end the compactly supported basis functions are also much more forgiving. I recite some estimates of the condition number in the Appendix.

Local, piecewise polynomial RBFs are generally dimension dependent in that they are not positive definite for $d > d_0$ in $\mathbb{R}^d$ where $d_0$ depends on the function. For this application $d_0 = 3$ is required and reflected in the chosen Wendland function $\phi_{3,1}$.

The Wendland functions (see Wendland 1995) are positive definite and of minimal degree with respect to the space dimension and smoothness. For the surface interpolation I use the twice continuously differentiable function and it's derivative in table tbl. 2.2

Table 2.1: RBF functions with global support

| function | name | definiteness |
|---|---|---|
| $e^{-r^2}$ | gaussian | pd |
| $\sqrt{r^2+1}$ | multiquadratics | pd |
| $1/\sqrt{r^2+1}$ | inverse multiquadratics | pd |
| $r^3$ | polyharmonic spline | cpd |

Table 2.2: Local RBF functions introduced by Wendland (Wendland 1995)

| function | name | definiteness | smoothness |
|---|---|---|---|
| $(1-r)_+^2$ | $\phi_{3,0}(r)$ | pd | $C^0$ |
| $(1-r)_+^4(4r+1)$ | $\phi_{3,1}(r)$ | pd | $C^2$ |
| $(1-r)_+^6(35r^2+18r+3)$ | $\phi_{3,2}(r)$ | pd | $C^4$ |
| $(1-r)_+^8(32r^3+25r^2+8r+1)$ | $\phi_{3,3}(r)$ | pd | $C^6$ |

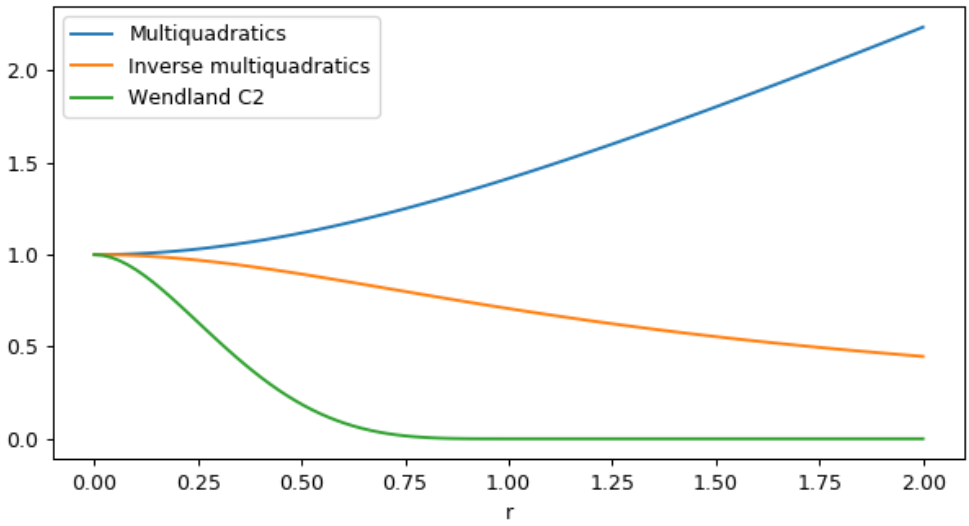



Figure 2.4: Comparison of different RBF functions. Note that some RBFs can grow to infinity. However, the Wendland functions become zero after r=1

### 2.3.4 Scaling of RBF functions, ambiguities and interpolation properties

The Wendland RBF functions have a fixed support radius of 1 as seen in fig. 2.4. Since spacing of the interpolation data is not fixed between models, a scaling of the radial argument needs to be introduced that scales $r$ such that the RBFs extend into the space between the data sites. Otherwise the interpolant might just have, in the exteme case, spikes at the sites to attain the required values. To this end I scale $r$ with a scale parameter $c > 0$ as $r' = r/c$ since that makes the Wendland functions extend to exactly the value of $r'$.

This scaling parameter, in general could be nonuniform over the interpolated values but this comes with uncertainty for the solvability of the interpolation system.

Moreover, as previously noted, it cannot be generally assessed which value of a scale parameter is more accurate in an interpolation unless there is a target to which the interpolant can be compared. See fig. 2.5 for an illustration.

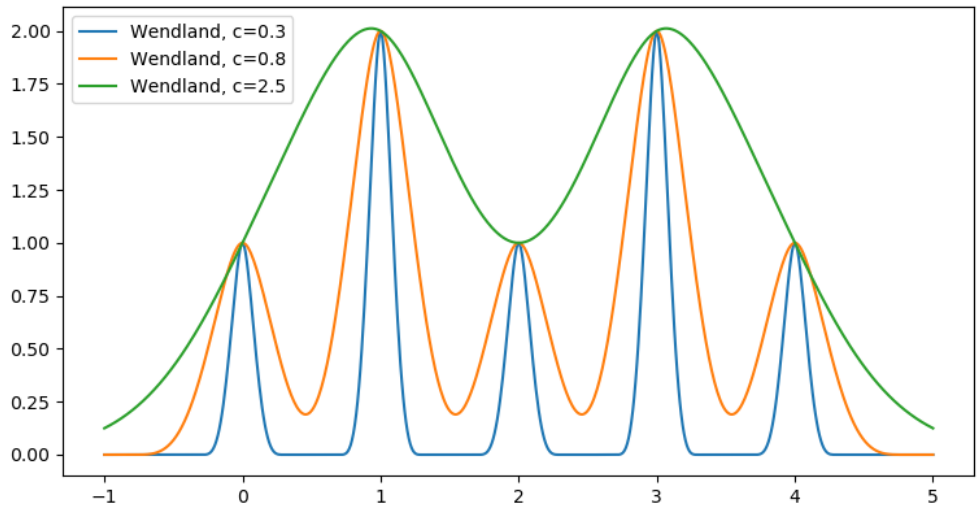



Figure 2.5: Wendland C2 functions for different scaling parameters c. The interpolation values were set to (1,2,1,2,1) at (0,1,2,3,4).

fig. 2.6 aims to illustrate the validity of the interpolation condition for a special case of a 2d interpolant.

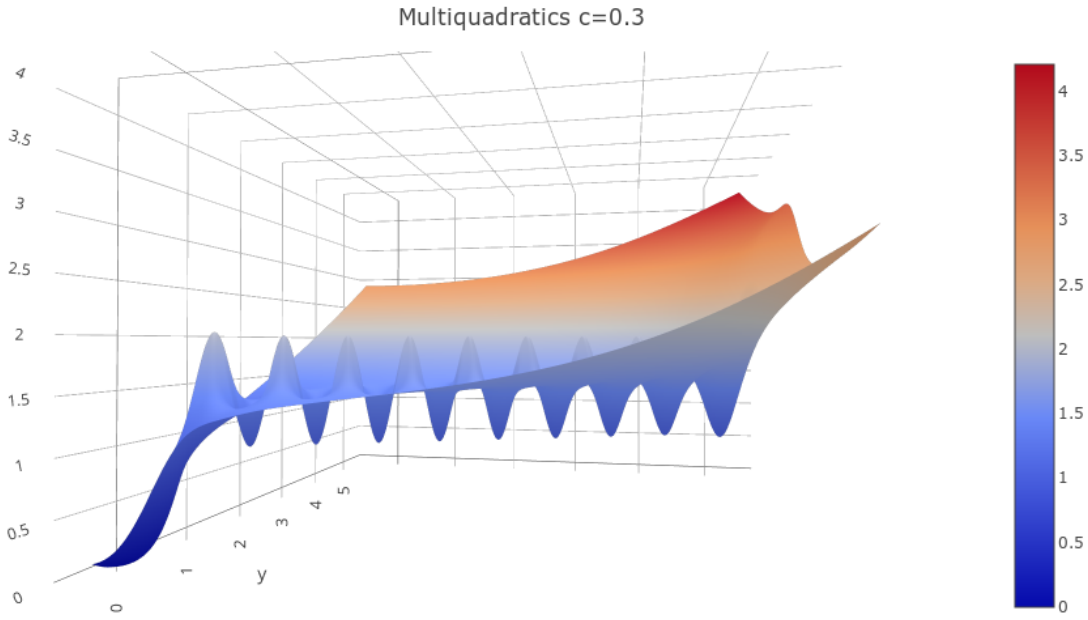



Figure 2.6: Different Radial-Basis-Funtions may have different behaviours for off-site values. Multiquadratics can even grow toward infinity. Displayed is a an alternating 1-2 sequence of along a line in $\mathbb{R}^2$

### 2.3.5 SURFACE INTERPOLATION

Surface descriptions are either explicit or implicit. Explicit means that the surface is the graph of a function $F : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$ which can be very difficult to construct. Especially for complicated topologies, this can usually be only done via 2d-parametric patches of the surface which are difficult to match at the boundaries. Implicit surfaces on the other hand are defined via a functions level set (usually the zero level i.e. $F(x) = 0$) which is easier to construct but is harder to visualise. Common methods for visualization include marching-cubes and raytracing methods.

For the surface interpolation with an implicit function this translates to the interpolant being zero at the data sites: $S(x_i) = 0$. Since the zero function would be a trivial solution to this, off-surface constraints must be given. This is usually done with points generated from unit-normalvectors to the original surface if such a surface exists. The pointvalues are then assigned the values of the signed distance function to the surface:

$$S(\mathbf{x}_i + \epsilon \mathbf{n}_i) = F(\mathbf{x}_i + \epsilon \mathbf{n}_i) = \epsilon \tag{2.24}$$

For a stronger falloff(rise) of the interpolant multiples of $\epsilon$ may be chosen as interpolation values on the right hand side. If normalvectors are not available, they can be generated from a cotangent plane that is constructed via a principal component analysis of nearest neighbors (Wendland 2005 p. 3).

In my case the vectors could be obtained from an average of the normals of the adjacent triangles that a vertex takes part in scaled with the inverse of the distance to the respective triangle centroids:

$$\mathbf{n}_\mathbf{v} = \sum_{T \in \mathcal{N}_T} \frac{1}{\left\| \mathbf{v} - \mathbf{v}^\mathbf{T}_{\text{cent}} \right\|} \mathbf{n}_T$$

where $\mathbf{v}^\mathbf{T}_{\text{cent}}$ is the centroid of the triangle.

These offset-points were generated for every vertex of the original mesh and in both directions (on the inside and on the outside) such as to have a guaranteed area of convergence for a gradient descent projection.

## 2.4 Remeshing operations

Different approaches exist to remesh a surface. Most fall into one of the following categories:

- triangulate a commpletely new mesh, usually with delauney triangulation and go from there
- incremental triangulation, with new nodes inserted or removed one at a time.
- local mesh modifications / pliant remeshing

Additionally most methods utilize some form of vertex-smoothing as this is an straightforward iterative procedure that improves the mesh globally and is guaranteed to converge.

The approach used here falls into the last category above and uses consecutive loops of local mesh modifications of the following kinds:

- Edge collapse
- Edge split
- Edge flip
- Vertex smoothing

Which of the modifications is applied depends on the length of an edge in comparison to a target length. These modifications are now exemplified shortly.

### 2.4.1 Edge collapse

Edge collapse, as the name suggests, removes an edge from the mesh thereby deleting two adjacent triangles and removing one point. Special conditions have to be checked as there are certain configurations that would result in an illegal triangulation. See fig. 2.8 and fig. 2.9. To avoid having to project a new midpoint to the surface, the two vertices of the edge are joined at either one of them.

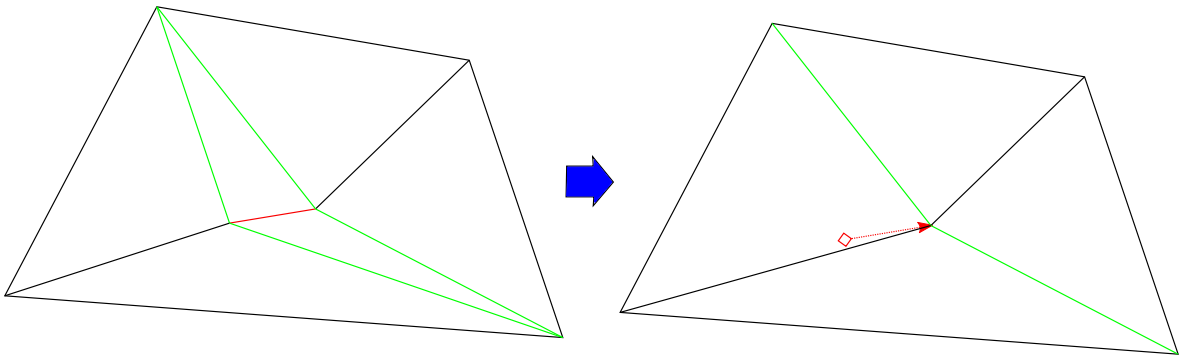


Figure 2.7: Edge collapse with the new point at one of the endpoints.

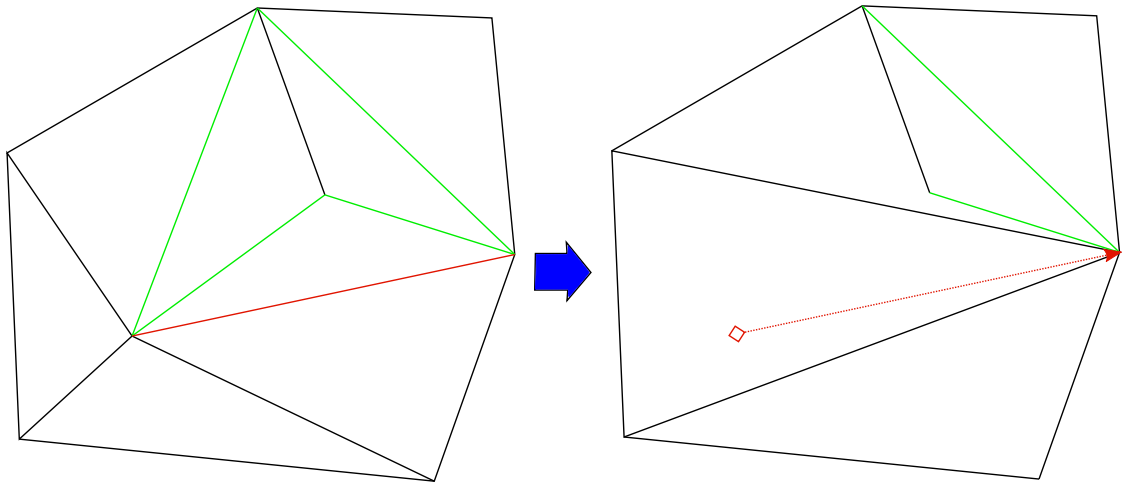


Figure 2.8: Illegal edge collapse with more than two common neighbors of the edges endpoints.

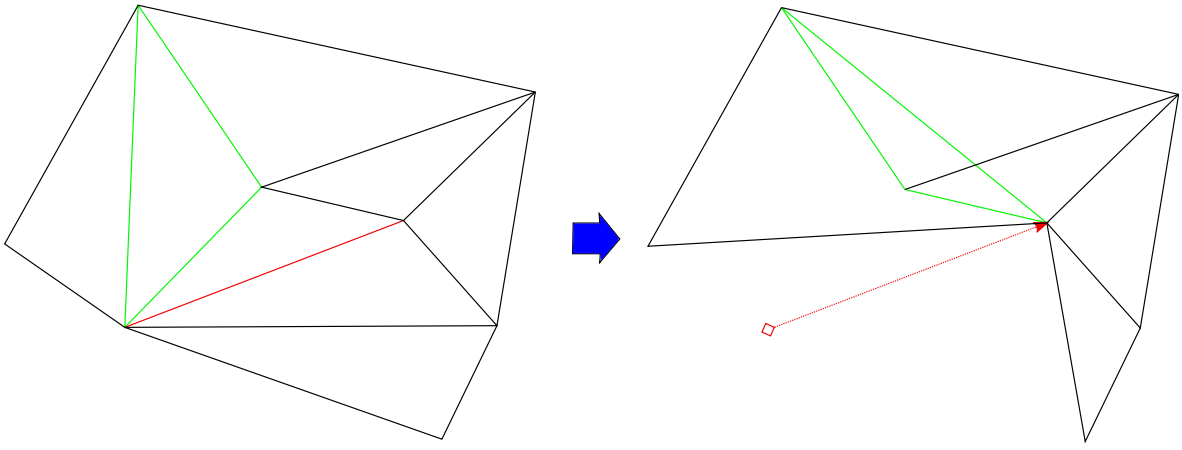


Figure 2.9: Illegal edge collapse with a triangle flip.

Both cases are easily cared for. The case of more than two common neighbors can be

checked in a graph datastructure and to check if a triangle was flipped the normals before and after the operation have to be compared.

### 2.4.2 Edge split

The edge split is a straightforward operation as no special cases have to be taken care of. A new vertex is put at the surface projected midpoint of the existing edge and 4 new edges as well as 4 new triangles replace the split edge and it's adjacent triangles.

The only pitfall than can occur is that the projection of the midpoint with a gradient descent can sometimes project inside another triangle therefore yielding flipped triangles. This again needs to be checked with a normal-flip check.

### 2.4.3 Edge flip

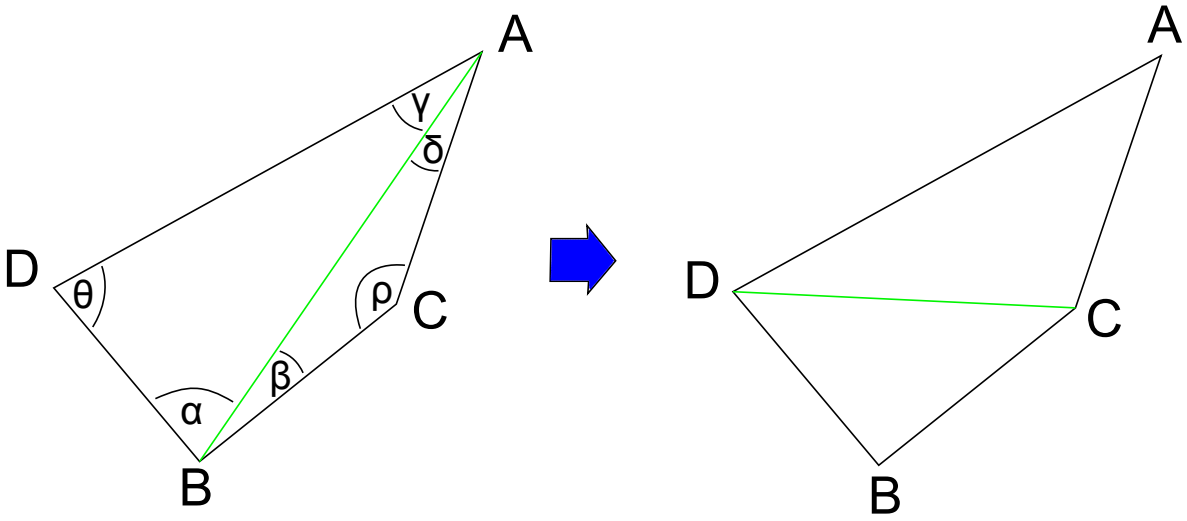



Figure 2.10: Edge flip.

An edge flip can dramatically increase the aspect ratio of a triangle if the right conditions are met. Consider the setup in fig. 2.10. The edge AB is flippable if:

- The edge does not belong to the boundary of the mesh
- The edge CD does not already belong to the mesh

- $\alpha + \beta < \pi \quad \text{and} \quad \gamma + \delta < \pi$
- The angle between the normals of the triangles is not too big to not cast "ridges"

I do a flip according to (Dassi et al. 2016) if the 6d versions of the angles opposite the to-be flipped edge. namely $\theta$ and $\rho$ together are larger than $\pi$:

$$\theta^{6d} + \rho^{6d} \geq \pi$$

For the definition of 6d angles see sec. 2.5.

### 2.4.4 Vertex smoothing

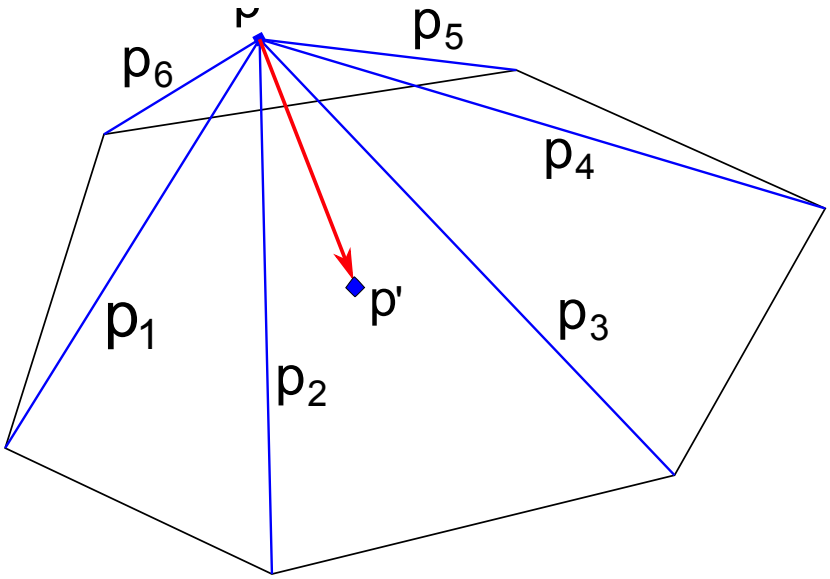



Figure 2.11: Vertex smoothing.

Vertex smoothing finds a new position for a given vertex based on the distance to its neighbors according to the following formula:

$$\mathbf{p}' = \mathbf{p} + \alpha \sum_{j \in \mathcal{N}} f(\|\mathbf{p} - \mathbf{p}_j\|)(\mathbf{p} - \mathbf{p}_j)$$

Wherein $\mathcal{N}$ stands for the neighbors, $\alpha$ is a normalization constant and $f$ is a weight

function. Different weights have been investigated in (Bossen and Heckbert 1998) where they constructed a weight function that results in very isotropic grids. Given a target edge length $t$ and an actual edge length $l$ a normalized edge length is defined as $d = l/t$ and the weight function reads:

$$f(d) = (1 - d^4) \cdot e^{-d^4}$$

This function pushes if $l < t$ and slightly pulls if $t > l$ as opposed to the frequently used Laplace-smoothing which pulls proportional to the distance. The function is plotted in fig. 2.12 versus the Laplace weights. Additionally, I clip the shiftdistance $p' - p$ to 80% of the minimal heights of the adjacent triangles. This is done because moves that exceed this distance are likely to cause triangles that are excessively tilted against the surface or even flipped. These then cause problems in later operations.

One of those problems occurs due to the pushing nature of the smoothing that tends to squish larger triangles into thin strips when attached short edges are pushing the vertex into the direction of least resistance. Thin strip triangles should then usually be flipped and consequently push back but this can fail due to the requirements for the edge flip and consequently very thin triangles that I call thin strip triangles may persist.

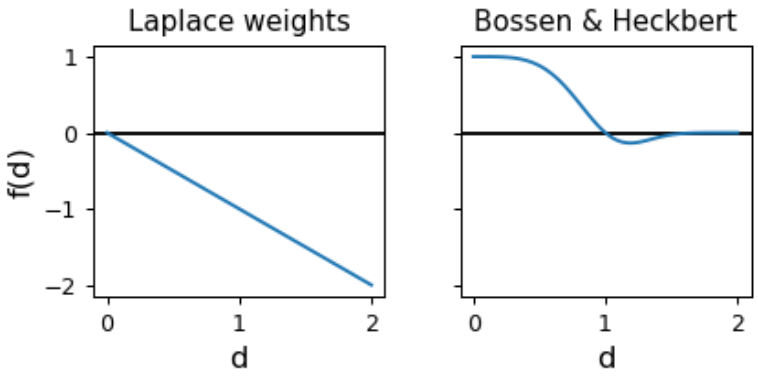



Figure 2.12: The weight function used compared to the Laplace weights.

### 2.4.5 Projection of vertices onto the surface

Both in an edge split as well as in vertex smoothing a constructed new vertex must be projected onto the surface. For this I use a gradient descent iteration with a fixed steplength of one. This steplength has proven much faster convergence than the exact steplength and is due to the fact that around the surface the slope of the interpolant is one by construction. The projection algorithm works as follows.

**Algorithm 1:** Project a vertex $x_0$ onto the surface

**Input :** $x_0$ and *eps* and interpolant $f$

```
1  i=0
2  while i < steplimit do
3  |  if i mod 3==0 then
4  |  |  calculate ∇f(x_i)
5  |  calculate f(x_i)
6  |  if f(x_i) < eps then
7  |  |  return x_i
8  |  stepsize = f / ||∇f(x_i)||
9  |  clip stepsize
10 |  x_{i+1} = x − stepsize · ∇f(x_i)
11 end
```

## 2.5 Higher dimensional embedding

The higher dimensional embedding causes a change in geometry that effectuates the isotropy of the mesh. Namely, the pointnormals are included in an edges length calculation as to enlarge the edge when the normals differ. Thereby, the the enlarged edges are remeshed more finely. Formally this reads as follows. Given a vertex $x$ on the surface, it is concatenated with the surface normal $n$ at this point to a 6 dimensional vector via the following embedding:

$$\Psi(x) = (x, y, z, \sigma n_x, \sigma n_y, \sigma n_z)^T$$

Here $\sigma$ is a parameter of the embedding and in effect controls how much an edge with

differing normals will be enlarged. With $\Psi$ the edgelength between two points $a$ and $b$ are defined with the 6d Euclidean scalar product as:

$$l_{ab}^{6d} = \|\Psi(a) - \Psi(b)\|_{6d} = \sqrt{(\Psi(a) - \Psi(b), \Psi(a) - \Psi(b))_{6d}}$$

And in the same manner an angle between the points $a, b, c$ is defined via:

$$cos(\theta_{abc}^{6d}) = \frac{(\Psi(a) - \Psi(c), \Psi(b) - \Psi(c))_{6d}}{l_{ac}^{6d} l_{bc}^{6d}}$$

These edgelengths are subsequently used as the regulator for the local mesh modifications in the remeshing algorithm while the 6d angles are used in the edge flips.

The overall idea of this embedding is then that a curvature adapted mesh corresponds to an isotropic mesh in the 6d space.

# Chapter 3

# Algorithm description and implemenation details

## 3.1 interpolation

The vertices of the extracted isosurface are the principal points for the interpolation.

As stated in sec. 2.3.5, additionally, the off-surface values both in the positive as well as the negative direction are incorporated into the interpolation. These are calculated via the original meshes vertices $v$ and triangle normals $n_T$ as follows:

$$\mathbf{v}_{\text{off}} = \mathbf{v} \pm \varepsilon \frac{\mathbf{n_v}}{\|\mathbf{n_v}\|} \tag{3.1}$$

where $\mathbf{n_v}$ is assembled of the triangles $\mathcal{N}_T$ containing $v$ as a vertex:

$$\mathbf{n_v} = \sum_{T \in \mathcal{N}_T} \frac{1}{\|\mathbf{v} - \mathbf{v}^{\mathbf{T}}_{\text{cent}}\|} \mathbf{n}_T$$

Here $\mathbf{v}^{\mathbf{T}}_{\text{cent}}$ is the centroid of the triangle.

Since the normals of the triangles point outward of the structure as in the stl specification we have higher interpolant values outside the object and a gradient pointing outwards.

Experimentation with $\varepsilon$ only yielded surfaces that were vertiable for projection for small values. I settled with a general forumla setting $\varepsilon$ as the average between the longest edge in the mesh and the smallest edge divided by 10, i.e.:

$$\varepsilon = \frac{e_{\text{longest}} + e_{\text{shortest}}}{20}$$

For the actual RBF interpolation the scale factor of the Wendland function was set to 2.5 times the longest edge in the mesh where lower values did reduce the convergence-rate of successfull projection onto the surface.

## 3.2 surface conditioning

A principal consideration is that of the values of the interpolant in between the datasites. In the case of an implicitly defined surface this influences not only the shape of the zero level set but also the slope at the zero crossing. The latter is plotted in fig. 3.1. Displayed are the interpolant values along the triangle normals of the initial model which were centered at each triangles centroid. The zero crossing mostly occur in the neighborhood of zero where shifts are expected from the smoothness of the surface. However, for some normal traces or plots a stronger deviation can be made out and the existence of a zero crossing is questionable. If in fact the scale parameter is chosen too small then there might be no zero crossings at all for some triangles which in the best case will only inhibit refinement in that area as the vertex projection cannot converge. In the worst case though the projection might yield a vertex that results in an invalid mesh (flipped triangle or non-manifold surface or self intersection).

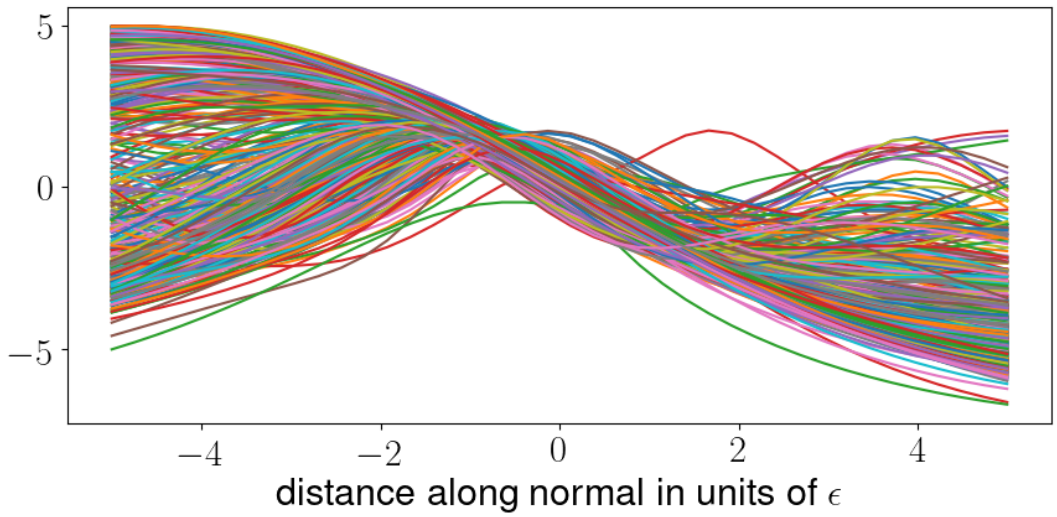
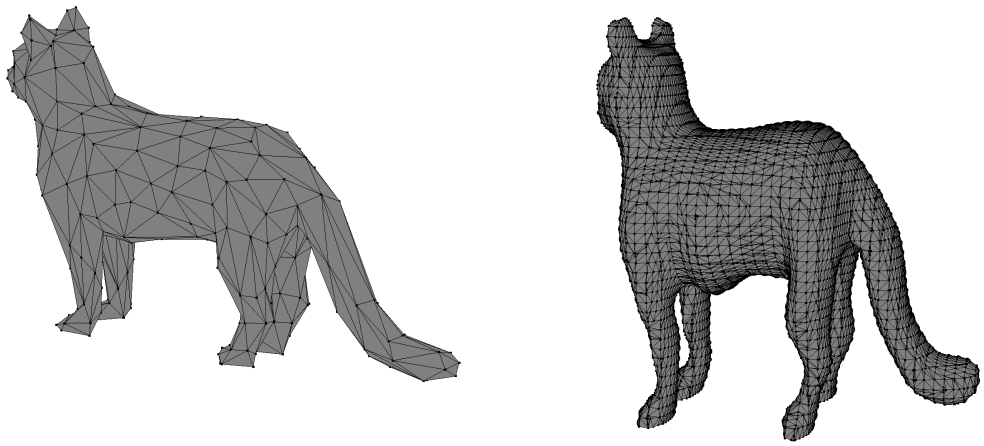



Figure 3.1: Interpolant values along the normal direction sampled at the triangle centroids for the cat model.

The interpolation matrix 2.22 is then constructed as a sparse matrix with a scale factor $c$ as explained in sec. 2.3.4 and the system 2.23 is subsequently solved for the coefficients.

Throughout the construction of the algorithm a cat model was used for testing that is pictured in figure 3.2 together with it's interpolated surface (the isosurface was generated with marching cubes). The smoothness of the interpolant can be assessed visually.



(a) (b)

Figure 3.2: An isosurface extracted via marching cubes of an interpolated cat model. a)input mesh, b)interpolated mesh.

An intersection through the cat surface is shown in fig. 3.3. The isolines show that the surface is generally well behaved with the isolines being mostly parallel. Deviations can occur in areas of high curvature of the original mesh (the feet) and where other parts of the model are in proximity (this is no issue in this case).

For models with very close proximity of parts (in relation to the spacing of the original mesh) the slope around zero given by equation (3.1) should be adapted.

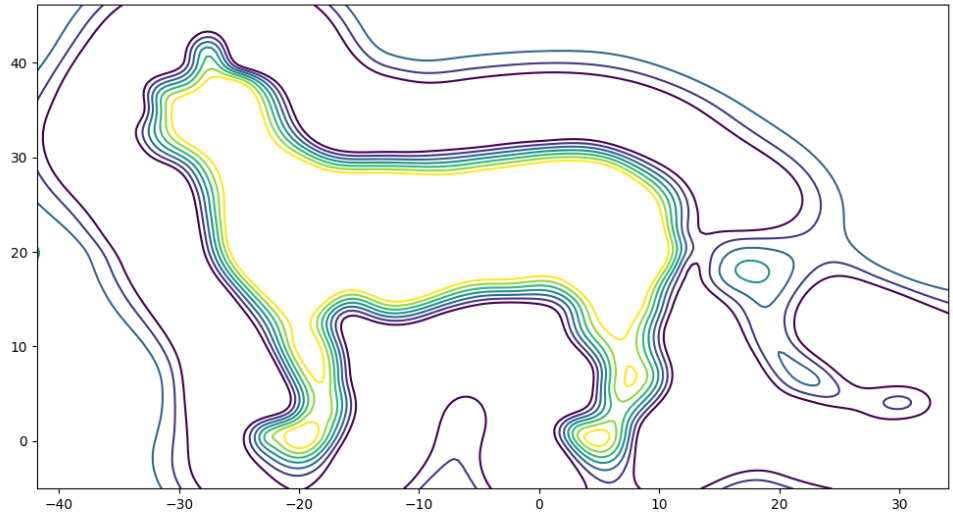



Figure 3.3: An 2d section through the interpolant of the cat model with 7 isolines from -1 to 1.

## 3.3 Smoothing

The resulting isosurfaces of the topology optimization generally had rough surfaces with seemingly random small surface features like bumbs and dents. These features are deemed unphysical but would be preserved in an interpolation. They would also be agressively refined since the normals are irregular.

To circumvent this, a mesh smoothing method was employed prior to the interpolation to smooth the isosurface mesh. Since the often used Laplace smoothing diminishes volume I opted for Taubin smoothing as described in (Taubin 1995).

The smoothing is very similar to the vertex-smoothing introduced above. For a vertex $v_i$ and neighbors $v_j$, the position of the vertex is shifted with a weighed average of the neigbors positions:

$$\Delta v_i = \sum_{j \in \mathcal{N}_i} w_{ij}(v_j - v_i)$$

where the weights $w_{ij}$ are just set as the inverse number of neighbors $w_{ij} = 1/|\mathcal{N}_i|$.

The shift is then added partially to the original vertex.

$$v_i' = v_i + \lambda \Delta v_i \qquad 0 < \lambda < 1$$

For Taubin smoothing a second smoothing step is introduced with a negative $\lambda$ i.e. a roughening. The coefficient for this is denoted $\mu$ with the restriction that $0 < \lambda < -\mu$. In (Taubin 1995) Taubin shows that repeated iteration of these two step act as a lowpass filter and limit the shrinkage of the model.

For the application I used values of $\lambda = 0.40$, $\mu = -0.50$ and ran 40 smoothing iterations. The smoothing effect of this is displayed in fig. 3.4.

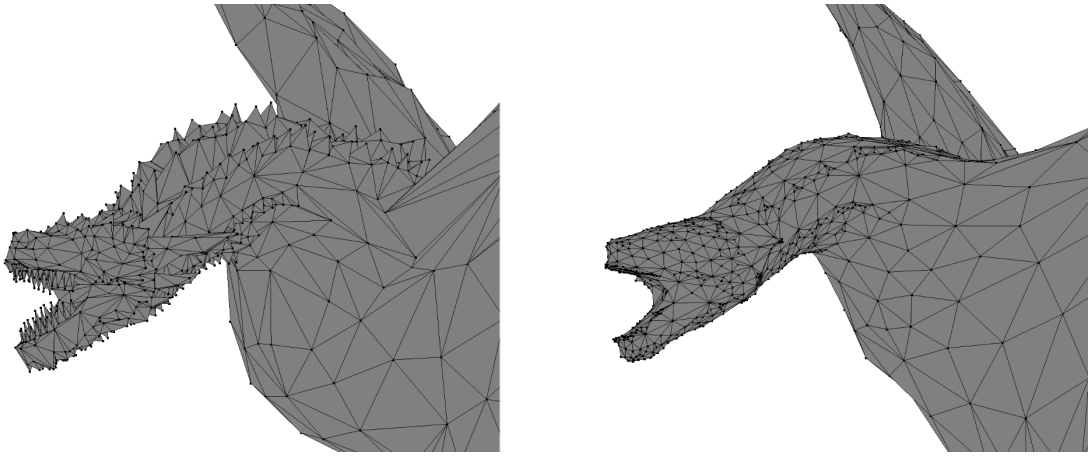


Figure 3.4: The mesh smoothing method applied to a dragon test model

## 3.4 Projection step

Inserting a new vertex in an edge split and smoothing a vertex requires a projection of a (mid-)point onto the surface. For sufficiently small distances to the surface this projection can be assumed to be orthogonal since the gradient has no components in the tangent plane to the isosurface and the tangent planes are mostly parallel. This is seen in fig. 3.3. However this no requirement since consecutive vertex smoothing will adjust the positions of the vertices on the surface.

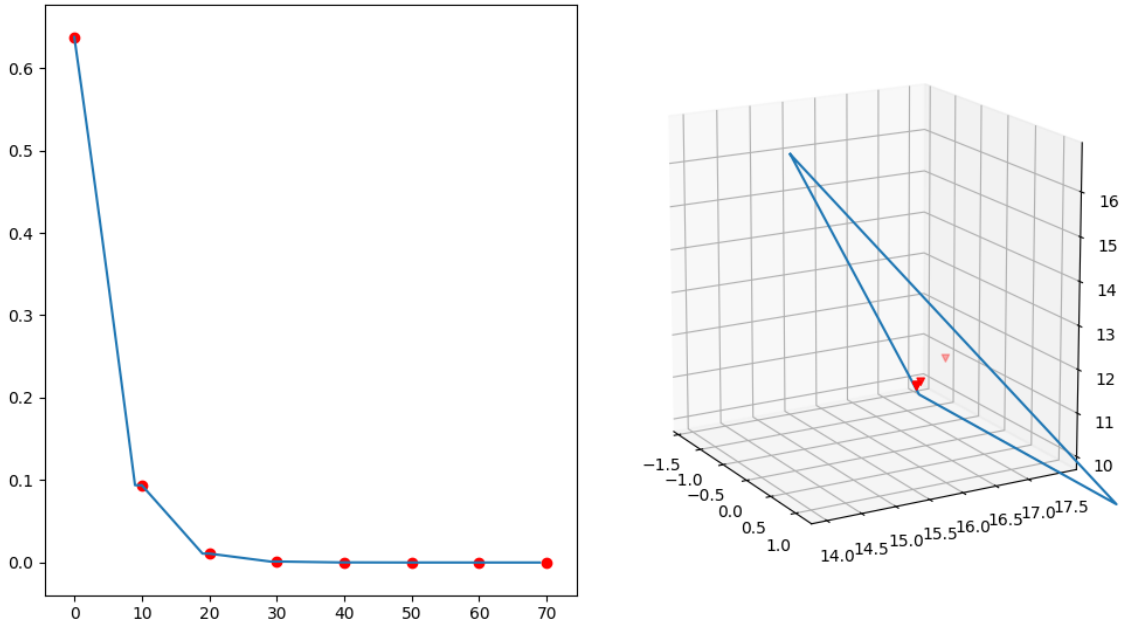



Figure 3.5: Fast convergence of the gradient descent onto the surface for a steplength of 1. On the right side the corresponding triangle is shown.

The convergence of this projection was generally fast and stayed within 8 iterations for all test cases to give an interpolant value of $1e^{-4}$. The convergence is depicted in fig. 3.5.

## 3.5 Remeshing

After the smoothed mesh is interpolated the remeshing begins.

Generally, points on the boundary are not included in the remeshing procedure as the coplanarity of points on the boundary could not be preserved. Also those points present an interface that should be fixed. If a reforming of the boundary was needed a seperate 2d interpolant could be constructed.

The remeshing algorithm is implemented as in (Dassi et al. 2016):

**Algorithm 2:** The remeshing procedure

**Input :** Target edge length $l_{6d}$ and $\sigma$ for HDE

```
i=0
while i <maxiter do
    smalledges={l^e_6d < 0.5 · l_6d for e in edges }
    j=0
    while smalledges and j < 10 do
        collapse smalledges
        smooth random 30% of vertices
        flip all edges
        update smalledges
        j+=1
    end
    longedges={l^e_6d > 1.5 · l_6d for e in edges }
    split long edges
    flip all edges; smooth all vertices
    flip all edges
end
```

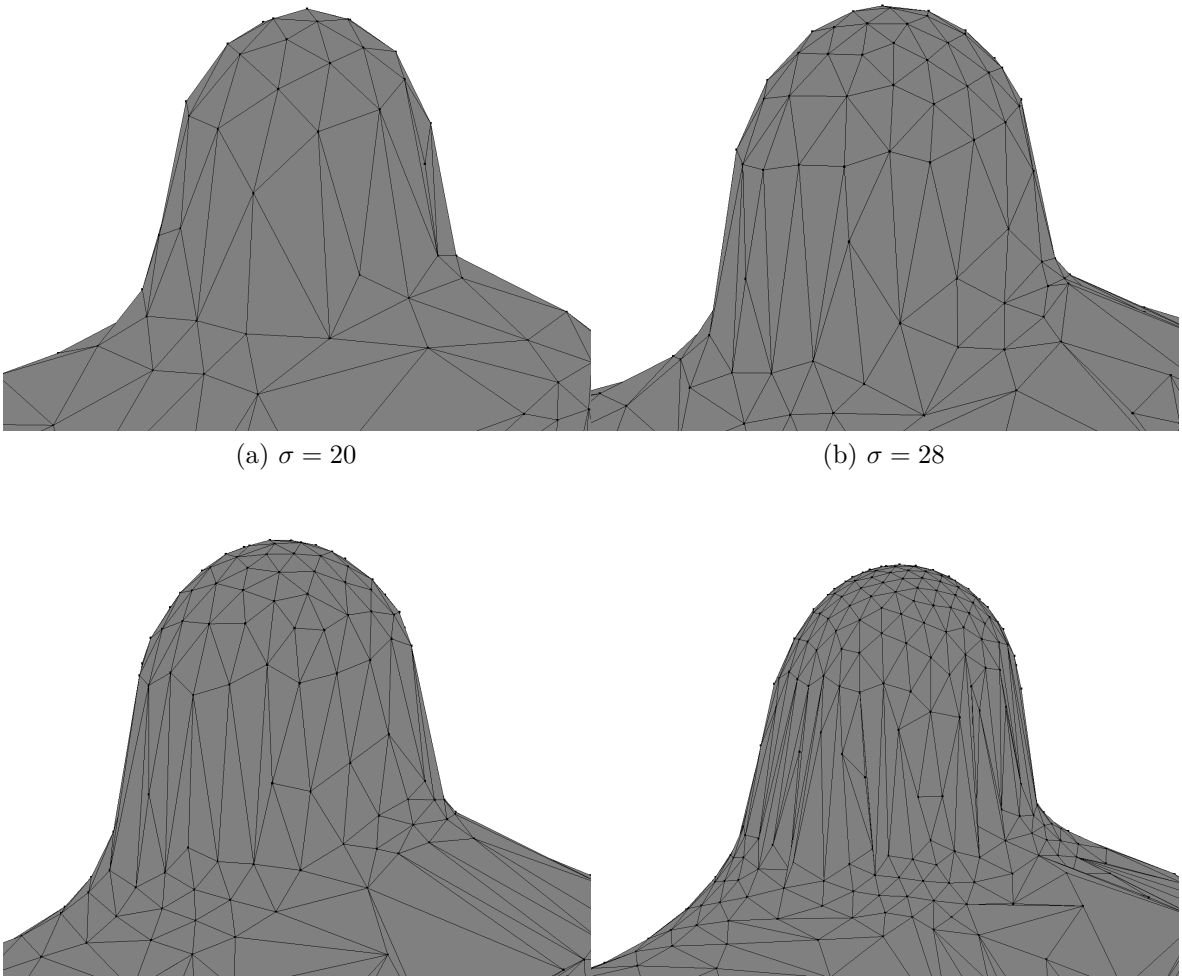
## 3.6 Higher dimensional embedding

The parameter for the embedding is $\sigma$. By the nature of the extension, the additional edgelength due to the normals is independent of the scaling of the original mesh. Therefor the values of $\sigma$ depend on the input mesh. A good default value is set with the following formula utilizing an enclosing box (or bounding box) $B$ of a model $m$.

$$\sigma_{default} = \frac{\text{size}_x(B_m) + \text{size}_y(B_m) + \text{size}_z(B_m)}{10}$$

The value 10 was obtained heuristically but depending on the model proportions and desired refinement, values as low as 5 or as high as 15 might be used.

The effect of different $\sigma$ are shown in fig. 3.6.



(a) $\sigma = 20$ (b) $\sigma = 28$

(c) $\sigma = 36$ (d) $\sigma = 50$

Figure 3.6: Greater values of $\sigma$ yield a refinement of curved surfaces. However, due to small variation in normals irregularities can occur. Notice also that some thin strip triangles can be seen in (d).

## 3.7 Implementation details

The programming language Python was used to build a datastructure for the mesh (that is named Trimesh in the code) and implement the remeshing operations from sec. 2.4 with it. This datastructure is an undirected graph made up of points with associated pointnormals and references to their neighbors. Additionally each point stores references to the edges and triangles it takes part in.

Also in this structure are lists of edge and triangle objects that each hold references to their points and to each other. A triangle object has labels A,B,C for its points and a,b,c for its edges where the lower letter edge is located opposite to the respective capital letter Point.

Also, the order of points in a triangle not only determines the normal vector but is used in finding triangles left and right to an edge (looking from the outside). Therefor the correctness of these information is of fundamental importance.

The richness[1] of this datastructure has the benefit that the remeshing operations could be written in a relatively compact form. The code for this datastructure and the remeshing algorithm can be found on the accompanying DVD or, should it be approved by the supervisors on github under this link.

To then optimize the performance of this datastructure it was ported to the C-language with the cython transpiler where additional datatypes were added.

However, performance was not a consideration from the beginning. And it was discovered that unstructured graphs of relatively large python objects have a bad cache performance. Adding to this, for large input meshes, the evaluation of the interpolant takes a lot of time since the distance to each center needs to be computed. And for vertex smoothing this needs to be done multiple times for all vertices in the mesh. That is to say, the topology optimization meshes that were remeshed in the following section took several hours to remesh even though smaller meshes of only a few thousand vertices could be remeshed in acceptable times from around 5 to 10 minutes.

I will reconsider this issue in the outlook section and give recommendations on how to improve this.

For the implementation the following third party libraries were used:

- numpy (array package for python)
- numpy-stl (stl mesh reader)
- glumpy (openGL wrapper)
- scipy (for sparse matrix linalg)
- cython (Python to C transpiler)

[1] All relations are stored directly and are not deduced on the fly

# Chapter 4

# Results

## 4.1 The topology optimization models

The implementations for the following calculations were provided by Moritz Ebeling-Rump from the Weierstraß Institute. They are computed using the pdelib library developed at the Weierstraß Institute and contain contributions from multiple authors. The parameters $\gamma$ and $\varepsilon$ as well as the Lamé-coefficients were set according to (Ebeling-Rump et al. 2019).

For reference, the models were calculated with the Lamé-coefficients set to those of the 3d-printing plastic PLA:

$$\lambda = 1599 \cdot 10^6 \quad \mu = 685 \cdot 10^6$$

For the calculations the gravitational volume force was set to zero since it had no visible influence for the small sized structures that were calculated. The time stepping parameter $\tau$ was generally set to $\tau = 0.01$.

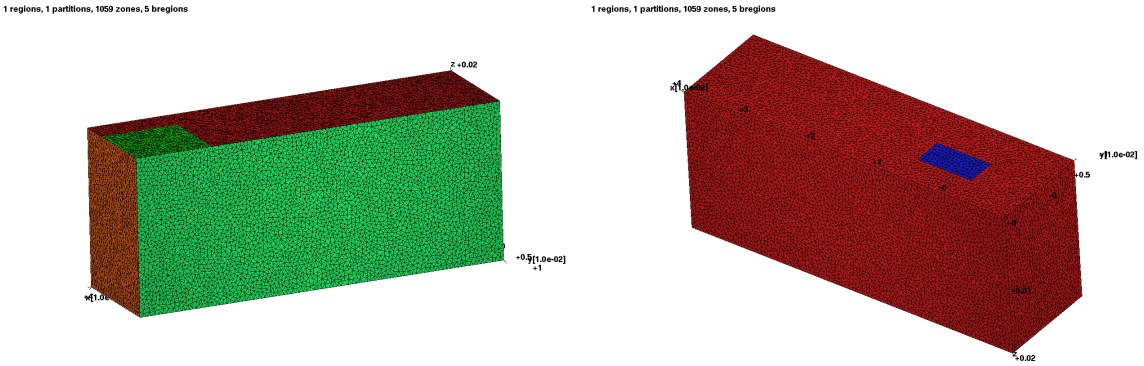
### 4.1.1 The bridge

The mesh for the bridge model posesses an x-y mirror symmetry and has a force square on top. The force square with a Neumann load is depicted as the green square on the

top of the domain in fig. 4.1a. The symmetry is imposed by a homogeneous Dirichlet condition in the x direction on the orange and in the y-direction on the green part of the domain also visible in the first picture. fig. 4.1bshows the Dirichlet clamp on the bottom (in blue) where x and y displacements are penalized.

The dimensions of the domain are 5cm in the x-direction, 1.5cm in the y-direction and 2cm in the z-direction. The force square has a size of 1 by 1 cm and a load density of 2400 N/m$^2$ in the z-direction resulting in a total load of 0.24N or approximately 24g.

The model was calculated using 294231 tetrahedra with the parameters for the Ginzburg-Landau term set to: $\gamma = 3.125e^{-5}$ and $\epsilon = 0.00175$




(a) (b)

Figure 4.1: The mesh for the bridge model from the front and the back

After 60 iterations the mesh in fig. 4.2 was extracted. In the closeup fig. 4.3 the it can be seen that the triangle size and shape is very unregular.

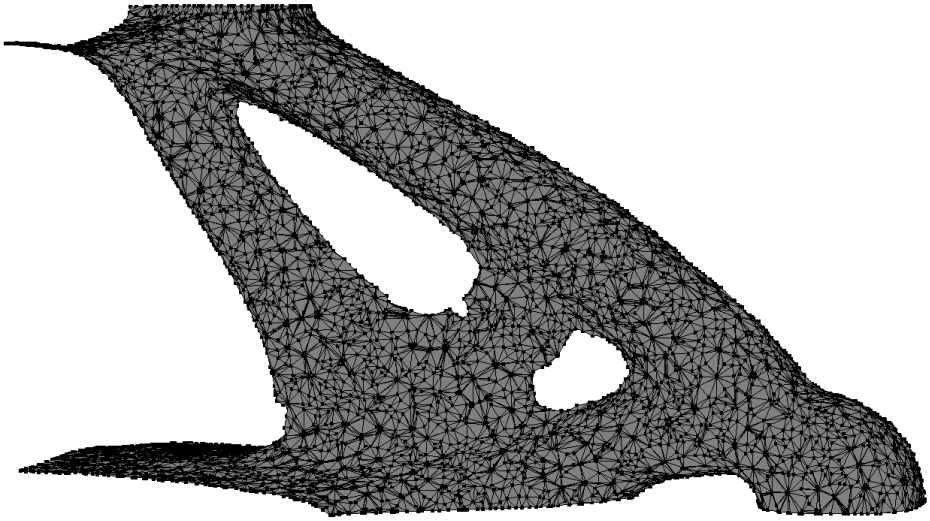


Figure 4.2: Isosurface of the bridge model after extraction. The Dirichlet clamp is located on the bottom right part

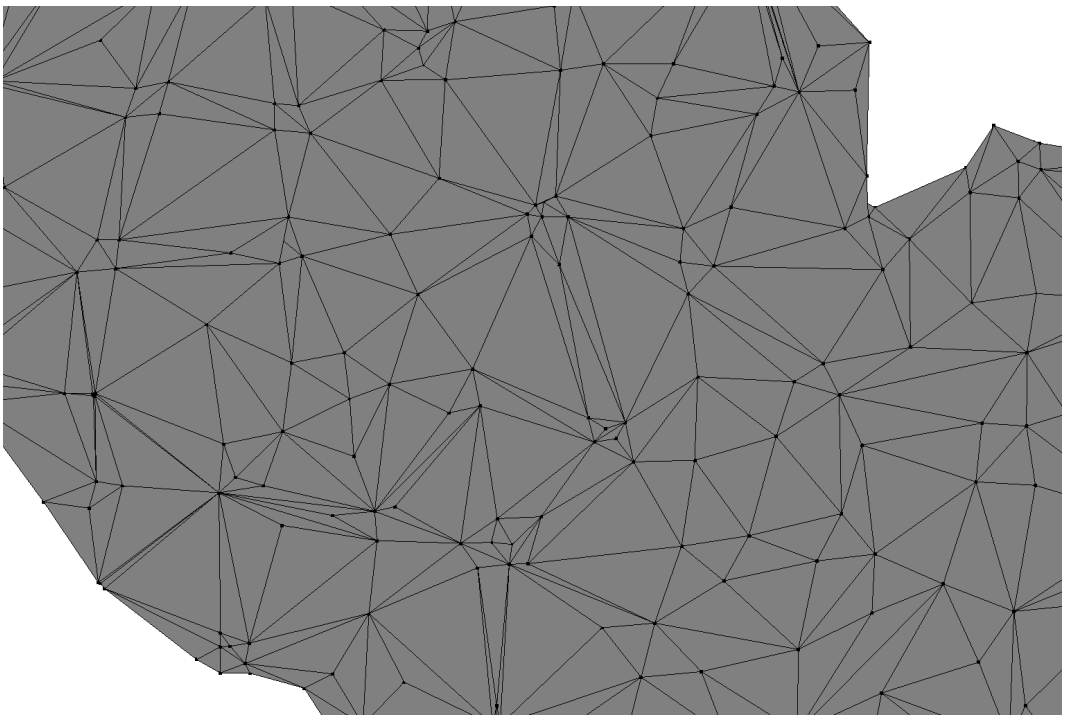


Figure 4.3: Closeup of the bridges isosurface. Some irregularities can be made out and the use of triangles is inefficient.

Subsequently the mesh was mirrored at the symmetry axes and merged into one stl model visible in fig. 4.4.

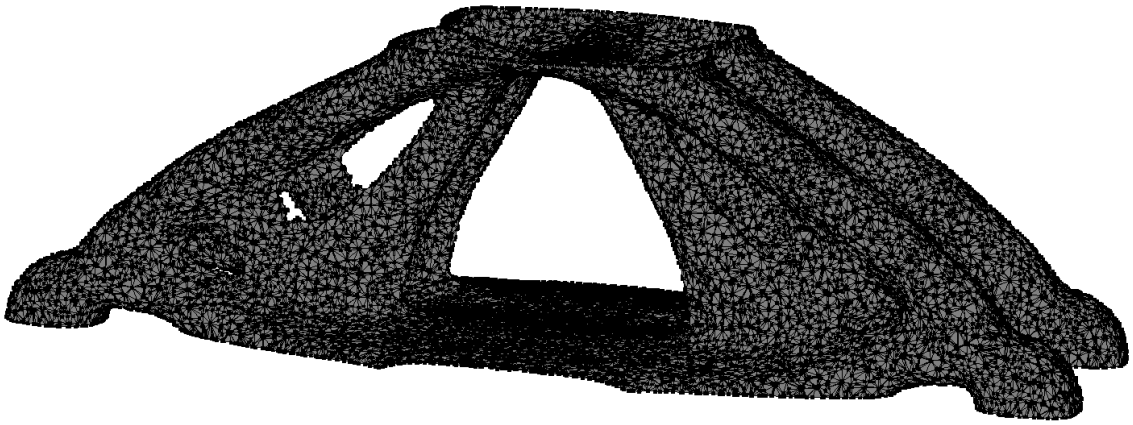


Figure 4.4: Bridge model after accounting for the mirror symmetry

The mesh was remeshed with 10 iterations and the target edge length set to two times that of the original mesh. The remeshed mesh is shown in fig. 4.5 where the target edge length is reflected in the triangle sizes of the flat areas. Again in

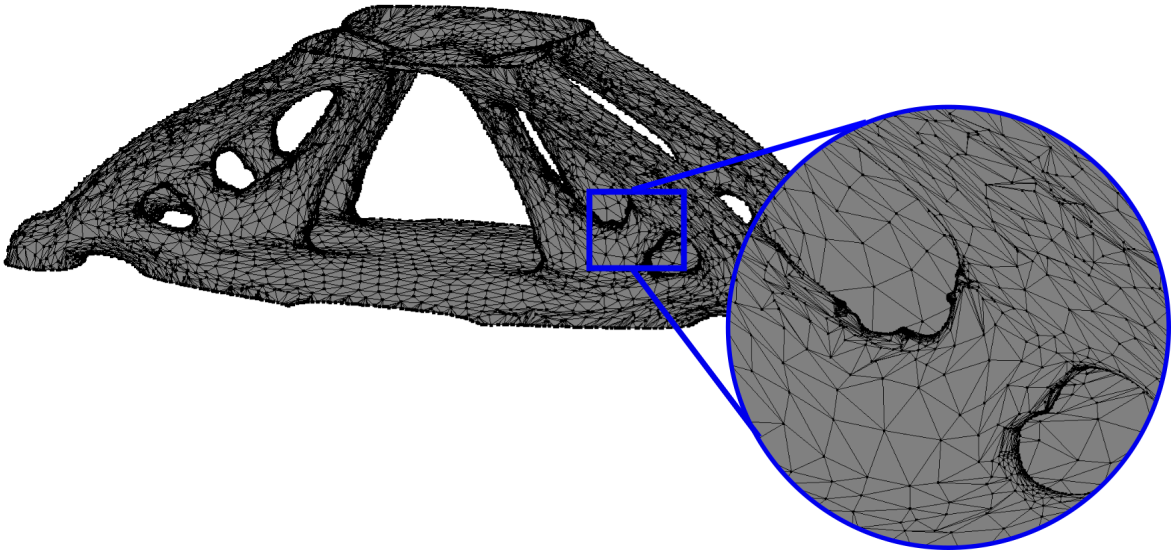
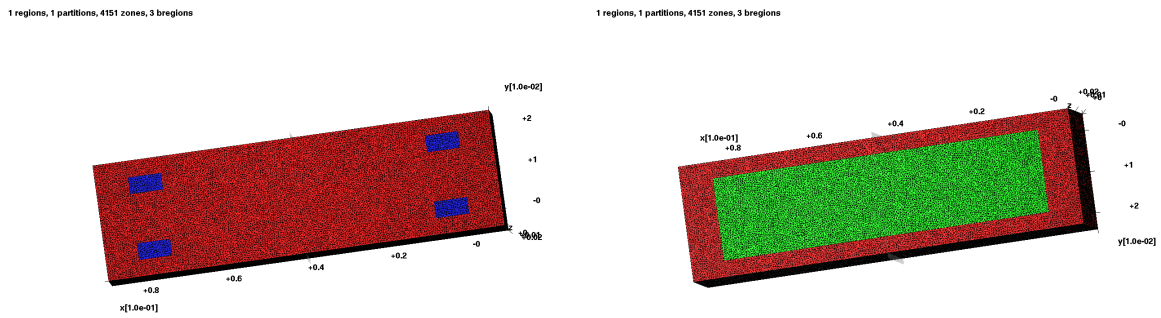


Figure 4.5: After remeshing with $l^{6d}$ = 2x longest edge and doing 10 iterations

### 4.1.2 The table

The table model has a large force rectangle with non-homogeneous neumann conditions on top and 4 Dirichlet conditions on the bottom. On the Dirichlet part only the z-direction was penalized(clamped) which results in the inclusion of x-y support structures. This is called a sliding condition.

The domain has size 9.6cm(x) by 2.8cm(y) by 2cm(z) and the force square is 8 x 2cm with a force density of 2400 N/m$^2$ resulting in a load of 3.84N or approximately 384g in graviational terms.




(a) (b)

Figure 4.6: The table mesh with a large force rectangle on the top and 4 x-y sliding Dirichlet conditions on the bottom.

The resulting isosurface is depicted in fig. 4.7. The model was calculated using 1120591 tetrahedra with the parameters for the Ginzburg-Landau term set to: $\gamma = 0.008$ and $\epsilon = 0.0035$

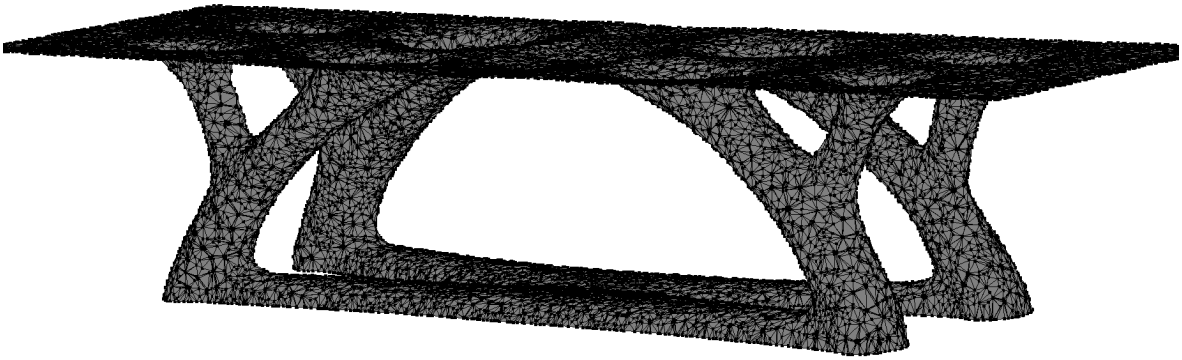


Figure 4.7: The extracted table isosurface

The remeshed surface did display erronous behaviour. oughening and

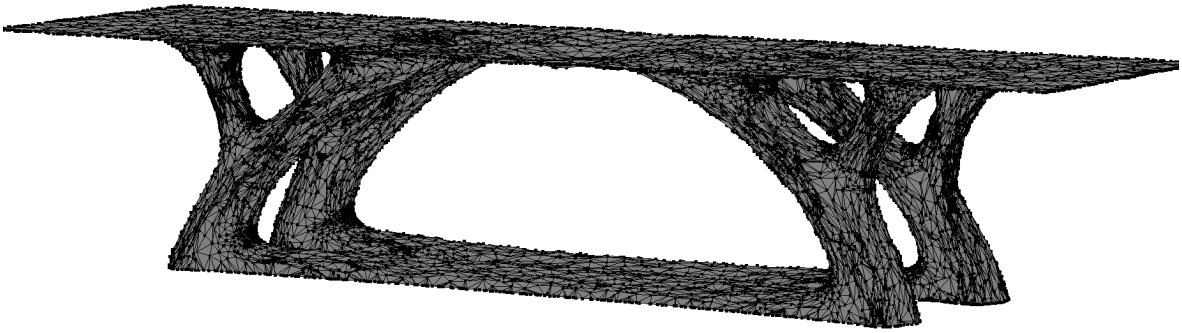


Figure 4.8: The remeshed table model.

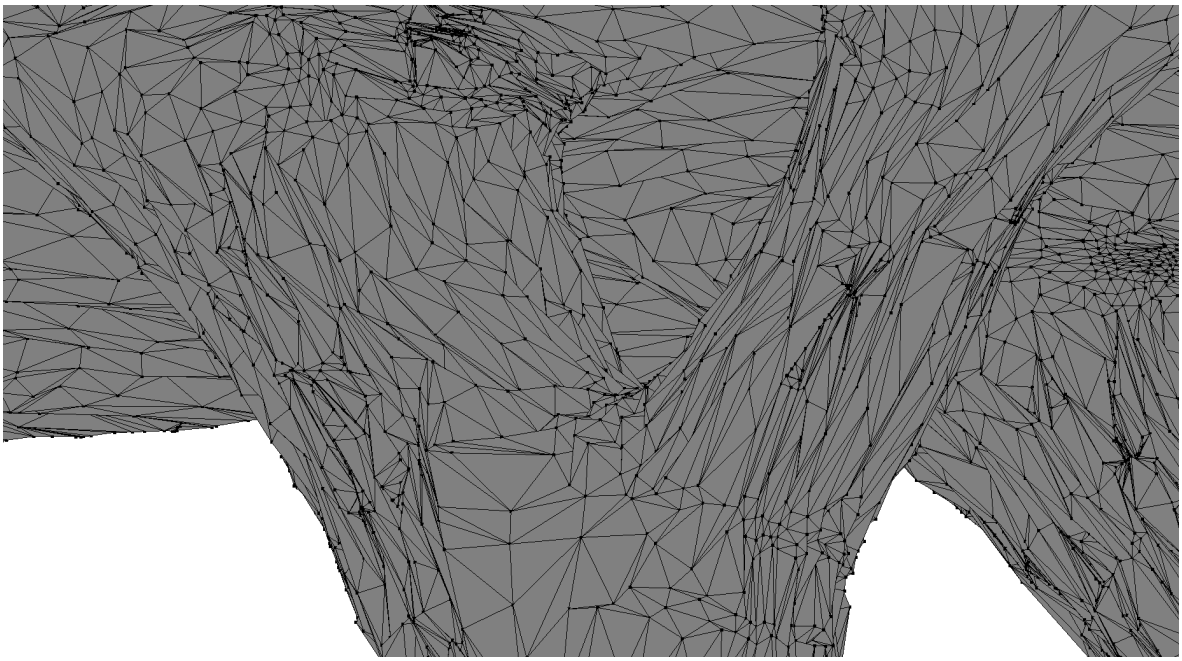
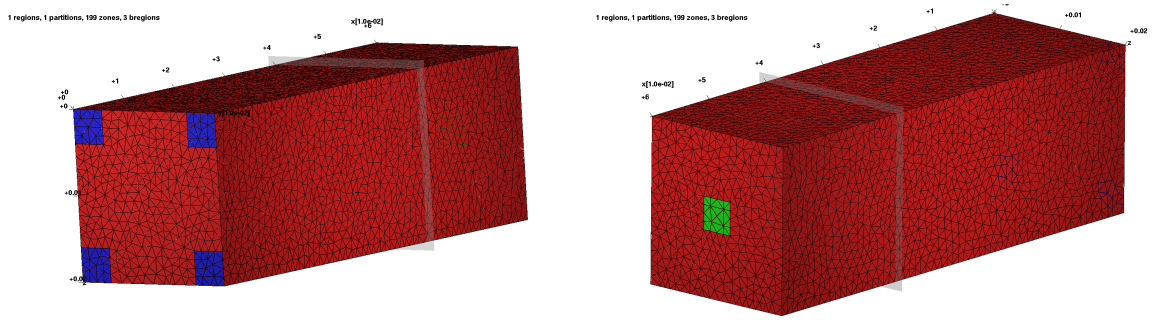


Figure 4.9: Closeup of the problematic remeshed table mesh.

### 4.1.3 The tower

The tower model has a similar layout to the table model above. A small force square on top applies a pressing force in the negative x-direction and 4 clamped Dirichlet squares function as force sinks.

The dimensions of the domain are 6cm by 2cm by 2cm and the force and Dirichlet squares were each sized .4cm by .4cm. The load density was set to 240000 N/m$^2$ such that again a resulting load of 3.84N or ~384g was applied.



(a) From the bottom (b) From the top

Figure 4.10: The grid for the tower model.

## 4.2 Analysis of the results and problem

### 4.2.1 General problems with normals as refinement markers

A general problem with normal based refinement is that normals are not a reliable marker for the mesh error i.e. the difference between interpolant and the triangular mesh. To see this consider the ear of the cat model from sec. 3.2 in fig. 4.11. The normals at the bottom of the ear point in the same direction as the ones on the top of the ear. Furthermore the long neighboring edges then exibit a sawtooth or cliff like pattern stemming from strong normal variations of the triangle normals. This feature is depicted in fig. 4.12.

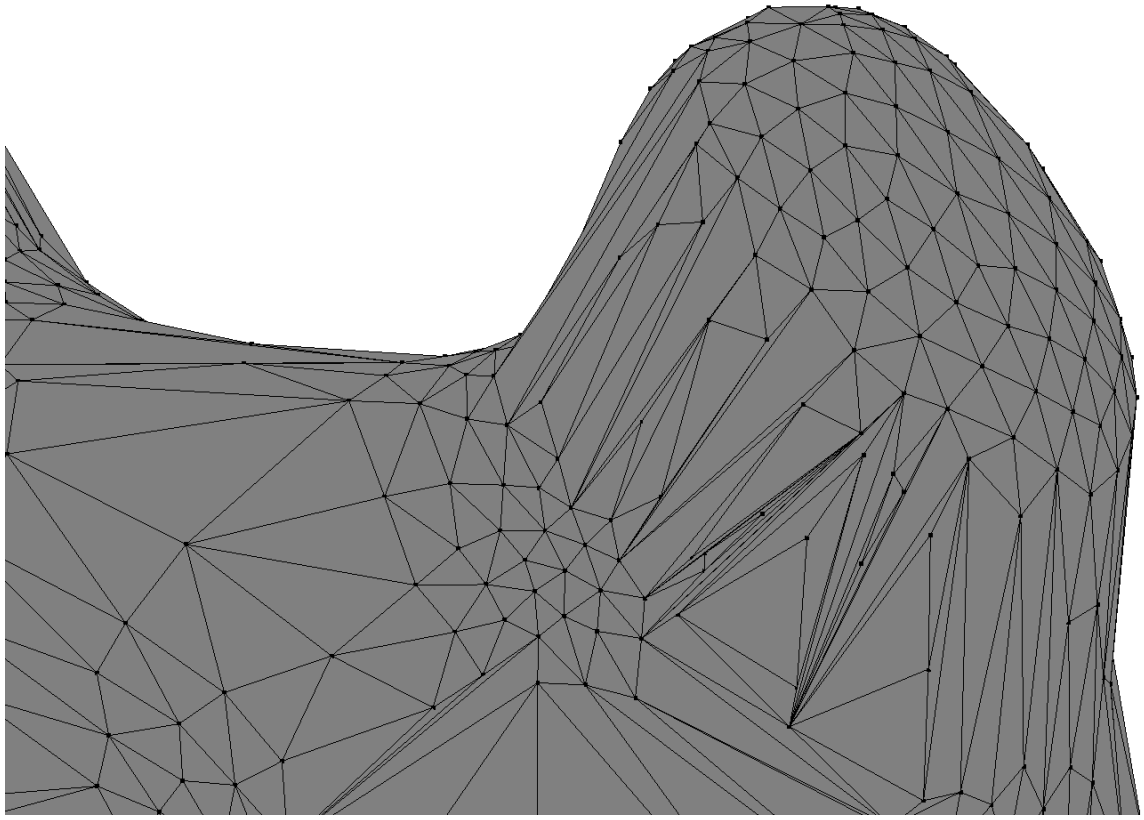


Figure 4.11: Coinciding normals at the bottom and the top of the cat ear.

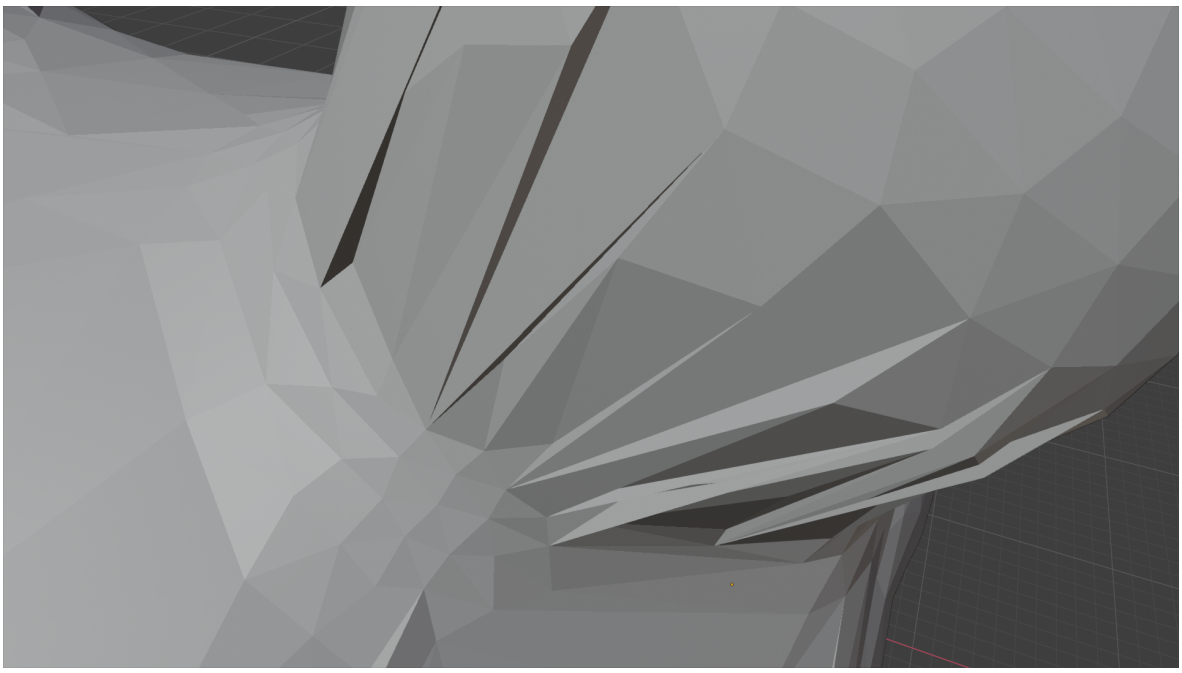


Figure 4.12: The cat ear in rendered with triangle normal based lightning.

Another possibility to generate an adaptive mesh without relying on the normals would be to use the interpolants values as an indicator for accuracy of meshpoints. This is possible because close to the surface the interpolant has the shape of the signed distance function. Using quadrature points on triangles/edges the volume/area under the surface defined by the approximant/interpolant and the mesh entity could be calculated. However this would rely heavily on evaluations of the interpolant.

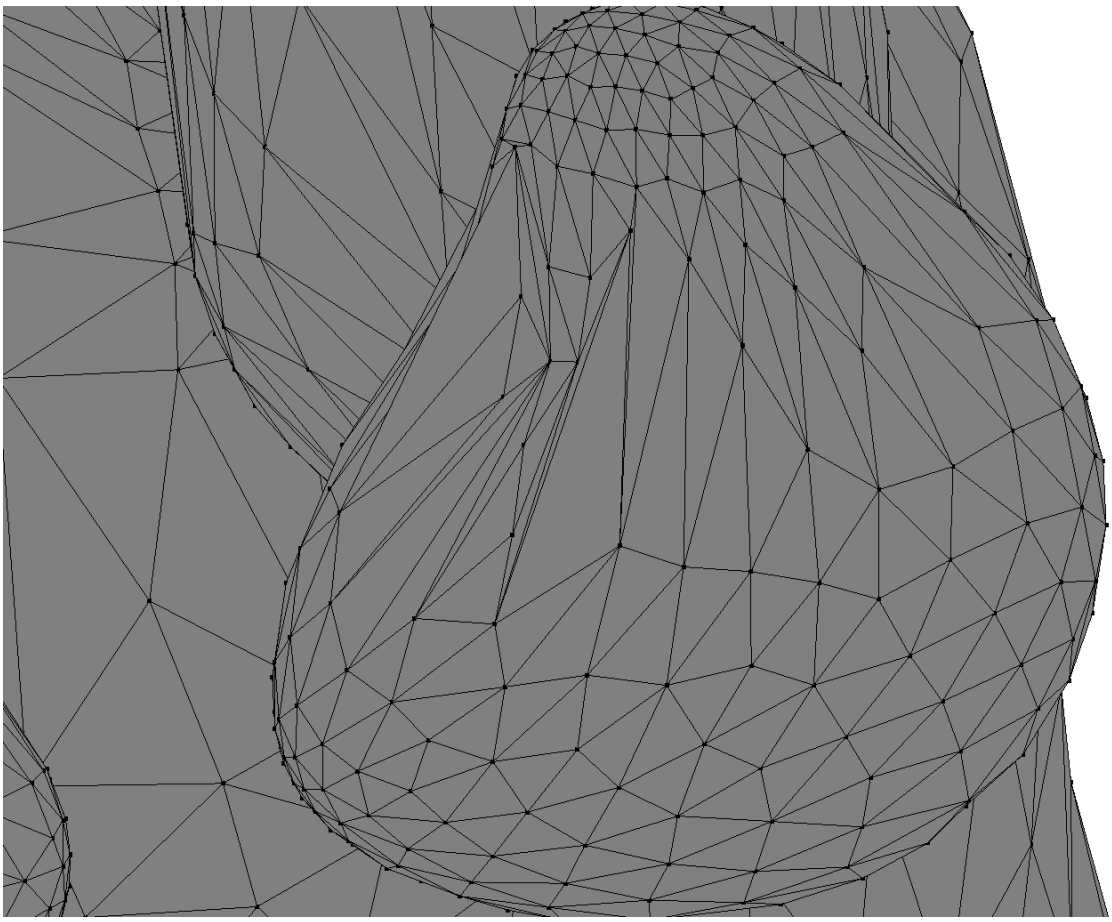


Figure 4.13: In the case of parallel normals a refinement is inhibited and the mesh error pertains.

# Chapter 5

# Problems, outlook and future work

As already stated in the algorithm section, the performance of the remeshing algorithm posed to be a major drawback for larger meshes. The easiest way to influence this is not to interpolate the input mesh but rather approximate the mesh with fewer functions. This would also provide a superior smoothing than the Taubin smoothing used here and would allow to use a dense interpolation matrix. In (Carr et al. 2001) this approach was realized with a greedy algorithm that starts with few interpolation centers and iteratively includes more centers into the quasi-interpolant where the error to the original mesh is largest. They can fit detailed meshes with small erros with 80.000 centers. As a comparison, the bridge mesh was interpolated with over 10 million centers.

Another improvement can be made by evaluating radial basis functions iteratively via the Fast Multipole method (see Buhmann 2003 chap. 7.3).

As the pliant remeshing relies on local mesh modifications that include deleting, manipulating and adding entities, an optimised datastructure can give a good performance benefit. With smaller data sizes and by reusing the memory of deleted objects, less fragmentation can be achieved which gives a better cache performance. An approach to this that I would recommend is to utilize an unrolled linked list with points, edges and triangles stored side-by side rather than using individual datastructures.

Also the vertices could be stored in a on-demand resized array to then utilize a vectorized(parallelised) version of the vertex smoothing. This would be ideally implemented in a statically typed programming language like C/C++ or the more modern Julia.

The surface interpolation with the local Wendland radial basis functions has worked well and was only dependent on choosing a fitting $\varepsilon$. The projection onto that surface was robust and converged fast.

Another possibility to generate an adaptive mesh without relying on the higher dimensional embedding is to use the interpolants values as an indicator for accuracy of meshpoints. This is possible because close to the surface the interpolant has the shape of the signed distance function. Using quadrature points on triangles/edges the volume/area under the surface defined by the approximant/interpolant and the mesh entity could be calculated.

- The unstructured graph datastructure scaled very poorly to larger meshes ie. excessive cache misses yielded a poor performance. This is a fundamental problem for pliant remeshing and may be adressed with good datastructures that preserve neighborhood to some degree.
- fitting with RBFs would have been more appropiate to incorporate the smoothing and reduce the number of RBFs. However, projections might not work so well without offset constraints.
- 6d flips work poorly for refinement due to mesh normals being a limited indicator of mesh accuracy -> some long edges remain due to equal normals.
- surface interpolation works well and projections were fast and reliable.

Immediate requirement for a practicable method: faster datastructure and mesh fitting.

# Appendix 1

## 5.1 Non manifold-errors of meshes

Triangluar meshes can have a multitude of minor errors that unfortunately are very much tolerated in cad-programs and 3D-software but can cause issues with

## 5.2 Interpolation with conditionally positive definite functions

**Definition:** A continuous even function $\Phi : \mathbb{R}^d \mapsto \mathbb{R}$ is conditionally positive (semi) definite of order m if and only if for all pairwise distinct $X = x_1, ..., x_N \subset \mathbb{R}^d$ and all $\alpha \in \mathbb{R}^d \setminus \{0\}$ that satisfy

$$\sum_j^N \alpha_j p(x_j) = 0 \qquad \forall p \in \Pi^m_{\mathbb{R}^d}$$

the quadratic form

$$\sum_{j,k}^N \alpha_j \alpha_k \Phi(x_j - x_k)$$

is (nonnegative) positive. Here $\Pi^m_{\mathbb{R}^d}$ is the space of polynomials of maximal order m.

**Theorem:** Let a univariate function $\phi \in C[0,\infty) \bigcap C^\infty(0,\infty)$ be given.
The mutlivariate function $\Phi$ defined by $\Phi = \phi(\|\cdot\|_2^2)$ is conditionally positive semi definite of order $m \in \mathbb{N}$ on every $\mathbb{R}^d$ if and only if $(-1)^m \phi^{(m)}$ is completely monotone on $(0,\infty)$ (Wendland 2005 theorem 8.19)

The interpolation condition is then to be modified and the the interpolant takes the

following form:

$$S(x) = \sum_{j=1}^{N} \alpha_j \Phi(x - x_j) + \sum_{k=1}^{Q} \beta_k p_k(x)$$

Where $Q$ is the dimension of the polynomial space and the $p_k$ form a basis. The interpolation condition is complemented by the the condition:

$$\sum_{j=1}^{N} \alpha_j p_k(x_j) = 0 \qquad 1 \leq k \leq Q$$

The linear interpolation system can then be written in the compact form:

$$\begin{pmatrix} A_\Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{5.1}$$

With $P = p_k(k_j) \in \mathbb{R}^{Q \times N}$ This system is solvable if $\Phi$ is conditionally positive definite (Wendland 2005 theorem 8.21)

# References

**Barles et al.** (**1993**) G. Barles et al. Front propagation and phase field theory. *SIAM Journal on Control and Optimization*. 31, 2 (Mar.-1993), 439–469. doi: 10.1137/0331021.

**Bendsoe and Sigmund** (**2004**) Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: Theory, methods, and applications*. Springer-Verlag. doi: 10.1007/978-3-662-05086-6.

**Blank et al.** (**2014**) Luise Blank et al. Relating phase field and sharp interface approaches to structural topology optimization. *Control, Optimisation and Calculus of Variations (ESAIM-COCV)*. 20, (2014), 1025–1058. Retrieved 5-Oct.-2019, from https://epub.uni-regensburg.de/34578/.

**Blank et al.** (**2010**) Luise Blank et al. Phase-field approaches to structural topology optimization. (2010). Retrieved 5-Oct.-2019, from https://epub.uni-regensburg.de/14656/.

**Blank et al.** (**2013**) Luise Blank et al. Primal-dual active set methods for allen–cahn variational inequalities with nonlocal constraints. *Numerical Methods for Partial Differential Equations*. 29, 3 (2013), 999–1030. doi: 10.1002/num.21742.

**Bossen and Heckbert** (**1998**) Frank Bossen and Paul Heckbert. A pliant method for anisotropic mesh generation. *Proceedings of the 5th International Meshing Roundtable*. (Oct.-1998).

**Bossen and Heckbert** (**n.d.**) Frank J. Bossen and Paul S. Heckbert. A pliant method for anisotropic mesh generation. 12.

**Bourdin and Chambolle** (**2003**) Blaise Bourdin and Antonin Chambolle. Design-

dependent loads in topology optimization. *ESAIM: Control, Optimisation and Calculus of Variations*. 9, (Jan.-2003), 19–48. doi: 10.1051/cocv:2002070.

**Braess** (**2013**) Dietrich Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer Spektrum.

**Buhmann** (**2003**) Martin D. Buhmann. *Radial basis functions*. Cambridge University Press.

**Carr et al.** (**2001**) J. C. Carr et al. Reconstruction and representation of 3D objects with radial basis functions. *Proceedings of the 28th annual conference on computer graphics and interactive techniques* (New York, NY, USA, 2001), 67–76. doi: 10.1145/383259.383266.

**Ciarlet** (**1990**) Philippe Ciarlet. Mathematical elasticity, volume i: Three-dimensional elasticity. *Acta Applicandae Mathematica*. 18, 2 (Feb.-1990), 190–195. doi: 10.1007/BF00046568.

**Dassi et al.** (**2016**) Franco Dassi et al. A novel surface remeshing scheme via higher dimensional embedding and radial basis functions. (2016).

**Ebeling-Rump et al.** (**2019**) Moritz Ebeling-Rump et al. Topology optimization subject to additive manufacturing constraints. *WIAS Preprints*. (Oct.-2019).

**Micchelli** (**1986**) Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*. 2, 1 (Dec.-1986), 11–22. doi: 10.1007/BF01893414.

**Müller and Wehle** (**1999**) Heinrich Müller and Michael Wehle. Visualization of implicit surfaces using adaptive tetrahedrizations. *Dagstuhl '97, scientific visualization* (Washington, DC, USA, 1999), 243–250. Retrieved 12-Nov.-2019, from http://dl.acm.org/citation.cfm?id=647367.723598.

**Takezawa et al.** (**2010**) Akihiro Takezawa et al. Shape and topology optimization based on the phase field method and sensitivity analysis. *Journal of Computational Physics*. 229, 7 (Apr.-2010), 2697–2718. doi: 10.1016/j.jcp.2009.12.017.

**Taubin** (**1995**) G. Taubin. Curve and surface smoothing without shrinkage. *Proceedings of IEEE international conference on computer vision* (Jun.-1995), 852–857. doi: 10.1109/ICCV.1995.466848.

**Treece et al.** (**1999**) G. M. Treece et al. Regularised marching tetrahedra: Improved iso-surface extraction. *Computers & Graphics*. 23, 4 (Aug.-1999), 583–598. doi: 10.1016/S0097-8493(99)00076-X.

**Wendland** (**1995**) Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*. 4, (1995), 389–396.

**Wendland** (**2005**) Holger Wendland. *Scattered data approximation*. Cambridge University Press.